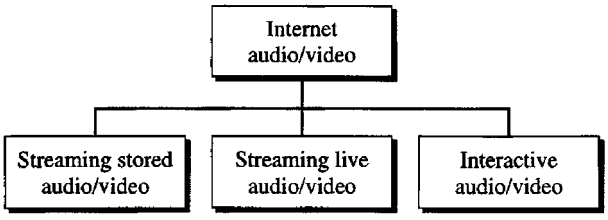
CHAPTER 29

Multimedia

Recent advances in technology have changed our use of audio and video. In the past, we listened to an audio broadcast through a radio and watched a video program broadcast through a TV. We used the telephone network to interactively communicate with another party. But times have changed. People want to use the Internet not only for text and image communications, but also for audio and video services. In this chapter, we concentrate on applications that use the Internet for audio and video services.

We can divide audio and video services into three broad categories: streaming stored audio/video, streaming live audio/video, and interactive audio/video, as shown in Figure 29.1. Streaming means a user can listen to (or watch) the file after the downloading has started.

Figure 29.1 *Internet audio/video*



In the first category, streaming stored audio/video, the files are compressed and stored on a server. A client downloads the files through the Internet. This is sometimes referred to as on-demand audio/video. Examples of stored audio files are songs, symphonies, books on tape, and famous lectures. Examples of stored video files are movies, TV shows, and music video clips.

Streaming stored audio/video refers to on-demand requests for compressed audio/video files.

In the second category, streaming live audio/video, a user listens to broadcast audio and video through the Internet. A good example of this type of application is the Internet radio. Some radio stations broadcast their programs only on the Internet; many broadcast them both on the Internet and on the air. Internet TV is not popular yet, but many people believe that TV stations will broadcast their programs on the Internet in the future.

Streaming live audio/video refers to the broadcasting of radio and TV programs through the Internet.

In the third category, interactive audio/video, people use the Internet to interactively communicate with one another. A good example of this application is Internet telephony and Internet teleconferencing.

Interactive audio/video refers to the use of the Internet for interactive audio/video applications.

We will discuss these three applications in this chapter, but first we need to discuss some other issues related to audio/video: digitizing audio and video and compressing audio and video.

29.1 DIGITIZING AUDIO AND VIDEO

Before audio or video signals can be sent on the Internet, they need to be digitized. We discuss audio and video separately.

Digitizing Audio

When sound is fed into a microphone, an electronic analog signal is generated which represents the sound amplitude as a function of time. The signal is called an *analog audio signal*. An analog signal, such as audio, can be digitized to produce a digital signal. According to the Nyquist theorem, if the highest frequency of the signal is f, we need to sample the signal $2f$ times per second. There are other methods for digitizing an audio signal, but the principle is the same.

Voice is sampled at 8000 samples per second with 8 bits per sample. This results in a digital signal of 64 kbps. Music is sampled at 44,100 samples per second with 16 bits per sample. This results in a digital signal of 705.6 kbps for monaural and 1.411 Mbps for stereo.

Digitizing Video

A video consists of a sequence of frames. If the frames are displayed on the screen fast enough, we get an impression of motion. The reason is that our eyes cannot distinguish the rapidly flashing frames as individual ones. There is no standard number of frames per second; in North America 25 frames per second is common. However, to avoid a

condition known as flickering, a frame needs to be refreshed. The TV industry repaints each frame twice. This means 50 frames need to be sent, or if there is memory at the sender site, 25 frames with each frame repainted from the memory.

Each frame is divided into small grids, called picture elements or pixels. For black-and-white TV, each 8-bit pixel represents one of 256 different gray levels. For a color TV, each pixel is 24 bits, with 8 bits for each primary color (red, green, and blue).

We can calculate the number of bits in 1 s for a specific resolution. In the lowest resolution a color frame is made of 1024 x 768 pixels. This means that we need

$$2 \times 25 \times 1024 \times 768 \times 24 = 944 \text{ Mbps}$$

This data rate needs a very high data rate technology such as SONET. To send video using lower-rate technologies, we need to compress the video.

Compresswn is needed to send video over the Internet.

29.2 AUDIO AND VIDEO COMPRESSION

To send audio or video over the Internet requires compression. In this section, we discuss audio compression first and then video compression.

Audio Compression

Audio compression can be used for speech or music. For speech, we need to compress a 64-kHz digitized signal; for music, we need to compress a 1.41 I-MHz signal. Two categories of techniques are used for audio compression: predictive encoding and perceptual encoding.

Predictive Encoding

In predictive encoding, the differences between the samples are encoded instead of encoding all the sampled values. This type of compression is normally used for speech. Several standards have been defined such as GSM (13 kbps), G.729 (8 kbps), and G.723.3 (6.4 or 5.3 kbps). Detailed discussions of these techniques are beyond the scope of this book.

Perceptual Encoding: MP3

The most common compression technique that is used to create CD-quality audio is based on the perceptual encoding technique. As we mentioned before, this type of audio needs at least 1.411 Mbps; this cannot be sent over the Internet without compression. MP3 (MPEG audio layer 3), a part of the MPEG standard (discussed in the video compression section), uses this technique.

Perceptual encoding is based on the science of psychoacoustics, which is the study of how people perceive sound. The idea is based on flaws in our auditory system: Some sounds can mask other sounds. Masking can happen in frequency and time. In

frequency masking, a loud sound in a frequency range can partially or totally mask a softer sound in another frequency range. For example, we cannot hear what our dance partner says in a room where a loud heavy metal band is performing. In temporal masking, a loud sound can numb our ears for a short time even after the sound has stopped.

MP3 uses these two phenomena, frequency and temporal masking, to compress audio signals. The technique analyzes and divides the spectrum into several groups. Zero bits are allocated to the frequency ranges that are totally masked. A small number of bits are allocated to the frequency ranges that are partially masked. A larger number of bits are allocated to the frequency ranges that are not masked.

MP3 produces three data rates: 96 kbps, 128 kbps, and 160 kbps. The rate is based on the range of the frequencies in the original analog audio.

Video Compression

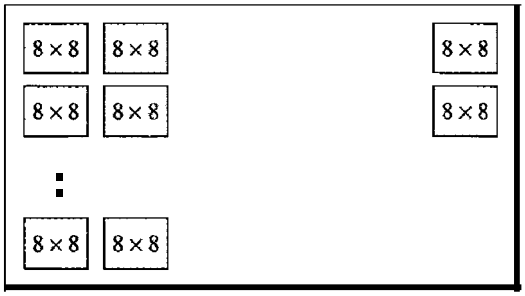
As we mentioned before, video is composed of multiple frames. Each frame is one image. We can compress video by first compressing images. Two standards are prevalent in the market. Joint Photographic Experts Group (JPEG) is used to compress images. Moving Picture Experts Group (MPEG) is used to compress video. We briefly discuss JPEG and then MPEG.

Image Compression: lPEG

As we discussed previously, if the picture is not in color (gray scale), each pixel can be represented by an 8-bit integer (256 levels). If the picture is in color, each pixel can be represented by 24 bits (3 x 8 bits), with each 8 bits representing red, blue, or green (RBG). To simplify the discussion, we concentrate on a gray scale picture.

In JPEG, a gray scale picture is divided into blocks of 8 x 8 pixels (see Figure 29.2).

Figure 29.2 *lPEG gray scale*

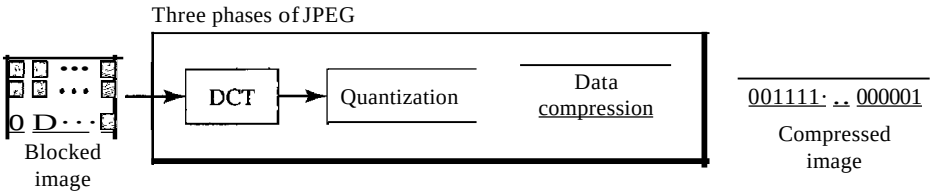


The purpose of dividing the picture into blocks is to decrease the number of calculations because, as you will see shortly, the number of mathematical operations for each picture is the square of the number of units.

