Streaming Video over the Internet: Approaches and Directions

Dapeng Wu, *Student Member, IEEE*, Yiwei Thomas Hou, *Member, IEEE*, Wenwu Zhu, *Member, IEEE*, Ya-Qin Zhang, *Fellow, IEEE*, and Jon M. Peha, *Senior Member, IEEE*

***Abstract*—Due to the explosive growth of the Internet and increasing demand for multimedia information on the web, streaming video over the Internet has received tremendous attention from academia and industry. Transmission of real-time video typically has bandwidth, delay, and loss requirements. However, the current best-effort Internet does not offer any quality of service (QoS) guarantees to streaming video. Furthermore, for video multicast, it is difficult to achieve both efficiency and flexibility. Thus, Internet streaming video poses many challenges. To address these challenges, extensive research has been conducted. This special issue is aimed at dissemination of the contributions in the field of streaming video over the Internet. To introduce this special issue with the necessary background and provide an integral view on this field, we cover six key areas of streaming video. Specifically, we cover video compression, application-layer QoS control, continuous media distribution services, streaming servers, media synchronization mechanisms, and protocols for streaming media. For each area, we address the particular issues and review major approaches and mechanisms. We also discuss the tradeoffs of the approaches and point out future research directions.**

***Index Terms*—Application-layer QoS control, continuous media distribution services, Internet, protocol, streaming video, streaming server, synchronization, video compression.**

I. Introduction

RECENT advances in computing technology, compression technology, high-bandwidth storage devices, and high-speed networks have made it feasible to provide real-time multimedia services over the Internet. Real-time multimedia, as the name implies, has timing constraints. For example, audio and video data must be played out continuously. If the data does not arrive in time, the playout process will pause, which is annoying to human ears and eyes.

Real-time transport of live video or stored video is the predominant part of real-time multimedia. In this paper, we are concerned with video streaming, which refers to real-time transmission of *stored video*. There are two modes for transmission of stored video over the Internet, namely the download mode and the streaming mode (i.e., video streaming). In the download mode, a user downloads the entire video file and then plays back the video file. However, full file transfer in the download mode usually suffers long and perhaps unacceptable transfer time. In contrast, in the streaming mode, the video content need not be downloaded in full, but is being played out while parts of the content are being received and decoded. Due to its real-time nature, video streaming typically has bandwidth, delay and loss requirements. However, the current best-effort Internet does not offer any quality of service (QoS) guarantees to streaming video over the Internet. In addition, for multicast, it is difficult to efficiently support multicast video while providing service flexibility to meet a wide range of QoS requirements from the users. Thus, designing mechanisms and protocols for Internet streaming video poses many challenges.

To address these challenges, extensive research has been conducted. This special issue is aimed at dissemination of the contributions in the field of streaming video over the Internet. To introduce this special issue with the necessary background and give the reader a complete picture of this field, we cover six key areas of streaming video, namely: video compression, application-layer QoS control, continuous media distribution services, streaming servers, media synchronization mechanisms, and protocols for streaming media. Each of the six areas is a basic building block, with which an architecture for streaming video can be built. The relations among the six basic building blocks can be illustrated in Fig. 1.

Fig. 1 shows an architecture for video streaming. In Fig. 1, raw video and audio data are pre-compressed by *video compression* and audio compression algorithms and then saved in storage devices. Upon the client's request, a *streaming server* retrieves compressed video/audio data from storage devices and then the *application-layer QoS control* module adapts the video/audio bit-streams according to the network status and QoS requirements. After the adaptation, the transport *protocols* packetize the compressed bit-streams and send the video/audio packets to the Internet. Packets may be dropped or experience excessive delay inside the Internet due to congestion. To improve the quality of video/audio transmission, *continuous media distribution services* (e.g., caching) are deployed in the Internet. For packets that are successfully delivered to the receiver, they first pass through the transport layers and are then processed by the application layer before being decoded at the video/audio decoder. To achieve synchronization between video and audio presentations, *media synchronization mechanisms* are required. From Fig. 1, it can be seen that the six areas are closely related and they are coherent constituents of the video streaming architecture. We briefly describe the six areas as follows.

