Chapter 1

FUNDAMENTALS OF STREAMING MEDIA SYSTEMS

Multimedia (MM) systems utilize audio and visual information, such as video, audio, text, graphics, still images, and animations to provide effective means for communication. These systems utilize multi-human senses in conveying information, and they play a major role in educational applications (such as e-learning and distance learning), library information systems (such as digital library systems), entertainment systems (such as Video-On-Demand and interactive TV), communication systems (such as mobile phone multimedia messaging), military systems (such as Advanced Leadership Training Simulation), etc. Due to the exponential improvements of the past few years in solid state technology (i.e., processor and memory) as well as increased bandwidth and storage capacities of modern magnetic disk drives, it has been technically feasible to implement these systems in ways we only could have dreamed about a decade ago.

A challenging task when implementing MM systems is to support the sustained bandwidth required to display *Streaming Media* (SM) objects, such as video and audio objects. Unlike traditional data types, such as records, text and still images, SM objects are usually large in size. For example, a two-hour MPEG-2 encoded movie requires approximately 3.6 gigabytes (GByte) of storage (at a display rate of 4 megabits per second (Mb/s)). Figure 1.1 compares the space requirements for ninety minute video clips encoded in different industry standard digital formats. Second, the isochronous nature of SM objects requires timely, real-time display of data blocks at a pre-specified rate. For example, the NTSC video standard requires that 30 video frames per second be displayed to a viewer. Any deviation from this real-time requirement may result in undesirable artifacts, disruptions, and jitters,

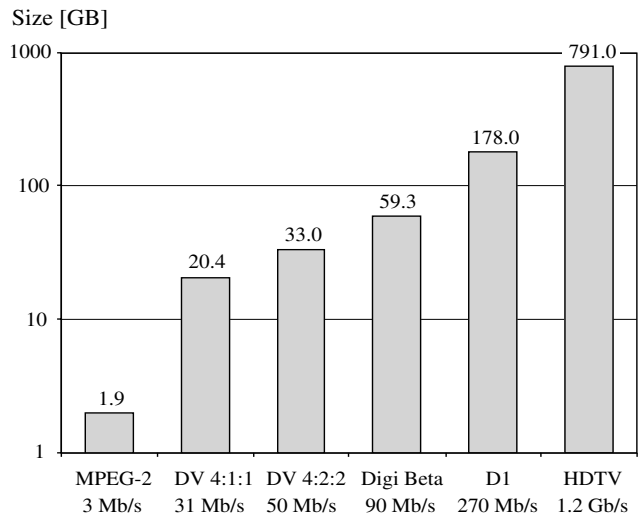

Figure 1.1. Storage requirements for a ninety minute video clip digitized in different industry standard encoding formats.

collectively termed *hiccups*.

In the remainder of this chapter, the following topics are covered. First, in Section 1.1, we introduce different display paradigms for SM objects. Next, in Section 1.2, we consider the overall SM system architecture. After that, in Section 1.3, we briefly overview data compression techniques. Subsequently, in Section 1.4, we describe a number of networking protocols that ensure real-time streaming over the Internet. Finally, in Section 1.5, we present an outline for the entire textbook.

1.1 Introduction to Streaming Media Display

Many applications that traditionally were the domain of analog audio and video are evolving to utilize digital audio and video. For example, satellite broadcast networks, such as *DirectTV*^{TM}, were designed from the ground up with a completely digital infrastructure. The proliferation of digital audio and video have been facilitated by the wide acceptance of standards for compression and file formats. Consumer electronics are also adopting these standards in products such as the digital versatile disk (DVD[1] and digital

[1]DVD is a standard for optical discs that feature the same form-factor as CD-ROMs but holds 4.7 GByte of data or more.

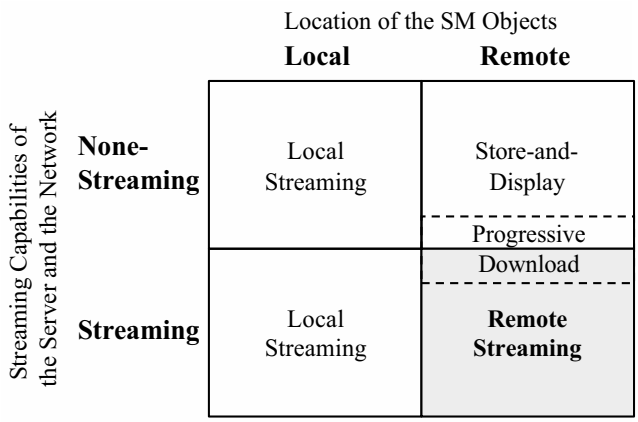

Figure 1.2. SM Display Paradigms.

VHS[2] (D-VHS)). Furthermore, increased network capacity for local area networks (LAN) and advanced streaming protocols (e.g., RTSP[3]) allow remote viewing of SM clips.

There are a number of possible display paradigms for SM objects, as shown in Figure 1.2. These display paradigms are defined according to:

1. The location of the SM object (local vs. remote), and

2. The capabilities of the network and the servers (none-streaming vs. streaming).

SM objects can either be displayed from a local machine (i.e., *local streaming*) or from a remote server. In the former approach, the SM objects are available locally in their entirety (e.g., stored on a hard disk, or available on DVDs and/or CD-ROMs). When the user requests the display of an object, the data blocks are retrieved from the local storage, passed to the *video-card/graphics-card* for decoding, and subsequently displayed to the user. Modern personal computers (PCs) and consumer electronic devices (such as Sony's *PlayStation*^{TM} and Microsoft's *Xbox*^{TM}) can display a single SM object without much difficulty. Local streaming does not require any guarantees from the remote servers and the network (for obvious reasons), and hence, it occupies the upper and lower left quadrants of Figure 1.2.

[2]Video Home System: a half-inch video cassette format.

[3]The Real Time Streaming Protocol is an Internet Engineering Task Force (IETF) proposed standard for control of SM on the Internet.

