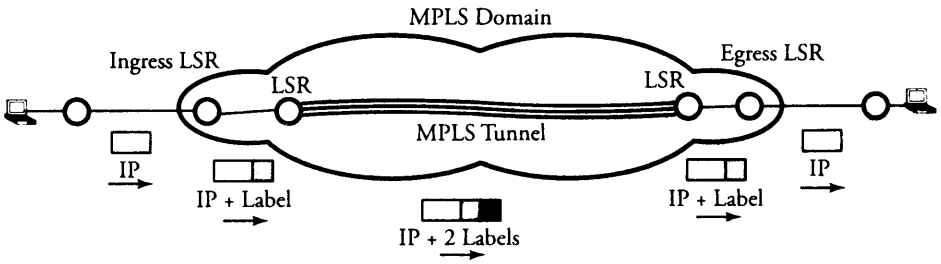


**Figure 16.8** An IP packet labeled in an MPLS domain and tunneled to reach the other end of the domain

label at the top of the stack determines the forwarding decision. The egress LSR strips the label, reads the IP packet header, and forwards the packet to its final destination.

## 16.2.3 Tunneling and Use of FEC

In an MPLS operation, any traffic is grouped into FECs. FEC implies that a group of IP packets are forwarded in the same manner—for example, over the same path or with the same forwarding treatment. A packet can be mapped to a particular FEC, based on the following criteria:

- Source and/or destination IP address or IP network addresses
- TCP/UDP port numbers
- Class of service
- Applications

As mentioned earlier, labels have only local significance. This fact removes a considerable amount of the network-management burden. An MPLS packet may carry as many labels as required by a network sender. The process of labeled packets can always be performed based on the top label. The feature of label stack allows the aggregation of LSPs into a single LSP for a portion of the route, creating an MPLS *tunnel*. Figure 16.8 shows an IP packet moving through an MPLS domain. When the labeled packet reaches the ingress LSR, each incoming IP packet is analyzed and classified into different FECs. This traffic-classification scheme provides the capability to partition the traffic for service differentiation.

Route selection can be done either hop by hop or by *explicit routing*. With hop-by-hop routing, each LSR can independently choose the next hop for each FEC. Hop-by-hop routing does not support traffic engineering, owing to limited available resources.

Explicit routing can provide all the benefits of traffic engineering. With explicit routing, a single LSR determines the LSP for a given FEC. For explicit routing, LSRs in the LSP are identified, whereas in an explicit routing, only some of the LSRs in an LSP are specified.

With the introduction of constraint-based routing, FEC can segregate the traffic into different levels of QoS, each with different service constraints, to support a variety of services, such as latency-based voice traffic and security-based VPN. At the beginning of the tunnel, an LSR assigns the same label to packets from a number of LSPs by pushing the label onto each packet's stack. At the other side of the tunnel, another LSR pops the top element from the label stack, revealing the inner label.

#### Label Distribution Protocol (LDP)

The *Label Distribution Protocol* (LDP) is a set of rules by which an LSR informs another LSR of an FEC. LDP enables two LSRs to understand each other's MPLS capabilities. LSP schemes are either *downstream on demand* or *downstream unsolicited*. With the downstream-on-demand scheme, an upstream node explicitly requests a label from a downstream node, and the downstream node forms the requested label. With the downstream-unsolicited scheme, a downstream node advertises a label mapping even without receiving any advance requests. Both types of LDPs can be used in explicit and hop-by-hop routing; however, a simple LDP can function using the routing protocol, such as OSPF, to design routes, since hop-by-hop routing does not follow the traffic engineering.

### 16.2.4 Traffic Engineering

High-quality connections can be expensive in an Internet service provider domain. *Traffic engineering* enables an ISP to route high-quality traffic to offer the best service to users in terms of throughput and delay. This way, traffic engineering reduces the cost of a network connection. Traffic engineering substitutes the need to manually configure network devices to set up explicit routes. In MPLS, traffic engineering is an automated scheme for control signaling and link bandwidth assignment and has a dynamic adaptation mechanism.

Traffic engineering can be either *traffic oriented* or *resource oriented*. Traffic-oriented traffic engineering relates to the optimization of such traffic performance parameters as the minimization of packet loss and delay and quick fault recovery when a node or a link fails. The resource-oriented technique engages in the optimization of network resource utilization.

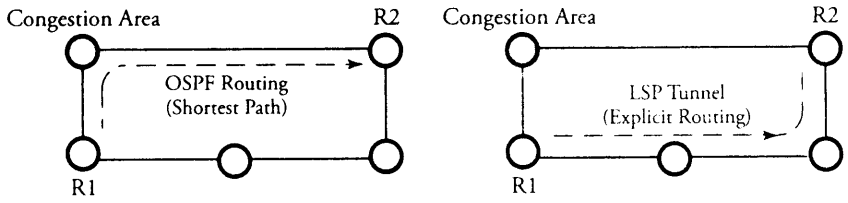


**Figure 16.9** A traffic engineering scenario

**Example.** Figure 16.9 shows an example of traffic engineering in MPLS. Assume that router R1 has a packet to send to R2. OSPF routes the packet through the shortest path, regardless of whether R3 is experiencing congestion. In an MPLS network, an LSP can be set up explicitly to avoid the congested node; if a constraint-based routing algorithm is used, an LSP avoiding the congested node is set up dynamically, even though the routing path is longer. This path-management capability is very appealing for traffic engineering.

**Example.** Figure 16.10 shows an example of traffic engineering in an MPLS network. The routing table is shown for router R2, where R2 advertises to R3, R5, and R6 that it can route to all three destinations C, D, and E. Hence, any frame with labels 31, 4, and 21, respectively, are switched toward these destinations.

### 16.2.5 MPLS-Based VPNs

Routine operations of virtual private networks require the use of both wide-area intradomain routing and interdomain routing schemes. A VPN's request to form a tunnel can be processed at the edge routers. For example, multiprotocol-based Border Gateway Protocol (BGP) makes MPLS-based VPN easier to manage VPN sites and VPN membership, mainly owing to the traffic engineering feature of MPLS. In an MPLS network, VPNs can be deployed by delivering the service using MPLS-aware subscriber equipment on the same infrastructure used for deploying Internet services.

An MPLS network domain acts as a backbone network between VPN users. Also, core LSRs act as *providing routers*, and edge routers act as *costumer edge routers*. Costumer edge routers distribute VPN information through MPLS-BGP to other providing routers. In order to forward an IP packet encapsulated for VPN through an MPLS backbone, the top label of the MPLS label stack is used to indicate the outgoing interface, and the second-level label is used to indicate the BGP next hop. When it receives a normal encapsulated IP packet from a router, an ingress costumer edge router

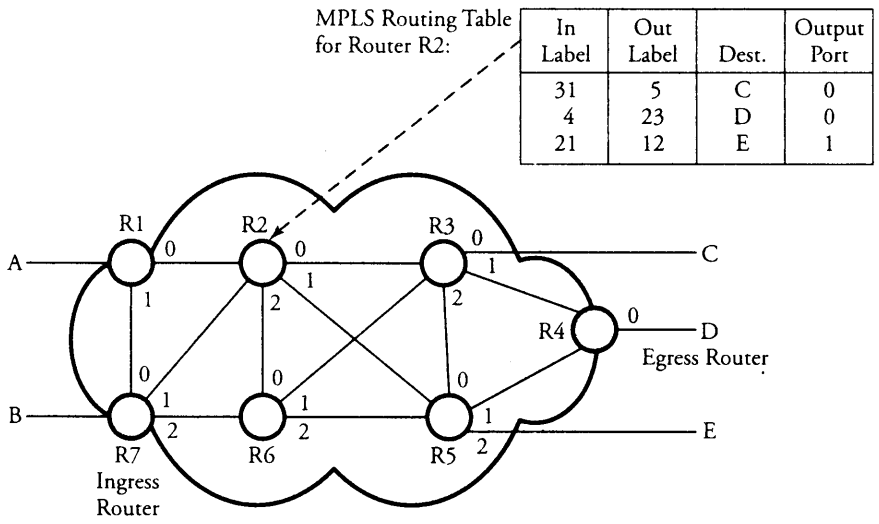


**Figure 16.10** Layer 3 routing in an MPLS network

performs an "IP longest match" and finds the next hop corresponding to the packet's home VPN. The second-to-last MPLS router forwards the packet and pops the top label so that the costumer edge router can forward the packet, based on the second-level label, which gives the VPN.

## 16.3 Overlay Networks

An *overlay network* is an application-specific computer network built on top of another network. In other words, an overlay network creates a virtual topology on top of the physical topology. This type of network is created to protect the existing network structure from new protocols whose testing phases require Internet use. Such networks protect packets under test while isolating them from the main networking infrastructure in a test bed.

Figure 16.11 shows an overlay network configured over a wide area network. Nodes in an overlay network can be thought of as being connected by logical links. In Figure 16.11, for example, routers $R_4$, $R_5$, $R_6$, and $R_1$ are participating in creating an overlay network where the interconnection links are realized as overlay logical links. Such a logical link corresponds to a path in the underlying network. An obvious example of these networks is the *peer-to-peer network*, which runs on top of the Internet. Overlay networks have no control over how packets are routed in the underlying network

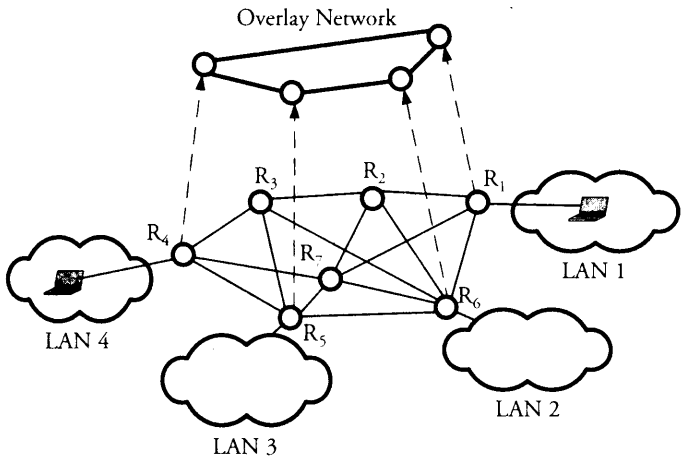


**Figure 16.11** An overlay network for connections between two LANs associated with routers $R_1$ and $R_4$

between a pair of overlay source/destination nodes. However, these networks can control a sequence of overlay nodes through a message-passing function before reaching the destination.

For various reasons, an overlay network might be needed in a communication system. An overlay network permits routing messages to destinations when the IP address is not known in advance. Sometimes, an overlay network is proposed as a method to improve Internet routing as implemented in order to achieve higher-quality streaming media. Sometimes, for the implementation of such techniques as DiffServ and IP multicast, modification of all routers in the network is required. In such cases, an overlay network can be deployed on end hosts running the overlay protocol software, without cooperation from Internet service providers.

Overlay networks are *self-organized*. When a node fails, the overlay network algorithm should provide solutions that let the network recover and recreate an appropriate network structure. Another fundamental difference between an overlay network and an unstructured network is that overlays' look-up routing information is on the basis of identifiers derived from the content of moving frames.

### 16.3.1 Peer-to-Peer (P2P) Connection

As an overlay network resembles a system consisting of various applications running on a single operating system, it could also resemble a set of tunnels that interconnect resources and users. The interconnects are carried out by *peer-to-peer* (P2P) protocols.

Let $\delta$ be the time required to establish a connection and $t_f$ be the time to finish the service as soon as the connection establishes. Assuming that the requests arrive at random to a peer node, the service time $s$ is

$$s = \begin{cases} t_f & \text{if the peer is connected} \\ t_f + \delta & \text{if the peer is not connected} \end{cases} \tag{16.1}$$

When a request arrives, the peer can be connected with probability $p_c$ and not connected with probability $1 - p_c$. Realizing that $s$ is a continuous random variable (see Appendix C) and is discrete for its two cases, discussed in Equation (16.1), the expected value of average service time is derived by

$$E[S] = (1 - p_c)t_f + p_c(t_f + \delta) = t_f + p_c\delta. \tag{16.2}$$

Let $\rho = \frac{\lambda}{\mu}$ be the utilization of the peer node, where $\lambda$ is the request arrival rate, and $\mu$ is the average service rate. Thus, a fraction $\rho$ of any given time that either "the peer uses for connecting" or "the connection is used for service" is expressed by

$$u_s = \rho \left( \frac{(1 - p_c)\delta)}{E[S]} \right). \tag{16.3}$$

Similarly, the fraction $1 - \rho$ that the same mentioned given time is idle can be derived by

$$u_i = (1 - \rho)p_c, \tag{16.4}$$

where the idle time occurs when the peer is either disconnected or connected but not using the connection. We can now derive an expression for the peer connection efficiency, $u$, as follows:

$$u = 1 - (u_s + u_i). \tag{16.5}$$

The connection efficiency of the peer can be used to determine the overall efficiency of the P2P connection.

## 16.4 Summary

Our discussion of *tunneling* issues began with *virtual private networks* (VPNs), which are virtually used by a private-sector entity over a public network. Tunneling is an encapsulation of a packet data segment to move from one protocol to another protocol at the same or higher layer. An organization using a VPN uses a service provider's IP network and builds a private network and runs its own traffic.

Tunneling has two forms: *remote-access* tunneling, which is a user-to-LAN connection, and *site-to-site* tunneling, whereby an organization can connect multiple fixed sites over a public network. Employees who are located beyond their main campus can use *point-to-point* (PPP) connections to create tunnels through the Internet into the organization's resources. Both PPTP and L2TP depend on PPP to frame tunneled packets.

*Multiprotocol label switching* (MPLS) improves the overall performance of traditional IP routing, especially for the establishment of more effective VPNs. In MPLS, multiple labels can be combined in a packet to form a header used by an LSR for efficient tunneling. The *Label Distribution Protocol* (LDP) is a set of rules by which an LSR informs another LSR. The MPLS *traffic engineering* feature is an automated scheme for a control-signaling process and link-bandwidth assignment to improve the quality of network management.

*Overlay networks* create a virtual topology on top of an existing physical topology on a public network. Overlay networks are *self-organized*; thus, if a node fails, the overlay network algorithm can provide solutions that let the network recreate an appropriate network structure.

The next chapter starts a new topic. We focus on multimedia networking, starting with the introductory concept of *compression of digital voice and video.*

## 16.5 Exercises

1. Label routing in MPLS networks is similar to VPN tunneling. Compare these two schemes in terms of
   (a) Traffic engineering capability
   (b) Security
2. Traffic engineering in MPLS networks can be done in the following two locations. Compare the advantages of each approach.
   (a) Egress nodes estimate routing to and from all ingress nodes.
   (b) A preassigned router estimates routing and propagates to all LSRs.
3. Consider the MPLS network shown in Figure 16.10; suppose that we want to use traffic engineering. Assume that the table of egress router R4 contains 26 (In Label), (Out Label), D (dest.), and 0 (Output Port). We want to route packets received at the ingress router R7 to egress router R4 through R1-R2-R5-R4.
   (a) Show the MPLS routing table for R1.
   (b) Show the MPLS routing table for R3.

4. Consider an overlay network that consists of five connected nodes. Compare two methods for a peer-to-peer connection on top of the Internet, using
   (a) A ring topology
   (b) A star topology
5. Suppose that two peers are connected to a four-node overlay network interconnected with a ring topology, as shown in Figure 16.11. There are 200 requests per second arriving at one of the peers, and the average service rate is 230 requests per second. Assuming that the time required to establish a connection is 10 ms, that the time to finish the service as soon as the connection establishes is 120 ms, and that there is an equal chance that the peer is connected or not connected:
   (a) Find the average service time of the peer.
   (b) Find the total utilization of the peer.
6. *Computer simulation project*. Carry the computer program you developed for the simulation of the seven-node network in Chapter 7 and add a subprogram to simulate the operation of overlay networks. To do this, use the computer program, and construct the seven-router network shown in Figure 16.11.
   (a) Simulate a packet transfer through the original network from $R_1$ to $R_4$.
   (b) Compare the results of part (a) to the case, using the overlay network shown in the figure.

CHAPTER 17

# Compression of Digital Voice and Video

This chapter looks at a new advanced topic: *multimedia networking*. This important topic requires a thorough understanding of the process of preparing compressed voice and video. A raw piece of information, whether voice or video, must be converted to digital form and then compressed to save link bandwidth. This chapter discusses methods of preparing digital voice and video for multimedia networks, including the analysis of information sources, source coding, and limits of data compression. Typical voice and video streaming compression techniques, such as JPEG, MPEG, and MP3, are explained. The major topics of this chapter are

- *Overview of data compression*
- *Digital voice and compression*
- *Still images and JPEG compression*
- *Moving images and MPEG compression*
- *Limits of compression with loss*
- *Compression process without loss*
- *Case study: FAX compression for transmission*

Our discussion starts with an overview of data preparation and compression, focusing on voice, image, and moving-image data. Considerable effort is required to turn analog voice into digital form. This process of converting from raw voice to

compressed binary form—sampling, quantization, and encoding—is known as *pulse code modulation* (PCM).

We then take a quick look at some practical compression methods without loss. We review the compression schemes that are used after the binary voice or image is prepared. We discuss both JPEG and MPEG compression techniques for still images and moving images, respectively. Still images or moving images might contain redundant or repeated elements that can be eliminated and codes substituted for future decoding process. We also summarize Shannon's limits of compression with loss. At the end of the discussion on compression, we review lossless compression techniques, such as *Huffman encoding* and *run-length encoding*. The chapter ends with a case study on FAX compression for transmission.

## 17.1 Overview of Data Compression

The benefits of data compression in high-speed networks are obvious. Following are those that are especially important for the compressed version of data.

- Less transmission power is required.
- Less communication bandwidth is required.
- System efficiency is increased.

There are, however, certain trade-offs with data compression. For example, the encoding and decoding processes of data compression increase the cost, complexity, and delay of data transmission. Both of the two processes of data compression are required for producing multimedia networking information: *compression with loss* and *compression without loss*.

In the first category of data compression, some less valuable or almost similar data must be eliminated permanently. The most notable case of compression with loss is the process of signal sampling. In this category, for example, is voice sampling (Section 17.2). With data compression without data loss, the compressed data can be recovered and converted back to its original form when received. This method of compression is typically applied to digital bits after sampling.

Figure 17.1 shows the basic information process in high-speed communication systems. Any type of "source" data is converted to digital form in a long *information-source* process. The outcome is the generation of digital words. Words are encoded in the *source coding* system to result in a compressed form of the data.

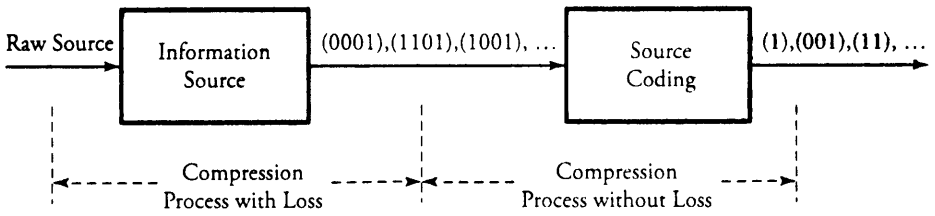


**Figure 17.1** Overview of information process and compression in multimedia networks

## 17.2 Digital Voice and Compression

Our discussion starts with the voice as a simple real-time signal. We first review the fundamentals of voice digitization and sampling.

### 17.2.1 Signal Sampling

In the process of digitalizing a signal, analog signals first go through a *sampling process*, as shown in Figure 17.2. The sampling function is required in the process of converting an analog signal to digital bits. However, acquiring samples from an analog signal and eliminating the unsampled portions of the signal may result in some permanent loss of information. In other words, the sampling resembles an *information-compression* process with loss.

Sampling techniques are of several types:

- *Pulse amplitude modulation* (PAM), which translates sampled values to pulses with corresponding amplitudes
- *Pulse width modulation* (PWM), which translates sampled values to pulses with corresponding widths
- *Pulse position modulation* (PPM), which translates sampled values to identical pulses but with corresponding positions to sampling points

PAM is a practical and commonly used sampling method; PPM is the best modulation technique but is expensive. PWM is normally used in analog remote-control systems. The sampling rate in any of these schemes obeys the *Nyquist theorem*, according to which at least two samples on all components of the spectrum are needed in order to reconstruct a spectrum:

$$f_S \geq 2f_H, \tag{17.1}$$

where $f_H$ is the highest-frequency component of a signal, and $f_S$ is the sampling rate.

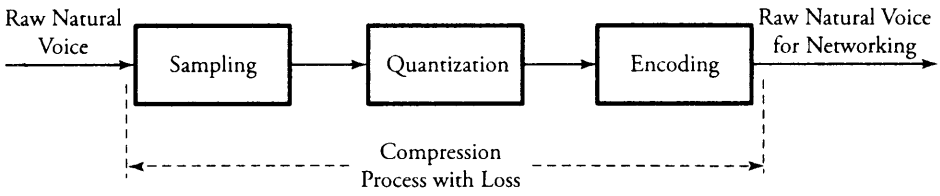


**Figure 17.2** Overview of digital voice process

## 17.2.2 Quantization and Distortion

Samples are real numbers—decimal-point values and integer values—and, thus, up to infinite bits are required for transmission of a raw sample. The transmission of infinite bits occupies infinite bandwidth and is not practical for implementation. In practice, sampled values are rounded off to available quantized levels. However, rounding off the values loses data and generates *distortion*. A measure is needed to analyze this distortion. The distortion measure should show how far apart a signal denoted by $x(t)$ is to its reproduced version, denoted by $\hat{x}(t)$. The distortion measure of a single source is the difference between source sample $X_i$ and its corresponding quantized value $\hat{X}_i$, denoted by $d(X, \hat{X})$, and is widely known as *squared-error distortion*:

$$d(X, \hat{X}) = (x - \hat{x})^2. \tag{17.2}$$

Note that $\hat{X}_i$ is noninvertible, since lost information cannot be recovered. The distortion measure of $n$ samples is based on the values of source samples obtained at the sampler output. As a result, the collection of $n$ samples forms a random process:

$$\mathbf{X}_n = \{X_1, X_2, \cdots, X_n\}. \tag{17.3}$$

Similarly, the reconstructed signal at the receiver can be viewed as a random process:

$$\hat{\mathbf{X}}_n = \{\hat{X}_1, \hat{X}_2, \cdots, \hat{X}_n\}. \tag{17.4}$$

The distortion between these two sequences is the average between their components:

$$d(\mathbf{X}_n, \hat{\mathbf{X}}_n) = \frac{1}{n} \sum_{i=1}^{n} d(X_i, \hat{X}_i). \tag{17.5}$$

Note that $d(X_n, X_n)$ itself is a random variable, since it takes on random numbers. Thus, the total distortion between the two sequences is defined as the expected value of $d(X_n, X_n)$:

$$\begin{aligned} D &= E[d(X_n, X_n)] = E\left[\frac{1}{n}\sum_{i=1}^{n} d(X_i, X_i)\right] \\ &= \frac{1}{n} E[d(X_1, X_1) + d(X_2, \hat{X}_2) + \cdots + d(X_n, \hat{X}_n)]. \end{aligned} \tag{17.6}$$

If all samples are expected to have approximately the same distortion denoted by $d(X, X)$, $d(X_1, X_1) = d(X_2, X_2) = \ldots = d(X_n, X_n) = d(X, X)$. By using squared error distortion, we obtain the total distortion:

$$D = \frac{1}{n}(nE[d(X, X])) = E[d(X, X)] = E[(X - X)^2] \tag{17.7}$$

Let $R$ be the minimum number of bits required to reproduce a source and guarantee that the distortion be less than a certain distortion bound $D_b$. Clearly, if $D$ decreases, $R$ must increase. If $X$ is represented by $R$ bits, the total number of different values $X_i$ takes is $2^R$. Each single-source output is quantized into $N$ levels. Each level 1, 2, ... , $N$ is encoded into a binary sequence. Let $\Re$ be the set of real numbers $\Re_1, \ldots, \Re_k, \ldots, \Re_N$, and let $\hat{X}_k$ be the quatized value belonging to subset $\Re_k$. Note that $\hat{X}_k$ is a quantized version of $X_k$. Apparently, $R = \log_2 N$ bits are required to encode $N$ quantized levels. Figure 17.3 shows a model of $N$-level quantization: For the subsets $\Re_1 = [-\infty, a_1]$, $\Re_1 = [a_1, a_2]$, ... , $\Re_8 = [a_{N-1}, \infty]$, the quantized values are $\hat{X}_1, \hat{X}_2, \cdots, \hat{X}_n$, respectively. We can use the definition of expected value to obtain $D$, as follows:

$$D = \int_{-\infty}^{+\infty} (x - x)^2 f_X(x) dx = \sum_{i=1}^{N} \int_{\Re_i} (x - x)^2 f_X(x) dx. \tag{17.8}$$

Typically, a distortion bound denoted by $D_b$ is defined by designers to ensure that $D_b \geq D$.