Manuscript received June 15, 2000; revised December 7, 2000. This paper was recommended by Guest Editor A. Luthra.

D. Wu and J. M. Peha are with the Deptartment of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

Y. T. Hou is with the Fujitsu Laboratories of America, Sunnyvale, CA 94085 USA (e-mail: thou@fla.fujitsu.com).

W. Zhu and Y.-Q. Zhang are with the Microsoft Research China, Haidian District, Beijing 100080, China.

Publisher Item Identifier S 1051-8215(01)02237-6.

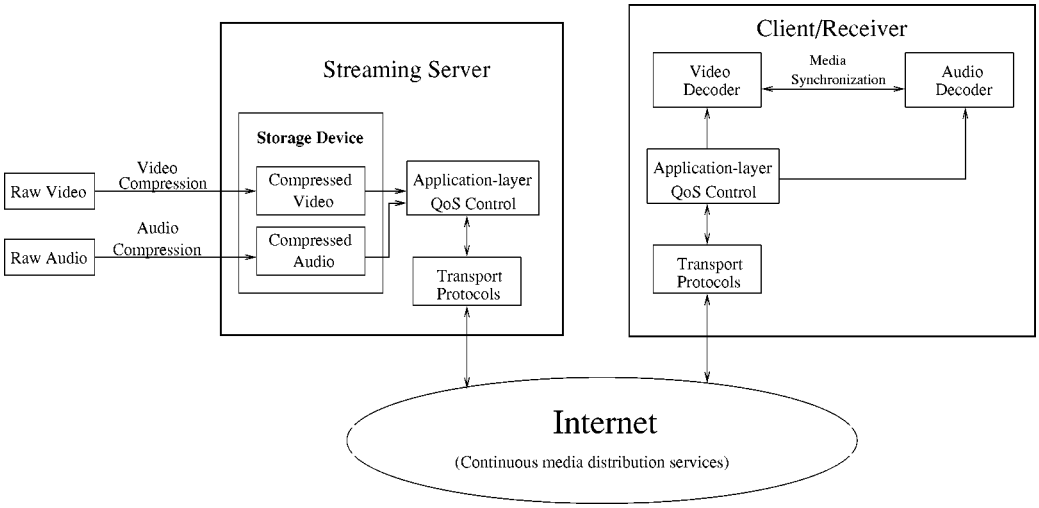

Fig. 1. Architecture for video streaming.

1) *Video compression:* Raw video must be compressed before transmission to achieve efficiency. Video compression schemes can be classified into two categories: scalable and nonscalable video coding. Since scalable video is capable of gracefully coping with the bandwidth fluctuations in the Internet [43], we are primarily concerned with scalable video coding techniques. We will also discuss the requirements imposed by streaming applications on the video encoder and decoder.
2) *Application-layer QoS control:* To cope with varying network conditions and different presentation quality requested by the users, various application-layer QoS control techniques have been proposed [17], [60], [66], [79]. The application-layer techniques include congestion control and error control. Their respective functions are as follows. Congestion control is employed to prevent packet loss and reduce delay. Error control, on the other hand, is to improve video presentation quality in the presence of packet loss. Error control mechanisms include forward error correction (FEC), retransmission, error-resilient encoding, and error concealment.
3) *Continuous media distribution services:* In order to provide quality multimedia presentations, adequate network support is crucial. This is because network support can reduce transport delay and packet loss ratio. Built on top of the Internet (IP protocol), continuous media distribution services are able to achieve QoS and efficiency for streaming video/audio over the best-effort Internet. Continuous media distribution services include network filtering, application-level multicast, and content replication.
4) *Streaming servers:* Streaming servers play a key role in providing streaming services. To offer quality streaming services, streaming servers are required to process multimedia data under timing constraints and support interactive control operations such as pause/resume, fast forward, and fast backward. Furthermore, streaming servers need to retrieve media components in a synchronous fashion. A streaming server typically consists of three subsystems, namely, a communicator (e.g., transport protocols), an operating system, and a storage system.
5) *Media synchronization mechanisms:* Media synchronization is a major feature that distinguishes multimedia applications from other traditional data applications. With media synchronization mechanisms, the application at the receiver side can present various media streams in the same way as they were originally captured. An example of media synchronization is that the movements of a speaker's lips match the played-out audio.
6) *Protocols for streaming media:* Protocols are designed and standardized for communication between clients and streaming servers. Protocols for streaming media provide such services as network addressing, transport, and session control. According to their functionalities, the protocols can be classified into three categories: network-layer protocol such as Internet protocol (IP), transport protocol such as user datagram protocol (UDP), and session control protocol such as real-time streaming protocol (RTSP).