The whole idea of JPEG is to change the picture into a linear (vector) set of numbers that reveals the redundancies. The redundancies (lack of changes) can then be removed

by using one of the text compression methods. A simplified scheme of the process is shown in Figure 29.3.

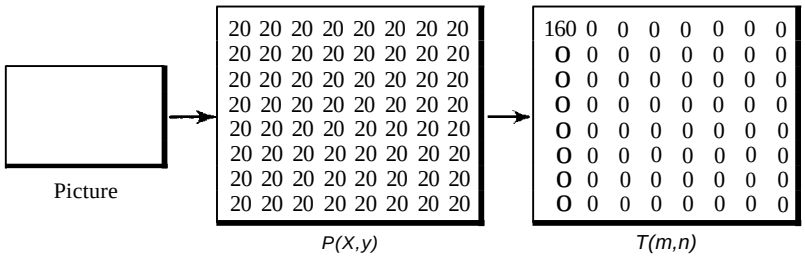
Figure 29.3 *lPEG process*



Discrete Cosine Transform (DCT) In this step, each block of 64 pixels goes through a transformation called the discrete cosine transform (DCT). The transformation changes the 64 values so that the relative relationships between pixels are kept but the redundancies are revealed. We do not give the formula here, but we do show the results of the transformation for three cases.

Case 1 In this case, we have a block of uniform gray, and the value of each pixel is 20. When we do the transformations, we get a nonzero value for the first element (upper left comer); the rest of the pixels have a value of 0. The value of *T(O,O)* is the average (multiplied by a constant) of the *P(x,y)* values and is called the *de value* (direct current, borrowed from electrical engineering). The rest of the values, called *ae values,* in *T(m,n)* represent changes in the pixel values. But because there are no changes, the rest of the values are 0s (see Figure 29.4).

Figure 29.4 *Case* 1: *uniform gray scale*



Case 2 In the second case, we have a block with two different uniform gray scale sections. There is a sharp change in the values of the pixels (from 20 to 50). When we do the transformations, we get a dc value as well as nonzero ac values. However, there are only a few nonzero values clustered around the dc value. Most of the values are 0 (see Figure 29.5).

Case 3 In the third case, we have a block that changes gradually. That is, there is no sharp change between the values of neighboring pixels. When we do the transformations, we get a de value, with many nonzero ac values also (Figure 29.6).

Figure 29.5 *Case* 2: *two sections*

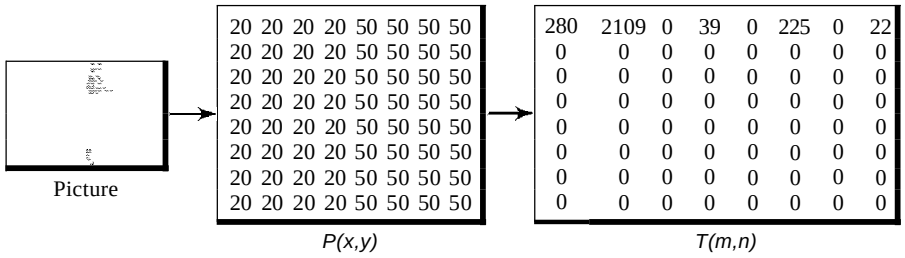
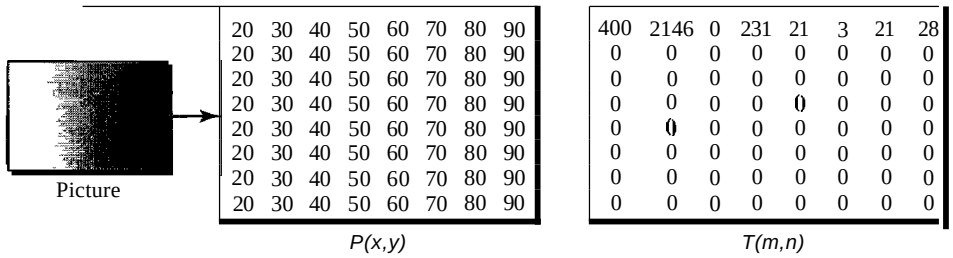


Figure 29.6 *Case* 3: *gradient gray scale*



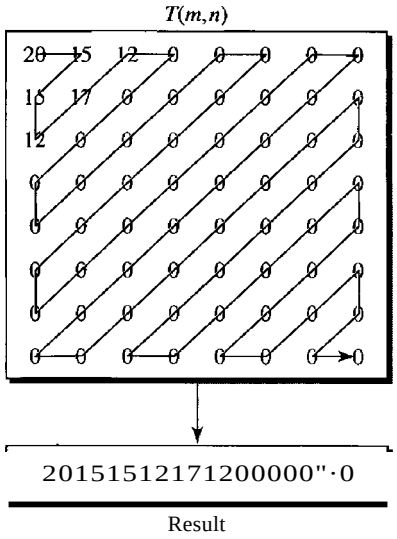
From Figures 29.4,29.5, and 29.6, we can state the following:

- The transformation creates table *T* from table *P.*
- The dc value is the average value (multiplied by a constant) of the pixels.
- The ac values are the changes.
- Lack of changes in neighboring pixels creates Os.

Quantization Mter the Ttable is created, the values are quantized to reduce the number of bits needed for encoding. Previously in quantization, we dropped the fraction from each value and kept the integer part. Here, we divide the number by a constant and then drop the fraction. This reduces the required number of bits even more. In most implementations, a quantizing table (8 x 8) defines how to quantize each value. The divisor depends on the position of the value in the *T* table. This is done to optimize the number of bits and the number of Os for each particular application. Note that the only phase in the process that is not reversible is the quantizing phase. We lose some information here that is not recoverable. As a matter of fact, the only reason that lPEG is called *lossy compression* is because of this quantization phase.

Compression After quantization, the values are read from the table, and redundant Os are removed. However, to cluster the Os together, the table is read diagonally in a zigzag fashion rather than row by row or column by column. The reason is that if the picture changes smoothly, the bottom right corner of the *T* table is all Os. Figure 29.7 shows the process.

Figure 29.7 *Reading the table*



Video Compression: MPEG

The Moving Picture Experts Group method is used to compress video. In principle, a motion picture is a rapid flow of a set of frames, where each frame is an image. In other words, a frame is a spatial combination of pixels, and a video is a temporal combination of frames that are sent one after another. Compressing video, then, means spatially compressing each frame and temporally compressing a set of frames.

Spatial Compression The spatial compression of each frame is done with IPEG (or a modification of it). Each frame is a picture that can be independently compressed.

Temporal Compression In temporal compression, redundant frames are removed. When we watch television, we receive 50 frames per second. However, most of the consecutive frames are almost the same. For example, when someone is talking, most of the frame is the same as the previous one except for the segment of the frame around the lips, which changes from one frame to another.

To temporally compress data, the MPEG method first divides frames into three categories: I-frames, P-frames, and B-frames.

- I-frames. An intracoded frame (I-frame) is an independent frame that is not related to any other frame (not to the frame sent before or to the frame sent after). They are present at regular intervals (e.g., every ninth frame is an I-frame). An I-frame must appear periodically to handle some sudden change in the frame that the previous and following frames cannot show. Also, when a video is broadcast, a viewer may tune in at any time. If there is only one I-frame at the beginning of the broadcast, the viewer who tunes in late will not receive a complete picture. I-frames are independent of other frames and cannot be constructed from other frames.
- P-frames. A predicted frame (P-frame) is related to the preceding I-frame or P-frame. In other words, each P-frame contains only the changes from the preceding

frame. The changes, however, cannot cover a big segment. For example, for a fast-moving object, the new changes may not be recorded in a P-frame. P-frames can be constructed only from previous 1- or P-frames. P-frames carry much less information than other frame types and carry even fewer bits after compression.

- B-frames. A bidirectional frame (B.frame) is related to the preceding and following I-frame or P-frame. In other words, each B-frame is relative to the past and the future. Note that a B-frame is never related to another B-frame.

Figure 29.8 shows a sample sequence of frames.