**Example.** Consider that each sample of a sampled source is a Gaussian random variable with a given probability distribution function $f_X(x) = 0.01e^{-\frac{1}{800}x^2}$. We want eight levels of quantization over the regions $\{a_1 = -60, a_2 = -40, \ldots, a_7 = 60\}$ and $\{x_1 = -70, x_2 = -50, \cdots, x_8 = 70\}$. Assuming that the distortion bound for this signal is $D_b = 7.2$, find out how four the real distortion, $D$ is to $D_b$.

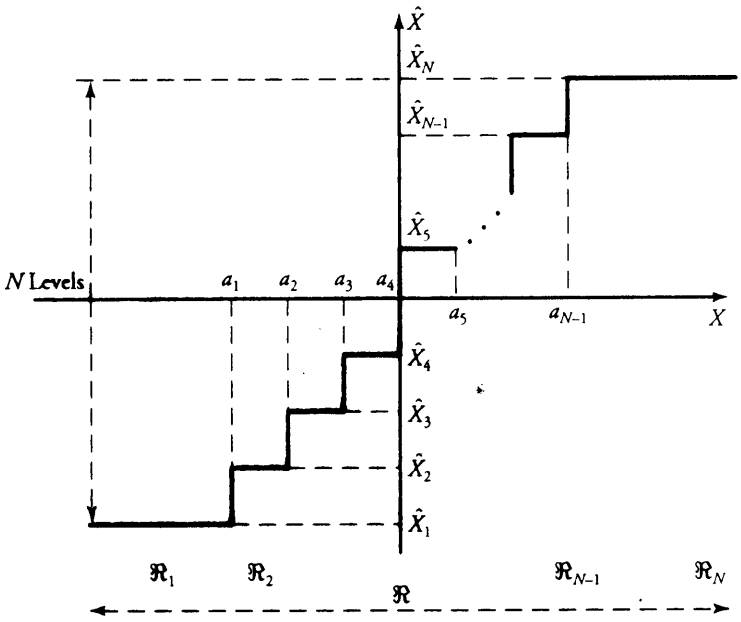


**Figure 17.3** $N$-level quantization

***Solution.*** Since $N = 8$, the number of bits required per sample is $R = \log_2 8 = 3$. Using this, $D$ can be developed further:

$$\begin{aligned} D &= \sum_{i=1}^{8} \int_{\Re_i} (X - \hat{X})^2 f_X(x)dx \\ &= \int_{-\infty}^{a_1} (x - \hat{x})^2 (0.01e^{-\frac{1}{800}x^2})dx + \sum_{i=2}^{7} \int_{a_{i-1}}^{a_7} (x - \hat{x})^2 (0.01e^{-\frac{1}{800}x^2})dx \\ &\quad + \int_{a_7}^{\infty} (x - \hat{x}_8)^2 (0.01e^{-\frac{1}{800}x^2})dx = 16.64. \end{aligned}$$

We note here that the total distortion as a result of eight-level quantization is 16.64, which is considerably greater than the given distortion bound of 7.2.

The conclusion in the example implies that the quantization chosen for that source may not be optimal. A possible reason may be inappropriate choices of $(a_1, \cdots, a_7)$ and/or $(\hat{x}_1, \cdots, \hat{x}_8)$. This in turn means that $R$ is not optimal.

**Optimal Quantizers**

Let $\Delta$ be the length of each region equal to $a_{i+1} - a_i$. Thus, regions can be restated as $(-\infty, a_1) \ldots (a_{N-1}, +\infty)$. Clearly, the limits of the upper region can also be shown by $(a_1 + (N-2)\Delta, +\infty)$. Then, the total distortion can be rewritten as

$$D = \int_{-\infty}^{a_1} (x - x) f_X(x) dx + \sum_{i=1}^{N-2} \int_{a_1+(i-1)\Delta}^{a_1+i\Delta} (x - x_{i+1})^2 f_X(x) dx + \int_{a1+(N-2)\Delta}^{\infty} (x - x_N)^2 f_X(x) dx. \tag{17.9}$$

For $D$ to be optimized, we must have $\frac{\partial D}{\partial a_i} = 0$, $\frac{\partial D}{\partial \Delta} = 0$ and given the following arrangements when $N$ is odd or even.

$$a_i = -a_{N-i} = -\left(\frac{N}{2} - i\right)\Delta \quad 1 \le i \le \frac{N}{2} \tag{17.10}$$

and

$$\hat{x}_i = -\hat{x}_{N+1-i} = -\left(\frac{N}{2} - i + \frac{1}{2}\right)\Delta. \tag{17.11}$$

For $N$ = odd: the optimal values, $\Delta_{opt}$ and $D_{opt}$, can be obtained as results of solving the two differential equations.

## 17.3 Still Images and JPEG Compression

This section investigates algorithms that prepare and compress still and moving images. The compression of such data substantially affects the utilization of bandwidths over the multimedia and IP networking infrastructures. We begin with a single visual image, such as a photograph, and then look at video, a motion image. The *Joint Photographic Experts Group* (JPEG) is the compression standard for still images. It is used for grayscale and quality-color images. Similar to voice compression, JPEG is a lossy processes. An image obtained after the decompression at a receiving end may not be the same as the original.

Figure 17.4 gives an overview of a typical JPEG process , which consists of three process: *discrete cosine transform* (DCT), *quantization*, and *compression*, or encoding.

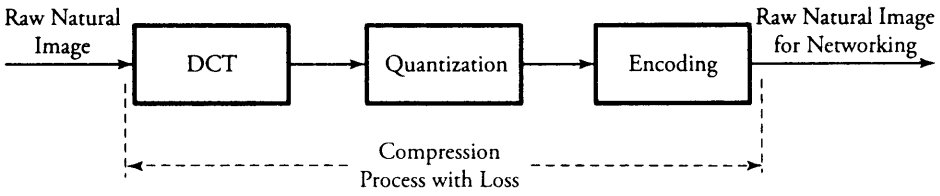


**Figure 17.4** A typical JPEG process for production and compression of still images

The DCT process is complex and converts a snapshot of a real image into a matrix of corresponding values. The quantization phase converts the values generated by DCT to simple numbers in order to occupy less bandwidth. As usual, all quantizing processes are lossy. The compression process makes the quantized values as compact as possible. The compression phase is normally lossless and uses standard compression techniques. Before describing these three blocks, we need to consider the nature of a digital image.

## 17.3.1 Raw-Image Sampling and DCT

As with a voice signal, we first need samples of a raw image: a *picture*. Pictures are of two types: *photographs*, which contain no digital data, and *images*, which contain digital data suitable for computer networks. An image is made up of $m \times n$ blocks of picture units, or *pixels*, as shown in Figure 17.5. For FAX transmissions, images are made up of 0s and 1s to represent black and white pixels, respectively. A *monochrome image*, or *black-and-white image*, is made up of various shades of gray, and thus each pixel must be able to represent a different shade. Typically, a pixel in a monochrome image consists of 8 bits to represent $2^8 = 256$ shades of gray, ranging from white to black, as shown in Figure 17.6.

### JPEG Files

Color images are based on the fact that any color can be represented to the human eye by using a particular combination of the base colors red, green, and blue (RGB). Computer monitor screens, digital camera images, or any other still color images are formed by varying the intensity of the three primary colors at pixel level, resulting in the creation of virtually any corresponding color from the real raw image. Each intensity created on any of the three pixels is represented by 8 bits, as shown in Figure 17.6. The intensities of each 3-unit pixel are adjusted, affecting the value of the 8-bit word to produce the desired color. Thus, each pixel can be represented by using 24 bits, allowing $2^{24}$ different colors. However, the human eye cannot distinguish all colors

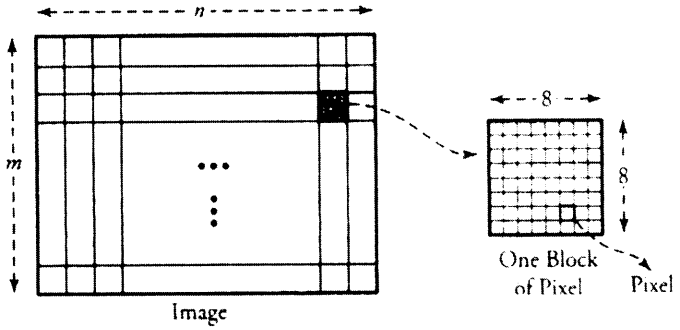


**Figure 17.5** A digital still image

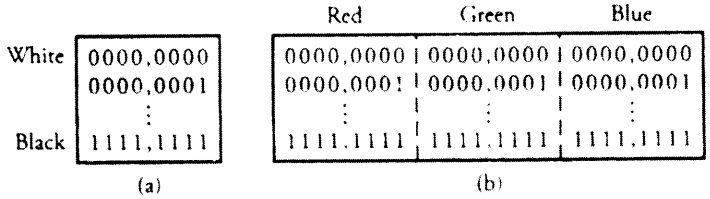


**Figure 17.6** Still image in bits (a) monochrome codes for still image; (b) color codes for still image

among the $2^{24}$ possible colors. The number of pixels in a typical image varies with the image size.

**Example.** A JPEG-based computer screen can consist of 1,024 × 1,280 pixels. Consequently, this computer image requires (1,024 × 1,280) × 24 = 31,457,280 bits. If a video consists of 30 images per second, a 943 Mb/s bandwidth is recquired.

### GIF Files

JPEG is designed to work with full-color images up to $2^{24}$ colors. The *graphic interchange format* (GIF) is an image file format that reduces the number of colors to 256. This reduction is the number of possible colors is a trade-off between the quality of the image and the transmission bandwidth. GIF stores up to $2^8 = 256$ colors in a table and covers the range of colors in an image as closely as possible. Therefore, 8 bits are used to represent a single pixel. GIF uses a variation of Lempel-Ziv encoding (Section 17.6.3) for compression of an image. This technology is used for images whose color detailing is not important, such as cartoons and charts.

### DCT Process

The *discrete cosine transform* (DCT) is a lossy compression process that begins by dividing a raw image into a series of standard $N \times N$ *pixel* blocks. For now, consider a monochrome block of pixels. With a standard size of $N = 8$, $N$ is the number of pixels per row or column of each block. Let $x, y$ be the position of a particular pixel in a block where $0 \le x \le N-1$ and $0 \le y \le N-1$. Hence, a gray scale of a given pixel $x, y$ can get an integer value in $\{0, 1, 2, \ldots, 255\}$. For an $N \times N$ pixel block, the DCT process is summarized in two steps.

1. Form a $\mathbf{P}[x][y]$ matrix to represent the collection of light-intensity values taken from various points of a real row image.
2. Convert the values of $\mathbf{P}[x][y]$ matrix to matrix with normalized and reduced values denoted by $\mathbf{T}[i][j]$ obtained as follows.

The objective of matrix $\mathbf{T}[i][j]$ is to create as many 0s as possible instead of small numbers in the $\mathbf{P}[x][y]$ matrix in order to reduce the overall bandwidth required to transmit the image. Similarly, matrix $\mathbf{T}[i][j]$ is a two-dimensionl array with $N$ rows and $N$ columns, where $0 \le i \le N-1$ and $0 \le j \le N-1$. The elements of $\mathbf{T}[i][j]$ are known as *spatial frequencies* and are obtained from

$$T[i][j] = \frac{2}{N} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} P[x][y] \cos\left(\frac{\pi i(2x+1)}{2N}\right) \cos\left(\frac{\pi i(2y+1)}{2N}\right), \quad (17.14)$$

where

$$C(i) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } i = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$C(j) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } j = 1 \\ 1 & \text{otherwise} \end{cases}$$

**Example.** An $8 \times 8$ matrix $\mathbf{P}[x][y]$ for an image is formed as shown in Figure 17.7 (a). This matrix is converted to matrix $\mathbf{T}[i][j]$ by using Equation (17.14). This example clearly shows that a matrix as $\mathbf{P}[x][y]$ consisting of 64 values can be converted to $\mathbf{T}[i][j]$ with 9 values and 55 zeros. It is easy to figure out the advantages of this conversion. The 55 zeros can be compressed, conveniently resulting in significant reduction of transmission bandwidth.

$$\begin{bmatrix} 22 & 31 & 41 & 50 & 60 & 69 & 80 & 91 \\ 29 & 42 & 52 & 59 & 71 & 80 & 90 & 101 \\ 40 & 51 & 59 & 70 & 82 & 92 & 100 & 110 \\ 51 & 62 & 70 & 82 & 89 & 101 & 109 & 119 \\ 60 & 70 & 82 & 93 & 100 & 109 & 120 & 130 \\ 70 & 82 & 90 & 100 & 110 & 121 & 130 & 139 \\ 79 & 91 & 100 & 110 & 120 & 130 & 140 & 150 \\ 91 & 99 & 110 & 120 & 130 & 140 & 150 & 160 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 716 & -179 & 0 & -19 & 0 & -6 & 0 & -1 \\ -179 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b)

**Figure 17.7** Matrix examples: (a) **P**[*x*][*y*]; (b) **T**[*i*][*j*]

The spatial frequencies depend directly on how much the pixel values change as functions of their positions in a pixel block. Equation (17.14) is set up such that the generated matrix **T**[*i*][*j*] contains many 0s, and the values of the matrix elements generally become smaller as they get farther away from the upper left position in the matrix. The values of **T**[*i*][*j*] elements at the receiver can be converted back to matrix **P**[*x*][*y*] by using the following function:

$$\mathbf{P}[x][y] = \frac{2}{N} C(i)C(j) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} T[i][j] \cos\left(\frac{\pi i(2i+1)}{2N}\right) \cos\left(\frac{\pi i(2j+1)}{2N}\right). \quad (17.15)$$

Note that as values become farther away from the upper-left position, they correspond to a fine detail in the image.

## 17.3.2 Quantization

To further scale down the values of **T**[*i*][*j*] with fewer distinct numbers and more consistent patterns to get better bandwidth advantages, this matrix is *quantized* to another matrix, denoted by **Q**[*i*][*j*]. To generate this matrix, the elements of matrix **T**[*i*][*j*] are divided by a standard number and then rounded off to their nearest integer. Dividing **T**[*i*][*j*] elements by the same constant number results in too much loss. The values of elements on the upper-left portion of the matrix must be preserved as much as possible, because such values correspond to less subtle features of the image. In contrast, values of elements in the lower-right portions correspond to the highest detail of an image. To preserve as much information as possible, the elements of **T**[*i*][*j*] are

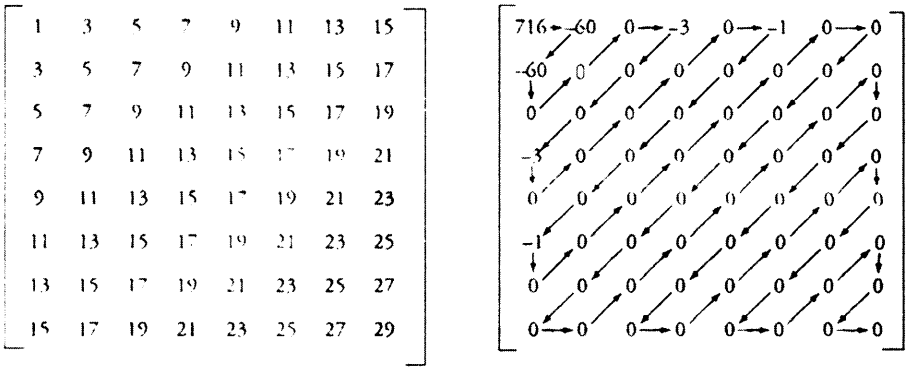

**Figure 17.8** (a) Divisor matrix **D**[*i*][*j*] and the quantization of a still image to produce; (b) matrix **Q**[*i*][*j*] and the order of matrix elements for transmission

divided by the elements of an $N \times N$ matrix denoted by $D[i][j]$, in which the values of elements decrease from the upper-left portion to the lower-right portion.

**Example.** Consider again the $8 \times 8$ matrix $\mathbf{P}[x][y]$ already converted to matrix $\mathbf{T}[i][j]$ shown in Figure 17.7. Figure 17.8 (a) shows a divisor matrix $\mathbf{D}[i][j]$. Figure 17.8 (b) shows the corresponding matrix $\mathbf{Q}[i][j]$ resulting from the quantization process. Note particularly that big values, such as −179 have become smaller and that some such values, as −1 in the upper-right corner turn into 0, making it easier for compression.

We notice here that the process of quantization, as discussed at the beginning of this chapter, is not quite reversible. This means that the value of $Q[i][j]$ cannot be exactly converted back to $T[i][j]$, owing mainly to rouding the values after dividing them by $D[i][j]$. For this reason, the quantization phase is a lossy process.

## 17.3.3 Encoding

In the last phase of the JPEG process, encoding finally does the task of compression. In the quantization phase, a matrix with numerous 0s is produced. Consider the example shown in Figure 17.8 (b). The **Q** matrix in this example has produced 57 zeros from the original raw image. A practical approach to compressing this matrix is to use run-length coding (Section 17.6.1). If run-length coding is used, scanning matrix $\mathbf{Q}[i][j]$ row by row may result in several phrases.

A logical way to scan this matrix is in the order illustrated by the arrows in Figure 17.8 (b). This method is attractive because the larger values in the matrix tend to collect

in the upper-left corner of the matrix, and the elements representing larger values tend to be gathered together in that area of the matrix. Thus, we can induce a better rule: Scanning should always start from the upper-left corner element of the matrix. This way, we get much longer runs for each phrase and a much lower number of phrases in the run-length coding.

Once the run-length coding is processed, JPEG uses some type of Huffman coding or arithmetic coding for the nonzero values. Note here that, so far, only a block of $8 \times 8$ pixels has been processed. An image consists of numerous such blocks. Therefore, the speed of processing and transmission is a factor in the quality of image transfer.

## 17.4 Moving Images and MPEG Compression

A *motion image*, or video is a rapid display of still images. Moving from one image to another must be fast enough to fool the human eye. There are different standards on the number of still images comprising a video clip. One common standard produces a motion image by displaying still images at a rate of 30 frames per second. The common standard that defines the video compression is the *Moving Pictures Expert Group* (MPEG), which has several branch standards:

- MPEG-1, primarily for video on CD-ROM
- MPEG-2, for multimedia entertainment and *high-definition television* (HDTV) and the satellite broadcasting industry
- MPEG-4, for object-oriented video compression and videoconferencing over low-bandwidth channels
- MPEG-7, for a broad range of demands requiring large bandwidths providing multimedia tools
- MPEG-21 for interaction among the various MPEG groups

Logically, using JPEG compression for each still picture does not provide sufficient compression for video as it occupies a large bandwidth. MPEG deploys additional compression. Normally, the difference between two consecutive frames is small. With MPEG, a base frame is sent first, and successive frames are encoded by computing the differences. The receiver can reconstruct frames based on the first base frame and the submitted differences. However, frames of a completely new scene in a video may not be compressed this way, as the difference between the two scenes is substantial. Depending on the relative position of a frame in a sequence, it can be compressed through one of the following types of frames:

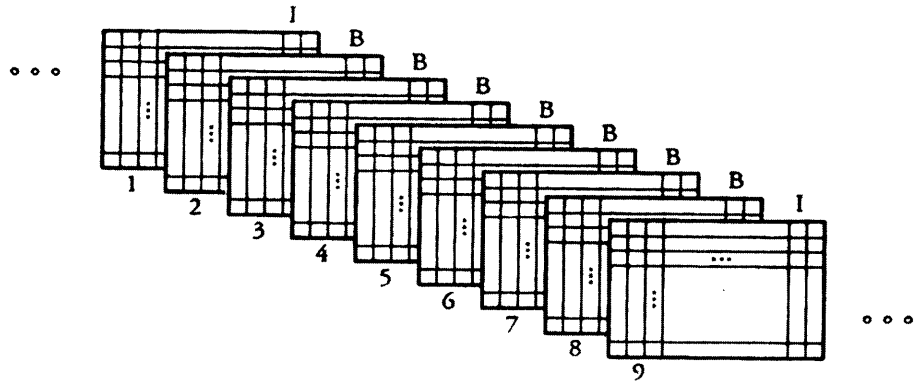


**Figure 17.9** Snapshot of moving frames for MPEG compression

- *Interimage* (I) frames. An 1 frame is treated as a JPEG still image and compressed using DCT.
- *Predictive* (P) frames. These frames are produced by computing differences between a current and a previous I or P frame.
- *Bidirectional* (B) frames. A B frame is similar to a P frame, but the B frame considers differences between a previous, current, and future frames.

Figure 17.9 illustrates a typical grouping of frames. With I,P, and B frames forming a sequence. In any frame sequence, I frames appear periodically as the base of the scene. Normally, there is a P frame between each two groups of B frames.

### 17.4.1 MP3 and Streaming Audio

Section 17.2 explained how an audible sound or a human voice ranging between 20 Hz and 20 KHz can be converted into digital bits and eventually into packets for networking. A signal is sampled, quantized, and encoded, a process called PCM. A variety of methods of compressing such encoded products at the output of the PCM are available. However, Huffman compression of the processed signal may not be sufficient for transmission over IP networks.

The MPEG-1 layer 3 (MP3) technology compresses audio for networking and producing CD-quality sound. The sampling part of PCM is performed at a rate of 44.1 KHz to over the maximum of 20 KHz of audible signals. Using the commonly used 16-bit encoding for each samples the maximum total bits required for audio is $16 \times 44.1 = 700$ kilobits and 1.4 megabits for two channels if the sound is processed

in a stereo fashion. For example a 60-minute CD (3,600 seconds) requires about $1.4 \times 3{,}600 = 5{,}040$ megabits, or 630 megabytes. This amount may be acceptable for recording on a CD but is considered extremely large for networking, and thus a carefully designed compression technique is needed.

MP3 combines the advantages of MPEG with "three" layers of audio compressions. MP3 removes from a piece of sound all portions that an average ear may not be able to hear, such as weak background sounds. On any audio streaming, MP3 specifies what humans are not able to hear, removes those components, and digitizes the remaining. By filtering some part of an audio signal, the quality of compressed MP3 is obviously degraded to lower than the original one. Nonetheless, the compression achievement of this technology is remarkable.

## 17.5 Limits of Compression with Loss

Hartely, Nyquist, and Shannon are the founders of *information theory*, which has resulted in the mathematical modeling of information sources. Consider a communication system in which a source signal is processed to produce sequences of $n$ words, as shown in Figure 17.10. These sequences of digital bits at the output of the information source can be compressed in the *source encoder* unit to save the transmission link bandwidth. An information source can be modeled by a *random process* $\mathbf{X_n} = (X_1, \ldots, X_n)$, where $X_i$ is a random variable taking on values from a set of values as $\{a_1, \cdots, a_N\}$, called *alphabet*. We use this model in our analysis to show the information process in high-speed networks.

### 17.5.1 Basics of Information Theory

The challenge of data compression is to find the output that conveys the most information. Consider a single source with random variable $X$, choosing values in $\{a_1, \cdots a_N\}$.

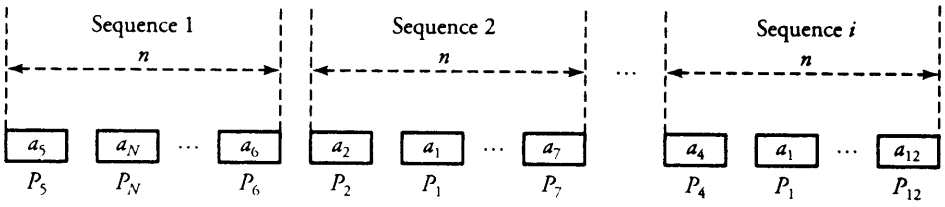


**Figure 17.10** A model of data sequences

If $a_i$ is the most likely output and $a_j$ is the least likely output, clearly, $a_j$ conveys the most information and $a_i$ conveys the least information. This observation can be rephrased as an important conclusion: *The measure of information for an output is a decreasing and continuous function of the probability of source output*. To formulate this statement, let $P_{k_1}$ and $P_{k_2}$ be the probabilities of an information source's outputs $a_{k_1}$ and $a_{k_2}$, respectively. Let $I(P_{k_1})$ and $I(P_{k_2})$ be the information content of $a_{k_1}$ and $a_{k_2}$, respectively. The following four facts apply.