The remainder of this paper is devoted to the exposition of the above six areas. Section II discusses video compression techniques. In Section III, we present application-layer QoS control mechanisms for streaming video. Section IV describes continuous media distribution services. In Section V, we discuss key issues on design of streaming servers. Section VI presents various media synchronization mechanisms. In Section VII, we overview key protocols for streaming video. Section VIII summarizes this paper and points out future research directions.

II. Video Compression

Since raw video consumes a large amount of bandwidth, compression is usually employed to achieve transmission efficiency. In this section, we discuss various compression approaches and requirements imposed by streaming applications on the video encoder and decoder.

Basically, video compression schemes can be classified into two approaches: scalable and nonscalable video coding. For

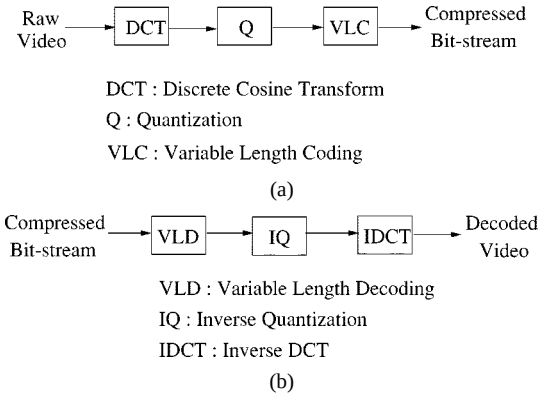

Fig. 2. (a) Nonscalable video encoder. (b) Nonscalable video decoder.

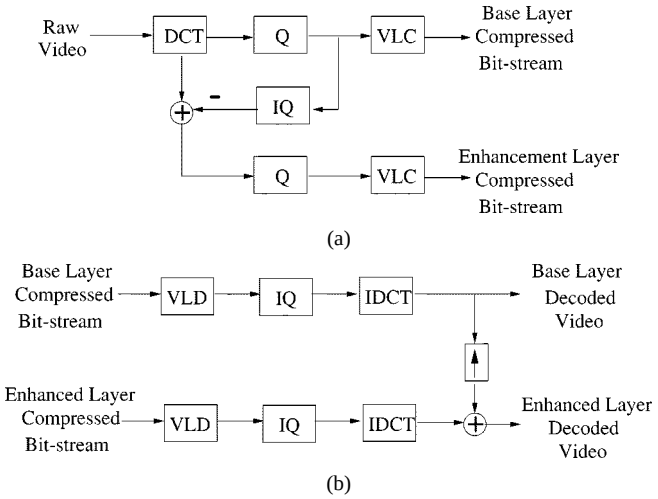

Fig. 3. (a) SNR-scalable encoder. (b) SNR-scalable decoder.

simplicity, we will only show the encoder and decoder in intra-mode[1] and only use discrete cosine transform (DCT). For wavelet-based scalable video coding, please refer to [36], [62], [73], [74] and references therein.