Figure 29.8 *MPEG frames*

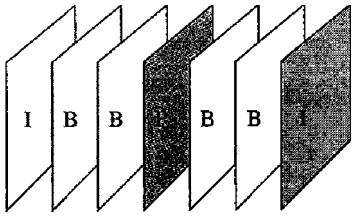
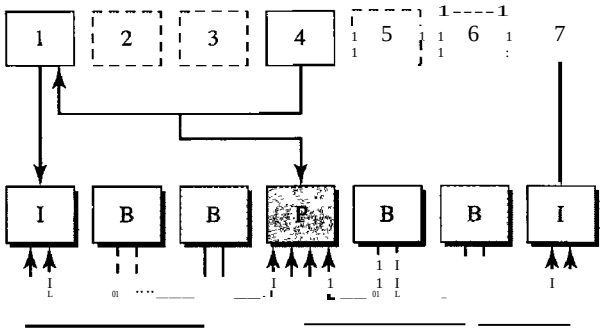


Figure 29.9 shows how 1-, P-, and B-frames are constructed from a series of seven frames.

Figure 29.9 *MPEG frame construction*



MPEG has gone through two versions. MPEG1 was designed for a CD-ROM with a data rate of 1.5 Mbps. MPEG2 was designed for high-quality DVD with a data rate of 3 to 6 Mbps.

29.3 STREAMING STORED AUDIONIDEO

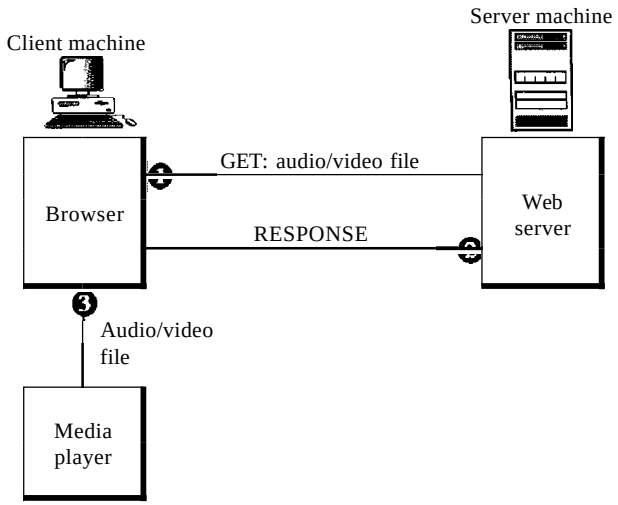
Now that we have discussed digitizing and compressing audio/video, we turn our attention to specific applications. The first is streaming stored audio and video. Downloading these types of files from a Web server can be different from downloading other

types of files. To understand the concept, let us discuss four approaches, each with a different complexity.

First Approach: Using a Web Server

A compressed audio/video file can be downloaded as a text file. The client (browser) can use the services of HTTP and send a GET message to download the file. The Web server can send the compressed file to the browser. The browser can then use a help application, normally called a media player, to play the file. Figure 29.10 shows this approach.

Figure 29.10 *Using a Web server*



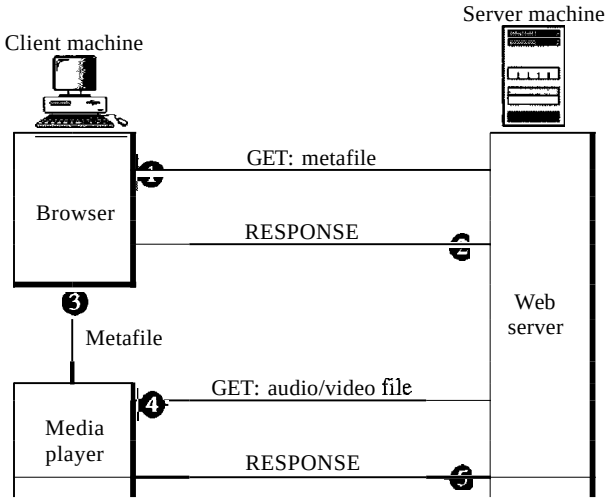
This approach is very simple and does not involve *streaming*. However, it has a drawback. An audio/video file is usually large even after compression. An audio file may contain tens of megabits, and a video file may contain hundreds of megabits. In this approach, the file needs to download completely before it can be played. Using contemporary data rates, the user needs some seconds or tens of seconds before the file can be played.

Second Approach: Using a Web Server with Metafile

In another approach, the media player is directly connected to the Web server for downloading the audio/video file. The Web server stores two files: the actual audio/video file and a metafile that holds information about the audio/video file. Figure 29.11 shows the steps in this approach.

1. The HTTP client accesses the Web server by using the GET message.
2. The information about the metafile comes in the response.
3. The metafile is passed to the media player.
4. The media player uses the URL in the metafile to access the audio/video file.
5. The Web server responds.

Figure 29.11 *Using a Web server with a metafile*

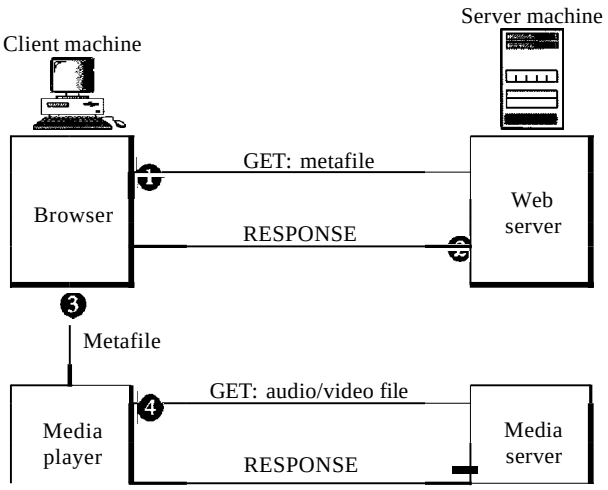


Third Approach: Using a Media Server

The problem with the second approach is that the browser and the media player both use the services of HTIP. HTIP is designed to run over TCP. This is appropriate for retrieving the metafile, but not for retrieving the audio/video file. The reason is that TCP retransmits a lost or damaged segment, which is counter to the philosophy of streaming. We need to dismiss TCP and its error control; we need to use UDP. However, HTTP, which accesses the Web server, and the Web server itself are designed for TCP; we need another server, a media server. Figure 29.12 shows the concept.

1. The HTTP client accesses the Web server by using a GET message.
2. The information about the metafile comes in the response.

Figure 29.12 *Using a media server*

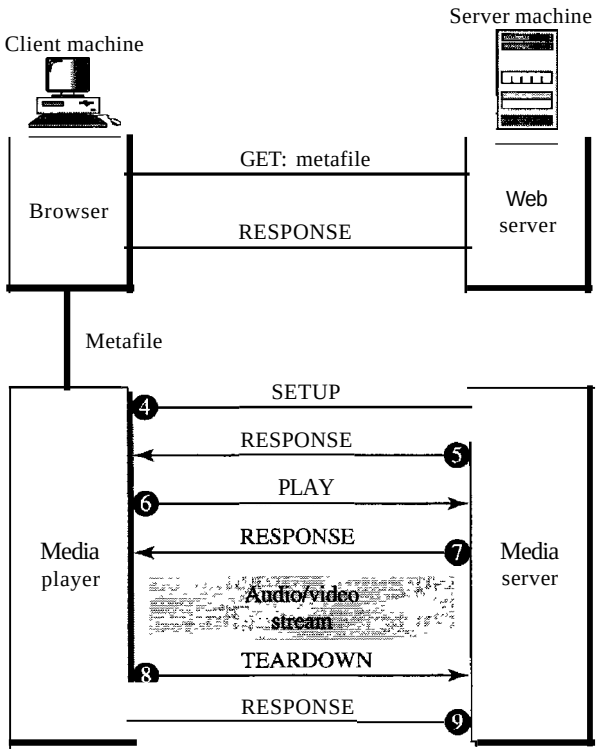


3. The metafile is passed to the media player.
4. The media player uses the URL in the metafile to access the media server to download the file. Downloading can take place by any protocol that uses UDP.
5. The media server responds.

Fourth Approach: Using a Media Server and RTSP

The Real-Time Streaming Protocol (RTSP) is a control protocol designed to add more functionalities to the streaming process. Using RTSP, we can control the playing of audio/video. RTSP is an out-of-band control protocol that is similar to the second connection in FTP. Figure 29.13 shows a media server and RTSP.