1. As discussed, $I(P_k)$ depends on $P_k$.
2. $I(P_k)$ = a continuous function of $P_k$.
3. $I(P_k)$ = a decreasing function of $P_k$.
4. $P_k = P_{k_1}.P_{k_2}$ (probability of two outputs happen in the same time).
5. $I(P_k) = I(P_{k_1}) + I(P_{k_2})$ (sum of two pieces of information).

These facts lead to an important conclusion that can relate the probability of a certain data to its information content:

$$I(P_k) = -\log_2 P_k = \log_2\left(\frac{1}{P_k}\right). \tag{17.16}$$

The log function has a base 2, an indication of incorporating the binary concept of digital data.

## 17.5.2 Entropy of Information

In general, *entropy* is a measure of uncertainty. Consider an information source producing random numbers, $X$, from a possible collection of $\{a_1, \cdots, a_N\}$ with corresponding probabilities of $\{P_1, \cdots, P_N\}$ and information content of $\{I(P_1), \cdots, I(P_N)\}$, respectively. In particular, the entropy, $H(x)$, is defined as the average information content of a source:

$$\begin{aligned} H_X(x) &= \sum_{k=1}^{N} P_k I(P_k) \\ &= \sum_{k=1}^{N} -P_k \log_2 P_k = \sum_{k=1}^{N} P_k \log_2\left(\frac{1}{P_k}\right). \end{aligned} \tag{17.17}$$

**Example.** A source with bandwidth 8 KHz is sampled at the Nyquist rate. If the result is modeled using any value from $\{-2,-1,0,1,2\}$ and corresponding probabilities $\{0.05, 0.05, 0.08, 0.30, 0.52\}$, find the entropy.

***Solution.***

$$H_X(x) = -\sum_{i=1}^{N} P_i \log_2 P_i = 1.74 \text{ bits/sample.}$$

The information rate in samples/sec = 8,000 × 2 = 16,000, and the rate of information produced by the source = 16,000 × 1.74 = 27,840 bits.

**Joint Entropy**

The Joint entropy of two discrete random variables $X$ and $Y$ is defined by

$$H_{X,Y}(x,y) = -\sum_{x,y} P_{X,Y}(x,y) \log_2 P_{X,Y}(x,y) \tag{17.18}$$

where $P_{X,Y}(x,y)$ = Prob[$X = x$ and the same time $Y = y$] and is called the *joint probability mass function* of two random variables. In general, for a random process $\mathbf{X}_n = (X_1, \ldots, X_n)$ with $n$ random variables:

$$H_{X_n}(X_n) = \sum_{X_1 \ldots X_n} P_{X_1,\ldots,X_n}(x_1,\ldots,x_n) \log_2 P_{X_1,\ldots,X_n}(x_1,\ldots,x_n), \tag{17.19}$$

where $PX_1, \ldots, X_n(x_1, \ldots, x_n)$ is the joint probability mass function (J-PMF) of the random process $X_n$. For more Information on J-PME, see Appendix C.

## 17.5.3 Shannon's Coding Theorem

This theorem limits the rate of data compression. Consider again Figure 17.10, which shows a discrete source modeled by a random process that genearates sequences of length $n$ using values in set $\{a_1, \ldots, a_n\}$ with probabilities $\{P_1, \ldots, P_2\}$ Respectively. If $n$ is larger enough, the number of times a value $a_i$ is repeated in a given sequences = $nP_i$, and the number of values in a typical sequences is therefore $n(P_1 + \ldots + P_N)$.

We defined the *typical sequences* as one in which any value $a_i$ is repeated $nP_i$ times. Accordingly, the probablity that $a_i$ is repeated $np_i$ times is obviously $P_iP_i \ldots P_i = P_i^{npi}$, resulting in amore general statement: The probability of a typical sequence is the probability [($a_1$ is repeated $np_1$)] × the probability [($a_2$ is repeated $np_2$] × .... this can be shown by $P_1^{np1}P_2^{np2} \ldots P_N^{npN}$, or

$$\text{Prob(Typical Sequence)} = \prod_{i=1}^{N} P_i^{np_i}.$$

Knowing $P_i^{nP_i} = 2^{nP_i \log_2 P_i}$, we can obtain the probability of a typical sequence $P_t$ as follows:

$$\begin{aligned} P_t &= \prod_{i=1}^{N} 2^{nP_i \log_2 P_i} \\ &= 2^{(nP_1 \log_2 P_1 + \ldots nP_N \log_2 P_N)} \\ &= 2^{n(P_1 \log_2 P_1 + \ldots P_N \log_2 P_N)} \\ &= 2^{n\{\sum_{i=1}^{N} P_i \log_2 P_i\}}. \end{aligned} \tag{17.21}$$

This last expression results in the well-known Shannon's theorem, which expresses the probability that a typical sequence of length $n$ with entropy $H_X(x)$ is equal to

$$P_t = 2^{-nH_X(x)}. \tag{17.22}$$

**Example.** Assume that a sequence size of 200 of an information source chooses values from the set $\{a_1, \ldots, a_5\}$ with corresponding probabilities {0.05, 0.05, 0.08, 0.30, 0.52}. Find the probability of a typical sequence.

***Solution.*** In the previous example, we calculated the entropy to be $H_X(x) = 1.74$ for the same situation. With $n = 200$, $N = 5$, probability of a typical sequence is the probability of a sequence in which $a_1$, $a_2$, $a_3$, $a_4$, and $a_5$ are repeated, respectively, $200 \times 0.05 = 10$ times, 10 times, 16 times, 60 times, and 104 times. Thus, the probability of a typical sequences is $P_t = 2^{-nH_X(x)} = 2^{-200(1.74)}$.

The fundamental Shannon's theorem leads to an important conclusion. As the probability of a typical sequence is $2^{-nH_X(x)}$ and the sum of probabilities of all typical sequence is 1, the number of typical sequences is obtained by $= \frac{1}{2^{-nH_X(x)}} = 2^{nH(x)}$. Knowing that the total number of all sequences, including typical and nontypical ones, is $N^n$, it is sufficient, in all practical cases when $n$ is large enough, to transmit only the set of typical sequences rather than the set of all sequences. This is the essence of data compression: The total number of bits required to represent $2^{nH_X(x)}$ sequences is $nH_X(x)$ bits, and the average number of bits for each source = $H_X(x)$.

**Example.** Following the previous example, in which the sequence size for an information source is 200, find the ratio of the number of typical sequences to the number of all types of sequences.

***Solution.*** We had $n = 200$ and $N = 5$; thus, the number of typical sequences is $2^{nH_X(x)} = 2^{200 \times 1.74}$, and the total number of all sequences is $2^{348}$. This ratio is almost zero, which may cause a significant loss of data if it is compressed, based on Shannon's

theorem. It is worth mentioning that the number of bits required to represent $2^{nH_X(x)}$sequences is $nH_X(x) = 104$ bits.

### 17.5.4 Compression Ratio and Code Efficiency

Le $\bar{R}$ be the average length of codes, $l_i$ be the length of code word $i$, and $P_i$ be the probability of code word $i$:

$$\bar{R} = \sum_{i=0}^{N} P_i l_i. \tag{17.23}$$

A *compression ratio* is defined as

$$C_r = \frac{R}{\bar{R}_x}, \tag{17.24}$$

where $\bar{R}_x$ is the length of a source output before coding. It can also be shown that

$$H_X(x) \leq R < H_X(x) + 1. \tag{17.25}$$

*Code efficiency* is a measure for understanding how close code lengths are to the corresponding decoded data and is defined by

$$\eta_{code} = \frac{H_X(x)}{\bar{R}}. \tag{17.26}$$

When data is compressed, part of the data may need to be removed in the process.

## 17.6 Compression Methods Without Loss

Some types of data, including text, image, and video, might contain redundant or repeated elements. If so, those elements can be eliminated and some sort of codes substituted for future decoding. In this section, we focus on techniques that do not incur any loss during compression:

- Arithemetic encoding
- Run-length encoding
- Huffman encoding
- Lempel-Ziv encoding

Here, we ignore arithmetic encoding and consider only the last three encoding techniques.

**Table 17.1** Statistics obtained for run-length compression of a 1,000-character piece of text

| Number of Repeated Characters | Average Length of Repeated Characters | Compression Ratio $C_r$ |
|---|---|---|
| 10 | 4 | 0.99 |
| 10 | 10 | 0.93 |
| 20 | 4 | 0.98 |
| 20 | 10 | 0.85 |
| 30 | 4 | 0.97 |
| 30 | 10 | 0.79 |

## 17.6.1 Run-Length Encoding

One of the simplest data-compression techniques is *run-length encoding*. This technique is fairly effective for compression of plaintext and numbers, especially for facsimile systems. With run-length code, repeated letters can be replaced by a run length, beginning with $C_c$ to express the compression letter count.

**Example.** Assume a system that represents b as a blank. Find the compressed version of the following sentence:

THISSSSSSbISbbbbANbEXAMPLEbOFbRUN——LENGTHbCODE

***Solution.*** According to the conventions stated, the compressed version of that sentence turns into:

THIS$C_c$6SbIS$C_c$b4bANbEXAMPLEbOFbRUN$C_c$-5LENGTHbCODE

It is obvious that the longer the text, the smaller the compression ratio becomes, as shown in Table 17.1. The statistics obtained in this table are based on a 1,000-character piece of text.

## 17.6.2 Huffman Encoding

*Huffman encoding* is an efficient frequency-dependent coding technique. With this algorithm, source values with smaller probabilities appear to be encoded by a longer word.

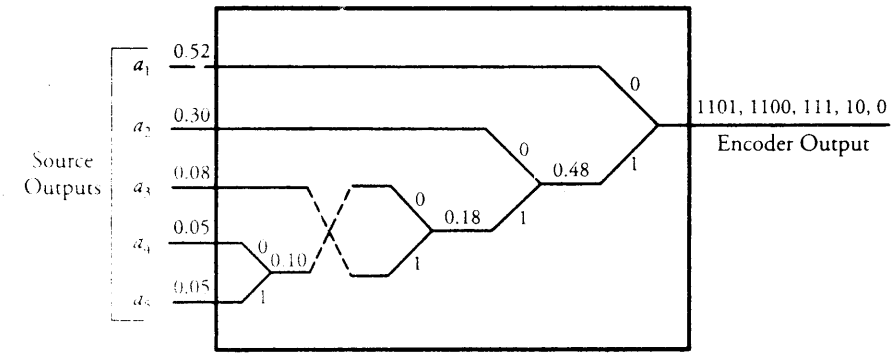


Figure 17.11 Huffman encoding

This technique reduces the total number of bits, leading to an efficient compression of data. The algorithm that implements such a technique is as follows.

**Begin Huffman Encoding Algorithm**

1. Sort outputs of the source in decreasing order of their probabilities. For example, 0.7, 0.6, 0.6, 0.59, · · ·, 0.02, 0.01.
2. Merge the two least probabilistic outputs into a single output whose probability is the sum of corresponding probability, such as 0.02 + 0.01 = 0.03.
3. If the number of remaining outputs is 2, go to the next step; otherwise, go to step 1.
4. Assign 0 and 1 as codes on the diagram.
5. If a new output is the result of merging two outputs, append the code word with 0 and 1; otherwise, stop. ■

**Example.** Design a Huffman encoder for a source generating $\{a_1, a_2, a_3, a_4, a_5\}$ and with probabilities {0.05, 0.05, 0.08, 0.30, 0.52}.

***Solution.*** Following the algorithm, the output of the information source shown in Figure 17.11, the information related to $\{a_1, a_2, a_3, a_4, a_5\}$ is compressed to 1100, 1101, 111, 10, 0, respectively.

## 17.6.3 Lempel-Ziv Encoding

*Lempel-Ziv* codes are independent of the source statistics. This coding technique is normally used for UNIX compressed files. The algorithm that converts a string of logical bits into a Lempel-Ziv code is summarized as follows.

**Begin Lempel-Ziv Encoding Algorithm**

1. Any sequence of source output is passed in a phrase of varying length. At the first step, identify phrases of the smallest length that have not appeared so far. Note that all phrases are different, and lengths of words grow as the encoding process proceeds.
2. Phrases are encoded using code words of equal length. If $k_1$ = number of bits are needed to describe the code word and $k_2$ = the number of phrases, we must have

$$k_1 = \log_2 \lceil k_2 \rceil_2.$$

3. A code is the location of the prefix to the phrases.
4. A code is followed by the last bit of parser output to double-check the last bit. ■

**Example.** For the following string of bits, find the encoded Lempel-Ziv words:

$$11110111011000001010010001111010101100$$

***Solution.*** Implementing step 1 on the string, there are 14 phrases, as follows:

$$1-11-10-111-0-110-00-001-01-0010-0011-1101-010-1100$$

Thus, $k_2 = 14$ and $k_1 = \log_2 \lceil 14 \rceil_2 = 4$. Table 17.2 shows the encoded words as steps 3 and 4 are applied on the parser output.

## 17.7 Case Study: FAX Compression for Transmission

A *FAX* picture is scanned and compressed in two steps: run-length encoding and then Huffman encoding. First, the transmission of the digital line scan is replaced by the transmission of a quantity count of each of the successive runs of black or white elements.

Consider a document of standard size 8.5 inches by 11 inches. The picture is first partitioned into *pixels*. If the desired resolution is 200 × 200 pixels per square inch, the total number of pixels per picture is exactly $200^2 \times (8.5 \times 11)$ = 37,400,000 pixels.

### Fax Process Algorithm

As mentioned earlier, processing a FAX picture requires both run-length coding and Huffman coding. Since black and white always alternate, no special characters are

**Table 17.2** An example of Lempel-Ziv coding process

| Parser Output | Location | Encoded output |
|---|---|---|
| 1 | 0001 | 00001 |
| 11 | 0010 | 00011 |
| 10 | 0011 | 00010 |
| 111 | 0100 | 00101 |
| 110 | 0110 | 00100 |
| 0 | 0101 | 01010 |
| 00 | 0111 | 01010 |
| 001 | 1000 | 01111 |
| 0011 | 1001 | 10000 |
| 00110 | 1001 | 10000 |
| 0011 | 1011 | 10001 |
| 010 | 1101 | 10010 |
| 0107 | 1101 | 10010 |
| 1100 | 1110 | 01100 |

needed to specify that a run is black or white. Thus, the encoded data stream is a string of numbers indicating the lengths of the alternating black and white runs. The algorithm for the first phase is as follows.

1. Identify the first row out of the $n$-row document.
2. At any row $i$, start at the first pixel of the row. If the pixel is black, assign code 1; if the pixel is white, assign code 0.
3. At any step of counting $j$, let $X_j$ be the number of consecutive 0s before a 1 appears. Then assign code $C_c X_j 0$ to this string of 0s. Do the same thing for 1s, and code it with $C_c X_j 1$.

At this point, the document is converted into a number of $C_c X_j 0$s and $C_c X_j 1$. In phase 2 coding, we need the statistics on the frequencies of a certain run-length code in order to compress it further, using the Huffman algorithm. Table 17.3 shows practical statistics for a black-and-white FAX document.

**Table 17.3** Statistics on frequency of occurrences for strings obtained after run-length coding for black-and-white FAX document

| Number of Repeated Pixels ($C_cX$) | Huffman Code for White Pixels | Huffman Code for Black Pixels |
|---|---|---|
| $C_c1$ | 000111 | 010 |
| $C_c2$ | 0011 | 01 |
| $C_c10$ | 00011 | 1000011 |
| $C_c50$ | 00111001 | 000001101111 |

## 17.8 Summary

A number of algorithms effectively compress voice, still images, and moving images. A raw voice signal is converted to a binary-encoded form known as *pulse code modulation* (PCM). In this process, sampling and quantization both present some sort of loss. Compression schemes used for the preparation of an image include the *Joint Photographic Experts Group* (JPEG), a compression standard of still images. JPEG consists of three processes: *discrete cosine transform* (DCT), *quantization*, and *compression*. In particular, DCT converts a snapshot of a real image into a matrix of corresponding values, and the quantization phase converts the values generated by DCT to simple numbers.

A *motion image*, or video, is the rapid display of still images to fool the human eye. Standards differ on the number of still images that make a video clip. The common standard that defines video compression is the *Moving Pictures Expert Group* (MPEG). MPEG-1 layer 3 (MP3) is a technology for compressing audio for networking and producing CD-quality sound.

Compression has limits, as presented by Shannon. Shannon's theorem expresses the probability of a typical sequence of length $n$ with entropy $H_X(x)$ to be equal to $2^{-nH_X(x)}$. Although some compression processes entail loss of data, others do not, such as Huffman or run-length encoding techniques.

The next chapter looks at multimedia networking, voice over IP (VIP), and streaming video.

## 17.9 Exercises

1. A sinusoidal signal $g(t)$ with period 10 ms is to be sampled by a sampler $s(t)$ with period $T_s = 1$ ms and pulse width $\tau = 0.5$ ms. The maximum voltages for both

signals are 1 volt., Let the sampled signal be $g_s(t)$; compute and sketch $g(t)$, $s(t)$, $g_s(t)$, $G(f)$, $S(f)$, and $G_s(f)$, (the range of $nT_s$ in $s(t)$, is $[-2T_s, +2T_s]$).

2. Consider a pulse signal $g(t)$ being 1 volt in intervals: ... [−4 and −2], [−1 and +1], [+2 and +4] ... ms. This signal is to be sampled by an impulse sampler $s(t)$ being generated at ... ,−3, 0, +3, ... ms, with pluse width $\tau$ = 0.5ms. Compute and sketch all the processes from analog signal $g(t)$ to sampled version $g_s(t)$ in both time and frequency domains.

3. Assume that a normal-distributed source with zero mean and variance of 2 is to be transmitted via a channel that can provide a transmission capacity of 4 bits/each source output.
   (a) What is the minimum mean-squared error achievable?
   (b) What is the required transmission capacity per source output if the maximum tolerable distortion is 0.05?

4. Let $X(t)$ denote a normal (Gaussian) source with $\sigma^2 = 10$, for which a 12-level optimal uniform quantizer is to be designed.
   (a) Find optimal quantization intervals ($\Delta$).
   (b) Find optimal quantization boundaries ($a_i$).
   (c) Find optimal quantization levels ($\hat{x}_i$).
   (d) Find the optimal total resulting distortion.
   (e) Compare the optimal total resulting distortion with the result obtained from the rate-distortion bound that achieves the same amount of distortion.

5. Consider the same information sourced discusse in exercise 4. This time, apply a 16-level optimal uniform quantizer.
   (a) Find optimal quantization boundaries ($a_i$).
   (b) Find optimal quantization intervals ($\Delta$).
   (c) Find optimal quantization levels ( $\hat{x}_i$ ).
   (d) Find the optimal total resulting distortion.

6. To encode two random signals $X$ and $Y$ that are uniformly distributed on the region between two squares, let the marginal PDF of random variables be

$$f_X(x) = \begin{cases} 0.25 & -2 \le X < -1 \\ 0.25 & -1 \le X < 0 \\ 0.25 & 0 \le X < +1 \\ 0.25 & +1 \le X < +2 \end{cases}$$

and

$$f_Y(y) = \begin{cases} 0.3 & -2 \le Y < -1 \\ 0.1 & -1 \le Y < 0 \\ 0.4 & 0 \le Y < +1 \\ 0.2 & +1 \le Y < +2 \end{cases}$$

Assume that each of the random variables $X$ and $Y$ is quantized using four-level uniform quantizers.

(a) Calculate the joint probability $P_{xy}(x,y)$.

(b) Find quantization levels $x_1$ through $x_4$ if $\Delta = 1$

(c) Without using the optimal quantization table, find the resulting total distortion.

(d) Find the resulting number of bits per $(X,Y)$ pair.

**7.** The sampling rate of a certain CD player is 80,000, and samples are quantized using a 16bits/sample quantizer. Determine the resulting number of bits for a piece of music with a duration of 60 minutes.

**8.** The PDF of source is defined by $fx(x) = 2\text{^}(x)$. This source is quantized using an four-level uniform quantizer described as follows:

$$Q(x) = \begin{cases} +1.5 & 1 < x \le 2 \\ +0.5 & 0 < x \le 1 \\ -0.5 & -1 < x \le 0 \\ -1.5 & -2 < x \le -1 \end{cases}$$

Find the PDF of random variable representing the quantization error $X-Q(X)$.

**9.** Using logic gates, design a PCM encoder using 3-bit gray codes.

**10.** To preserve as much information as possible, the JPEG elements of $T[i][j]$ are divided by the elements of an $N \times N$ matrix denoted by $\mathbf{D}[i][j]$, in which the values of elements increase from the upper left portion to the lower-right portion. Consider the matrix $\boldsymbol{T}[i][j]$ and $\boldsymbol{D}[i][j]$, in Figure 17.12.

(a) Find the quantized matrix $Q[i]$ $[j]$.

(b) Obtain a run-length compression on $Q[i]$ $[j]$.

$$\begin{bmatrix} 1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 \\ 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \\ 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 \\ 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 \\ 9 & 11 & 13 & 15 & 17 & 19 & 21 & 23 \\ 11 & 13 & 15 & 17 & 19 & 21 & 23 & 25 \\ 13 & 15 & 17 & 19 & 21 & 23 & 25 & 27 \\ 15 & 17 & 19 & 21 & 23 & 25 & 27 & 29 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 513 & -138 & 0 & -17 & 0 & -6 & 0 & -1 \\ -138 & 1 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -17 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b)

**Figure 17.12** Exercise 10 matrices for applying (a) divisor matrix **D**[*i*][*j*] on (b) matrix **T**[*i*][*j*] to produce an efficient quantization of a JPEG image to produce matrix **Q**[*i*][*j*]

11. Find the differential entropy of the continuous random variable $X$ with a PDF defined by

$$f_X(x) = \begin{cases} x+1 & -1 \le x \le 0 \\ -x+1 & 0 < x \le 1 \\ 0 & \text{else} \end{cases}$$

12. A source has an alphabet $\{a_1, a_2, a_3, a_4, a_5\}$ with corresponding probabilities $\{0.23, 0.30, 0.07, 0.28, 0.12\}$.
    (a) Find the entropy of this source.
    (b) Compare this entropy with that of a uniformly distributed source with the same alphabet.

13. We define two random variables $X$ and $Y$ for two random voice signals in a multimedia network, both taking on values in alphabet $\{1, 2, 3\}$. The joint probability mass function (JPMF), $P_{X,Y}(x, y)$, is given as follows:

$$\begin{cases} P_{X,Y}(1, 1) = P(X = 1, Y = 1) = 0.1 \\ P_{X,Y}(1, 2) = P(X = 1, Y = 2) = 0.2 \\ P_{X,Y}(2, 1) = P(X = 2, Y = 1) = 0.1 \\ P_{X,Y}(1, 3) = P(X = 1, Y = 3) = 0.4 \\ P_{X,Y}(2, 3) = P(X = 2, Y = 3) = 0.2 \end{cases}$$

    (a) Find the two marginal entropies, $H(X)$ and $H(Y)$.

(b) Conceptually, what is the meaning of the marginal entropy?
(c) Find the joint entropy of the two signals, $H(X, Y)$.
(d) Conceptually, what is the meaning of the joint entropy?

14. We define two random variables $X$ and $Y$ for two random voice signals in a multimedia network.
(a) Find the conditional entropy, $H(X|Y)$, in terms of joint and marginal entropies.
(b) Conceptually, what is the meaning of the joint entropy?

15. Consider the process of a source with a bandwidth $W = 50$ Hz sampled at the Nyquist rate. The resulting sample outputs take values in the set of alphabet $\{a_0, a_1, a_2, a_3, a_4, a_5, a_6\}$ with corresponding probabilities {0.06, 0.09, 0.10, 0.15, 0.05, 0.20, 0.35} and are transmitted in sequences of length 10.
(a) Which output conveys the most information? item What is the information content of outputs $a_1$ and $a_5$ together?
(b) Find the least-probable sequence and its probability, and comment on whether it is a typical sequence.
(c) Find the entropy of the source in bits/sample and bits/second.
(d) Calculate the number of nontypical sequences.