A nonscalable video encoder [see Fig. 2(a)] generates one compressed bit-stream. In contrast, a scalable video encoder compresses a raw video sequence into multiple substreams [see Fig. 3(a)]. One of the compressed substreams is the base substream, which can be independently decoded and provide coarse visual quality. Other compressed substreams are enhancement substreams, which can only be decoded together with the base substream and can provide better visual quality. The complete bit-stream (i.e., combination of all the substreams) provides the highest quality. Specifically, compared with decoding the complete bit-stream [Fig. 4(a)], decoding the base substream or multiple substreams produces pictures with degraded quality [Fig. 4(b)], or a smaller image size [Fig. 4(c)], or a lower frame rate [Fig. 4(d)]. The scalabilities of quality, image sizes, or frame rates, are called SNR, spatial, or temporal scalability, respectively. These three scalabilities are basic scalable mechanisms. There can be combinations of the basic mechanisms, such as spatiotemporal scalability [27].

[1]Intra-mode coding refers to coding a video unit (e.g., a macroblock) without any reference to previously coded data.

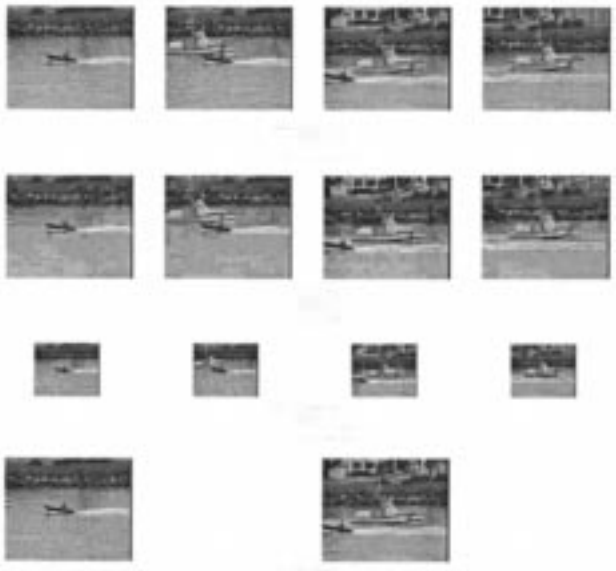

Fig. 4. Scalable video: (a) video frames reconstructed from the complete bit-stream; (b) video frames with degraded quality; (c) video frames with a smaller image size; and (d) video frames with a lower frame rate.

To provide more flexibility in meeting different demands of streaming (e.g., different access link bandwidths and different latency requirements), a new scalable coding mechanism, called fine granularity scalability (FGS), was proposed to MPEG-4 [37]–[39]. As shown in Fig. 5(a), an FGS encoder compresses a raw video sequence into two substreams, i.e., a base layer bit-stream and an enhancement bit-stream. Different from an SNR-scalable encoder, an FGS encoder uses bitplane coding[2] to represent the enhancement stream (see Fig. 6). With bitplane coding, an FGS encoder is capable of achieving continuous rate control for the enhancement stream. This is because the enhancement bit stream can be truncated anywhere to achieve the target bit-rate.

A variation of FGS is progressive fine granularity scalability (PFGS) [72]. PFGS shares the good features of FGS, such as fine granularity bit-rate scalability and error resilience. Unlike FGS, which only has two layers, PFGS could have more than two layers. The essential difference between FGS and PFGS is that FGS only uses the base layer as a reference for motion prediction while PFGS uses multiple layers as references to reduce the prediction error, resulting in higher coding efficiency.

Next, we describe various requirements imposed by streaming applications on the video encoder and decoder and briefly discuss some techniques that address these requirements.

1) *Bandwidth:* To achieve acceptable perceptual quality, a streaming application typically has minimum bandwidth requirement. However, the current Internet does not provide bandwidth reservation to support this requirement. In addition, it is desirable for video streaming applications to employ congestion control to avoid congestion,

[2]Bitplane coding uses embedded representations [60]. For example, a DCT coefficient can be represented by 7 bits (i.e., its value ranges from 0 to 127). There are 64 DCT coefficients. Each DCT coefficient has a most significant bit (MSB) and all the MSBs from the 64 DCT coefficients form Bitplane 0 (see Fig. 6). Similarly, all the second most-significant-bits form Bitplane 1.

which happens when the network is heavily loaded. For video streaming, congestion control takes the form of rate control; that is, adapting the sending rate to the available bandwidth in the network. Compared with nonscalable video, scalable video is more adaptable to the varying available bandwidth in the network.