When the user requests the display of an object that resides on a remote server, the data blocks are retrieved from the remote server over a network (e.g., the Internet or a corporate intranet), and passed to the client for display. Depending on the capabilities of the server and the network, there are two alternative display paradigms: 1) *store-and-display* paradigm, and 2) *remote streaming* paradigm. With the store-and-display paradigm, the SM objects are first downloaded in their entirety from a remote server to a local storage before initiating the display process (which is similar to local streaming). This paradigm is favored when the server and/or the network cannot guarantee the isochronous nature of SM objects (upper right quadrant of Figure 1.2). With the remote streaming paradigm, the data blocks are retrieved from the remote server over a network and processed by the client as soon as they are received; data blocks are not written to the local storage (although they might be buffered). This paradigm is possible only when both the server and the network can guarantee the isochronous nature of the SM object (lower right quadrant of Figure 1.2). In the remainder of this book, we will be referring to remote streaming as *streaming*.

There are several advantages of streaming paradigm versus store-and-display paradigm:

1. No waiting for downloads (well, not much waiting).

2. No physical copies of the content are stored locally. reducing the possibility of copyright violations (if any).

3. No storage (or limited storage) requirements at the client side.

4. Support of live events.

On the other hand, there are a number of limitations to the streaming paradigm:

1. It requires real-time guarantees from the server and the network.

2. Lost or damaged packets (blocks) or missed deadlines may cause a hiccup in the display of the SM object.

It is important to note that streaming is not a progressive download, where the display starts as soon as enough of the data is buffered locally while the remainder of the file is being retrieved and stored on the local disk (i.e., at the end of the display the entire SM object is stored locally). Progressive download can be identified as a special case of the steaming paradigm (if the

system can guarantee the hiccup-free display of the SM objects), or a special case of store-and-display paradigm.

In this book, we focus on the design of Steaming Media Servers (*SM Servers*) that guarantee the continuous display of SM objects, assuming that the network delivers the data blocks in a timely fashion. We ignore for the most part design issues related to SM encoding standards (such as MPEG[4] standards) and SM networking[5].

1.2 Streaming Media System Architecture

To support the streaming paradigm, servers and the network must guarantee the continuous display of the SM object without disruptions (i.e., hiccups). To illustrate, assume a SM object X that consists of n equi-sized blocks: X_0, X_1, ..., X_{n-1}, and resides on a SM server, as shown in Figure 1.3. There are some important observations when streaming this object:

1. The display time of each block is a function of the display requirements of each object and the size of the block. For example, if the display requirement of the object X is 4 Mb/s and the size of each block, X_i, is 1 MByte, then the display time of each block, X_i, is 2 seconds. However, it is important to note that SM objects can be encoded using either *constant bit-rate* (CBR) or *variable bit-rate* (VBR) schemes. As the name implies, the consumption rate of a CBR media is constant over time, while VBR streams use variable rates to achieve maximum compression. We will assume CBR media throughout the book to provide a focused discussion, unless indicated otherwise.

2. The retrieval time of each block is a function of the transfer rate of the SM server, and it is primarily related to the speed of the storage subsystem.

3. The delivery time of each block from the SM server to the display station is a function of the network speed, traffic, and protocol used.

Assume that a user requests the display of object X from the SM server, as depicted in Figure 1.3. Using the above information, the SM server schedules the retrieval of the blocks, while the network ensures the timely delivery

[4]Motion Picture Encoding Group (MPEG) (http://mpeg.telecomitalialab.com/) has standardized several audio and video compression formats, such as MPEG-1, MPEG-2, and MPEG-4.

[5]In this chapter, we present an overview of SM encoding standards and introduce a number of networking technologies that make SM possible over the Internet.

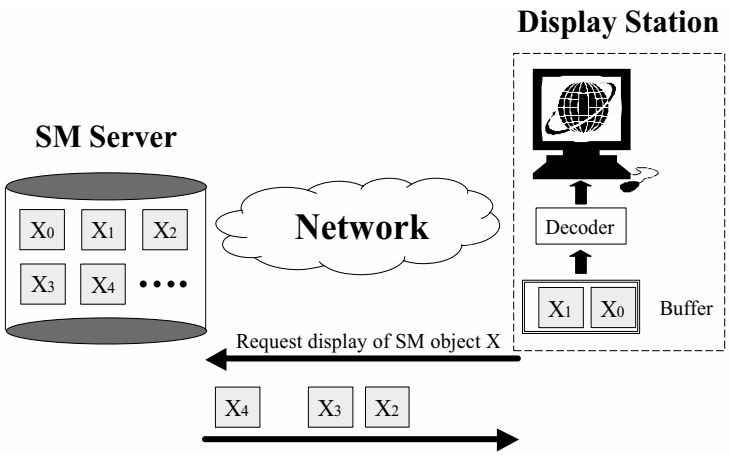

Figure 1.3. System Architecture.

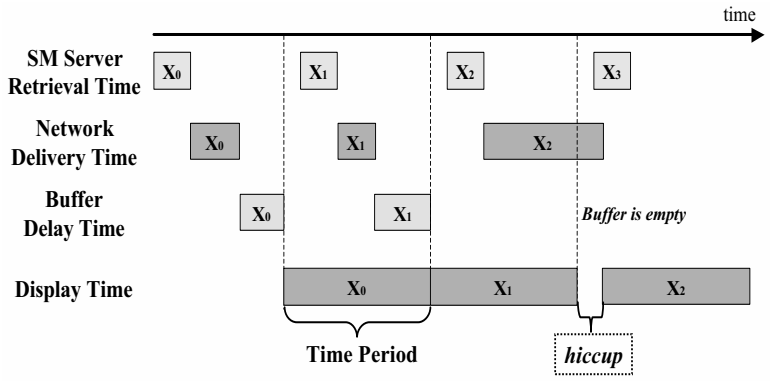

Figure 1.4. Timing of retrieval, delivery, and display of object X.

of these blocks to the display station. The SM server stages a block of X (say X_i) from the disk into main memory and initiates its delivery to the display station (via the network). The SM server schedules the retrieval and the delivery of X_{i+1} prior to the completion of the display of X_i. This ensures a smooth transition between the two blocks in support of continuous display. This process is repeated until all blocks of X have been retrieved, delivered, and displayed. The periodic nature of the data retrieval and display process gives rise to the definition of a *time period* (T_p): it denotes the time required to display one data block. Note that the display time of a block is in general

significantly longer than its retrieval and delivery time from the SM server to the display station. Thus, the SM server can be multiplexed among several displays.