Figure 29.13 *Using a media server and RTSP*



1. The HTTP client accesses the Web server by using a GET message.
2. The information about the metafile comes in the response.
3. The metafile is passed to the media player.
4. The media player sends a SETUP message to create a connection with the media server.
5. The media server responds.
6. The media player sends a PLAY message to start playing (downloading).
7. The audio/video file is downloaded by using another protocol that runs over UDP.

8. The connection is broken by using the TEARDOWN message.
9. The media server responds.

The media player can send other types of messages. For example, a PAUSE message temporarily stops the downloading; downloading can be resumed with a PLAY message.

29.4 STREAMING LIVE AUDIONIDEO

Streaming live audio/video is similar to the broadcasting of audio and video by radio and TV stations. Instead of broadcasting to the air, the stations broadcast through the Internet. There are several similarities between streaming stored audio/video and streaming live audio/video. They are both sensitive to delay; neither can accept retransmission. However, there is a difference. In the first application, the communication is unicast and on-demand. In the second, the communication is multicast and live. Live streaming is better suited to the multicast services of IP and the use of protocols such as UDP and RTP (discussed later). However, presently, live streaming is still using TCP and multiple unicasting instead of multicasting. There is still much progress to be made in this area.

29.5 REAL-TIME INTERACTIVE AUDIONIDEO

In real-time interactive audio/video, people communicate with one another in real time. The Internet phone or voice over IP is an example of this type of application. Video conferencing is another example that allows people to communicate visually and orally.

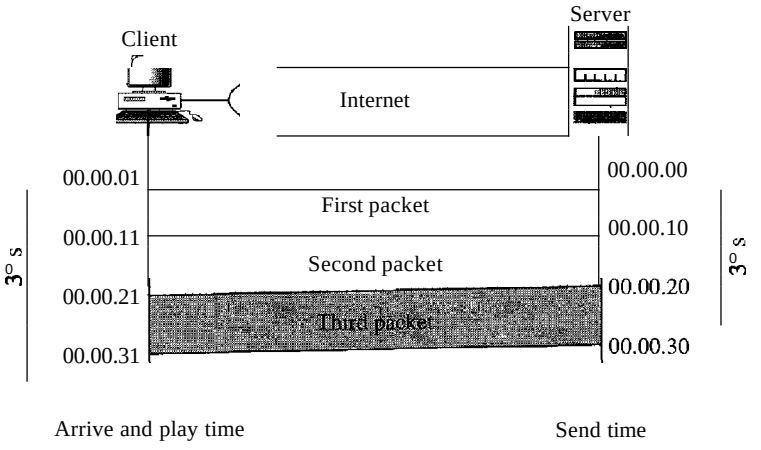
Characteristics

Before addressing the protocols used in this class of applications, we discuss some characteristics of real-time audio/video communication.

Time Relationship

Real-time data on a packet-switched network require the preservation of the time relationship between packets of a session. For example, let us assume that a real-time video server creates live video images and sends them online. The video is digitized and packetized. There are only three packets, and each packet holds lOs of video information. The first packet starts at 00:00:00, the second packet starts at 00:00: 10, and the third packet starts at 00:00:20. Also imagine that it takes 1 s (an exaggeration for simplicity) for each packet to reach the destination (equal delay). The receiver can play back the first packet at 00:00:01, the second packet at 00:00:11, and the third packet at 00:00:21. Although there is a I-s time difference between what the server sends and what the client sees on the computer screen, the action is happening in real time. The time relationship between the packets is preserved. The I-s delay is not important. Figure 29.14 shows the idea.

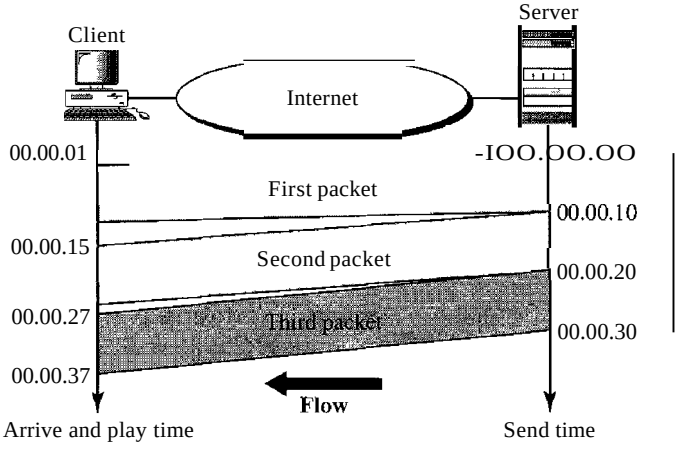
Figure 29.14 *Time relationship*



But what happens if the packets arrive with different delays? For example, say the first packet arrives at 00:00:01 (l-s delay), the second arrives at 00:00: 15 (5-s delay), and the third arrives at 00:00:27 (7-s delay). If the receiver starts playing the first packet at 00:00:01, it will finish at 00:00: 11. However, the next packet has not yet arrived; it arrives 4 s later. There is a gap between the first and second packets and between the second and the third as the video is viewed at the remote site. This phenomenon is called jitter. Figure 29.15 shows the situation.

Jitter is introduced in real-time data by the delay between packets.

Figure 29.15 *Jitter*

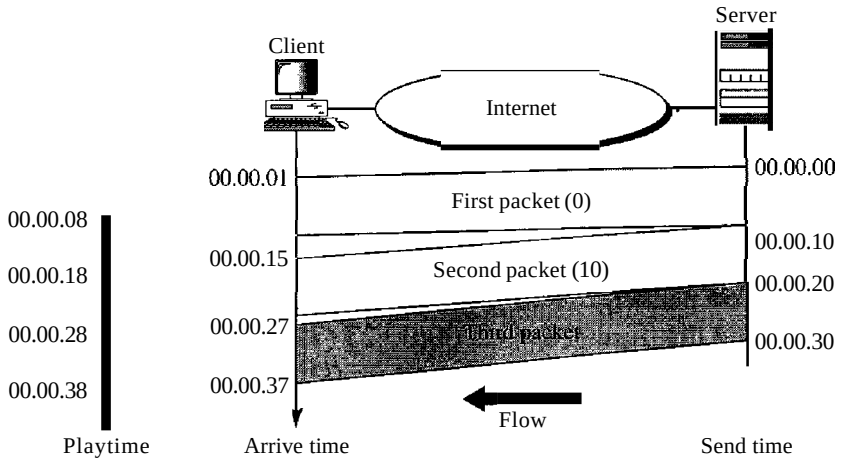


Timestamp

One solution to jitter is the use of a timestamp. If each packet has a timestamp that shows the time it was produced relative to the first (or previous) packet, then the receiver

can add this time to the time at which it starts the playback. In other words, the receiver knows when each packet is to be played. Imagine the first packet in the previous example has a timestamp of 0, the second has a timestamp of 10, and the third has a timestamp of 20. If the receiver starts playing back the first packet at 00:00:08, the second will be played at 00:00: 18 and the third at 00:00:28. There are no gaps between the packets. Figure 29.16 shows the situation.

Figure 29.16 *Timestamp*



To prevent jitter, we can time-stamp the packets and separate the arrival time from the playback time.

Playback Buffer

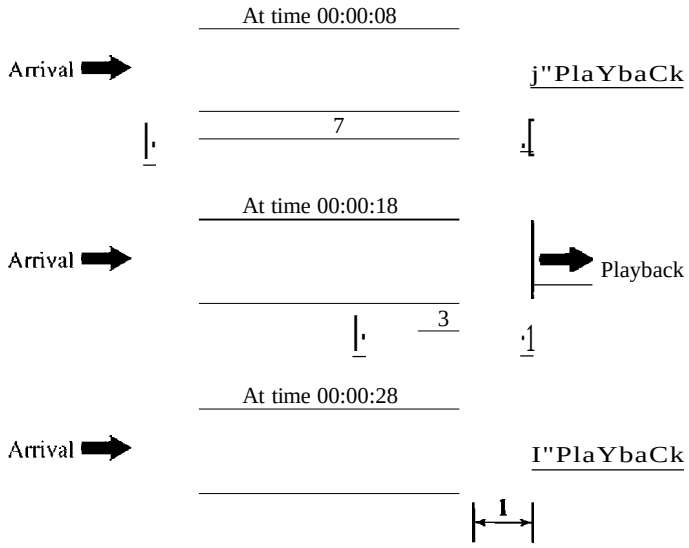
To be able to separate the arrival time from the playback time, we need a buffer to store the data until they are played back. The buffer is referred to as a playback buffer. When a session begins (the first bit of the first packet arrives), the receiver delays playing the data until a threshold is reached. In the previous example, the first bit of the first packet arrives at 00:00:01; the threshold is 7 s, and the playback time is 00:00:08. The threshold is measured in time units of data. The replay does not start until the time units of data are equal to the threshold value.