16. A source with the output alphabet $\{a_1, a_2, a_3, a_4\}$ and corresponding probabilities {0.15, 0.20, 0.30, 0.35} produces sequences of length 100.
(a) What is the approximate number of typical sequences in the source output?
(b) What is the ratio of typical sequences to nontypical sequences?
(c) What is the probability of a typical sequence?
(d) What is the number of bits required to represent only typical sequences?
(e) What is the most probable sequence, and what is its probability?

17. For a source with an alphabet $\{a_0, a_1, a_2, a_3, a_4, a_5, a_6\}$ and with corresponding probabilities {0.55, 0.10, 0.05, 0.14, 0.06, 0.08, 0.02}:
(a) Design a Huffman encoder.
(b) Find the code efficiency.

18. A voice information source can be modeled as a band-limited process with a bandwidth of 4,000 Hz. This process is sampled at the Nyquist rate. In order to provide a guard band to this source, 200 Hz is added to the bandwidth for which a Nyquist rate is not needed. It is observed that the result-

ing samples take values in the set $\{-3, -2, -1, 0, 2, 3, 5\}$, with probabilities $\{0.05, 0.1, 0.1, 0.15, 0.05, 0.25, 0.3\}$.

(a) What is the entropy of the discrete time source in bits/output?
(b) What is the entropy in b/s?
(c) Design a Huffman encoder.
(d) Find the compression ratio and code efficiency for Part (c).

**19.** Design a Lempel-Ziv encoder for the following source sequence:

0101000010001111100101010111110100101010101

**20.** Design a Lempel-Ziv encoder for the following source sequence:

1111100010101010111011111000101010100011111010100001

**21.** *Computer simulation project*. Using a computer program, implement Equation (17.15) to obtain $\mathbf{T}[i][j]$ matrix for a JPEG compression process.

CHAPTER 18

# VoIP and Multimedia Networking

The discussion in the previous chapter on compressed voice and video sets the stage for the discussion in this chapter on multimedia networking. The communication industry has spent considerable effort in designing an IP-based media transport mechanism: *voice over* IP (VoIP), which can deliver voice-band telephony with the quality of telephone networks. Internet phone services are less expensive and have more features, such as video conferencing, online directory services, and Web incorporation. *Multimedia networking* is one of the most effective Internet developments. In addition to data, the Internet is used to transport phone calls, audio, and video. This chapter looks at the transportation of real-time signals along with the signaling protocols used in voice telephony, video streaming, and multimedia networking, covering the following major topics:

- *Overview of IP telephony*
- *VoIP signaling protocols*
- *Real-time media transport protocols*
- *Distributed multimedia networking*
- *Stream Control Transmission Protocol (SCTP)*
- *Self-similarity and non-Markovian streaming analysis*

This chapter focuses on transport mechanisms for the delivery of media streams with the highest possible quality. After reviewing how sampled and digitized streams

of voice are treated in networks, we look at two VoIP protocols—*Session Initiation Protocol* (SIP) and the *H.323 series of protocols*—and explain their session signaling and numbering. We then present *real-time media transport protocols*, by which a sender sends a stream of data at a constant rate. The most widely applied protocol for real-time transmission is the *Real-Time Transport Protocol* (RTP), which is available with its companion version, *Real-Time Control Protocol* (RTCP).

As detailed in our discussion on video streaming, a video in a single server can be streamed from a video server to a client for each client request. However, when a high-bit-rate video stream must pass through many Internet domains, a significant delay may result. We then present a solution by using *content distribution networks* (CDNs), which can bring multimedia content to users. The *Stream Control Transmission Protocol* (SCTP) provides a general-purpose protocol for transporting stream traffic. The chapter ends with a detailed streaming source modeling and analysis and a detailed traffic modeling of streaming sources, using *self-similarity* patterns.

## 18.1 Overview of IP Telephony

An IP *telephone* can be used to make telephone calls over IP networks. *Voice over* IP (VoIP), or IP telephony, uses packet-switched networks to carry voice traffic in addition to data traffic. The basic scheme of IP telephony starts with *pulse code modulation*, discussed in Chapter 17. The encoded data is transmitted as packets over packet-switched networks. At a receiver, the data is decoded and converted back to analog form. The packet size must be properly chosen to prevent large delays. The IP telephone system must also be able to handle the signaling function of the call setup, mapping of phone number to IP address, and proper call termination.

Basic components of an IP telephone system include IP *telephones*, the *Internet backbone*, and *signaling servers*, as shown in Figure 18.1. The IP telephone can also be a laptop or a computer with the appropriate software. An IP telephone connects to the Internet through a wired or a wireless medium. The signaling servers in each domain are analogous to the central processing unit in a computer and are responsible for the coordination between IP phones. The hardware devices required to deploy packet-switched networks are less expensive than those required for the connection-oriented public-switched telephone networks. On a VoIP network, network resources are shared between voice and data traffic, resulting in some savings and efficient use of the available network resources.

A VoIP network is operated through two sets of protocols: *signaling protocols* and *real-time packet-transport protocols*. Signaling protocols handle call setups and are

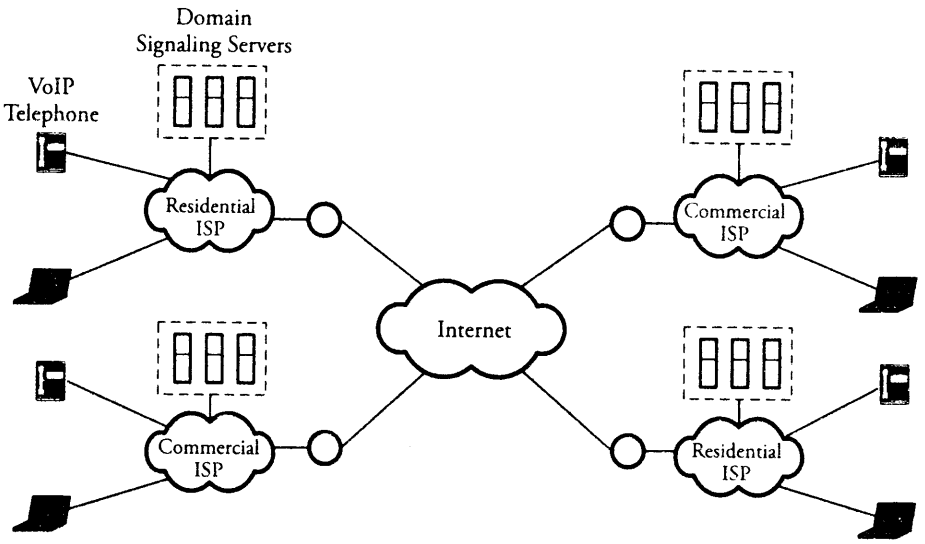


**Figure 18.1** Voice over IP system

controlled by the signaling servers. Once a connection is set, RTP transfers voice data in real-time fashion to destinations. RTP runs over UDP because TCP has a very high overhead. RTP builds some reliability into the UDP scheme and contains a sequence number and a real-time clock value. The sequence number helps RTP recover packets from out-of-order delivery. Two RTP sessions are associated with each phone conversation. Thus, the IP telephone plays a dual role: an RTP sender for outgoing data and an RTP receiver for incoming data.

### 18.1.1 VoIP Quality-of-Service

A common issue that affects the QoS of packetized audio is *jitter*. Voice data requires a constant packet interarrival rate at receivers to convert data into a proper analog signal for playback. The variations in the packet interarrival rate lead to jitter, which results in improper signal reconstruction at the receiver. Typically, an unstable sine wave reproduced at the receiver results from the jitter in the signal. Buffering packets can help control the interarrival rate. The buffering scheme can be used to output the data packets at a fixed rate. The buffering scheme works well when the arrival time of the next packet is not very long. Buffering can also introduce a certain amount of delay.

Another issue having a great impact on real-time transmission quality is *network latency*, or delay, which is a measure of the time required for a data packet to travel

from a sender to a receiver. For telephone networks, a round-trip delay that is too large can result in an *echo* in the earpiece. Delay can be controlled in networks by assigning a higher priority for voice packets. In such cases, routers and intermediate switches in the network transport these high-priority packets before processing lower-priority data packets.

Congestion in networks can be a major disruption for IP telephony. Congestion can be controlled to a certain extent by implementing weighted random early discard, whereby routers begin to intelligently discard lower-priority packets before congestion occurs. The drop in packets results in a subsequent decrease in the window size in TCP, which relieves congestion to a certain extent.

A VoIP connection has several QoS factors:

- *Packet loss* is accepted to a certain extent.
- *Packet delay* is normally unacceptable.
- *Jitter*, as the variation in packet arrival time, is not acceptable after a certain limit.

*Packet loss* is a direct result of the queueing scheme used in routers. VoIP can use *priority queueing*, *weighted fair queuing*, or *class-based weighted fair queuing*, whereby traffic amounts are also assigned to classes of data traffic. Besides these well-known queueing schemes, voice traffic can be handled by a *custom queuing*, in which a certain amount of channel bandwidth for voice traffic is reserved.

Although the packet loss is tolerable to a certain extent, packet delay may not be tolerable in most cases. The variability in the time of arrival of packets in packet-switched networks gives rise to *jitter* variations. This and other QoS issues have to be handled differently than in conventional packet-switched networks. QoS must also consider connectivity of packet-voice environment when it is combined with traditional telephone networks.

## 18.2 VoIP Signaling Protocols

The IP telephone system must be able to handle signalings for call setup, conversion of phone number to IP address mapping, and proper call termination. Signaling is required for call setup, call management, and call termination. In the standard telephone network, signaling involves identifying the user's location given a phone number, finding a route between a calling and a called party, and handling the issue of call forwarding and other call features.

<table>
<tr><td rowspan="2">Layer</td><td colspan="8">Protocol</td></tr>
<tr><td colspan="2">SIP</td><td colspan="6">H.323</td></tr>
<tr><td rowspan="3">5</td><td rowspan="2">Other Signals</td><td rowspan="2">Media Transport</td><td rowspan="2">Registration</td><td colspan="2">Media Transport</td><td rowspan="2">Security</td><td rowspan="2">Signaling</td><td rowspan="2">Data</td></tr>
<tr><td>Voice Codec</td><td>Video Codec</td></tr>
<tr><td></td><td></td><td>H.225.0 RAS</td><td>G.711<br>H.263<br>G.722<br>G.723<br>G.728</td><td>H.261<br>H.323</td><td>H.235</td><td>H.225.9-Q.931<br>H.250<br>H.245<br>H.250</td><td>T.120</td></tr>
<tr><td></td><td>RTP, RTCP</td><td></td><td colspan="2">RTP, RTCP</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td colspan="5">UDP</td><td colspan="3">TCP</td></tr>
<tr><td>3</td><td colspan="8">IP, RSVP, and IGMP</td></tr>
</table>

**Figure 18.2** Main protocols for VoIP and corresponding layers of operation

IP telephone systems can use either a distributed or a centralized signaling scheme. The distributed approach enables two IP telephones to communicate using a client/server model, as most Internet applications do. The distributed approach works well with VoIP networks within a single company. The centralized approach uses the conventional model and can provide some level of guarantee. Three well-known signaling protocols are

1. *Session Initiation Protocol* (SIP)
2. H.323 *protocols*
3. *Media Gateway Control Protocol* (MGCP)

Figure 18.2 shows the placement of VoIP in the five-layer TCP/IP model. SIP, H.323, and MGCP run over TCP or UDP; real-time data transmission protocols, such as RTP, typically run over UDP. Real-time transmissions of audio and video traffic are implemented over UDP, since the real-time data requires lower delay and less overhead. Our focus in this chapter is on the two signaling protocols SIP and H.323 and on RTP and RTCP.

## 18.2.1 Session Initiation Protocol (SIP)

The *Session Initiation Protocol* (SIP) is one of the most important VoIP signaling protocols operating in the application layer in the five-layer TCP/IP model. SIP can

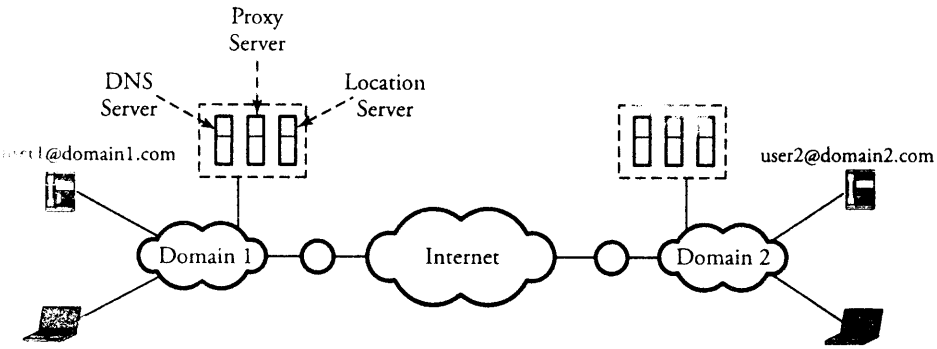


**Figure 18.3** Overview of SIP

perform both unicast and multicast sessions and supports user mobility. SIP handles signals and identifies user location, call setup, call termination, and busy signals. SIP can use multicast to support conference calls and uses the *Session Description Protocol* (SDP) to negotiate parameters.

### SIP Components

Figure 18.3 shows an overview of SIP. A call is initiated from a *user agent*: the user's IP telephone system, which is similar to a conventional phone. A user agent assists in initiating or terminating a phone call in VoIP networks. A user agent can be implemented in a standard telephone or in a laptop with a microphone that runs some software. A user agent is identified using its associated domain. For example, user1@domain1.com refers to user 1, who is associated with the domain1.com network.

SIP consists of the following five servers:

1. *DNS server.* The *Domain Name System* (DNS) server maps the domain name to an IP address in the *user information database* (UID). The UID database contains such user information as preferences and the services to which it has subscribed. The UID also stores information on the related IP addresses. Each user is normally configured with more than one DNS server.
2. *Proxy server.* The proxy server forwards requests from a user agent to a different location and handles authorizations by checking whether the caller is authorized to make a particular call.
3. *Location server.* This server is responsible for UID management. The location server interacts with the database during call setup. Each proxy server is normally configured with more than one location server.

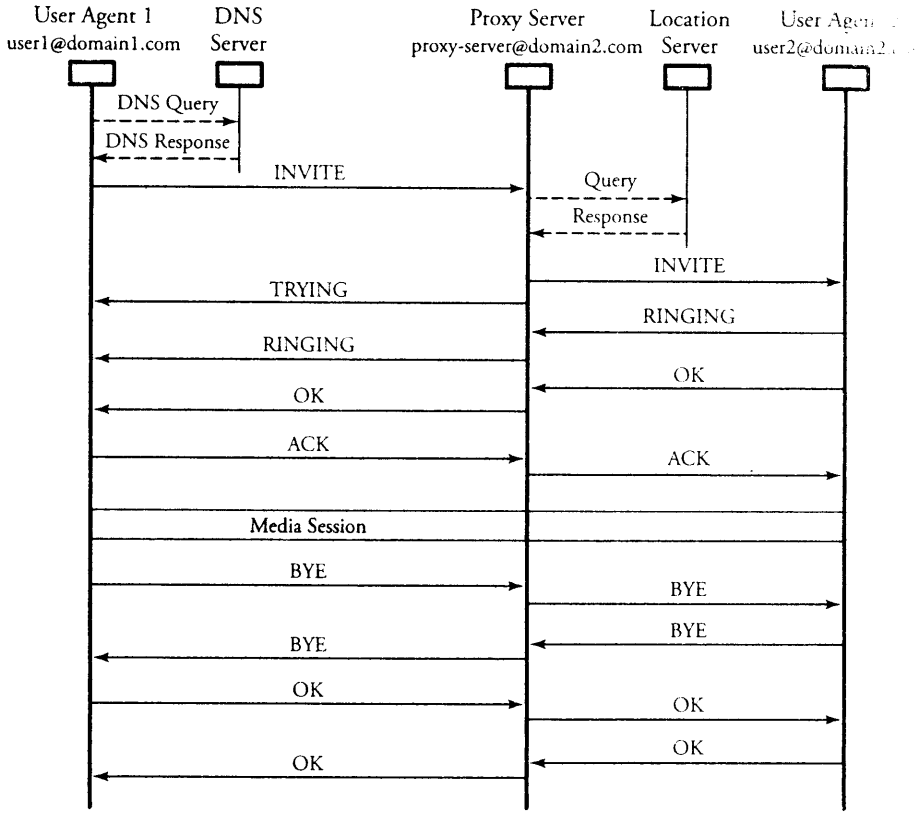


**Figure 18.4** Overview of SIP signaling

4. *Redirect server.* This server performs call forwarding and provides alternative paths for the user agent.
5. *Registrar server.* This server is responsible for registering users in the system and updating the UID that the location server consults. Requests for registration must be authenticated before the registration is granted.

### Session Signaling and Numbering

Figure 18.4 shows a SIP session between user agents 1 and 2: user1@domain1.com and user2@domain2.com, respectively. In this example, user 1 places a call to contact user 2. User 1 first communicates with its DNS server to map the domain name to an IP address in the SIP UID. The DNS server then communicates with the proxy server of the called party. At this point, user 1 has resolved the name of user2@domain2.com into an IP address through DNS query and response.

SIP uses IP addresses for numbering. Locating users in an integrated network consisting of different networks is a complex task. In addition, an optimal route to the user after the user is located must be found. The location servers use a the *Telephone Routing Over IP* (TRIP) protocol to locate a user in an integrated network. TRIP advertises routes, exchanges routing information, and divides the globe into *IP telephone administrative domains* (ITAD). Location servers exchange information about routes with signaling gateways connecting to various ITADs.

The first signal to establish this call is the INVITE message, which is used for session creation. The INVITE message contains such information fields as *from*, *to*, *via*, and *call-id*, in addition to routing information. The proxy server in turn communicates with a location server of the called party. The end point of the called party, user2@domain2.com, is sent an INVITE message to initiate a session. Once user 2 receives a connection query, the TRYING signal is propagated to user 1 from the proxy server, indicating that the call is being routed. This signal is also used to keep track of the call process. A RINGING signal is transmitted from user 2 back to user 1.

When user 2 accepts the call, an OK signal is issued back to user 1 to indicate that the called party has accepted the call. This last signal is acknowledged by an ACK message without any response. The called party picks up the phone, and the two IP phones communicate directly through a *media session*, using the real-time protocol, as shown in the figure. At the end of the conversation, the BYE message is used for a session termination, ending the call.

A few other message signals can be used for other services. The CANCEL message is used to cancel a pending request. The REGISTER message is used to register the user's location. After registration, the user can be reached at a specific URL. Finally the OPTIONS message is used to learn about the parameters used by the called party.

## 18.2.2 H.323 Protocols

The H.323-series protocols are implemented in layer 5 of the TCP/IP model and run over either TCP or UDP. The H.323 protocols interact to provide ideal telephone communication, providing phone numbers to IP address mapping, handling digitized audio streaming in IP telephony, and providing signaling functions for call setup and call management. The H.323 series supports simultaneous voice and data transmission and can transmit binary messages that are encoded using *basic encoding rules*. From the security standpoint, the H.323 scheme provides a unique framework for security, user authentication, and authorization and supports conference calls and multipoint connections, as well as accounting and call-forwarding services.

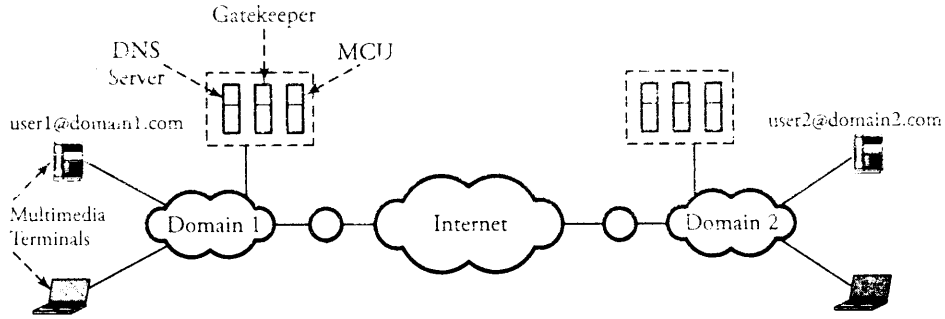


**Figure 18.5** Overview of H.323 protocol connection

### H.323 Components

Figure 18.5 shows an overview of the H.323 protocol connections. The H.323 scheme defines the following five components:

1. *Multimedia terminal*. A multimedia terminal is designed to support video and data traffic and to provide support for IP telephony.
2. *DNS server*. As in SIP, a DNS server maps a domain name to an IP address.
3. *Gateway*. The gateway is a router that serves as an interface between the IP telephone system and the traditional telephone network.
4. *Gatekeeper*: The gatekeeper is the control center that performs all the location and signaling functions. The gatekeeper monitors and coordinates the activities of the gateway. The gateway also performs some signaling functions.
5. *Multicast or multipoint control unit* (MCU). This unit provides some multipoint services, such as conference calls.

A gatekeeper can send signals to other gatekeepers of different zones to access users of those domains. This feature supports distributed locations of multinational systems.

### Session Signaling and Numbering

Figures 18.6 and 18.7 show the details of two IP telephone communications using H.323-series protocols. Assume that two user agents 1 and 2: user1@domain1.com and user2@domain2.com, respectively. In this example, user 1 places a call to contact user 2.