The display station in turn might buffer enough data (i.e., blocks) in its memory, such that any delays in data delivery due to network or SM server delays are tolerated[6] (see Figure 1.3). To guarantee the continuous display of object X without any hiccups, it is critical that prior to the completion of the display of block X_i, block X_{i+1} is available in the buffer at the display station. For example, consider the retrieval times, delivery times, and the display times shown in Figure 1.4. Block X_1 is retrieved and delivered to the display station, and the display station delays the display of the block to reduce the probability of hiccups (e.g., due to network delays). For block X_1, the network delivers the block in a shorter time, and hence, the block is buffered for a longer period of time. If the network delay becomes longer than anticipated, as shown in Figure 1.4 for block X_2, then the display of the object might suffer a hiccup. Conversely, if the delivery of the blocks becomes faster than the display rate of the object, then the display station's buffer might overflow. In this case, the display station might have to either: 1) discard blocks, or 2) send a signal to the SM server to slow down or to pause the retrieval. The former will cause retransmissions of the blocks (increasing the network traffic) and extra load on the SM server, or it might cause hiccups in the display. The later will affect retrieval schedule at the SM server.

Due to the size and the isochronous characteristic of SM objects, the design of servers in support of SM has been different from that of conventional file servers and associated storage systems [175], [59]. The fundamental functionality of SM servers is a hiccup-free display of SM. However, just supporting a continuous display is not enough in the design of SM servers because many real applications, especially commercial ones such as movie-on-demand systems that concurrently service multiple users, require maximizing the performance of servers for cost-effective solutions. Thus, the following performance metrics are important: 1) the number of simultaneous displays that can be supported by a SM server, i.e., *throughput*, and 2) the amount of time elapsed from when a display request arrives at the system until the time the actual display is initiated by the system on behalf of this request, i.e., *startup latency*. Throughput, in general, is closely related to another important metric of SM servers, *cost per stream*. If a technique supports a higher throughput with fixed resources than others, it provides for a more

[6] Note that network and SM server delays can either be deterministic or otherwise.

cost-effective solution. Throughout this book, we compare alternative designs based on these metrics.

1.3 Data Compression

A ninety minute uncompressed video clip digitized in High Definition Television (HDTV) format at 1.2 gigabits per second (Gb/s) requires 791 GByte of storage. Even though modern storage devices, such as magnetic disk drives, provide large storage capacities, it might not be economical to store SM objects uncompressed. Moreover, the data transfer rate of magnetic disks is not high enough to retrieve multiple high bit rate objects in real time. A more serious problem arises when transferring a large number of SM objects over the network. Even though network speeds are rapidly increasing, it is not economically feasible to handle the simultaneous display of a large number of SM objects over existing networks.

To resolve these problems and to efficiently handle large number of SM objects, we need to compress these objects, where a smaller SM object requires less disk storage space and less network bandwidth. In the remainder of this section, we briefly overview data compression techniques.

1.3.1 Information vs. Data

Data is an individual fact or multiple facts, or a value, or a set of values. For example, a collection of digitally captured pixel values of an image is data. Information is data in a usable form, usually interpreted in a predefined way. For example, the content of an image is information. In general, information is meaningful to us and it is stored and handled as data in computer systems. Thus, the main goal of compression techniques is to reduce the size of data while maintaining as much information as possible.

A popular approach in multimedia data compression is *perceptual coding* that hides errors where humans will not see or hear them. Based on the studies of human hearing and vision to understand how we see/hear, perceptual coding exploits the limit of human perception. For example, human audio perception ranges between 20 Hz and 20 KHz but most voice sounds are in low frequencies (below 4 KHz). Thus, audio signals above 4 KHz are eliminated in telephone systems. Human visual perception is strongly influenced by edges and low frequency information such as big objects. Thus, many detailed patterns in an image can be ignored in the coding process. Perception coding takes advantage of this fact and encodes only those signals that will be perceived by humans, eliminating lots of imperceptible signals.

1.3.2 Coding Overview

Good compression algorithms should satisfy the following objectives:

- Achieve high compression ratios. It is obvious that a higher compression ratio is more beneficial.

- Ensure a good quality of the reconstructed information, i.e., it is important to maintain a good or acceptable quality while providing a high compression ratio.

- Minimize the complexity of the encoding and decoding process, i.e., if it takes too long to encode and/or decode the SM objects, then the usage of the algorithm might become limited. For real time encoding and decoding, coding and decoding processes must be fast.

Some other general requirements include independence of specific size and frame rate and support various data rates of an object.

Depending on the importance of the original information, there are two types of compression: *lossless compression* and *lossy compression*. Using lossless compression, the reconstructed information is mathematically equivalent to the original information (i.e., reconstruction is perfect). Thus, it does not lose or alter any information in the process of compression and decompression. For example, documents, databases, program executable files, and medical images, to name a few, do not tolerate a loss or alteration of information. Lossless compression reduces the size of data for these important types of information by detecting repeated patterns and compressing them. However, it achieves only a modest level of compression (about a factor of 5), which may not be sufficient for SM objects.

While lossless compression is desirable for information that cannot tolerate a loss of even a bit, some media types are not very sensitive to the loss of information. Human eyes usually do not detect minor color changes between the original image and the reconstructed one, and usually such small changes do not pose a problem in many practical applications. Thus, *lossy compression* achieves a very high degree of compression (compression ratios up to 200) by removing less important information such as imperceptible audio signals in music. Using lossy compression, reconstructed images may demonstrate some degradation in the quality of the image. The practical objective of lossy compression is to maximize the degree of compression while maintaining the quality of the information to be *virtually lossless*.