Data are stored in the buffer at a possibly variable rate, but they are extracted and played back at a fixed rate. Note that the amount of data in the buffer shrinks or expands, but as long as the delay is less than the time to play back the threshold amount of data, there is no jitter. Figure 29.17 shows the buffer at different times for our example.

Ordering

In addition to time relationship information and timestamps for real-time traffic, one more feature is needed. We need a *sequence number* for each packet. The timestamp alone cannot inform the receiver if a packet is lost. For example, suppose the timestamps

Figure 29.17 *Playback buffer*



A playback buffer is required for real-time traffic.

are 0, 10, and 20. If the second packet is lost, the receiver receives just two packets with timestamps 0 and 20. The receiver assumes that the packet with timestamp 20 is the second packet, produced 20 s after the first. The receiver has no way of knowing that the second packet has actually been lost. A sequence number to order the packets is needed to handle this situation.

A sequence number on each packet is required for real-time traffic.

Multicasting

Multimedia playa primary role in audio and video conferencing. The traffic can be heavy, and the data are distributed by using multicasting methods. Conferencing requires two-way communication between receivers and senders.

Real-time traffic needs the support of multicasting.

Translation

Sometimes real-time traffic needs translation. A translator is a computer that can change the format of a high-bandwidth video signal to a lower-quality narrow-bandwidth signal. This is needed, for example, for a source creating a high-quality video signal at 5 Mbps and sending to a recipient having a bandwidth of less than 1 Mbps. To receive the signal, a translator is needed to decode the signal and encode it again at a lower quality that needs less bandwidth.

Translation means changing the encoding of a payload to a lower quality to match the bandwidth of the receiving network.

Mixing

If there is more than one source that can send data at the same time (as in a video or audio conference), the traffic is made of multiple streams. To converge the traffic to one stream, data from different sources can be mixed. A mixer mathematically adds signals coming from different sources to create one single signal.

Mixing means combining several streams of traffic into one stream.

Support from Transport Layer Protocol

The procedures mentioned in the previous sections can be implemented in the application layer. However, they are so common in real-time applications that implementation in the transport layer protocol is preferable. Let's see which of the existing transport layers is suitable for this type of traffic.

TCP is not suitable for interactive traffic. It has no provision for time-stamping, and it does not support multicasting. However, it does provide ordering (sequence numbers). One feature of TCP that makes it particularly unsuitable for interactive traffic is its error control mechanism. In interactive traffic, we cannot allow the retransmission of a lost or corrupted packet. If a packet is lost or corrupted in interactive traffic, it must be ignored. Retransmission upsets the whole idea of time-stamping and playback. Today there is so much redundancy in audio and video signals (even with compression) that we can simply ignore a lost packet. The listener or viewer at the remote site may not even notice it.

TCP, with all its sophistication, is not suitable for interactive multimedia traffic because we cannot allow retransmission of packets.

UDP is more suitable for interactive multimedia traffic. UDP supports multicasting and has no retransmission strategy. However, UDP has no provision for time-stamping, sequencing, or mixing. A new transport protocol, Real-time Transport Protocol (RTP), provides these missing features.

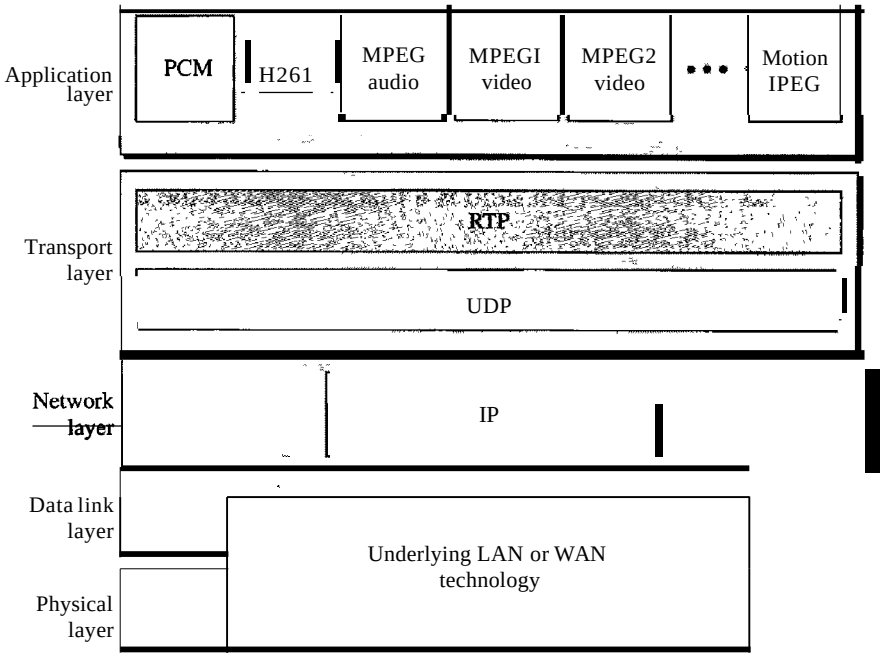
UDP is more suitable than TCP for interactive traffic. However, we need the services of RTP, another transport layer protocol, to make up for the deficiencies of UDP.

29.6 RTP

Real-time Transport Protocol (RTP) is the protocol designed to handle real-time traffic on the Internet. RTP does not have a delivery mechanism (multicasting, port numbers, and so on); it must be used with UDP. RTP stands between UDP and the application

program. The main contributions of RTP are time-stamping, sequencing, and mixing facilities. Figure 29.18 shows the position ofRTP in the protocol suite.

Figure 29.18 *RTP*

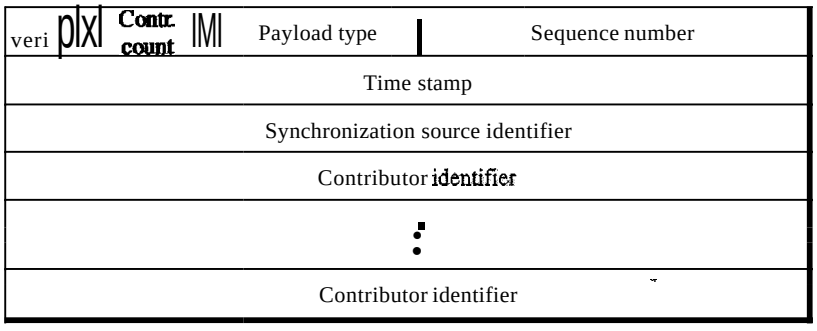


RTP Packet Format

Figure 29.19 shows the format of the RTP packet header. The format is very simple and general enough to cover all real-time applications. An application that needs more information adds it to the beginning of its payload. A description of each field follows.

OVer. This 2-bit field defines the version number. The current version is 2.

Figure 29.19 *RTP packet header format*



- P. This 1-bit field, if set to 1, indicates the presence of padding at the end of the packet. In this case, the value of the last byte in the padding defines the length of the padding. Padding is the norm if a packet is encrypted. There is no padding if the value of the P field is 0.
- X. This 1-bit field, if set to 1, indicates an extra extension header between the basic header and the data. There is no extra extension header if the value of this field is 0.
- Contributor count. This 4-bit field indicates the number of contributors. Note that we can have a maximum of 15 contributors because a 4-bit field only allows a number between 0 and 15.
- M. This 1-bit field is a marker used by the application to indicate, for example, the end of its data.
- Payload type. This 7-bit field indicates the type of the payload. Several payload types have been defined so far. We list some common applications in Table 29.1. A discussion of the types is beyond the scope of this book.