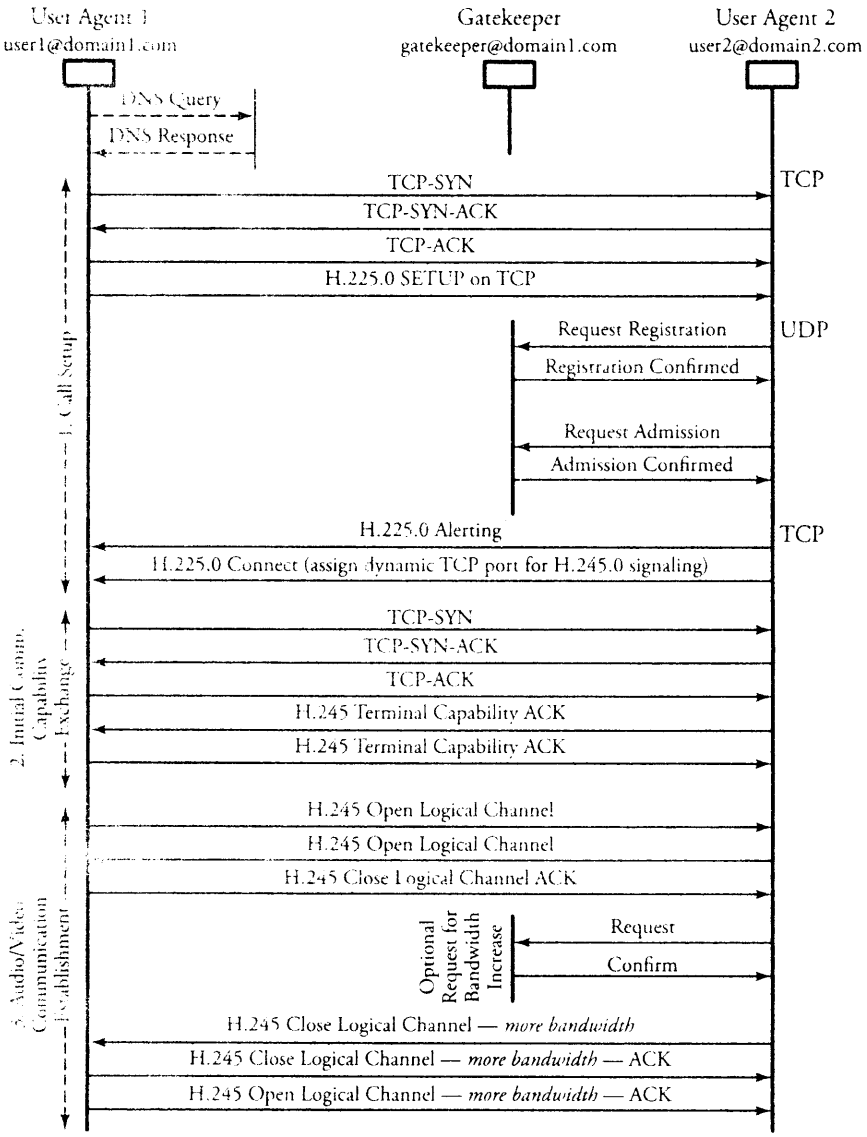


**Figure 18.6** H.323 protocol signaling: steps 1, 2, and 3

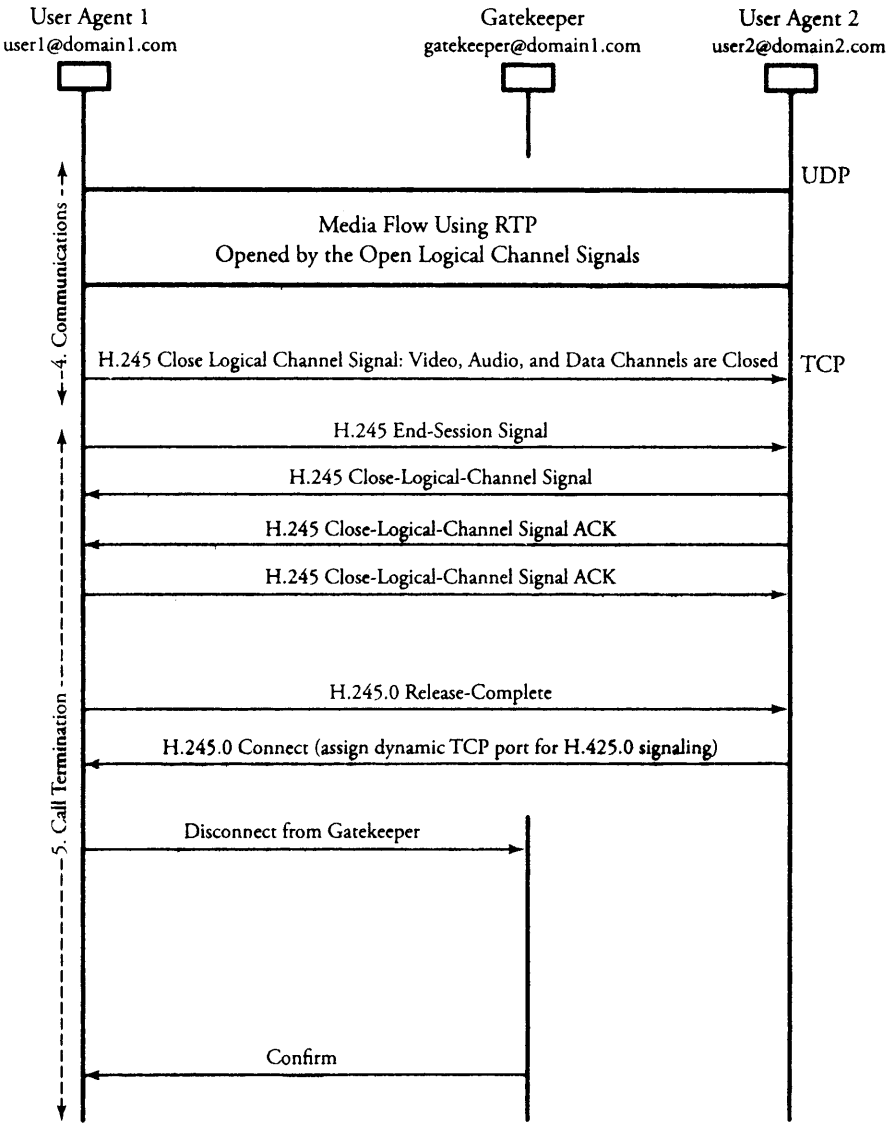


**Figure 18.7** H.323 protocol signaling: steps 4 and 5

First, user 1 communicates with its DNS server, as explained earlier. The signaling uses both UDP and TCP, and partitioned into the following five steps:

1. Call setup
2. Initial communication capability
3. Audio/video communication establishment
4. Communications
5. Call termination

At step 1, when user1 dials user 2's telephone number, the first set of signals are exchanged between these two users in conjunction with opening a TCP connection. TCP-SYN, TCP-SYN-ACK, and then TCP-ACK signals are generated between the two users. At this point, the H225.0 SETUP ON TCP signal informs the called party that the connection can be set up on TCP. The users can now request a certain bandwidth from the associated gatekeeper server of the called party. The requested bandwidth is granted if sufficient bandwidth is left over on the connection; otherwise, the call has to find another gatekeeper to register with. This phase is handled by H.225 and H.245 protocols.

At step 2, all the end points' communication capabilities available over TCP are exchanged. This phase is necessary because the type of communication service requested depends on the communication capabilities of both end points. Step 3 implements the establishment of a logical channel, which in H.323 is unidirectional; therefore, a logical channel must be established in either direction in order to have two-way communications. At the end of this phase, two end points are set for communications. Meanwhile, more bandwidth can also be requested, as shown in the figure, before communications start.

Step 4 comprises the communications between the two users. This phase is handled using RTP over UDP. At this step, any kind of media flow can be considered, depending on the size and type of the channel established in step 4. Finally at step 5, the call is terminated by either user. In Figure 18.7, user 2 initiates the termination of the call by closing the logical channel in H.245 and disconnecting the call from the gatekeeper.

## 18.3 Real-Time Media Transport Protocols

In real-time applications, a stream of data is sent at a constant rate. This data must be delivered to the appropriate application on the destination system, using real-time protocols. The most widely applied protocol for real-time transmission is the *Real-Time*

*Transport Protocol* (RTP), including its companion version: *Real-Time Control Protocol* (RTCP).

UDP cannot provide any timing information. RTP is built on top of the existing UDP stack. Problems with using TCP for real-time applications can be identified easily. Real-time applications may use multicasting for data delivery. As an end-to-end protocol, TCP is not suited for multicast distribution. TCP uses a retransmission strategy for lost packets, which then arrive out of order. Real-time applications cannot afford these delays. TCP does not maintain timing information for packets. In real-time applications, this would be a requirement.

## 18.3.1 Real-Time Transport Protocol (RTP)

The *real-time transport protocol* (RTP) provides some basic functionalities to real-time applications and includes some specific functions to each application. RTP runs on top of the transport protocol as UDP. As noted in Chapter 8, UDP is used for port addressing in the transport layer and for providing such transport-layer functionalities as reordering. RTP provides application-level framing by adding application-layer headers to datagrams. The application breaks the data into smaller units, called *application data units* (ADUs). Lower layers in the protocol stack, such as the transport layer, preserve the structure of the ADU.

Real-time applications, such as voice and video, can tolerate a certain amount of packet loss and do not always require data retransmission. The mechanism RTP uses typically informs a source about the quality of delivery. The source then adapts its sending rate accordingly. If the rate of packet loss is very high, the source might switch to a lower-quality transmission, thereby placing less load on the network. A real-time application can also provide the data required for retransmission. Thus, recent data can be sent instead of retransmitted old data. This approach is more practical in voice and video applications. If a portion of an ADU is lost, the application is unable to process the data, and the entire ADU would have to be retransmitted.

### Real-Time Session and Data Transfer

The TCP/IP and OSI models divide the network functionalities, based on a layered architecture. Each layer performs distinct functions, and the data flows sequentially between layers. The layered architecture may restrict the implementation on certain functions out of the layered order. *Integrated layer processing* dictates a tighter coupling among layers. RTP is used to transfer data among *sessions* in real time. A session is a

logical connection between an active client and an active server and is defined by the following entities:

- *RTP port number*, which represents the destination port address of the RTP session. Since RTP runs over UDP, the destination port address is available on the UDP header.
- *IP address* of the RTP entity, which involves an RTP session. This address can be either a unicast or a multicast address.

RTP uses two relays for data transmission. A *relay* is an intermediate system that acts as both a sender and a receiver. Suppose that two systems are separated by a firewall that prevents them from direct communication. A relay in this context is used to handle data flow between the two systems. A relay can also be used to convert the data format from a system into a form that the other system can process easily. Relays are of two types: *mixer* and *translator*.

A *mixer relay* is an RTP relay that combines the data from two or more RTP entities into a single stream of data. A mixer can either retain or change the data format. The mixer provides timing information for the combined stream of data and acts as the source of timing synchronization. Mixers can be used to combine audio streams in real-time applications and can be used to service systems that may not be able to handle multiple RTP streams.

The *translator* is a device that generates one or more RTP packets for each incoming RTP packet. The format of the outgoing packet may be different from that of the incoming packet. A *translator relay* can be used in video applications in which a high-quality video signal is converted to a lower-quality in order to service receivers that support a lower data rate. Such a relay can also be used to transfer packets between RTP entities separated by an application-level firewall. Translators can sometimes be used to transfer an incoming multicast packet to multiple destinations.

### RTP Packet Header

RTP contains a fixed header and an application-specific variable-length header field. Figure 18.8 shows the RTP header format. The RTP header fields are:

- *Version* (V), a 2-bit field indicating the protocol version.
- *Padding* (P), a 1-bit field that indicates the existence of a padding field at the end of the payload. Padding is required in applications that require the payload to be a multiple of some length.
- *Extension* (X), a 1-bit field indicating the use of an extension header for RTP.

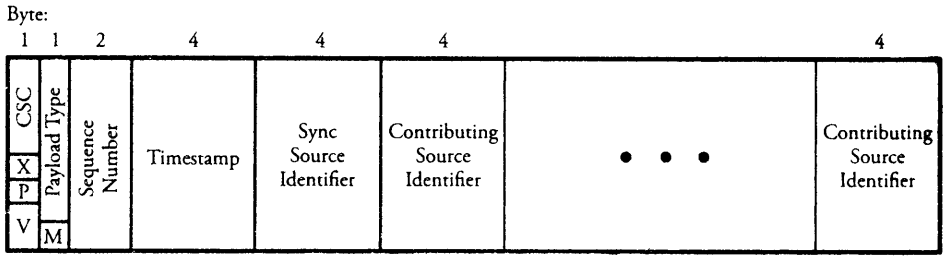


**Figure 18.8** Packet format for the real-time transport protocol

- *Contributing source count* (CSC), a 4-bit field that indicates the number of contributing source identifiers.
- *Marker* (M), a 1-bit field indicating boundaries in a stream of data traffic. For video applications, this field can be used to indicate the end of a frame.
- *Payload type*, A 7-bit field specifying the type of RTP payload. This field also contains information on the use of compression or encryption.
- *Sequence number*, a 16-bit field that a sender uses to identify a particular packet within a sequence of packets. This field is used to detect packet loss and for packet reordering.
- *Timestamp*, a 32-bit field enabling the receiver to recover timing information. This field indicates the timestamp when the first byte of data in the payload was generated.
- *Synchronization source identifier*, a randomly generated field used to identify the RTP source in an RTP session.
- *Contributing source identifier*, an optional field in the header to indicate the contributing sources for the data.

Overall, the main segment of an RTP header includes 12 bytes and is appended to a packet being prepared for multimedia application.

## 18.3.2 Real-Time Control Protocol (RTCP)

The Real-Time Transport Protocol (RTCP) also runs on top of UDP. RTCP performs several functions, using multicasting to provide feedback about the data quality to all session members. The session multicast members can thus get an estimate of the

performance of other members in the current active session. Senders can send reports about data rates and the quality of data transmission. Receivers can send information about packet-loss rate, jitter variations, and any other problems they might encounter. Feedback from a receiver can also enable a sender to diagnose a fault. A sender can isolate a problem to a single RTP entity or a global problem by looking at the reports from all receivers.

RTCP performs *source identification*. RTCP packets contain some information to identify the source of the control packet. The rate of RTCP packets must also be kept to less than 5 percent of the total session traffic. Thus, this protocol carries out "rate control" of RTCP packets. At the same time, all session members must be able to evaluate the performance of all other session members. As the number of active members in a session increases, the transmission rates of the control packets must be reduced. RTCP is also responsible for *session control* and can provide some session-control information, if necessary.

### Packet Type and Format

RTCP transmits control information by combining a number of RTCP packets in a single UDP datagram. The RTCP packet types are *sender reports* (SR), *receiver reports* (RR), *source descriptor* (SDES), *goodbye* (BYE), and *application-specific types*. Figure 18.9 shows some RTCP packet formats. The fields common to all packet types are as follows:

- *Version*, a 2-bit field that indicates the current version.
- *Padding*, a 1-bit field that indicates the existence of padding bytes at the end of the control data.
- *Count*, a 5-bit field that indicates the number of SR or RR reports or the number of source items in the SDES packet.
- *Packet type*, an 8-bit field that indicates the type of RTCP packet. (Four RTCP packet types were specified earlier.)
- *Length*, a 16-bit field that represents the length of packet in 32-bit words minus 1.
- *Synchronization source identifier*, a field common to the SR and RR packet types; it indicates the source of the RTCP packets.

Figure 18.9 also shows a typical format of a sender report. The report consists of the common header fields and a block of sender information. The sender report may

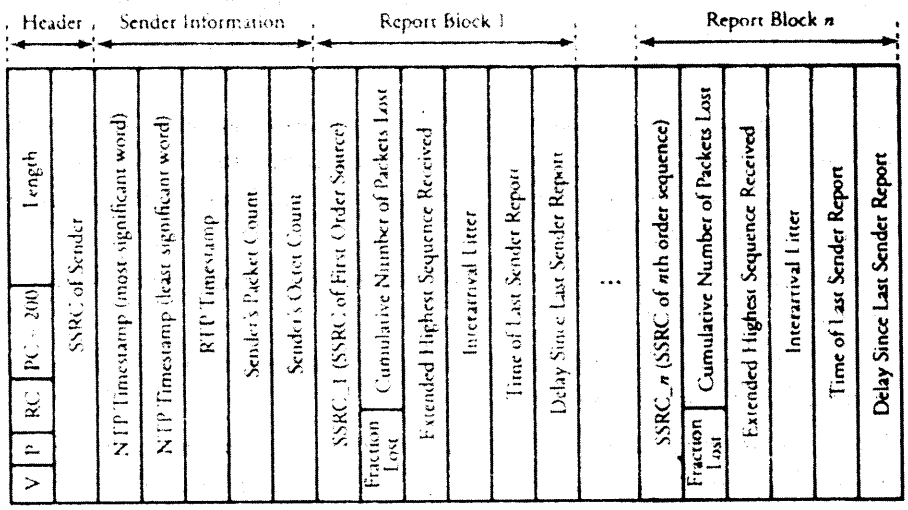


**Figure 18.9** Format of the SR packet in RTCP

also contain zero or more receiver report blocks, as shown in the figure. The fields in the sender information block are:

- *NTP timestamp*, a 64-bit field that indicates when a sender report was sent. The sender can use this field in combination with the timestamp field returned in receiver reports to estimate the round-trip delay to receivers.
- *RTP timestamp*, a 322-bit field used by a receiver to sequence RTP data packers from a particular source.
- *Sender's packet count*, a 32-bit field that represents the number of RTP data packets transmitted by the sender in the current session.
- *Sender's byte count*, a 32-bit field that represents the number of RTP data octets transmitted by the sender in the current session.

The SR packet includes zero or more RR blocks. One receiver block is included for each sender from which the member has received data during the session. The RR block includes the following fields:

- SSRC_*n*, a 32-bit field that identifies the source in the report block, where *n* is the number of sources.
- *Fraction lost*, an 8-bit field indicating the fraction of data packet loss from source SSRC_*n* since the last SR or RR report was sent.

- *Cumulative number of packets lost*, a 24-bit field that represents the total number of RTP data packets lost from the source in the current active session identified by SSRC_$n$.
- *Extended highest sequence number received*, the first 16 least-significant bits, used to indicate the highest sequence number for packets received from source SSRC_$n$. The first 16 most-significant bits indicate the number of times that the sequence number has been wrapped back to zero.
- *Interarrival jitter*, a 32-bit field used to indicate the jitter variations at the receiver for the source SSRC_$n$.
- *Last* SR *timestamp*, a 32-bit field indicating the timestamp for the last SR packet received from the source SSRC_$n$.
- *Delay since last* SR, a 32-bit field indicating the delay between the arrival time of the last SR packet from the source SSRC_$n$ and the transmission of the current report block.

Receivers in RTCP can provide feedback about the quality of reception through a receiver report. A receiver that is also a sender during a session, it also sends the sender reports.

### 18.3.3 Estimation of Jitter in Real-Time Traffic

The jitter factor is a measure of the delay experienced by RTP packets in a given session. The average jitter can be estimated at the receiver. Let us define the following parameters at the receiver:

| | |
|---|---|
| $t_i$ | Timestamp of the RTP data packet $i$ indicated by the source. |
| $a_i$ | Arrival time of the RTP data packet $i$ at the receiver. |
| $d_i$ | Measure of difference between interarrival time of RTP packets at receiver and the one for packet departure from the source. This value represents the difference in packet spacing at source and receiver. |
| $E[i]$ | Estimate of average jitter until the time of packet $i$ arrival. |

The difference interval $d_i$ is given by

$$d_i = (a_i - a_{i-1}) - (t_i - t_{i-1}). \tag{18.1}$$

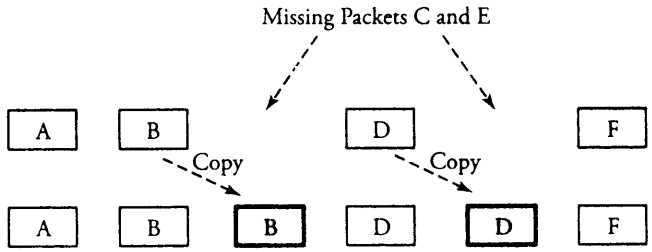


**Figure 18.10** Missing voice packets and reconstructing the data stream

The estimated average jitter until the time of packet $i$ arrival is given by

$$E[i] = k\left(E[i-1] + |d_i|\right), \tag{18.2}$$

where $k$ is a normalizing coefficient. The interarrival jitter value indicated in the sender report provides useful information on network conditions to the sender and the receiver. The jitter value can be used as an estimate to calculate the variation of network congestion.

RTP packet-sequence numbers are used to help a receiver sequence packets in order to recover lost packets. When packets arrive out of order at the receiver, the sequence number can be used to assemble data packets. Consider Figure 18.10. When certain packets are lost, the gaps are filled in with previously received data packets. As shown in the figure, packet D is replayed twice, since packet C is lost. This mechanism can help reduce the pips and clicks in voice signals owing to lost packets. This reconstructed data stream is sent to the receiver, with the lost packets replaced by previously received packets. This can significantly improve the latency.

## 18.4 Distributed Multimedia Networking

Video streaming presents a significant challenge to network designers. A video in a single server can be streamed from a video server to a client at the client request. The high-bit-rate video streaming must sometimes pass through many Internet service providers, leading to the likelihood of significant delay and loss on video. One practical solution to this challenge is to use *content distribution networks* (CDNs) for distributing stored multimedia content.

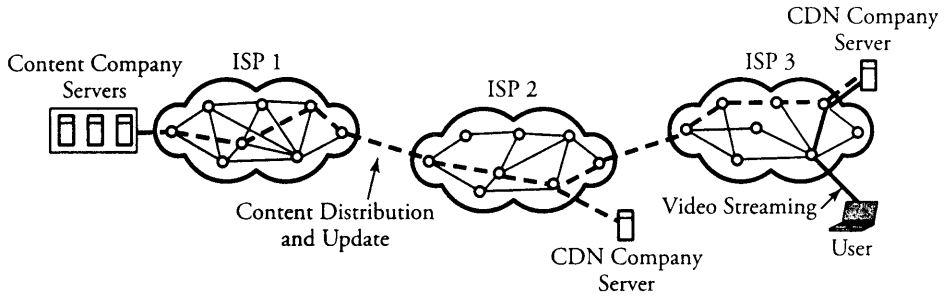


**Figure 18.11** Video streaming provided to a user in ISP domain 3 through a branch of a CDN company in the same domain while it interacts with its other servers in ISP 2 and ISP 1 domains. The source of the requested content is updated by the content company in ISP 1 domain.

## 18.4.1 Content Distribution Networks (CDNs)

A *content distribution network* (CDN) is a group of proxy servers located at a certain strategic location around the Internet service provider. CDNs ensure that a download request can always be handled from the nearest server. With CDNs, the content of streaming is geographically brought to a user unable to access the content at the desired data rate in the original location. Therefore, a user deals with a *content provider*, such as private TV broadcast companies, not ISPs. The content company hires a CDN company to deliver its content (streaming video) to the user. This does not mean that a CDN company cannot expose its server to ISPs.

A CDN company has several CDN centers around the Internet. Each group of servers is installed in proximity to ISP access points. At the request of a user, the content is provided by the closest CDN server that can best deliver the content. Figure 18.11 shows video streaming being provided to a user in ISP 3 domain through a branch of a CDN company in the same domain while this company interacts with its other servers in ISP 2 and ISP 1 domains. The source of the requested content is updated by the content company, which in this case is located in ISP 1 domain.

## 18.4.2 CDN Interactions with DNS

CDNs use Domain Name System (DNS) servers to direct browsers to the correct server. For example, assume that the URL of a content company is www.contentco.com, as shown in Figure 18.12. Assume that this company has produced a video movie named movie.mpg with MPEG format and that the content of this movie is placed on one

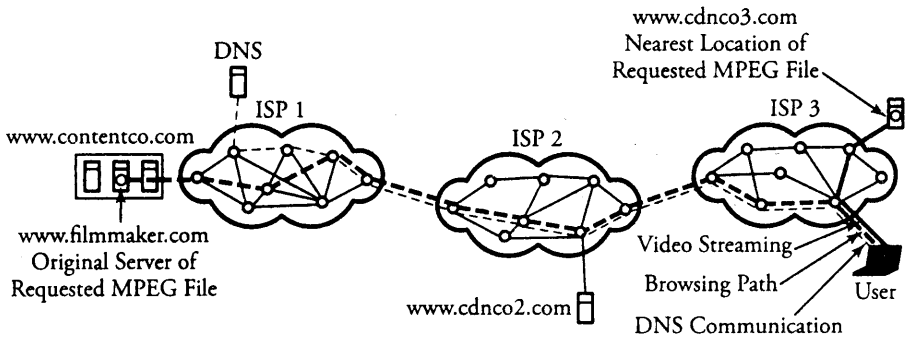


**Figure 18.12** Video streaming provided to a user in ISP 3 domain using domain name system (DNS) servers

of its servers, named www.filmmaker.com. Therefore, this MPEG file has a reference as www.filmmaker.com/movies/movie.mpg. Also, suppose that a CDN company interacting with the content company is accessed through www.cdnco.com. The content company replaces this reference with one affected by the CDN company's URL domain name as www.cdnco.com/www.filmmaker.com/movies/movie.mpg. Now, when a user like the one in Figure 18.12 requests the movie.mpg video object, its browser sends its request for the base object to the origin server, www.filmmaker.com. Then the browser finds the object at www.cdnco.com/www.filmmaker.com/movies/movie.mpg.

Now, the browser knows that www.cdnco.com is also involved and requests the corresponding DNS server for the location of the CDN company. The DNS server is set up to transfer all movie queries about www.contentco.com to the corresponding DNS server. Finally, the corresponding DNS server extracts the IP address of the requesting browser and returns the IP address of the closest CDN server to the browser, located in the ISP 3 domain in the previous example. Consequently, the look-up table of the DNS server finds this best (nearest) CDN's server URL at www.cdnco3.com/www.filmmaker.com/movies/movie.mpg. Note that the CDN's link for this example is www.cdnco3.com, corresponding to the CDN company's server in the ISP 3, as seen in the figure. This way, the user communicates with the CDN server located nearby with minimal congestion.

## 18.4.3 Providing QoS to Streaming

Consider Figure 18.13, in which three user ands user A, user B, and user C all belong to a local area network and are using the Internet simultaneously. User A is contacting

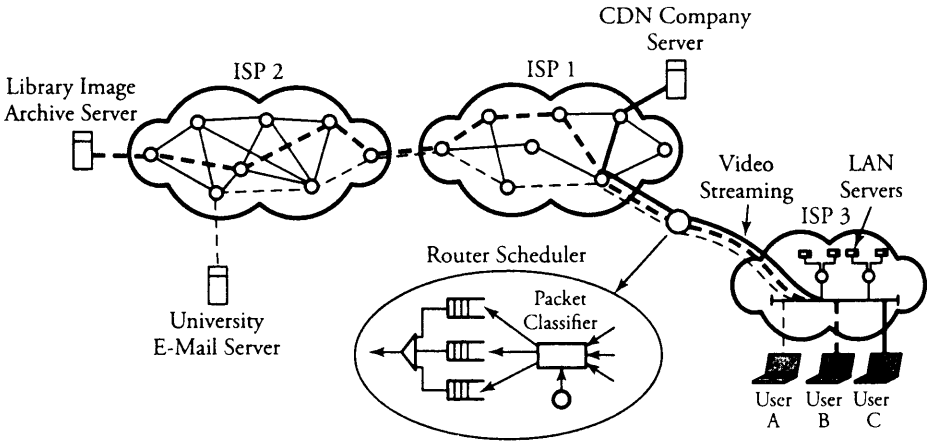


**Figure 18.13** Providing QoS at the main router connecting to an Internet service provider

a university e-mail server and requires small bandwidth. User B is searching images in a library archive server and requires a non-real-time but modest bandwidth. User C is using a video streaming service from a CDN server and requires substantial high-quality bandwidth. Packet flows from these users need to be scheduled as shown in the main router between the LAN and ISP 1.