Entropy and Source Encoding

Entropy is a measure of amount of information. If N states are possible, each characterized by a probability p_i, with $\sum_{i=1}^{N} p_i = 1$, then $S = -\sum_{i=1}^{N} p_i log_2 p_i$ is the entropy which is the lowest bound on the number of bits needed to describe all parts of the system. It corresponds to the information content of the system. For example, when there are only two symbols, S1 (0.9) and S2 (0.1), the entropy is $0.9log_20.9 - 0.1log_20.1 = 0.47$. Another typical example is as follows. In an image with uniform distribution of gray-level intensity, i.e., $p_i = 1/256$, the number of bits needed to code each gray level is 8 bits. Thus, the entropy of this image is 8.

Entropy encoding ignores semantics of input data and compresses media streams by regarding them as sequences of bits. Run-length encoding and Huffman encoding are popular entropy encoding schemes. On the contrary, *source encoding* optimizes the compression ratio by considering media-specific characteristics. It is also called *sematic-based coding*. Many advanced schemes such as DPCM, DCT, FFT, and Wavelet fall into this category. In reality, most practical compression algorithms employ a hybrid of the source and entropy coding such that a source encoding is applied at the early stage of compression and then an entropy encoding is applied to results in an attempt to further reduce the data size.

Run-length Encoding

This is one of the simplest compression techniques. It replaces consecutive occurrences of a symbol with the symbol followed by the number of times it is repeated. For example, the string, "a a a a a" can be represented by the following two bytes, "5a". The first byte shows the number of occurrences of the character, and the second byte represents the character itself. This scheme is naturally lossless and its compression factor ranges from 1/2 to 1/5 depending on the contents of objects. This scheme is most useful where symbols appear in long runs: e.g., for images that have areas where the pixels all have the same value, cartoons for example. However, this may not be efficient for complex objects where adjacent symbols are all different.

Relative Encoding

This is a technique that attempts to improve efficiency by transmitting only the difference between each value and its predecessor. In an image, based on the fact that neighboring pixels may be changing slowly in many cases, one can digitize the value at a pixel by using the values of the neighboring pixels and encode the difference between two. A differential PCM coder (DPCM)

quantizes and transmits the difference, $d(n) = x(n)x(n-1)$. The advantage of using difference $d(n)$ instead of the actual value $x(n)$ is the reduced number of bits to represent a sample. For example, the series of pixel values "60 105 161 129 78", can be represented by "60+45+56-32-51". By reducing the size of each value (in the example, 6 bits are enough to represent the difference while 8 bits are required to represent the original values), assuming the difference is far smaller than the original value, it can reduce the overall size of data. This scheme works well when the adjacent values are not greatly changing, such as voice signals. Furthermore, the transmitter can predict each value based on a mathematical model and transmit only the difference between the predicted and actual values, further reducing the size of required bits to represent a value (*predictive coding*).

Huffman Encoding

Huffman encoding is a popular compression technique that assigns variable length codes to symbols, so that the most frequently occurring symbols have the shortest codes. Thus, more frequently occurring values are assigned fewer bits exploiting the statistical distribution of the values within an object. To correctly decompress the encoded data, encoder and decoder must share the same codebook. Huffman coding is particularly effective where the data are dominated by a small number of symbols. Suppose that one wants to encode a source of N = 8 symbols: a,b,c,d,e,f,g,h. The probabilities of these symbols are: P(a) = 0.01, P(b) = 0.02, P(c) = 0.05, P(d) = 0.09, P(e) = 0.18, P(f) = 0.2, P(g) = 0.2, P(h) = 0.25. If we assign 3 bits per symbol ($N = 2^3 = 8$), the average length of the symbols is: $L = \sum_{i=1}^{8} 3P(i) = 3$ bits/symbol. The minimum average length we could ever achieve is equal to the entropy (according to Shannon's theorem): $S = -\sum_{i=1}^{8} P(i) log_2 P(i) =$ 2.5821 bits/symbol $= 0.86 \times L$.

The Huffman code assignment procedure is based on a binary tree structure. This tree is developed by a sequence of pairing operations in which the two least probable symbols are joined at a node to form two branches of a tree.

- Step 1. The list of probabilities of the source symbols are associated with the leaves of a binary tree.

- Step 2. Take the two smallest probabilities in the list and generate an intermediate node as their parent and label the branch from parent to one of the child nodes 1 and the branch from parent to the other child 0.

- Step 3. Replace the probabilities and associated nodes in the list by the single new intermediate node with the sum of the two probabilities. If the list contains only one element, quit. Otherwise, go to step 2.

It is very important to estimate the probability p_i, the relative frequency of the symbols. To decode the variable length codes, Huffman encoding uses prefix codes, which have the property that no codeword can be the prefix (i.e., an initial segment) of any other codeword. Note that there is no guarantee that the best possible codes always reduce the size of sources. In the worst case, there is no compression.

Transform Coding

In transform coding, using a mathematical transformation, the original data to be coded is converted into new data (transform coefficients) which is more suitable for compression. This process is reversible such that the original data can be recovered through inverse transformation. For example, Fourier transform and Cosine transform convert signals from space domain into frequency domain to obtain the frequency spectrum with which one can easily separate low frequency information.

Transform coefficients represent a proportion of energy contributed by different frequencies. In data compression using transform coding, it is important to choose a transformation so that only a subset of coefficients have significant values. In other words, energy is confined to a subset of important coefficients, known as energy compaction. Energy compaction is good for coding because we can consider only significant coefficients. If the number of significant coefficients is far smaller than the number of samples in original sequence, compression is possible. In practice, one can code significant coefficients accurately using a greater number of bits while allocating fewer or no bits to other less meaningful coefficients (which offer less perceptible information to humans). Moreover, many low energy coefficients can even be discarded through quantization. In practice, many algorithms apply transform coding at block level. For example, an N x N image is divided into several n x n blocks and each n x n block undergoes a reversible transformation.