Table 29.1 *Payload types*

Type	*Application*	*Type*	*Application*	*Type*	*Application*
a	PCMμ Audio	7	LPC audio	15	G728 audio
1	1016	8	PCMA audio	26	Motion JPEG
2	G721 audio	9	G722 audio	31	H.261
3	GSM audio	10-11	L16 audio	32	MPEGI video
5-6	DV14 audio	14	MPEGaudio	33	MPEG2 video

- Sequence number. This field is 16 bits in length. It is used to number the RTP packets. The sequence number of the first packet is chosen randomly; it is incremented by 1 for each subsequent packet. The sequence number is used by the receiver to detect lost or out-of-order packets.
- Timestamp. This is a 32-bit field that indicates the time relationship between packets. The timestamp for the first packet is a random number. For each succeeding packet, the value is the sum of the preceding timestamp plus the time the first byte is produced (sampled). The value of the clock tick depends on the application. For example, audio applications normally generate chunks of 160 bytes; the clock tick for this application is 160. The timestamp for this application increases 160 for each RTP packet.
- Synchronization source identifier. If there is only one source, this 32-bit field defines the source. However, if there are several sources, the mixer is the synchronization source and the other sources are contributors. The value of the source identifier is a random number chosen by the source. The protocol provides a strategy in case of conflict (two sources start with the same sequence number).
- Contributor identifier. Each of these 32-bit identifiers (a maximum of 15) defines a source. When there is more than one source in a session, the mixer is the synchronization source and the remaining sources are the contributors.

UDPPort

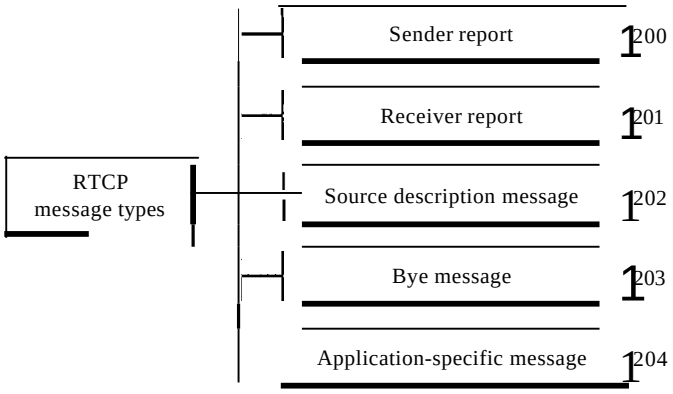
Although RTP is itself a transport layer protocol, the RTP packet is not encapsulated directly in an IP datagram. Instead, RTP is treated as an application program and is encapsulated in a UDP user datagram. However, unlike other application programs, no well-known port is assigned to RTP. The port can be selected on demand with only one restriction: The port number must be an even number. The next number (an odd number) is used by the companion of RTP, Real-time Transport Control Protocol (RTCP).

RTP uses a temporary even-numbered UDP port.

29.7 RTCP

RTP allows only one type of message, one that carries data from the source to the destination. In many cases, there is a need for other messages in a session. These messages control the flow and quality of data and allow the recipient to send feedback to the source or sources. Real-time Transport Control Protocol (RTCP) is a protocol designed for this purpose. RTCP has five types of messages, as shown in Figure 29.20. The number next to each box defines the type of the message.

Figure 29.20 *RTCP message types*



Sender Report

The sender report is sent periodically by the active senders in a conference to report transmission and reception statistics for all RTP packets sent during the interval. The sender report includes an absolute timestamp, which is the number of seconds elapsed since midnight on January 1, 1970. The absolute timestamp allows the receiver to synchronize different RTP messages. It is particularly important when both audio and video are transmitted (audio and video transmissions use separate relative timestamps).

Receiver Report

The receiver report is for passive participants, those that do not send RTP packets. The report infonns the sender and other receivers about the quality of service.

Source Description Message

The source periodically sends a source description message to give additional infonnation about itself. This infonnation can be the name, e-mail address, telephone number, and address of the owner or controller of the source.

Bye Message

A source sends a bye message to shut down a stream. It allows the source to announce that it is leaving the conference. Although other sources can detect the absence of a source, this message is a direct announcement. It is also very useful to a mixer.

Application-Specific Message

The application-specific message is a packet for an application that wants to use new applications (not defined in the standard). It allows the definition of a new message type.

UDPPort

RTCP, like RTP, does not use a well-known UDP port. It uses a temporary port. The UDP port chosen must be the number immediately following the UDP port selected for RTP. It must be an odd-numbered port.

RTCP uses an odd-numbered UDP port number that follows the port number selected for **RTP.**

29.8 VOICE OVER **IP**

Let us concentrate on one real-time interactive audio/video application: voice over IP, or Internet telephony. The idea is to use the Internet as a telephone network with some additional capabilities. Instead of communicating over a circuit-switched network, this application allows communication between two parties over the packet-switched Internet. Two protocols have been designed to handle this type of communication: SIP and H.323. We briefly discuss both.

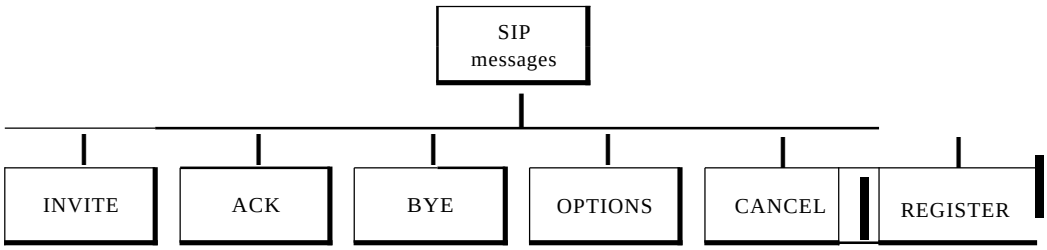
SIP

The Session Initiation Protocol (SIP) was designed by IETE It is an application layer protocol that establishes, manages, and terminates a multimedia session (call). It can be used to create two-party, multiparty, or multicast sessions. SIP is designed to be independent of the underlying transport layer; it can run on UDP, TCP, or SCTP.

Messages

SIP is a text-based protocol, as is HTTP. SIP, like HTTP, uses messages. Six messages are defined, as shown in Figure 29.21.

Figure 29.21 *SIP messages*



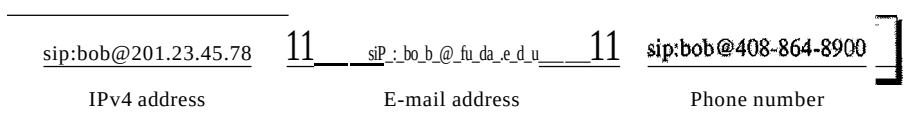
Each message has a header and a body. The header consists of several lines that describe the structure of the message, caller's capability, media type, and so on. We give a brief description of each message. Then we show their applications in a simple session.

The caller initializes a session with the INVITE message. After the callee answers the call, the caller sends an ACK message for confirmation. The BYE message terminates a session. The OPTIONS message queries a machine about its capabilities. The CANCEL message cancels an already started initialization process. The REGISTER message makes a connection when the callee is not available.

Addresses

In a regular telephone communication, a telephone number identifies the sender, and another telephone number identifies the receiver. SIP is very flexible. In SIP, an e-mail address, an IP address, a telephone number, and other types of addresses can be used to identify the sender and receiver. However, the address needs to be in SIP format (also called *scheme).* Figure 29.22 shows some common formats.