Video streaming, e-mail, and image packets in the best-effort Internet are mixed in the output queue of the main exit router of a domain. Under such circumstances, a burst of packets, primarily from the image file source, could cause IP video streaming packets to be excessively delayed or lost at the router. One solution in this case is to mark each packet as to which class of traffic it belongs to. This can be done by using the type-of-service (ToS) field in IPv4 packets. As seen in the figure, transmitted packets are first classified in terms of their priorities and are queued in a first in, first out (FIFO) order. The priority of an image file can be equal to or less than the one for video streaming, owing to the arrangement of purchased services.

## 18.5 Stream Control Transmission Protocol (SCTP)

The *Stream Control Transmission Protocol* (SCTP) provides a general-purpose transport protocol for message-oriented applications. It is a reliable transport protocol for transporting stream traffic, can operate on top of unreliable connectionless networks, and

offers acknowledged and nonduplicated transmission data on connectionless networks (datagrams). SCTP has the following features.

- The protocol is error free. A retransmission scheme is applied to compensate for loss or corruption of the datagram, using checksums and sequence numbers.
- It has ordered and unordered delivery modes.
- SCTP has effective methods to avoid flooding congestion and masquerade attacks.
- This protocol is *multipoint* and allows several streams within a connection.

In TCP, a stream is a sequence of bytes; in SCTP, a sequence of variable-sized messages. SCTP services are placed at the same layer as TCP or UDP services. Streaming data is first encapsulated into packets, and each packet carries several correlated chunks of streaming details. If an MPEG movie is displayed live over the Internet, a careful assignment of data per packet is required. An MPEG video consists of frames, each consisting of $n \times m$ blocks of pixels, with each pixel normally an $8 \times 8$ matrix. In this case, each block of pixels can be encapsulated into a chunk, where each row of the block is formatted as a packet.

## 18.5.1 SCTP Packet Structure

Figure 18.14 shows the structure of streaming packets used in SCTP. An SCTP packet is also called a *protocol data unit* (PDU). As soon as the streaming data is ready to be transmitted over IP, an SCTP packet forms the payload of an IP packet. Each packet consists of a *common header* and *chunks*. The streaming data is distributed over packets, and each packet carries correlated "chunks" of streaming data. Multiple chunks representing multiple portions of streaming information are in fact multiplexed into one packet up to the path-maximum packet size.

A chunk header starts with a chunk *type* field used to distinguish data chunks and any other types of control chunks. The *type* field is followed by a *flag* field and a chunk *length* field to indicate the chunk size. A chunk, and therefore a packet, may contain either control information or user data. The common header begins with the *source port number* followed by the *destination port number*. SCTP uses the same port concept as TCP or UDP does. A 32-bit *verification tag* field is exchanged between the end-point servers at startup to verify the two servers involved. Thus, two tag values are used in a connection. The common header consists of 12 bytes. SCTP packets are protected by a 32-bit checksum. The level of protection is more robust than the 16-bit checksum of TCP and UDP.

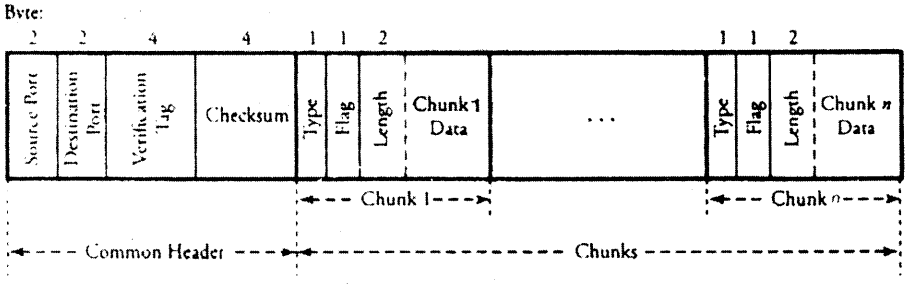


**Figure 18.14** The structure in the stream control transmission protocol (SCTP) streaming data is encapsulated into packets and each packet carries several correlated chunks of streaming details.

Each packet has $n$ chunks, and each chunk is of two types: *payload data chunk* for transmitting actual streaming data and *control chunks* for signaling and control. Signaling and control chunks are of different types, as follows:

- *Initiation*, to initiate an SCTP session between two end points
- *Initiation acknowledgment*, to acknowledge the initiation of an SCTP session
- *Selective acknowledgment*, to be transmitted to a peer end point to acknowledge received data chunks
- *Heartbeat request*, to probe the reachability of a particular destination transport address defined in the session
- *Heartbeat acknowledgment*, to respond to the heartbeat request chunk
- *Abort*, to close a session
- *Shutdown*, to initiate a graceful close of a session
- *Shutdown acknowledgment*, to acknowledge receipt of the shutdown chunk once the shutdown process is completed
- *Operation error*, to notify the other party of a certain error
- *State cookie*, sent by the source to its peer to complete the initialization process
- *Cookie acknowledgment*, to acknowledge receipt of a state cookie chunk
- *Shutdown complete*, to acknowledge receipt of the shutdown acknowledgment chunk at the completion of the shutdown process

SCTP can easily and effectively be used to broadcast live video clips or full-color video movies. The SCTP exercises at the end of this chapter explore SCTP further.

## 18.6 Self-Similarity and Non-Markovian Streaming Analysis

Multimedia applications are delay sensitive and loss tolerant, unlike static-content communications, which are delay and loss intolerant. Distributed multimedia networks must be able to support the exchange of multiple types of information, such as voice, video, and data among users while also satisfying the performance requirements of each individual application. Consequently, the expanding diversity of high-bandwidth communication applications calls for a unified, flexible, and efficient network to prevent any congestion.

A network handling heavy video streaming must reserve some resources based on the source QoS. With reservation techniques, lossless transmission is guaranteed for the entire duration of the block; otherwise, the block is lost. However, sudden changes in the total volume of traffic at a node can impact the performance of streaming transmission. Multimedia networks are expected to support a large number of *bursty sources* with different characteristics. This fact enforces the use of processes other than Poisson for describing network traffic. The aggregated arrivals of packets are assumed to form *stream batches* of packets.

### 18.6.1 Self-Similarity with Batch Arrival Models

Section 11.5.3 explained that some traffic patterns indicate significant "burstiness," or variations on a wide range of timescales. Bursty traffic, such as a video stream, can be viewed as a *batch* of traffic units and described statistically using *self-similarity patterns*. In a self-similar process, that packet loss and delay behavior are different from those in traditional network models using Poisson models. Self-similar traffic can be constructed by multiplexing a large number of ON/OFF sources that have ON and OFF intervals. This mechanism corresponds to a network of streaming servers, each of which is either silent or transferring video stream at a constant rate. Using this traffic, the distributions of transmission times and quiet times for any particular session are ***heavy tailed***, which is an essential characteristic of traffic self-similarity.

The discrete-time representation of a communication system is the natural way to capture its behavior. In most communication systems, the input process to a queue is not *renewal* but correlated. A renewal process is a process in which the interval between consecutive occurrences of a certain event are independently and identically distributed. For example, the Poisson process is a renewal case with an exponential distribution. In a practical environment, the input process of packetized voice, data, or video traffic to a multiplexer does not form a renewal process but is bursty and correlated.

In streaming-traffic analysis, a batch of packets may arrive simultaneously. With the relatively more accurate traffic model being presented here, maximum and average traffic rates over *a number of given intervals* are determined. Using maximum rate and average rate parameters in our analysis can be efficient in capturing the burstiness characteristics of streaming sources. This method is especially effective for realistic sources, such as compressed streaming video. In such situations, a source can even transmit at its peak rates when sending its large-size frames immediately followed by smaller frames. In the performance analysis, the packet-loss probability and the impact of increase in switching speed to link-speed ratio on the throughput are of particular interest.

In this analysis, consider a small buffered multiplexer or router, as large queuing delays are not expected in a multimedia network with a real-time transmission. A bursty arrival is modeled as a *batch*, or packets with identical interarrivals. This model captures the multirate burstiness characteristic of realistic sources. One property of self-similarity is that an object as an image is preserved with respect to scaling in space or time. In this environment, the traffic-relational structure remains unchanged at varying timescales. For any time $t > 0$ and a real number $a > 0$, a self-similar process, $X(t)$, is a continuous-time stochastic (random) process with parameter $0.5 < H < 1$ if it satisfies

$$X(t) = \frac{X(at)}{a^H}, \tag{18.3}$$

where parameter $H$ is known as the *Hurst parameter*, or *self-similarity parameter*. The Hurst parameter is an important factor in bursty traffic, representing a measure of the dependence length in a burst. The closer $H$ is to its maximum, 1, the greater the persistence of long-range dependence.

The expected values of both sides in Equation (18.3) must then be related as

$$E[X(t)] = \frac{E[X(at)]}{a^H}. \tag{18.4}$$

This result indicates that a self-similar process when $H = 0.5$ can also be obtained from the *Brownian random process* discussed in Section C.4.2. Based on Equation (C.34), and for any time increment $\delta$, the increment of process, $X(t + \delta) - X(t)$, has the following distribution:

$$P[X(t+\delta) - X(t) \le x] = \frac{1}{\sqrt{2\pi\delta}} \int_{-\infty}^{x} e^{-y^2/2\delta} dy. \tag{18.5}$$

To better understand the behavior of aggregated batch sequences of traffic, we can also present the self-similar process, $X(t)$, in terms of a discrete-time version $X_{n,m}$,

defined at discrete points in time. In this process, $n \in \{1, 2, \ldots\}$ is discrete time and $m$ is the batch size. Thus:

$$X_{n,m} = \frac{1}{m} \sum_{i=(n-1)m+1}^{nm} X(i). \tag{18.6}$$

Not that for $X_{n,m}$, the corresponding aggregated sequence with a level of aggregation $m$, we divide the original series $X(t)$ into nonoverlapping blocks of size $m$ and average them over each block where index $n$ labels blocks.

**Example.** Consider traffic with batch size $m = 4$. Self-similar process averaging is expressed by

$$X_{n,4} = \frac{1}{4}(X(4n-3) + X(4n-2) + X(4n-1) + X(4n)). \tag{18.7}$$

### Heavy-Tailed Distributions

Self-similarity implies that traffic has similar statistical properties in a timescale range, such as milliseconds, seconds, minutes, hours, or even days. In a practical situation, in which bursts of streams are multiplexed, the resulting traffic tend to produce a bursty aggregate stream. In other words, the traffic has a long-range dependence characterized by a *heavy-tailed distribution*. A random variable has heavy-tailed distribution if for $0 < \alpha < 2$, its cumulative distribution function (CDF)

$$F_X(x) = P[X \leq x] \sim 1 - \frac{1}{x^{\alpha}} \tag{18.8}$$

as $x \to 0$.

Heavy-tailed distributions are typically used to describe the distributions of burst lengths. A simple example of heavy-tailed distributions i the *Pareto distribution*, which is characterized by the following CDF and probability density function (PDF):

$$\begin{cases} F_X(x) = 1 - \left(\frac{k}{x}\right)^{\alpha} \\ f_X(x) = \frac{\alpha k^{\alpha}}{x^{\alpha+1}} \end{cases}, \tag{18.9}$$

where $k$ is the smallest possible value of the random variable. For this distribution, if $\alpha \leq 1$, the distribution has infinite mean and variance; if $\alpha \leq 2$, the distribution has infinite variance. To define *heavy tailed*, we can now use a comparison over PDFs of a Pareto distribution and exponential distribution. Making this comparison shows how

the tail of the curve in a Pareto distribution takes much longer to decay. A random variable that follows a heavy-tailed distribution may be very large with a probability that cannot be negligible.

## 18.7 Summary

This chapter explored transport mechanisms for application delivery with the highest possible quality. We focuesed on media applications, such as streaming audio and video, one-to-many transmission of real-time audio and video, and real-time interactive audio and video.

We discussed how sampled and digitized voice are treated in networks and investigated two VoIP signaling session protocols: *Session Initiation Protocol* (SIP) and the *H.323 series of protocols*, showing how calling and numbering for these protocols can be achieved. SIP identifies user location signals, call setup, call termination, and busy signals. User agents assist in initiating or terminating a phone call in VoIP networks. User agents can be implemented in a standard telephone or in a laptop with a microphone that runs some software. The signaling in H.323 uses both UDP and TCP and is partitioned into *call setup*, *initial communication capability*, *audio/video communication establishment*, *communications*, and *call termination*. Various scheduling-policy mechanisms—*priority queueing*, *weighted fair queuing*, and *class-based weighted fair queuing*—can provide the foundation of a QoS networking architecture.

Senders use various *real-time media transport protocols* to send a stream of data at a constant rate. One of the protocols for real-time transmission is the *Real-Time Transport Protocol* (RTP), which provides application-level framing. Real-time applications, such as voice and video, can tolerate a certain amount of packet loss and do not always require retransmission of data. But if the rate of packet loss is very high, the source might use a lower-quality transmission, thereby placing less load on the network.

A video in a single server can be streamed from a video server to a client for each client request. *Content distribution networks* (CDNs) can be used for streaming data. A video streaming provided to a user in an ISP domain can use Domain Name System (DNS) servers to direct browsers to the correct server. The *Stream Control Transmission Protocol* (SCTP) is a general-purpose transport protocol for transporting stream traffic. SCTP offers acknowledged and nonduplicated transmission of basic units of data, or datagrams, on connectionless networks.

Finally, a *non-Markovian analysis of streaming traffic* was presented. A stream of packets generated from a server source can be modeled as a discrete sequence of events and defined as a discrete-time 0-1 valued process called self-similar traffic.

The last two chapters of the book consider two advanced and related subjects: *mobile ad hoc networks* and *wireless sensor networks*.

## 18.8 Exercises

1. We want to transmit a speaker's voice through a digital radio broadcast system that uses 8-bit code PCM (explained in Chapter 17) placed on the Internet.
   (a) How many bits are produced in each second?
   (b) Estimate the size of an encapsulated RTP/IP datagram (the basic transmission unit of data in the Internet) that carries a half second of PCM-encoded audio using UDP.
2. Two voice sources come to one server for live transmission together, using RTP packets. Each source bandwidth has been compressed to 31 Kb/s.
   (a) Show an overview of the encapsulated RTP/IP datagram, and estimate the size of each packet, utilizing an RTP packet using UDP.
   (b) How many packets are produced each 5 minutes?
3. We have five packets to transmit in real time. The estimated average jitter until the time of the first packet is 0.02 ms. Table 18.1 shows the timestamps of RTP data packets indicated by the source, $t_i$, and the arrival times of RTP packets at the receiver $a_i$. Assume that the normalizing coefficient $k$ is 0.2.
   (a) Estimate the average jitter until every packet has arrived.
   (b) What would be possible reason(s) that $t_i$ increases at a different rate from $i$?

**Table 18.1** Exercise 3 example of source timestamps and receiver arrival times for five packets

| $i$ | $a_i$ | $t_i$ |
|---|---|---|
| 1 | 43 | 69 |
| 1 | 45 | 74 |
| 1 | 47 | 73 |
| 1 | 49 | 91 |
| 1 | 51 | 99 |

**4.** SCTP is applied to transmit a color video clip between two points of an IP network requiring 4 minutes of network use. Each image of this video clip consists of 1,024 × 1,280 pixel blocks, and the video consists of 30 images per second. The video clip is not compressed but is passed through the quantization process, and each pixel can be a value among a sample space of 77 numbers. One-tenth of each row of a frame (image) is formatted by one packet.
(a) Find the size of each SCTP chunk, including its header.
(b) Find the total size of each SCTP packet, including its header.
(c) Find the total size of bits transmitted, based only on payload packets.
(d) Determine the required bandwidth between these two nodes.

**5.** Suppose that a live transmission of a compressed color video movie between two points of an IP network requires 2 hours of network use. We want to apply SCTP for this transmission. Each image of this video consists of 1,024 × 1,280 pixel blocks, and the video consists of 30 images per second. One option is to encapsulate each block of pixels in a chunk, allowing each row of a frame (image) to be formatted by one packet. Assume that each pixel block is compressed on average to 10 phrases and each phrase requires on average 5 bits.
(a) Find the size of each SCTP chunk, including its header.
(b) Find the total size of each SCTP packet, including its header.
(c) Find the total size of bits transmitted, based only on payload packets.
(d) Determine the required bandwidth between these two nodes.

**6.** Assume that a real-time bursty source is modeled using a Brownian motion process $X(t)$. Let $Z(t) = X(t) + 2t$.
(a) Find the probability distribution function (PDF) of $Z(t)$.
(b) Find the joint PDF of $Z(t)$ and $X(t + 1)$.

**7.** In Figure 18.15, a remote medical emergency unit is streaming the 70-cycle/minute heartbeat of a patient to a hospital, using SCTP. Each heart cycle has six peaks: P, Q, R, S, T, and U. Suppose that all information about each peak is formatted into one chunk of the SCTP data packet. Because of their importance and complexity, each of the Q, R, and S pulses requires four samples; P, T, or U require only one sample each. Assume each sample is encoded by 8 bits.
(a) Determine the required bandwidth between the unit and the hospital.
(b) If multiple samples are included in a packet, find the maximum number of heartbeat cycles that can be formatted into a packet.

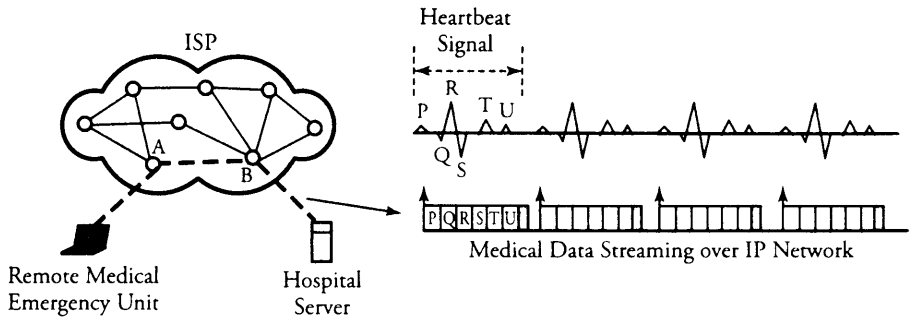


**Figure 18.15** Exercise 7 example of remote medical emergency unit streaming a patient's heartbeat of to a hospital. The heartbeat signal is converted to stream of packets.

(c) Evaluate the approach presented in part (b), and compare it with the original one.

8. For self-similar traffic, we have seen the relationship between the expected values in Equation (18.4), using the Hurst parameter, $H$.
   (a) Derive a relationship between the variances of the two sides of Equation (18.3).
   (b) Compare situations in which the Hurst parameter, $H$, takes values 0.5, 0.8, or 1 in Equation (18.3).
9. To better understand the behavior of bursty traffic, such as a video streaming source, assume that the traffic is described by a Pareto distribution with $k = 1$. Plot PDFs of the following two cases of $\alpha$, and compare them with that of an exponential distribution. Comment on the heavy-tailed phenomenon.
   (a) $\alpha = 0.8$
   (b) $\alpha = 3.8$
10. *Computer simulation project*. We want to simulate self-similar traffic. First, develop a random number generator. Then, develop source code that defines the following components, and integrate the these two programs to simulate the self-similar traffic:
    (a) Source ID
    (b) Priority of the packets
    (c) Elapsed time (in byte transmission times)
    (d) Packet size (bytes)
    (e) Number of packets remaining in current burst

CHAPTER 19

# Mobile Ad-Hoc Networks

*Mobile ad-hoc networks* (MANETs) have had a profound impact in the world of computer networks. Characterized by anytime/anywhere untethered establishment of a wireless network, the MANET infrastructure enables location-independent services. Ad-hoc networks do not need any fixed infrastructure to operate and support dynamic topology scenarios in which no wired infrastructure exists. This chapter covers the following topics on wireless mobile ad-hoc networks:

- *Overview of wireless ad-hoc networks*
- *Routing in ad-hoc networks*
- *Ad-hoc routing protocols for ad-hoc networks*
- *Security of ad-hoc networks*

A mobile user can act as a routing node, and a packet can be routed from a source to its destination without having any static router in the network. Two classes of routing strategies in ad-hoc networks are *table-driven routing protocols* and *source-initiated routing protocols*. Security of ad-hoc networks is a key issue. Ad-hoc networks are, by their nature, vulnerable to attacks. An intruder can easily attack ad-hoc networks by loading available network resources and disturbing the normal operation of routing protocols by modifying packets.

## 19.1 Overview of Wireless Ad-Hoc Networks

Wireless *mobile ad-hoc network* (MANET) technology is designed for the establishment of a network anywhere and anytime, without any fixed infrastructure to support the mobility of the users in the network. In other words, a wireless ad-hoc network is a collection of mobile nodes with a dynamic network infrastructure forming a temporary network. Such networks no central server or base station for providing connectivity, and all network intelligence must be placed inside the mobile user devices. Figure 19.1 gives overview of an ad-hoc network, where wireless mobile nodes have formed a network, with one node too far to reach.

In such an environment, each mobile user acts as a routing node, and a packet is routed from a source to its destination by incorporating of other network users. Since the topology of an ad-hoc network can change quickly and unpredictably, it should be adaptable to changes, such as when a link breaks, a node leaves the network, or a new node is attached to the network. Thus, unlike intradomain routing algorithm for regular networks, if a node leaves an ad-hoc network, all affected nodes can discover new routes. Ad-hoc networks have several types of applications

- *Rescue operations*. In an emergency public disaster, such as an earthquake, ad-hoc networks can be set up at a location where no infrastructure is present. Ad-hoc networks can be used to support network connectivity when no fixed network is available.

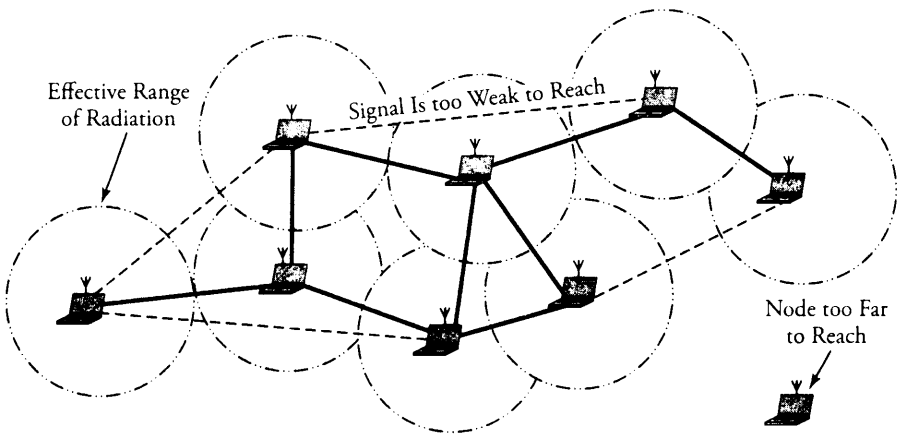


**Figure 19.1** Overview of an ad-hoc network

- *Military*. Ad-hoc networks can be used in a battle zone, for a military command and mobile units.
- *Law enforcement and security operations*. An ad-hoc network can be used in a temporary security operation, acting as a mobile surveillance network.
- *Home networks*. An ad-hoc network can be used to support seamless connectivity among various devices.
- *Conferencing*. Ad-hoc networks can be set up for a presentation. An audience can download a presentation, browse the slides on a portable device, print them on the local printer, or e-mail the presentation to an absent colleague.

Ad-hoc networks must possess several unique features. One is *automatic discovery* of available services. Each time a new service becomes available, an ad hoc networking device has to configure use of the new service. As an ad-hoc network lacks centralized administration, the network must be able to prevent network collapse when one of the mobile nodes moves out of transmitter range. In general, nodes should be able to enter or leave the network as they wish. Thus, every node acts as both a host and a router, and the network must be intelligent enough to handle network dynamics. This property is called *self-stabilization.*

One of the most common tasks of an ad-hoc network is to multicast a message to many users efficiently. In such an environment, networks are subject to severe blocking. Thus, the performance of an ad hoc system depends on the stability of the network architecture. The inclusion of all these features in ad-hoc networks requires considerable architectue sophistication.