1.3.3 JPEG

JPEG (Joint Photographic Expert Group) is an international compression standard for *continuous-tone* still images, both gray-scale and color. Its development was motivated by the fact that the compression ratio of lossless methods such as Huffman is not high enough for many applications such as

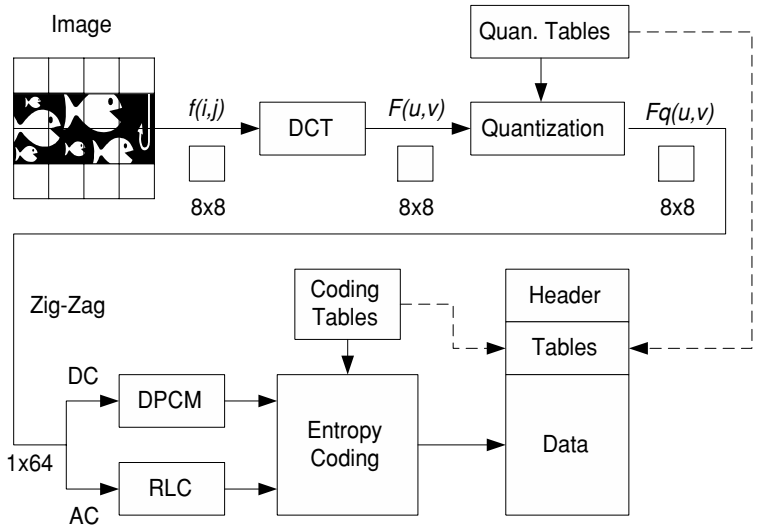

Figure 1.5. JPEG overview.

high quality gray-scale images, photographic images, and still video. JPEG achieves a high compression ratio by removing *spatial redundancy* (i.e., correlation between neighboring pixels) within an image, which is basically a lossy process.

Figure 1.5 shows the components and sequence of JPEG. JPEG utilizes discrete cosine transform (DCT), which is one of the best known transforms and is a close approximation to the optimal for a large class of images. DCT transforms data from a spatial domain to a frequency domain and separates high frequency information and low frequency information.

Discrete Cosine Transform (DCT)

As the first step in JPEG compression, an original image is partitioned into 8x8 pixel blocks and the algorithm independently applies DCT to each block. DCT transforms data from the spatial domain to the frequency domain in which they can be more efficiently encoded.

The definition of forward discrete cosine transform (FDCT) and inverse discrete cosine transform (IDCT) is as follows:

$$F(u,v) = \frac{1}{4}C(u)C(v)[\sum_{i=0}^{7}\sum_{j=0}^{7} f(i,j) * \cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}] \quad (1.1)$$

$$f(i,j) = \frac{1}{4}[\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v) * \cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}] \quad (1.2)$$

where: $C(u), C(v) = 1/\sqrt{2}$ for $u, v = 0$, $C(u), C(v) = 1$ otherwise.

Using FDCT, original pixel values, $f(i,j)$ where $0 \leq i, j < 8$, are transformed into $F(u,v)$ named DCT coefficient. The output DCT coefficient matrix represents information of the 8x8 block in frequency domain. DCT enhances compression by concentrating most of the energy in the signal in the lower spatial frequencies. The left upper corner of the matrix represents low frequency information while the right lower corner represents high frequency information. IDCT restores original pixel values from DCT coefficients.

Quantization

The purpose of quantization is to achieve high compression by representing DCT coefficients with no greater precision than necessary by discarding information which is not visually significant to human perception. After FDCT, each of the 64 DCT coefficients is quantized: $F[u,v] = Round(F[u,v]/q[u,v])$. For example, assume a DCT coefficient, $101101 = 45$ (6 bits). If it is quantized by a quanta value, $q[u,v] = 4$, the original DCT value is truncated to 4 bits, $1011 = 11$, reducing the required number of bits to represent a DCT coefficient. Due to many-to-one mapping, quantization is a fundamentally lossy process, and is the principal source of lossiness in DCT-based encoders. There are two different types of quantization: *uniform* and *non-uniform*. Uniform quantization divides each $F[u,v]$ by the same constant N. Non-uniform quantization uses a quantization table from psychovisual experiments to exploit the limits of the human visual system.

Zig-Zag Sequence

After quantization, the 8x8 matrix of quantized coefficients is ordered into a one-dimensional array (1x64) using the zig-zag sequence (Figure 1.6). This is to facilitate entropy coding by placing low-frequency coefficients on the top of the vector (maps 8x8 to 1x64 vector).

Entropy Coding

In order to achieve additional compression losslessly by encoding the quantized DCT coefficients more compactly based on statistics, JPEG applies Differential Pulse Code Modulation (DPCM) on the DC component which is a measure of the average value of the 64 image samples. DPCM is useful because there is usually a strong correlation between the DC coefficients of

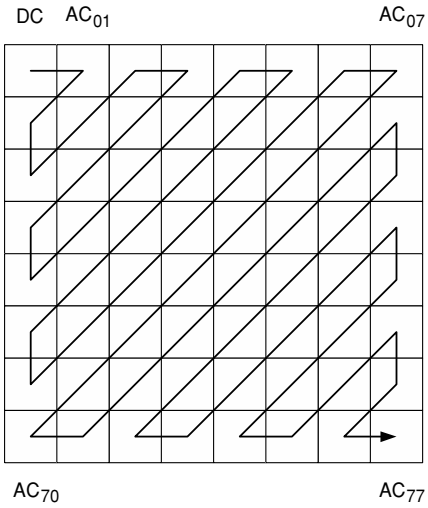

Figure 1.6. Zig-zag sequence.

adjacent 8x8 blocks. Run Length Encoding (RLE) is applied on AC components which usually have lots of zeros. For further compression, Huffman coding is used for both DC and AC coefficients at the last stage.

JPEG Operating Modes

JPEG supports the following operation modes to meet the various needs of many applications:

- Sequential Mode:
  Each image component is encoded in a single left-to-right, top-to-bottom scan as explained in this section.