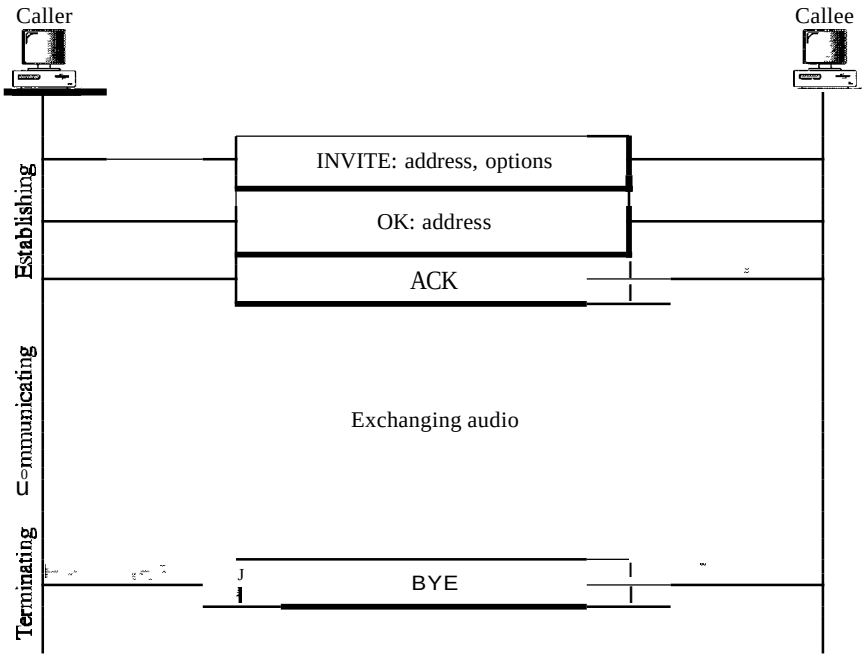
Figure 29.22 *SIP formats*



Simple Session

A simple session using SIP consists of three modules: establishing, communicating, and terminating. Figure 29.23 shows a simple session using SIP.

Figure 29.23 *SIP simple session*



Establishing a Session Establishing a session in SIP requires a three-way handshake. The caller sends an INVITE message, using UDP, TCP, or SCTP to begin the communication. If the callee is willing to start the session, she sends a reply message. To confirm that a reply code has been received, the caller sends an ACK message.

Communicating After the session has been established, the caller and the callee can communicate by using two temporary ports.

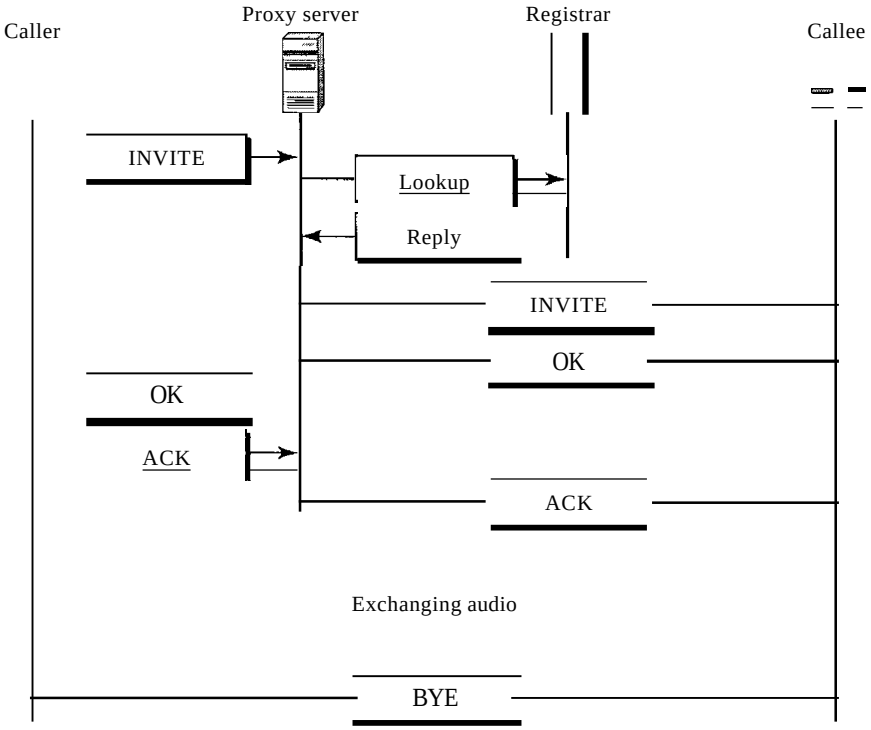
Terminating the Session The session can be terminated with a BYE message sent by either party.

Tracking the Callee

What happens if the callee is not sitting at her terminal? She may be away from her system or at another terminal. She may not even have a fixed IP address if DHCP is being used. SIP has a mechanism (similar to one in DNS) that finds the IP address of the terminal at which the callee is sitting. To perform this tracking, SIP uses the concept of registration. SIP defines some servers as registrars. At any moment a user is registered with at least one registrar server; this server knows the IP address of the callee.

When a caller needs to communicate with the callee, the caller can use the e-mail address instead of the IP address in the INVITE message. The message goes to a proxy server. The proxy server sends a lookup message (not part of SIP) to some registrar server that has registered the callee. When the proxy server receives a reply message from the registrar server, the proxy server takes the caller's INVITE message and inserts the newly discovered IP address of the callee. This message is then sent to the callee. Figure 29.24 shows the process.

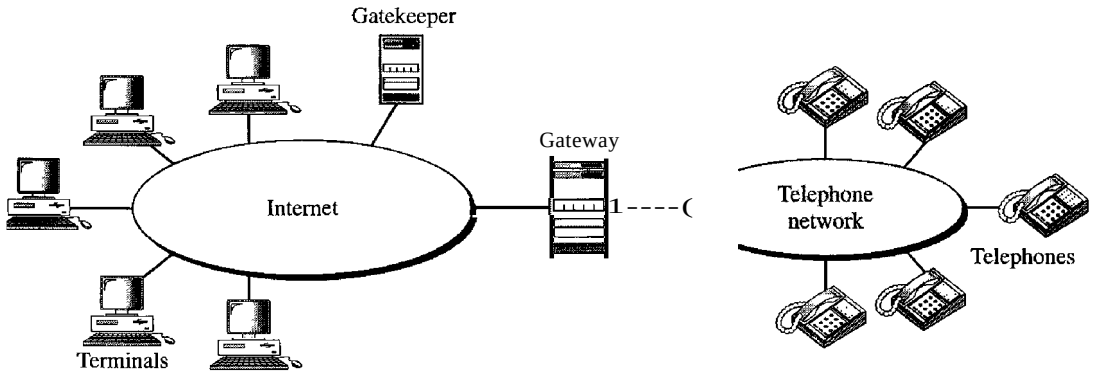
Figure 29.24 *Tracking the callee*



H.323

H.323 is a standard designed by lTV to allow telephones on the public telephone network to talk to computers (called *terminals* in H.323) connected to the Internet. Figure 29.25 shows the general architecture of H.323.

Figure 29.25 *H.323 architecture*



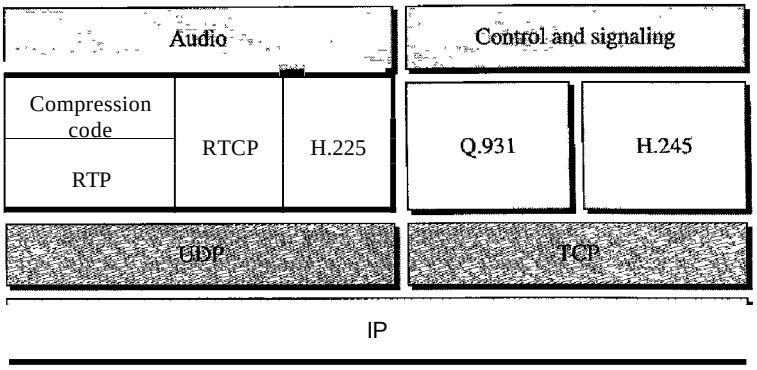
A gateway connects the Internet to the telephone network. In general, a gateway is a five-layer device that can translate a message from one protocol stack to another. The

gateway here does exactly the same thing. It transforms a telephone network message to an Internet message. The gatekeeper server on the local area network plays the role of the registrar server, as we discussed in the SIP.

Protocols

H.323 uses a number of protocols to establish and maintain voice (or video) communication. Figure 29.26 shows these protocols.