## 19.2 Routing in Ad-Hoc Networks

The lack of a backbone infrastructure makes packet routing in ad-hoc networks a challenging task. A routing protocol should be able to automatically recover from any problem in a finite amount of time without human intervention. Conventional routing protocols are designed for nonmoving infrastructures and assume that routes are bidirectional, which is not always the case for ad-hoc networks. Identification of mobile terminals and correct routing of packets to and from each terminal while moving are certainly challenging.

Since the topology of an ad-hoc network is dynamic, reserving resources and sustaining QoS are difficult. In an ad-hoc medium, it may not be possible to communicate bidirectionally, so ad-hoc routing protocols should assume that links are unidirectional. The power of a wireless device is another important factor. The routing protocol also

has to support node stand-by mode. Devices such as laptops are very limited in battery power; hence, the use of stand-by mode to save power is important.

### 19.2.1 Classification of Routing Protocols

Ad-hoc routing protocols can be classified into two broad categories:

1. *Centralized versus distributed*. In centralized routing protocols, the routing decision is made at a central node. In distributed routing protocols, the routing decision is made by all the network nodes. Routing protocols in most efficiently designed ad-hoc networks are distributed to increase the reliability of the network. In such cases, nodes can enter or leave the network easily, and each node can make routing decisions, using other collaborative nodes.
2. *Static versus adaptive*. In static routing protocols, a route of a source/destination pair does not change because of any traffic condition or link failure. In adaptive routing protocols, routes may change because of any congestion.

Whether a protocol is centralized, distributed, static, or adaptive, it can generally be categorized as either *table driven* or *source initiated.*

#### Table-Driven Routing Protocols

*Table-driven*, or *proactive*, *routing protocols* find routes to all possible destinations ahead of time. are needed. The routes are recorded in the nodes' routing tables and are updated within the predefined intervals. Proactive protocols are faster in decision making but need more time to converge to a steady state, causing problems if the topology of the network continually changes. However, maintaining routes can lead to a large overhead. Table-driven protocols require every node to maintain one or more tables to store updated routing information from every node to all other nodes. Nodes propagate updated tables all over the network such that the routing information in each table corresponds to topological changes in the network.

#### Source-Initiated Routing Protocols

*Source-initiated*, or *reactive*, *routing protocols*, are on-demand procedures and create routes only when requested to do so by source nodes. A route request initiates a *route-discovery process* in the network and is completed once a route is discovered. If it exists at the time of a request, a route is maintained by a route-maintenance procedure until either the destination node becomes irrelevant to the source or the route is no longer

needed. On-demand protocols are more suitable for ad-hoc networks. In most cases, reactive protocols are desired. This means that the network reacts only when needed and does not broadcast information periodically. However, the control overhead of packets is smaller than for proactive protocols.

## 19.3 Routing Protocols for Ad-Hoc Networks

This section discusses three table-driven protocols and four source-initiated protocols. The table-driven protocols are the *Destination-Sequenced Distance Vector* (DSDV) protocol, the *Cluster-Head Gateway Switch Routing* (CGSR) protocol, and the *Wireless Routing Protocol* (WRP). The source-initiated protocols are the *Dynamic Source Routing* (DSR) protocol, the *Associative-Based Routing* (ABR) protocol, *Temporally Ordered Routing Algorithm* (TORA), and *Ad-Hoc On-Demand Distance Vector* (AODV) protocol.

### 19.3.1 Destination-Sequenced Distance Vector (DSDV) Protocol

The *Destination-Sequenced Distance Vector* (DSDV) protocol is a table-driven routing protocol based on the improved version of classical Bellman-Ford routing algorithm. DSDV is based on the *Routing Information Protocol* (RIP), explained in Chapter 7. With RIP, a node holds a routing table containing all the possible destinations within the network and the number of hops to each destination. DSDV is also based on *distance vector routing* and thus uses bidirectional links. A limitation of DSDV is that it provides only one route for a source/destination pair.

#### Routing Tables

The structure of the routing table for this protocol is simple. Each table entry has a sequence number that is incremented every time a node sends an updated message. Routing tables are periodically updated when the topology of the network changes and are propagated throughout the network to keep consistent information throughout the network.

Each DSDV node maintains two routing tables: one for forwarding packets and one for advertising incremental routing packets. The routing information sent periodically by a node contains a new sequence number, the destination address, the number of hops to the destination node, and the sequence number of the destination. When the topology of a network changes, a detecting node sends an update packet to its

Routing Table for Node 2

| Destination | Next Hop | Metric | Dest. Seq. No. |
|---|---|---|---|
| 1 | 1 | 1 | 123 |
| 2 | 0 | 0 | 516 |
| 3 | 3 | 1 | 212 |
| 4 | 4 | 1 | 168 |
| 5 | 4 | 2 | 372 |
| 8 | 1 | INF | 432 |

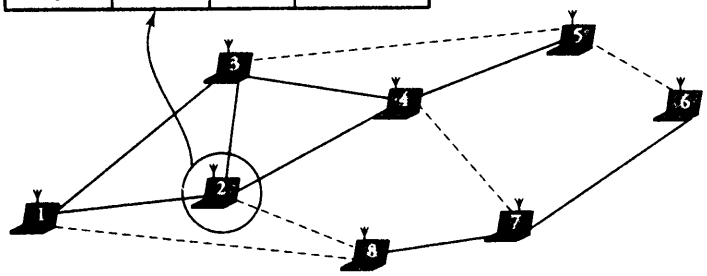


**Figure 19.2** A DSDV routing table

neighboring nodes. On receipt of an update packet from a neighboring node, a node extracts the information from the packet and updates its routing table as follows

### DSDV Packet Process Algorithm

1. If the new address has a higher sequence number, the node chooses the route with the higher sequence number and discards the old sequence number.
2. If the incoming sequence number is identical to the one belonging to the existing route, a route with the least cost is chosen.
3. All the metrics chosen from the new routing information are incremented.
4. This process continues until all the nodes are updated. If there are duplicate updated packets, the node considers keeping the one with the least-cost metric and discards the rest. ■

In case of a broken link, a cost of $\infty$ metric with a new sequence number (incremented) is assigned to it to ensure that the sequence number of that metric is always greater than or equal to the sequence number of that node. Figure 19.2 shows a routing table for node 2, whose neighbors are nodes 1, 3, 4, and 8. The dashed lines indicate no communications between any corresponding pair of nodes. Therefore, node 2 has no information about node 8.

The packet overhead of the DSDV protocol increases the total number of nodes in the ad-hoc network. This fact makes DSDV suitable for small networks. In large ad-hoc networks, the mobility rate—and therefore the overhead—increase, making the network unstable to the point that updated packets might not reach nodes on time.

## 19.3.2 Cluster-Head Gateway Switch Routing Protocol

The *Cluster-Head Gateway Switch Routing* (CGSR) protocol is a table-driven routing protocol. In a clustering system. each predefined number of nodes are formed into a *cluster* controlled by a *cluster head*, which is assigned using a distributed clustering algorithm. However, with the clustering scheme, a cluster head can be replaced frequently by another node, for several reasons, such as lower-level energy left in the node or a node moves out of contact.

With this protocol, each node maintains two tables: a *cluster-member table* and a *routing table*. The cluster-member table records the cluster head for each destination node, and the routing table contains the next hop to reach the destination. As with the DSDV protocol, each node updates its cluster-member table on receiving a new update from its neighbors.

### Clustering and Routing Algorithms

CGSR routing involves cluster routing, whereby a node is required to find the best route over cluster heads from the cluster-member table. Figure 19.3 shows an example of routing in an area in which six clusters have been formed. A node in cluster A is transmitting a packet to a node in cluster F. Nodes within each cluster route their packets to their own associated clusters. The transmitting node then sends its packet to the next hop, according to the routing table entry associated with that cluster head. The cluster head transmits the packet to another cluster head until the cluster head of the destination node is reached. The routing is made through a series of available cluster heads from A to F. Packets are then transmitted to the destination.

## 19.3.3 Wireless Routing Protocol (WRP)

The *Wireless Routing Protocol* (WRP) is a table-based protocol maintaining routing information among all nodes in the network. This protocol is based on the distributed Bellman-Ford algorithm. The main advantage of WRP is that it reduces the number

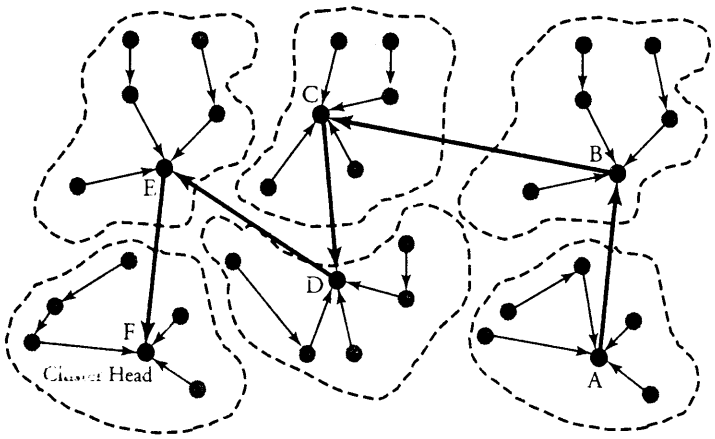


**Figure 19.3** Communication with cluster-head gateway switch routing (CGSR) protocol

of routing loops. With this protocol, each node in a network maintains four tables, as follows:

1. *Distance table*, which holds the destination, next hop, distance, and predecessors of each destination and each neighbor.
2. *Routing table*, which saves the destination address, next hop, distance, predecessor, and a marker for each destination, specifying whether that entry corresponds to a simple path.
3. *Link-cost table*, which provides the link cost to each neighbor and also the number of periodic update periods elapsed since the node received any error-free message from it.
4. *Message transmission-list table*, which records which updates in an update message are to be retransmitted and which neighbors need to acknowledge the retransmission. The table provides the sequence number of the update message, a retransmission counter, acknowledgments, and a list of updates sent in the update message.

Nodes should either send a message including the update message or a HELLO message to their neighbors. If a node has no message to send, it should send a HELLO message to ensure connectivity. If the sending node is new, it is added to the node's routing table, and the current node sends the new node a copy of its routing table content.

Once it detects a change in a route, a node sends the update message to its neighbors. The neighboring nodes then change their distance entries and look for new possible paths through other nodes. This protocol avoids the count-to-infinity issue present in most ad-hoc network protocols. This issue is resolved by making each node perform consistency checks of predecessor information reported by all its neighbors in order to remove looping and make a faster route convergence in the presence of any link or node failure.

### 19.3.4 Dynamic Source Routing (DSR) Protocol

The *Dynamic Source Routing* (DSR) protocol is an on-demand, or source-initiated, routing protocol in which a source node finds an unexpired route to a destination to send the packet. DSR quickly adapts to topological changes and is typically used for networks in which mobile nodes move with moderate speed. Overhead is significantly reduced with this protocol, since nodes do not exchange routing table information when there are no changes in the topology. DSR creates multiple paths from a source to a destination, eliminating route discovery when the topology changes. Similar to most ad-hoc networks, DSR has two phases: route discovery and route maintenance.

#### Route Discovery and Maintenance

Route discovery is initiated when a node wants to send packets to another node and no unexpired route to the destination is in its routing table. In such circumstances, the node first broadcasts a *route-request packet*, including the destination address, source address, and a unique identification number. When it receives a route-request packet, the neighboring node it looks at its table; if any route to the requested destination address is already present in the node's route record, the packet is discarded to avoid the looping issue. Otherwise, the node adds its own address to the preassigned field of the route-request packet and forwards it to its neighboring nodes.

When the route-request packet reaches a destination node or another intermediate node that has an unexpired route to the destination, this node generates a *route-reply packet*, which contains a route record of the sequence of nodes taken from the source to this node. Once the source receives all the route-reply packets, it updates its routing table with the best path to the destination and sends its packets through that selected route.

**Example.** Figure 19.4 shows an ad-hoc network with eight mobile nodes and a broken link (3-7). Node 1 wants to send a message to the destination, node 8. Node 1

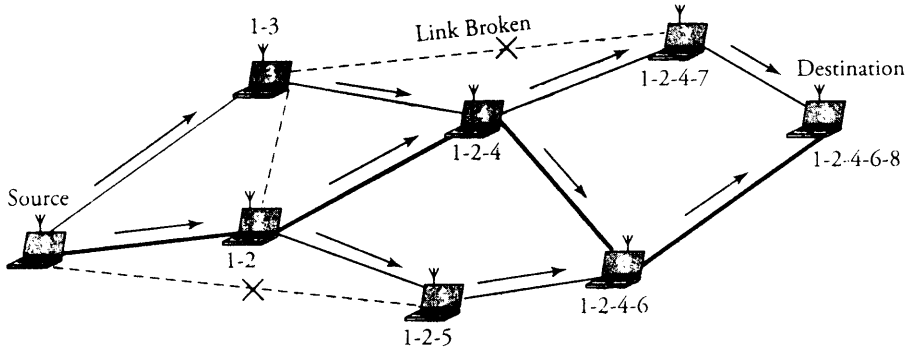


**Figure 19.4** Summary of DSR connection setup from node 1 to node 8

looks at its routing table, finds an expired route to node 8, and then propagates route-request packets to nodes 3 and 2. Node 3 finds no route to the destination and so appends the route record 1-3 to the route-request packet and forwards it to node 4. On receiving this packet, node 7 finds a route to the destination and so stops propagating any route-request packet and instead sends a route-reply packet to the source. The same happens when a route-request packet reaches the destination node 8 with a route record 1-2-4-6. When the source, node 1, compares all the route-reply packets, it concludes that the best route is 1-2-4-6-8 and establishes this path.

Route maintenance in this protocol is fast and simple. In case of a fatal error in the data-link layer, a *route-error packet* is generated from a failing node. When the route-error packet is received, the failing node is removed from its route cache, and all routes containing that node are truncated. Another route-maintenance signal is the acknowledgment packets sent to verify the correct operation of the route links.

### 19.3.5 Temporally Ordered Routing Algorithm (TORA)

The *Temporally Ordered Routing Algorithm* (TORA) is a source-initiated routing algorithm and, thus, creates multiple routes for any source/destination pair. The advantage of multiple routes to a destination is that route discovery is not required for every alteration in the network topology. This feature conserves bandwidth usage and increases adaptability to topological changes by reducing communication overhead.

TORA is based on the following three rules:

1. Route creation/discovery
2. Route maintenance
3. Route erasure

TORA uses three types of packets: *query packets* for route creation, *update packets* for both route creation and maintenance, and *clear packets* for route erasure. With TORA, nodes have to maintain routing information about adjacent nodes. This loop-free protocol is distributed based on the concept of *link reversal*.

Every node maintains a table describing the distance and status of all connected links. When a node has no route to a desired destination, a *route-creation process* starts. A query packet contains the destination ID, and an update packet contains the destination ID and the distance of the node. A receiving node processes a query packet as follows

- If the receiving node realizes that there are no further downstream links, the query packet is again broadcast. Otherwise, the node discards the query packet.
- If the receiving node realizes that there is at least one downstream link, the node updates its routing table to the new value and broadcasts an update packet.

Once it receives the update packet, a node sets its distance greater than the neighbor from which it received the packet and then rebroadcasts it. The update is eventually received by the source. When it realizes that there is no valid route to the destination, the node adjusts its distance and generates an update packet. TORA performs efficient routing in large networks and in mildly congested networks.

### 19.3.6 Associative-Based Routing (ABR) Protocol

*Associative-Based Routing* (ABR) is an efficient on-demand, or source-initiated, routing protocol. ABR's is better than WRPs' network-change adaptation feature, to the extent that it is almost free of loops and is free of packet duplications. In ABR, the destination node decides the best route, using node *associativity*. ABR is ideal for small networks, as it provides fast route discovery and creates shortest paths through associativity.

In ABR, the movements of nodes are observed by other nodes in the network. Each node keeps track of associativity information by sending messages periodically, identifying itself and updating the *associativity ticks* for its neighbors. If the associativity ticks exceed a maximum, a node has associativity stability with its neighbors. In other words, a low number of associativity ticks shows the node's high mobility, and high associativity indicates a node's sleep mode. The associativity ticks can be reset when a node or its neighbor moves to a new location.

Each point-to-point routing in ABR is connection oriented, with all nodes participating in setting up and computing a route. In a point-to-point route, the source node or any intermediate node decides the details of routes. If the communication must be of broadcast type, the source broadcasts the packet in a *connectionless routing* fashion.

*Route discovery* is implemented when a node wants to communicate with a destination with no valid route. Route discovery starts with sending a *query packet* and an *await-reply*. The broadcast query packet has the source ID, destination ID, all intermediate nodes' IDs, sequence number, *CRC*, *LIVE* field and a *TYPE* field to identify the type of message. When it receives a query packet, an intermediate node looks at its routing table to see whether it is the intended destination node; otherwise, it appends its ID to the list of all intermediate IDs and rebroadcasts the packet. When it receives all copies of the query packet, the destination node chooses the best route to the source and then sends a *reply packet* including the best route. This way, all nodes become aware of the best route and thereby make other routes to the destination invalid.

ABR is also able to perform *route reconstruction*. This function is needed for partial route discovery or invalid route erasure.

### 19.3.7 Ad-Hoc On-Demand Distance Vector (AODV) Protocol

The *Ad-Hoc On-Demand Distance Vector* (AODV) routing protocol is an improvement over DSDV and is a source-initiated routing scheme capable of both unicast and multicast routing. AODV establishes a required route only when it is needed as opposed to maintaining a complete list of routes, with DSDV. AODV uses an improved version of the *distance vector* algorithm, explained in Chapter 7, to provide on-demand routing. AODV offers quick convergence when a network topology changes because of any node movement or link breakage. In such cases, AODV notifies all nodes so that they can invalidate the routes using the lost node or link. This protocol adapts quickly to dynamic link conditions and offers low processing, memory overhead, and network utilization. Loop-free AODV is self-starting and handles large numbers of mobile nodes. It allows mobile nodes to respond to link breakages and changes in network topology in a timely manner. The algorithm's primary features are as follows.

- It broadcasts packets only when required.
- It distinguishes between local connectivity management and general maintenance.
- It disseminates information about changes in local connectivity to neighboring mobile nodes that need this information.
- Nodes that are not part of active paths neither maintain any routing information nor participate in any periodic routing table exchanges.

- A node does not have to find and maintain a route to another node until the two nodes communicate. Routes are maintained on all nodes on an active path. For example, all transmitting nodes maintain the route to the destination.

AODV can also form multicast trees that connect multicast group members. The trees are composed of group members and the nodes needed to connect them. This is similar to multicast protocols, explained in Chapter 15.

### Routing Process

A route is *active* as long as data packets periodically travel from a source to a destination along that path. In other words, an active route from a source to a destination has a valid entry in a routing table. Packets can be forwarded only on active routes. Each mobile node maintains a *routing table entry* for a potential destination. A routing table entry contains

- Active neighbors for a requested route
- Next-hop address
- Destination address
- Number of hops to destination
- Sequence number for the destination
- Expiration time for the route table entry (timeout)

Each routing table entry maintains the addresses of the active neighbors so that all active source nodes can be notified when a link along the route to the destination breaks. For each valid route, the node also stores a list of precursors that may transmit packets on this route. These precursors receive notifications from the node when it detects the loss of the *next-hop* link.

Any route in the routing table is tagged with the destination's *sequence number*, an increasing number set by a counter and managed by each originating node to ensure the freshness of the reverse route to a source. The sequence number is incremented whenever the source issues a new route-request message. Each node also records information about its neighbors with bidirectional connectivity. The insertion of a sequence number guarantees that no routing loops form even when packets are delivered out of order and under high node mobility. If a new route becomes available to the requested destination, the node compares the destination sequence number of the new incoming route with the one stored in the routing table for the current route. The existing entry

is updated by replacing the current entry with the incoming one if one of the following conditions exist.

- The current sequence number in the routing table is marked invalid.
- The new incoming sequence number is greater than the one stored in the table.
- The sequence numbers are the same, and the new hop count is smaller than the one for the current entry.

Once the source stops sending data packets on an established connection, the routes time out and are eventually deleted from the intermediate-node routing tables. At the reverse-path entry of any routing table, a *request-expiration timer* purges the reverse-path routing entries from the nodes that do not lie on the route from the source to the destination. When an entry is used to transmit a packet, the timeout for the entry is reset to the current time plus the active-route timeout. The routing table also stores the *routing caching time out*, which is the time after which the route is considered to be invalid.

### Route Discovery and Establishment

Suppose that a source node does not have information about a destination node, perhaps because it is unknown to the source node or a previously valid path to the destination expires or is marked invalid. In such cases, the source node initiates a *route-discovery* process to locate the destination. Route discovery is done by broadcasting a *route request* (RREQ) packet to neighbors, which in turn is forwarded to their neighbors until the destination node is reached. If it receives an already processed RREQ, a node discards the RREQ and does not forward it. Neighboring nodes update their information and set up backward pointers to the source node in their route tables. Each neighbor either responds to the RREQ by sending back a route-reply RREP to the source or increases the hop count and broadcasts the RREQ to its own neighbors; in this case, the process continues.

An RREQ packet contains the following information:

- Source address
- A unique RREQ
- Destination address
- Source sequence number
- Destination sequence number
- Hop count
- Lifetime

When an RREQ packet arrives at an intermediate node on a route, the node first updates the route to the previous hop. The node then checks whether the available route is current and completes this check by comparing the destination sequence number in its own route entry to the destination sequence number in the RREQ packet. If the destination sequence number is greater than the one available in the intermediate node's routing table, the intermediate node must not use its recorded route to respond to the RREQ. In this case, the intermediate node rebroadcasts the RREQ to its neighboring node.

On receipt of an RREQ, an intermediate node maintains the address of each neighbor from which the first copy of the packet is received in their route tables and establishes a reverse path. Therefore, if additional copies of the same RREQ are received, they are discarded. When the RREQ reaches the destination node, it responds by sending a *route reply* (RREP) packet back to the neighbor from which the RREQ was first received. As the RREP is routed back, nodes along this reverse path setup forward route entries in their routing tables, which indicates the node from which the RREP came.

Any intermediate node checks whether it has received an RREQ with the same source node IP address and RREQ within at least the last path-discovery time. If such an RREQ has been received, the node drops the newly received RREQ. The reverse-route entries are maintained long enough for the RREQ packet to pass through the network and produce a reply to the sender. For the RREQ that is not dropped, the node increments the hop count and then searches for a reverse route to the source. At this point, a routing timer associated with each route times out and deletes the entry if it has not received any RREP or is not used within a specified time. The protocol uses destination sequence-numbers to ensure that links are free of loops at all times and thus avoids counting to infinity. The destination address field is incremented every time a source issues a new RREQ.

**Example.** Figure 19.5 shows the process of signals with AODV from node 1 to node 8. To establish a connection, source node 1 searches in its table for a valid route to destination node 8. In the figure, an RREQ reaches the destination for the first time through path 1-2-4-6-8. The destination then issues an RREP packet to the source. After a while, the destination receives another RREQ, this time through path 1-3-7-8. The destination evaluates this path, and finds that path 1-3-7-8 is better, and then issues a new RREP packet, telling the source to discard the other reply.

As an RREP packet is propagated back to the source, involving nodes set up forward pointers to the destination. At this time, the hop-count field is incremented at each node. Thus, when the RREP packet reaches the source, the hop count represents the

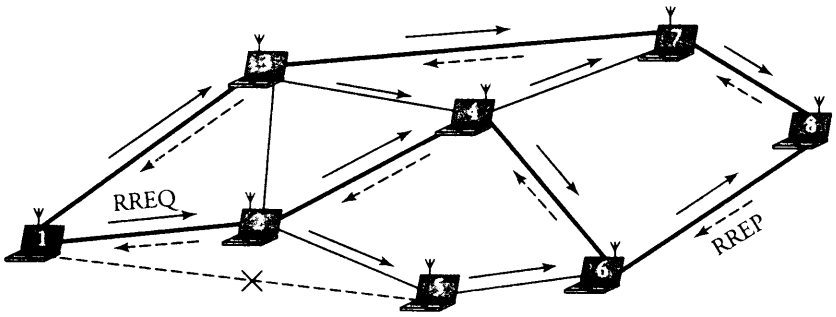


**Figure 19.5** AODV communication signaling from node 1 to node 8

distance of the destination from the originator. The source node starts sending the data packets once the first RREP is received. If it receives an RREP representing a better route to the destination, the source node then updates its routing information. This means that, while transmitting, if it receives a better RREP packet containing a greater sequence number or the same sequence number with a smaller hop count, the source may update its routing information for that destination and switch over to the better path. As a result, a node ignores all less-desired RREPs received while transmitting.