- Lossless Mode:
  A special case of the JPEG where indeed there is no loss. It does not use a DCT-based method. Instead, it uses a predictive (differential coding) method. Typical compression ratio is 2:1.

- Progressive Mode:
  Each image is coded in multiple scans with each successive scan refining the image until a desired quality of image is achieved. Display of the encoded image follows the same steps: a coarser image is displayed first, then more components are decompressed to provide a finer version of the image.

- Hierarchical Mode:
  An image is compressed to multiple resolution levels. This makes it possible to display a high resolution image on a low resolution monitor by accessing a lower resolution version without having to decompress the full version.

1.3.4 MPEG

JPEG achieves intra-frame compression for still images by exploiting the redundancy in images (spatial redundancy). This intra-frame compression is not enough for video because it doesn't consider inter-frame compression between successive frames. We need a different scheme for video to exploit both spatial redundancy and temporal redundancy. MPEG is a de facto compression standard for video, audio, and their synchronization. MPEG (Moving Picture Coding Experts Group) was established in 1988 to create standards for delivery of video and audio. MPEG achieves intra-frame encoding using DCT-based compression for the reduction of spatial redundancy (similar to JPEG). Its inter-frame encoding utilizes block-based motion compensation for the reduction of temporal redundancy. Specifically, MPEG uses bidirectional motion compensation.

Block-based Motion Compensation

To exploit the temporal redundancy between adjacent video frames, *difference coding* compares pixels in the current frame with ones in the previous frame so that only changed pixels are recorded. In this way, a fraction of the number of pixel values will be recorded for the current frame. In practice, pixels values are slightly different even with no movement of objects. This can be interpreted as lots of changes and result in less compression. Thus, difference coding needs to ignore small changes in pixel values and it is naturally lossy.

For more efficient compression, difference coding is done at the block level. Block-based difference coding receives a sequence of blocks rather than frames. For example, a 160 x 120 image (19200 pixels) can be divided into 300 8x8 blocks. If a block in the current frame is similar to the one in the same location of the previous frame, the algorithm skips it or stores only the difference. If they are different, a whole block of pixels is updated at once. Limitations of difference coding are obvious. It is useless where there is a lot of motion (few pixels unchanged). What if objects in the frame move fast? What if a camera itself is moving?

Motion compensation assumes that the current frame can be modelled as

a translation of the previous frame. As in block-based difference coding, the current frame is divided into uniform non-overlapping blocks. Each block in the current frame to be encoded is compared to areas of similar size from the preceding frame in order to find an area that is similar, i.e., the best matching block. The relative difference in locations is known as the *motion vector*. The matching process can be based on *prediction* or *interpolation*. Because fewer bits are required to code a motion vector than to code an actual block, compression is achieved. Motion compensation is the basis of most video compression algorithms.

For further and better compression, *bidirectional motion compensation* can be used. Areas just uncovered in the current frame are not predictable from the past, but they can be predicted from the future. Thus, bidirectional motion compensation searches in both past and future frames to find the best matching block. Moreover, the effect of noise and errors can be reduced by averaging between previous and future frames. Bidirectional interpolation also provides a high degree of compression. However, it requires that frames be encoded and transmitted in a different order from which they will be displayed. In reality, because exact matching is not possible, it is a lossy compression.

Group of Pictures

MPEG defines a set of pictures to form a group of pictures (GOP) consisting of the following types (Figure 1.7):

- I frame: Intra-coded picture
- P frame: Unidirectionally predicted picture
- B frame: Bidirectionally predicted picture

I-frames are encoded using intra-frame compression that is similar to JPEG. Thus, they can be decoded without other frames. I-frames are used as access points for random access. P-frames are predicted frames with reference to a previous I or P frame. B-frames are bidirectional frames encoded using the previous and the next I/P frames.

MPEG Standards

MPEG-1 (ISO/IEC 11172) is a standard for storing and playing video on a single computer at low bit-rates up to 1.5 Mb/s. MPEG-2 (ISO/IEC 13818) is a standard for high quality video such as digital TV (HDTV and DVD). MPEG-2 builds upon MPEG-1 standard and supports both field prediction

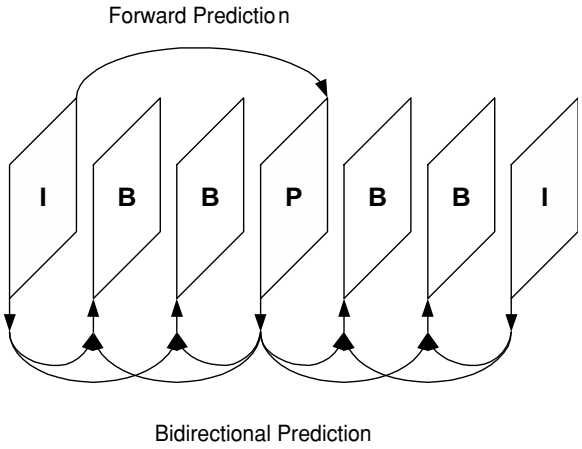

Figure 1.7. A group of pictures in MPEG.

and frame prediction (interlaced video format). While MPEG-2 aims at high quality video, MPEG-4 standard (ISO/IEC 14496) supports low bit-rate encoding of audio and video, user interactivity, and the special requirements for next generation broadcast services. MPEG-7 standard (ISO/IEC 15938) provides a set of standardized tools to describe multimedia content: metadata elements and their structure and relationships, that are defined by the standard in the form of Descriptors and Description Schemes. Officially, MPEG-7 is called *Multimedia Content Description Interface* [129]. MPEG-21 standard (ISO/IEC 21000) is being developed to define a multimedia framework in order to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities.

Hierarchical Coding

Hierarchical coding in MPEG encodes images in a manner that facilitates access to images at different quality levels or resolutions. This makes the progressive transmission possible: Partial image information is transmitted in stages, and at each stage, the reconstructed image is progressively improved. Hierarchical coding is motivated by the need for transmitting images over low-bandwidth channels. Progressive transmission can be stopped either if an intermediate version is of satisfactory quality or the image is found to be of no interest. Hierarchical coding is also very useful in multi-use environments where applications need to support a number of display devices with different resolutions. It could optimize utilization of storage server and

network resources.