Figure 29.26 *H.323 protocols*



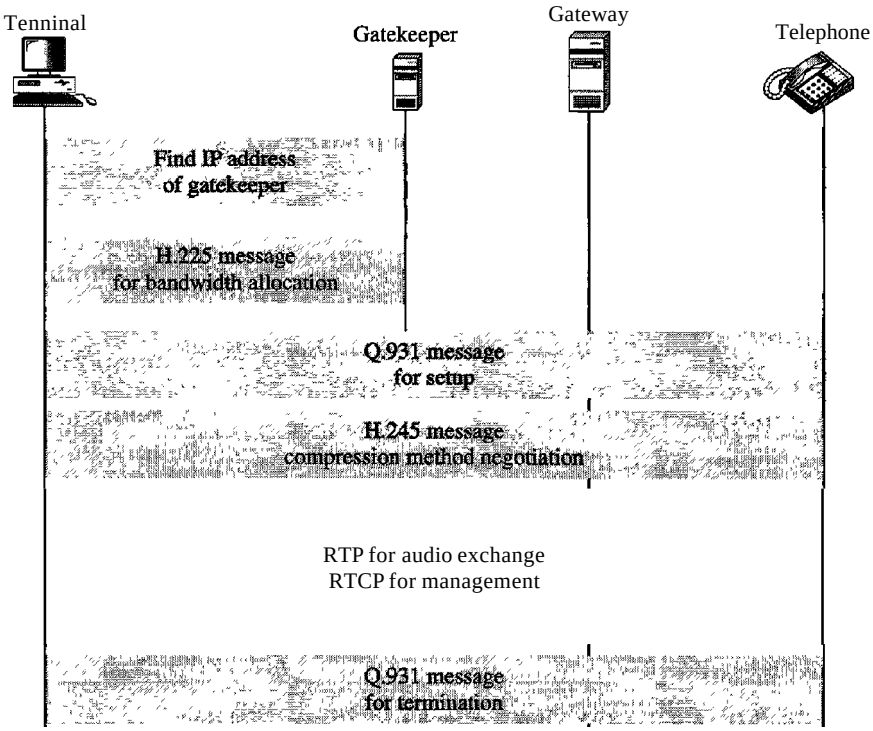
H.323 uses G.71 or G.723.1 for compression. It uses a protocol named H.245 which allows the parties to negotiate the compression method. Protocol Q.931 is used for establishing and terminating connections. Another protocol called H.225, or RAS (Registration!Administration!Status), is used for registration with the gatekeeper.

Operatio1l

Let us show the operation of a telephone communication using H.323 with a simple example. Figure 29.27 shows the steps used by a terminal to communicate with a telephone.

1. The terminal sends a broadcast message to the gatekeeper. The gatekeeper responds with its IP address.
2. The terminal and gatekeeper communicate, using H.225 to negotiate bandwidth.
3. The terminal, gatekeeper, gateway, and telephone communicate by using Q.931 to set up a connection.
4. The terminal, gatekeeper, gateway, and telephone communicate by using H.245 to negotiate the compression method.
5. The terminal, gateway, and telephone exchange audio by using RTP under the management of RTCP.
6. The terminal, gatekeeper, gateway, and telephone communicate by using Q.931 to terminate the communication.

Figure 29.27 *H.323 example*



29.9 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

Books

Multimedia communication is discussed in [HalOl] and in Section 7.4 of [Tan03]. Image and video compression are fully discussed in [Dro02].

Sites

- www.ietf.org/rfc.html Infonnation about RFCs

29.10 KEY TERMS

bidirectional frame (B-frame)
compression
discrete cosine transform (DCT)
frequency masking
gatekeeper
gateway
H.323
interactive audio/video
intracoded frame (I-frame)
jitter

Joint Photographic Experts Group (JPEG)
media player
media server
metafile
mixer
Moving Picture Experts Group (MPEG)
MP3
multicasting
on-demand audio/video
perceptual encoding
pixel
playback buffer
predicted frame (P-frame)
predictive encoding
quantization
Real-Time Streaming Protocol (RTSP)
real-time traffic
Real-time Transport Control Protocol (RTCP)
Real-time Transport Protocol (RTP)
registrar server
Session Initiation Protocol (SIP)
spatial compression
streaming live audio/video
streaming stored audio/video
temporal compression
temporal masking
timestamp
translation
voice over IP

29.11 SUMMARY

- Audio/video files can be downloaded for future use (streaming stored audio/video) or broadcast to clients over the Internet (streaming live audio/video). The Internet can also be used for live audio/video interaction.
- Audio and video need to be digitized before being sent over the Internet.
- Audio files are compressed through predictive encoding or perceptual encoding.
- Joint Photographic Experts Group (JPEG) is a method to compress pictures and graphics.
- The JPEG process involves blocking, the discrete cosine transform, quantization, and lossless compression.
- Moving Pictures Experts Group (MPEG) is a method to compress video.
- MPEG involves both spatial compression and temporal compression. The former is similar to JPEG, and the latter removes redundant frames.
- We can use a Web server, or a Web server with a metafile, or a media server, or a media server and RTSP to download a streaming audio/video file.
- Real-time data on a packet-switched network require the preservation of the time relationship between packets of a session.
- Gaps between consecutive packets at the receiver cause a phenomenon called jitter.
- Jitter can be controlled through the use of timestamps and a judicious choice of the playback time.
- A playback buffer holds data until they can be played back.
- A receiver delays playing back real-time data held in the playback buffer until a threshold level is reached.

- Sequence numbers on real-time data packets provide a form of error control.
- Real-time data are multicast to receivers.
- Real-time traffic sometimes requires a translator to change a high-bandwidth signal to a lower-quality narrow-bandwidth signal.
- A mixer combines signals from different sources into one signal.
- Real-time multimedia traffic requires both UDP and Real-time Transport Protocol (RTP).
- RTP handles time-stamping, sequencing, and mixing.
- Real-time Transport Control Protocol (RTCP) provides flow control, quality of data control, and feedback to the sources.
- Voice over IP is a real-time interactive audio/video application.
- The Session Initiation Protocol (SIP) is an application layer protocol that establishes, manages, and terminates multimedia sessions.
- H.323 is an ITU standard that allows a telephone connected to a public telephone network to talk to a computer connected to the Internet.

29.12 PRACTICE SET

Review Questions

1. How does streaming live audio/video differ from streaming stored audio/video?
2. How does frequency masking differ from temporal masking?
3. What is the function of a metafile in streaming stored audio/video?
4. What is the purpose of RTSP in streaming stored audio/video?
5. How does jitter affect real-time audio/video?
6. Discuss how SIP is used in the transmission of multimedia.
7. When would you use JPEG? When would you use MPEG?
8. In JPEG, what is the function of blocking?
9. Why is the DCT needed in JPEG?
10. What is spatial compression compared to temporal compression?

Exercises

11. In Figure 29.17 what is the amount of data in the playback buffer at each of the following times?
    a. 00:00:17
    b. 00:00:20
    c. 00:00:25
    d. 00:00:30
12. Compare and contrast TCP with RTP. Are both doing the same thing?
13. Can we say UDP plus RTP is the same as TCP?
14. Why does RTP need the service of another protocol, RTCP, but TCP does not?

15. In Figure 29.12, can the Web server and media server run on different machines?
16. We discuss the use of SIP in this chapter for audio. Is there any drawback to prevent using it for video?
17. Do you think H.323 is actually the same as SIP? What are the differences? Make a comparison between the two.
18. What are the problems for full implementation of voice over IP? Do you think we will stop using the telephone network very soon?
19. Can H.323 also be used for video?

Research Activities

20. Find the format of an RTCP sender report. Pay particular attention to the packet length and the parts repeated for each source. Describe each field.
21. Find the format of an RTCP receiver report. Pay particular attention to the packet length and the parts repeated for each source. Describe each field.
22. Find the format of an RTCP source description. Pay particular attention to the packet length and the parts repeated for each source. Describe each field.
23. Find the meaning of the source description items used in the RTCP source description packet. Specifically, find the meaning of CNAME, NAME, EMAIL, PHONE, LOC, TOOL, NOTE, and PRIY.
24. Find the format of an RTCP bye message. Pay particular attention to the packet length and the parts repeated for each source. Describe each field.