At the reverse-path entry of any routing table, a *request-expiration timer* purges the reverse-path routing entries from the nodes that do not lie on the route from the source to the destination. When an entry is used to transmit a packet, the timeout for the entry is reset to the current time plus the active-route timeout. The routing table also stores the *routing caching timeout*, which is the time after which the route is considered invalid. In Figure 19.6, a timeout has occurred. From the source node 1 to the destination node 8, two routes 1-2-4-6-8 and 1-3-7-8, are found. However, the RREP at the intermediate node 7 exceeds the time allowed to be released. In this case, route 1-3-7-8 is purged from the involving routing tables.

### Route Maintenance

After it knows how to establish a path, a network must maintain it. In general, each forwarding node should keep track of its continued connectivity to its active next hops. If a source node moves, it can reinitiate *route discovery* to find a fresh route to the destination. When a node along the route moves, its upstream neighbor identifies the move and propagates a link-failure notification message to each of its upstream neighbors. These nodes forward the link-failure notification to their neighbors, and so

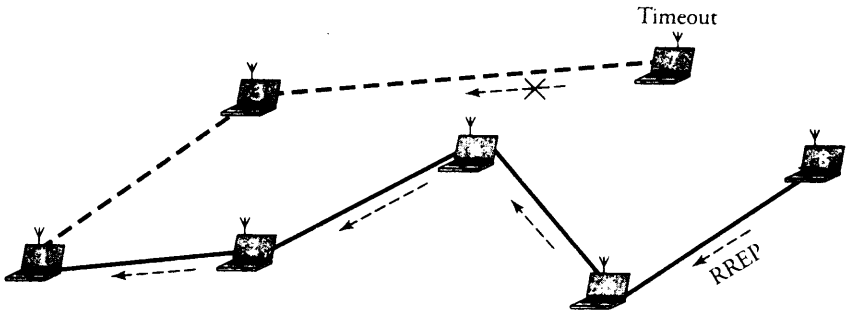


**Figure 19.6** Occurrence of a timeout

on, until the source node is reached. The source node may reinitiate *path-discovery* if a route is still desired.

When the local connectivity of a mobile node is required, each mobile node can get information about other nodes in its neighborhood by using local broadcasts known as HELLO messages. A node should use HELLO messages only if it is part of an active route. A node that does not receive a HELLO message from its neighbors along an active route sends a link-failure notification message to its upstream node on that route.

When it moves during an active session, a source node can start the route-discovery process again to find a new route to the destination node. If the destination or the intermediate node moves, a special RREP is sent to the affected source nodes. Periodic HELLO messages can normally be used to detect link failures. If a link failure occurs while the route is active, the upstream node of the breakage propagates a *route-error* (RERR) message. An RERR message can be either broadcast or unicast. For each node, if a link to the next hop is undetectable, the node should assume that the link is lost and take the following steps.

1. Make all related existing routes invalid.
2. List all destination nodes that would potentially be affected.
3. Identify all neighboring nodes that may be affected.
4. Send an RERR message to all such neighbors.

As shown in Figure 19.1, some next-hop nodes in a network might be unreachable. In such cases, the upstream node of the unreachable node propagates an unsolicited RREP with a fresh sequence number, and the hop count is set to infinity to all active upstream neighbors. Other nodes listen and pass on this message to their active

neighbors until all nodes are notified. AODV finally terminates the unreachable node (broken associated links).

### Joining a New Node to Network

A new node can join an ad-hoc network by transmitting a HELLO message containing its identity and sequence number. When a node receives a HELLO message from a neighbor, the node makes sure that it has an active route to it or, if necessary, creates one. After this update, the current node can begin using this route to forward data packets. In general, nodes in a network may learn about their neighbors when a node receives a normal broadcast message or HELLO message. If the HELLO message is not received from the next hop along an active path, the active neighbors using that next hop are sent notification of a link break.

A node receiving HELLO messages should be part of an active route in order to use them. In every predetermined interval, active nodes in a network check whether they received HELLO messages from their neighbors. If it does not receive any packets within the hello interval, a node broadcasts a HELLO message to all its neighbors. Neighbors that receive this packet update their local connectivity information. Otherwise, if it does not receive any HELLO messages for more than some predetermined time, a node should assume that the link to this neighbor failed.

## 19.4 Security of Ad-Hoc Networks

Because of dynamic topological changes, ad-hoc networks are vulnerable at the physical link, as they can easily be manipulated. An intruder can easily attack ad-hoc networks by loading available network resources, such as wireless links and energy (battery) levels of other users, and then disturb all users. Attackers can also disturb the normal operation of routing protocols by modifying packets. The intruder may insert spurious information into routing packets, causing erroneous routing table updates and thus misrouting. Some other security vulnerabilities of ad-hoc networks follow.

- *Limited computational capabilities*. Typically, nodes in ad-hoc networks are modular, independent, and limited in computational capability and therefore may become a source of vulnerability when they handle public-key cryptography during normal operation.
- *Limited power supply*. Since nodes normally use battery as power supply, an intruder can exhaust batteries by creating additional transmissions or excessive computations to be carried out by nodes.

- *Challenging key management*. Dynamic topology and movement of nodes in an ad-hoc network make *key management* difficult if cryptography is used in the routing protocol.

In any network, routing information can give an attacker access to relationships among nodes and their IP addresses. Especially in ad-hoc networks, an attacker may be able to bring the network down.

## 19.4.1 Types of Attacks

Attacks in ad-hoc networks are either *passive* or *active*. In a passive attack, the normal operation of a routing protocol is not interrupted. Instead, an intruder tries to gather information by listening. Active attacks can sometimes be detectable and thus are less important. In an active attack, an attacker can insert some arbitrary packets of information into the network to disable it or to attract packets destined to other nodes.

### Pin Attack

With the *pin*, or *black-hole*, *attack*, a malicious node pretends to have the shortest path to the destination of a packet. Normally, the intruder listens to a path set-up phase and, when learns of a request for a route, sends a reply advertising a shortest route. Then, the intruder can be an official part of the network if the requesting node receives its malicious reply before the reply from a good node, and a forged route is set up. Once it becomes part of the network, the intruder can do anything within the network, such as undertaking a denial-of-service attack.

### Location-Disclosure Attack

By learning the locations of intermediate nodes, an intruder can find out the location of a target node. The *location-disclosure attack* is made by an intruder to obtain information about the physical location of nodes in the network or the topology of the network.

### Routing Table Overflow

Sometimes, an intruder can create routes whose destinations do not exist. This type of attack, known as the *routing table overflow*. overwhelms the usual flow of traffic, as it creates too many dummy active routes. This attack has a profound impact on *proactive* routing protocols, which discover routing information before it is needed, but minimal impact on *reactive routing protocols*, which create a route only when needed.

**Energy-Exhaustion Attack**

Battery-powered nodes can conserve their power by transmitting only when needed. But an intruder may try to forward unwanted packets or request repeatedly fake or unwanted destinations to use up the energy of nodes' batteries.

## 19.4.2 Criteria for a Secure Routing Protocol

In order to prevent ad-hoc networks from attacks and vulnerability, a routing protocol must possess the following properties:

- *Authenticity*. When a routing table is updated, it must check whether the updates were provided by authenticated nodes and users. The most challenging issue in ad-hoc networks is the lack of a centralized authority to issue and validate certificates of authenticity.
- *Integrity of information*. When a routing table is updated, the information carried to the routing updates must be checked for eligibility. A misleading update may alter the flow of packets in the network.
- *In-order updates*. Ad-hoc routing protocols must contain unique sequence numbers to maintain updates in order. Out-of-order updates may result in the propagation of wrong information.
- *Maximum update time*. Updates in routing tables must be done in the shortest possible time to ensure the credibility of the update information. A timestamp or time-out mechanism can normally be a solution.
- *Authorization*. An unforgeable credential along with the certificate authority issued to a node can determine all the privileges that the node can have.
- *Routing encryption*. Encrypting packets can prevent unauthorized nodes from reading them, and only those routers having the decryption key can access messages.
- *Route discovery*. It should always be possible to find any existing route between two points in a network.
- *Protocol immunization*. A routing protocol should be immune to intruding nodes and be able identify them.
- *Node-privacy location*. The routing protocol must protect the network from spreading the location or other unpublic information of individual nodes.
- *Self-stabilization*. If the *self-stabilization* property of ad-hoc network performs efficiently, it must stabilize the network in the presence of damages continually received from malicious nodes.

- *Low computational load*. Typically, an ad hoc node is limited in powers as it uses a battery. As a result, a node should be given the minimal computational load to maintain enough power to avoid any denial-of-service attacks from low available power.

## 19.5 Summary

A wireless ad-hoc network supports "independent" wireless and mobile communication systems. A mobile user in fact acts as a routing node. Routing protocols can be *centralized versus distributed*, *static versus adaptive*, and *table driven versus source initiated routing*.

Table-driven routing protocols find routes to all possible destinations before they are needed. The routes are recorded in nodes' routing tables and are updated within predefined intervals. Examples of such protocols are DSDV, CGSR, and WRP. CGSR had a better converging capability than others do. Source-initiated routing protocols, such as DSR, ABR, TORA, and AODV, create routes only when they are requested by source nodes. AODV is very popular owing to its ability to stabilize the routing and its better security. Security is a critical issue in ad-hoc networks. Security vulnerability to various types of attacks can be minimized by meeting specific security criteria.

The next chapter explores a special version of ad-hoc network: *sensor network*.

## 19.6 Exercises

1. Compare table-driven and source-initiated routing algorithms in terms of
   (a) Speed of routing table updates
   (b) Point-to-point routing when there is a sudden change in the network topology
2. Suppose that we have the ad-hoc network shown in Figure 19.7. The number on each link indicates the strength of the signal on that link. At time $t_1$, the threshold value of 2.5 is applied on all links as the minimum good connection. Use the DSDV protocol to
   (a) Show the content of the routing table for each node for $t < t_1$
   (b) Perform the routing algorithm to connect A to F
   (c) Show the content of the routing table for each node in $t \geq t_1$
3. With the same conditions stated in exercise 2, use the AODV protocol to
   (a) Show the content of the routing table for each node for $t < t_1$

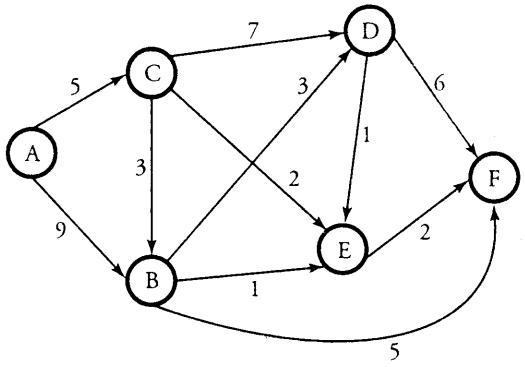


**Figure 19.7** Exercises 2 and 3 ad-hoc network

(b) Show the details of routing to connect A to F, including the communications with neighboring nodes

(c) Show all the detailed steps of updating the neighboring nodes for each node in $t \geq t_1$

4. Consider an ad-hoc network consisting of users spread randomly in locations with Poisson distribution, where there are $n$ users per square meter. Each user transmits with 40 percent probability and receives data with 60 percent probability in an assigned time slot. Make an assumption that the path loss exponent is $\beta$.

   (a) Find the distribution of interference power received in any randomly selected user.

   (b) Show that if $\beta > 2$, the average interference power is finite.

5. Consider Figure 19.3. We want to make a more detailed assignment to each node for routing from node A to F, using CGSR. Assume that the energy of cluster heads is normalized to A = 23, B = 18, C = 15, D = 16, E = 25, and F = 14. Find the best path from A to F. Assume that cluster heads have bidirectional links.

6. *Computer simulation project*. Consider the ad-hoc network shown in Figure 19.5. We want to implement AODV on this network. We first apply distance-vector routing to discover routes on an interconnected network. The primary distance vector routing uses the Bellman-Ford algorithm. Consider the network to be subject to topological changes at any time. You assign these times. Changes include a link to fail unexpectedly or a new link to be created or added. Let your program discover these changes, automatically update nodes' routing tables, and propagate these changes to the other nodes.

7. *Computer simulation project*. Now consider the network you analyzed in exercise 6. We want to test the AODV route-discovery procedure and the ability of AODV to handle link and node failures. At the network level, design a least-cost algorithm that gives the shortest path, taking failing nodes (malicious nodes) into account. If one of the nodes in the shortest path is a bad one, the next available shortest path must be found in the algorithm. In the algorithm you develop, the failure of a link should not affect the shortest-path determination. If a malicious node is in the network, that node is being discarded, and fresh routes are determined that have shortest path to the destination requested by the source node in the network. Assign faults deterministically, starting from the best path, and determine the shortest paths available. Determine how the best paths vary when the number of failing nodes is assigned randomly. Study how the network topology changes for a given $n$ nodes with $k$ failing nodes in the network. You will see that the total costs of the shortest paths between the sources and the destinations gradually increase with the number of fault nodes in the network if the fault nodes are assigned starting from the best path.

# CHAPTER 20
# Wireless Sensor Networks

*Self-organizing sensor networks* hold the potential to revolutionize many segments of public security, environmental monitoring, and manufacturing. Sensors can be networked to form a network to enhance sensing capability. Like a computer network, a sensor network has a packet with a header flowing in a network of nodes (sensor nodes). This chapter presents the architectures and protocols of such networks, discussing the following topics:

- *Sensor networks and protocol structures*
- *Communication energy model*
- *Clustering protocols*
- *Routing protocols*
- *Case study: Simulation of a sensor network*
- *Other related technologies*

We begin with an overview of sensor networks and explain some popular applications. We also provide an overview of a protocol stack for sensor networks and explain how the power factor distinguishes the routing protocols of sensor networks from those of computer networks. The protocol stack combines power efficiency and least-cost-path routing. Networking protocols and power efficientcy are integrated through the wireless medium, and cooperative efforts of sensor nodes are promoted.

*Clustering protocols* in sensor networks specify the topology of the hierarchical network partitioned into nonoverlapping *clusters* of sensor nodes. Typically, a robust clustering technique is essential for self-organizing sensor networks. Two clustering protocols are the *Low-Energy Adaptive Clustering Hierarchy* (LEACH) algorithm and the *Decentralized Energy-Efficient Cluster Propagation* (DEEP) protocol. After well-distributed cluster heads and clusters have been established in a network, energy-conscious routing is essential in order to set communication routes among cluster heads.

This last chapter also offers a detailed numerical case-study on the implementation of a clustering protocol, as well as a discussion of *ZigBee technology*, which is based on the IEEE 802.15.4 standard. This technology uses low-power nodes and is a well-known low-power standard.

## 20.1 Sensor Networks and Protocol Structures

Chemical, biological, or solar sensors can be networked together as a *sensor network* to strengthen the power of sensing. A sensor network is controlled through a software core engine. The network is typically wireless but may also be wired. Sensor networks are designed to be self-configuring such that they can gather information about a large geographical area or about movements of an object for surveillance purposes.

Sensor networks can be used for target tracking, environmental monitoring, system control, and chemical or biological detection. In military applications, sensor networks can enable soldiers to see around corners and to detect chemical and biological weapons long before they get close enough to cause harm. Civilian uses include environmental monitoring, traffic control, and providing health care monitoring for the elderly while allowing them more freedom to move about.

### 20.1.1 Clustering in Sensor Networks

The region being sensed is normally partitioned into equally loaded *clusters* of sensor nodes, as shown in Figure 20.1. A cluster in a sensor network resembles a domain in a computer network. In other words, nodes are inserted in the vicinity of a certain predefined region, forming a cluster. Different types of sensors can also be deployed in a region. Thus, a sensor network is typically cluster based and has irregular topology. The most effective routing scheme in sensor networks is normally based on the energy (battery level) of nodes. In such routing schemes, the best path has the highest amount of total energy. The network of such sensing nodes is constructed with identical sensor nodes, regardless of the size of the network. In Figure 20.1, three clusters are

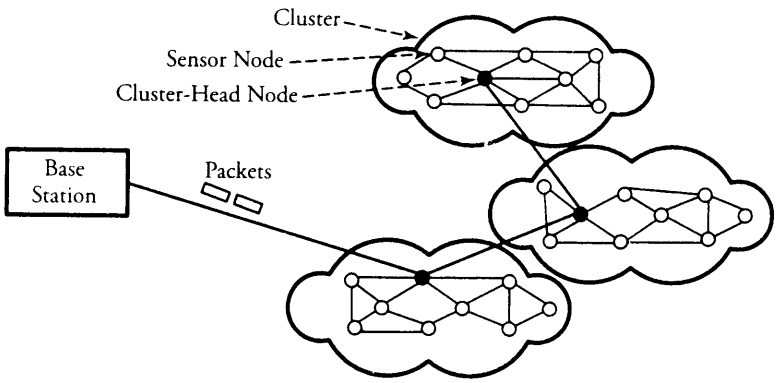


**Figure 20.1** A sensor network and its clusters

interconnected to the main base station, each cluster contains a *cluster head* responsible for routing data from its corresponding cluster to a *base station*.

Communicating nodes are normally linked by a wireless medium, such as radio. The wireless sensor node is equipped with a limited power source, such as a battery or even a *solar cell*, where there is enough sunlight exposure on the node. However, a solar cell may not be the best choice as a power supply, owing to its weight, volume, and expense. In some application scenarios, sensor-node lifetime depends on the battery lifetime. Removal of dead nodes can cause significant topological changes and may require packet rerouting. As a result, power management is a key issue in system design, node design, and communication protocol development. In summary, efficient energy-conscious clustering and routing algorithms can potentially prolong the network lifetime.

## 20.1.2 Protocol Stack

The algorithms developed for wireless ad-hoc networks cannot be used for sensor networks, for several reasons. One is that the number of sensor nodes is typically much more than in a typical ad-hoc network, and sensor nodes, unlike ad-hoc nodes, are prone to permanent failure. In addition, sensor nodes normally use broadcast rather than point-to-point communication with its limited power and memory. Unlike computer networks, sensor nodes do not have global ID, since a typical packet overhead can be too large for them.

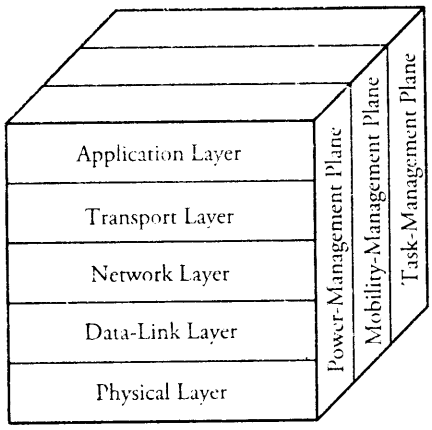


**Figure 20.2** Sensor network protocol stack architecture

Figure 20.2 shows a protocol architecture for sensor networks. The protocol stack combines power efficiency and least-cost-path routing. This protocol architecture integrates networking protocols and power through the wireless medium and promotes cooperative efforts of sensor nodes. The protocol stack consists of the physical layer, data-link layer, network layer, transport layer, and application layer, backed by a power-management plane, mobility-management plane, and task-management plane. The physical layer is responsible for robust modulation, transmission, and receiving signals. *Media access control* (MAC) at the data-link layer must minimize packet collision with neighboring nodes, as power is a restricted factor. The network layer routes packets provided by the transport layer. The application layer uses software for preparation of data on an event. The power-management plane monitors the sensor's power level among the sensor nodes and manages the amount of power a sensor node has used.

Most of the sensor network routing techniques and sensing tasks require an accurate knowledge of location. Thus, a sensor node commonly has a location-finding system. A mobilizer may sometimes be needed to move sensor nodes to carry out assigned tasks. Sensor network routing protocols must be capable of self-organizing. For these purposes, a series of energy-aware MAC, routing, and clustering protocols have been developed for wireless sensor networks. Most of the energy-aware MAC protocols aim to either *adjust the transmission power* or *keep transceivers off as long as possible*.

### 20.1.3 Sensor Node Structure

Figure 20.3 shows a typical sensor node. A node consists mainly of a sensing unit, a processing unit and memory, a self-power unit, and a wireless transceiver component,

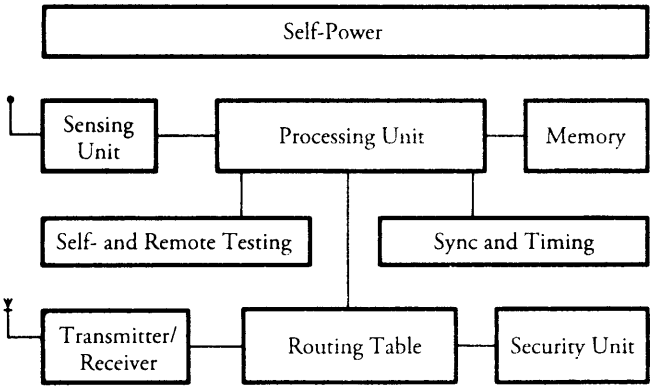


**Figure 20.3** A typical wireless sensor node

as well as a self- and remote-testing unit, a synchronizing and timing unit, a routing table, and security units. Since nodes in a network are not physically accessible once they are deployed in the field, they are not worth being brought under test. An option is an on-board remote self-testing unit for the node on a routine basis.

Each node must determine its location. This task is carried out by a location-finding system based on the *global positioning system* (GPS). All the processes within the sensor node are synchronized by a local clocking and synchronizing system. The communication and security protocol units are in fact part of the processing unit. These two units are responsible for computing the best path for networking and security of the data being transmitted. The three main blocks of the sensor node—sensing unit, processing and memory unit, and power unit—are described in more detail in the following subsections.

### Sensing Unit

The sensing unit consists of a sensor and an analog-to-digital converter. A smart sensor node consists of a combination of multiple sensors. The analog signals produced by the sensors, based on the observed event, are converted to digital signals by the converter and then fed into the processing unit. The sensing unit collects data externally and interacts with the central processor at the heart of the node.

### Processing and Memory Unit

The *processing unit* performs certain computations on the data and, depending on how it is programmed, may send the resulting information out to the network. The processing unit, which is generally associated with memory, manages the procedures

that make the sensor node collaborate with the other nodes to carry out the assigned sensing task. The central processor determines what data needs to be analyzed, stored, or compared with the data stored in memory. The streamed data from the sensor input is processed as it arrives. The database in memory stores an indexed data list to be used as a reference to detect an event. Since sensing nodes are typically tiny and many nodes are engaged in a network, the communication structure makes use of a hierarchically arranged self-routing network through cluster heads.

In smart wireless sensor networks, a tiny processor and a tiny database are used in a node. Thousands of such nodes are spread on fields to power up the sensing task, as in the deployment of numerous small intelligent sensor nodes in a battlefield to monitor enemy movements. By inserting self-organizing capability into a sensor network, a smart node can extract data, compare it with the data stored in its memory database, and process it for relevance before relaying it to its central base station.

### Self-Power Unit

A sensor node is supposed to be mounted in a small physical unit, limiting space for the battery. Moreover, the random distribution of sensors makes it impossible to periodically recharge or exchange batteries. In most types of sensor networks, the power unit in a sensor node is the most important unit of the node because the liveliness and existence of a node depend on the energy left in the node, and the routing in the sensor network is based on the algorithm that finds a path with the most energy. Thus, it is essential to use energy-efficient algorithms to prolong the life of sensor networks. The main task of the sensor node is to identify events, to process data, and then to transmit the data. The power of a node is consumed mainly in the transmitter and receiver unit. The sensor node can be supplied by a *self-power unit*, self-power unitbattery, or solar cells.

## 20.2 Communication Energy Model

IEEE standardsas 802.11a, b, and g provide a wide range of data rates: 54, 48, 36, 24, 18, 12, 9, and 6 Mb/s. This range reflects the trade-off between the transmission range and data rate intrinsic in a wireless communication channel. An accurate energy model is crucial for the development of energy-efficient clustering and routing protocols. The energy consumption, $E$, for all components of the transceiver in watts is summarized as

$$E = \theta + \eta\omega d^n, \tag{20.1}$$

where $\theta$ is the distance-independent term that accounts for the overhead of the radio electronics and digital processing, and $\eta\omega d^n$ is the distance-dependent term, in which