MPEG Issues

The following issues are commonly considered in the design of streaming applications using MPEG:

- Avoiding propagation of errors. Due to the dependency among successive frames, errors such as missing frames or packets transfer over multiple frames. We can avoid this problem by sending an I-frame every once in a while (e.g., by setting a reasonable group of pictures).

- Bit-rate control. In general, complicated images result in a less compression and a higher bit-rate than simple images. To regulate the output bit-rate, a simple feedback loop based on buffer fullness is used. If the buffer is close to full, MPEG increases the quantization scale factor to reduce the size of data.

- Constant Bit Rate (CBR) vs. Variable Bit Rate (VBR). MPEG streams can be encoded either as constant bit rate or as variable bit rate. CBR approach is more appropriate for video broadcasting through fixed bandwidth channels while VBR supports fixed quality of images such as DVD better than CBR.

1.4 Delivery of Streaming Media Over Internet

Due to its ubiquitous existence, the Internet has become the platform of most networking activities including SM applications, where users require the integration of multimedia services. However, as a shared datagram network, the Internet is not naturally suitable for real-time traffic, such as SM. Dedicated links are not practical in many ways for transmitting SM data over the Internet. Moreover, because of the store-and-forwarding in routers and internetworking devices, Internet Protocol (IP) has some fundamental problems in transmitting real-time data. Thus, delivery of SM over the Internet requires special installation and new software development so that packets experience as little delay as possible. ATM is promising in transmitting real-time data because it supports high bandwidth connection-oriented transmission and various quality of service (QoS) for different applications. However, it is expensive and not widely available at user sites.

In general, the following issues must be resolved to stream multimedia data over the Internet.

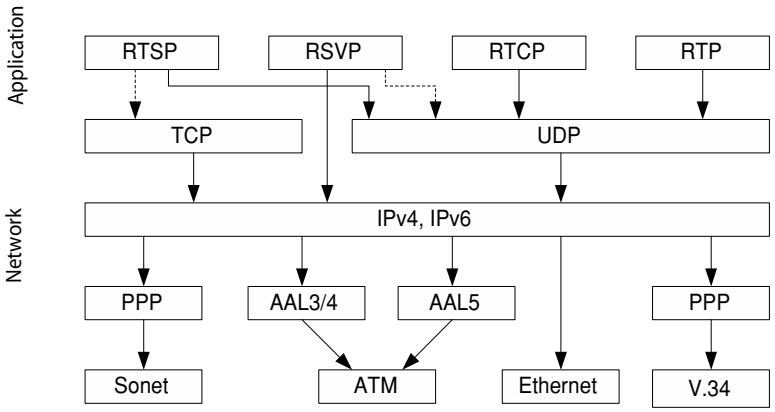

Figure 1.8. Network protocols for SM over the Internet.

- **High Bandwidth**
  The underlying hardware has to provide enough bandwidth to handle large size of multimedia data.

- **Multicast**
  Applications need to take into account multicast in order to reduce the traffic.

- **Guaranteed Bandwidth**
  There should be some mechanisms for real-time applications to reserve resources along the transmission path in order to guarantee QoS.

- **Out of Order Delivery**
  Applications need to take care of the timing issues so that audio and video data can be played back continuously with correct timing and synchronization. Applications also need to define standard operations for applications to manage the delivery and present the multimedia data.

This section explains several network protocols needed to support QoS over the Internet for many SM applications. Figure 1.8 shows their relations with other well-known protocols.

1.4.1 RTP

Media transport in many streaming applications is mainly implemented with RTP (Real-time Transport Protocol, RFC 1889) [148], which is an

IP-based transport protocol for audio and video conferences and other multi-participant real-time applications. It is a lightweight protocol without error correction or flow control functionality. Thus, it guarantees neither QoS nor resource reservation along the network path. RTP is also designed to work in conjunction with the auxiliary control protocol, RTCP, to get feedback on quality of data transmission and information about participants in the ongoing session. RTP is primarily designed for multicast of real-time data, but it can be also used in unicast. It can be used for one-way transport such as video-on-demand as well as interactive services such as Internet telephony.

Multimedia applications require appropriate timing in data transmission and playing back. To support real-time SM data transmission, RTP provides timestamping, sequence numbering, and other mechanisms. Through these mechanisms, RTP provides end-to-end transport for real-time data over the datagram network.

- *Timestamping* is the most important information for real-time applications. The sender sets the timestamp when the packet was sampled. The receiver uses the timestamp to reconstruct the original timing. It can be also used to synchronize different streams with timing properties, such as audio and video data in MPEG. However, RTP itself is not responsible for the synchronization. This has to be done at the application level.

- *Sequence numbers* are used to determine the correct order because UDP does not deliver packets in timely order. It is also used for packet loss detection. Notice that in some video formats, when a video frame is split into several RTP packets, all of them can have the same timestamp. So timestamp alone is not enough to put the packets in order.

- The *payload type identifier* specifies the payload format as well as the encoding/compression schemes. Using this identifier, receiving application knows how to interpret and play out the payload data. Example specifications include PCM, MPEG1/MPEG2 audio and video, et al. More payload types can be added by providing a profile and payload format specification. At any given time of transmission, an RTP sender can only send one type of payload, although the payload type may change during transmission, for example, to adjust to network congestion.

- *Source identification* allows the receiving application to know where the data is coming from. For example, in an audio conference, from the source identifier a user could tell who is talking.

RTP is typically run on top of UDP. RTP is primarily designed for multicast, the connection-oriented TCP does not scale well and therefore is not suitable. For real-time data, reliability is not as important as timely delivery. Even more, reliable transmission provided by retransmission as in TCP is not desirable. RTP and RTCP packets are usually transmitted using UDP/IP service.

1.4.2 RTCP

RTCP (RTP Control Protocol) is the control protocol designed to work in conjunction with RTP. In an RTP session, participants periodically send RTCP packets to convey feedback on quality of data delivery and information of membership:

- RR (receiver report): Receiver reports reception quality feedback about data delivery, including the highest packet number received, the number of packets lost, inter-arrival jitter, and timestamps to calculate the round-trip delay between the sender and the receiver.
- SR (sender report): Sender reports a sender information section, providing information on inter-media synchronization, cumulative packet counters, and number of bytes sent.
- SDES (source description items): Information to describe the sources.
- APP (application specific functions): It is now intended for experimental use as new applications and new features are developed.

Using the above messages, RTCP provides the following services to control data transmission with RTP:

- QoS monitoring and congestion control. RTCP provides feedback to an application about the quality of data distribution. Sender can adjust its transmission based on the receiver report feedback. Receivers can know whether a congestion is local, regional or global. Network managers can evaluate the network performance for multicast distribution.
- Source identification. RTCP SDES (source description) packets contain textual information called canonical names as globally unique identifiers of the session participants. They may include a user's name, telephone number, e-mail address and other information.
- Inter-media synchronization. RTCP sender reports contain an indication of real time and the corresponding RTP timestamp. This can be used in inter-media synchronization like lip synchronization in video.

- Control information scaling. RTCP packets are sent periodically among participants. In order to scale up to large multicast groups, RTCP has to prevent the control traffic from overwhelming network resources. RTP limits the control traffic to at most 5% of the overall session traffic. This is enforced by adjusting the RTCP generating rate according to the number of participants.

1.4.3 RTSP

RTSP (Real Time Streaming Protocol, RFC 2326) is a client-server multimedia presentation protocol to enable controlled delivery of streamed multimedia data over IP network. RTSP provides methods to realize commands (play, fast-forward, fast-rewind, pause, stop) similar to the functionality provided by CD players or VCRs. RTSP is an application-level protocol designed to work with lower-level protocols like RTP and RSVP to provide a complete streaming service over the Internet. It can act as a network remote control for multimedia servers and can run over TCP or UDP. RTSP can control either a single or several time-synchronized streams of continuous media.

RTSP provides the following operations:

- Retrieval of media from media server. The client can request a presentation description, and ask the server to setup a session to send the requested data.
- Invitation of a media server to a conference. The media server can be invited to the conference to play back media or to record a presentation.
- Adding media to an existing presentation. The server and the client can notify each other about any additional media becoming available.

In RTSP, each presentation and media stream is identified by an RTSP URL. The overall presentation and the properties of the media are defined in a presentation description file, which may include the encoding, language, RTSP URLs, destination address, port, and other parameters. The presentation description file can be obtained by the client using HTTP, e-mail or other means. RTSP aims to provide the same services for streamed audio and video just as HTTP does for text and graphics. But RTSP differs from HTTP in several aspects. First, while HTTP is a stateless protocol, an RTSP server has to maintain "session states" in order to correlate RTSP requests with a stream. Second, HTTP is basically an asymmetric protocol where the client issues requests and the server responds, but in RTSP both the media server and the client can issue requests. For example, the server can issue a request to set playback parameters of a stream.

1.4.4 RSVP

RSVP (Resource reSerVation Protocol) is the network control protocol that allows the data receiver to request a special end-to-end quality of service for its data flows. Thus, real-time applications can use RSVP to reserve necessary resources at routers along the transmission paths so that the requested bandwidth can be available when the transmission actually takes place. RSVP is a main component of the future Integrated Services Internet which can provide both best-effort and real-time service.

RSVP is used to set up reservations for network resources. When an application in a host (the data stream receiver) requests a specific quality of service (QoS) for its data stream, it uses RSVP to deliver its request to routers along the data stream paths. RSVP is responsible for the negotiation of connection parameters with these routers. If the reservation is set up, RSVP is also responsible for maintaining router and host states to provide the requested service. Each node capable of resource reservation has several local procedures for reservation setup and enforcement. Policy control determines whether the user has administrative permission to make the reservation (authentication, access control and accounting for reservation). Admission control keeps track of the system resources and determines whether the node has sufficient resources to supply the requested QoS.

RSVP is also designed to utilize the robustness of current Internet routing algorithms. RSVP does not perform its own routing; instead it uses underlying routing protocols to determine where it should carry reservation requests. As routing changes paths to adapt to topology changes, RSVP adapts its reservation to the new paths wherever reservations are in place.

1.5 Outline of the Book

The book is organized into the following chapters:

Chapter 2 concentrates on single disk platform SM server design, by presenting different techniques in support of a hiccup-free display. Even though a single disk server is not practical in many applications, however, it presents the fundamental concepts and techniques in designing SM servers.

Chapter 3 extends the discussion to multiple disk platform SM server design. It introduces possible design approaches: 1) cycle-based scheduling with round-robin data placement approach and 2) deadline-driven scheduling with unconstrained data placement approach. In addition, it presents a number of optimization techniques and an online reconfiguration process.

Chapter 4 deals exclusively with deadline-driven scheduling and uncon-

strained media placement approach. It quantifies the hiccup probability of this approach and presents techniques to reduce this probability.

Chapter 5 extends the discussion to a heterogenous disk platform. It presents a number of techniques that take advantage of the rapid development in disk storage devices.

Chapter 6 deals with fault-tolerance issues in SM servers. It presents techniques for homogenous disk platforms, then it extends the discussion to a heterogenous disk platform.

Chapter 7 is devoted to hierarchical storage system design for SM servers. It presents a pipelining technique to ensure the continuous display from tape jukeboxes and data placement techniques to improve the access time to tape resident SM objects.

In Chapter 8 and Chapter 9, the concept of distributed SM servers are introduced. Chapter 8 presents RedHi, a distributed SM server and its network components. Chapter 9 presents a super-streaming mechanism that takes advantage of a distributed system.

Chapter 10 presents a case study on the design of a second generation SM server, namely Yima, while Appendix A provides instructions on installing a personal version of Yima.