2) *Delay:* Streaming video requires bounded end-to-end delay so that packets can arrive at the receiver in time to be decoded and displayed. If a video packet does not arrive in time, the playout process will pause, which is annoying to human eyes. A video packet that arrives beyond its delay bound (e.g. its playout time) is useless and can be regarded as lost. Since the Internet introduces time-varying delay, to provide continuous playout, a buffer at the receiver is usually introduced before decoding [13].
3) *Loss:* Packet loss is inevitable in the Internet and can damage pictures, which is displeasing to human eyes. Thus, it is desirable that a video stream be robust to packet loss. Multiple description coding is such a compression technique to deal with packet loss [68].
4) *Video-cassette-recorder (VCR) like function:* Some streaming applications require VCR-like functions such as stop, pause/resume, fast forward, fast backward, and random access. Lin *et al.* [40] proposed a dual-bit-stream least-cost scheme to efficiently provide VCR-like functionality for MPEG video streaming.
5) *Decoding complexity:* Some devices such as cellular phones and personal digital assistants (PDAs) require low power consumption. Therefore, streaming video applications running on these devices must be simple. In particular, low decoding complexity is desirable. To address this issue, Lin *et al.* [40] employed a least-cost scheme to reduce decoding complexity.

We have discussed various compression mechanisms and requirements imposed by streaming applications on the video encoder and decoder. Next, we present the application-layer QoS control mechanisms, which adapt the video bit-streams according to the network status and QoS requirements.

III. Application-Layer QoS Control

The objective of application-layer QoS control is to avoid congestion and maximize video quality in the presence of packet loss. The application-layer QoS control techniques include congestion control and error control. These techniques are employed by the end systems and do not require any QoS support from the network.

We organize the rest of this section as follows. In Section III-A, we survey the approaches for congestion control. Section III-B describes mechanisms for error control.

A. Congestion Control

Bursty loss and excessive delay have a devastating effect on video presentation quality, and they are usually caused by network congestion. Thus, congestion-control mechanisms at end systems are necessary to help reducing packet loss and delay.

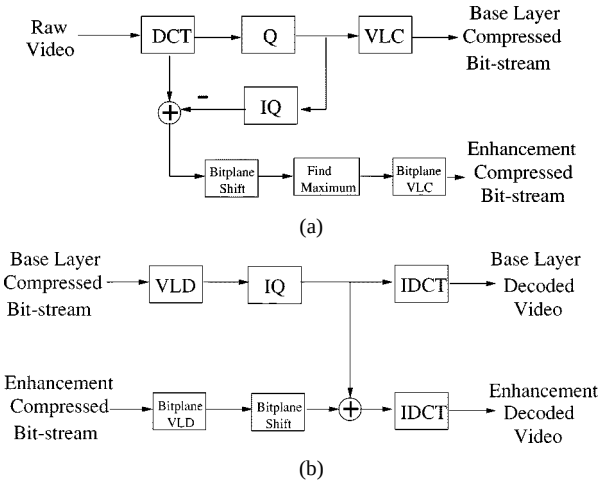

Fig. 5. (a) FGS encoder. (b) FGS decoder.

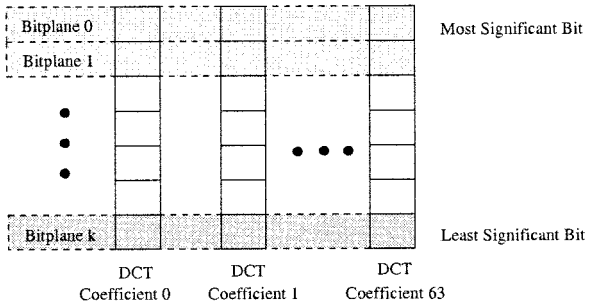

Fig. 6. Bitplanes of enhancement DCT coefficients.

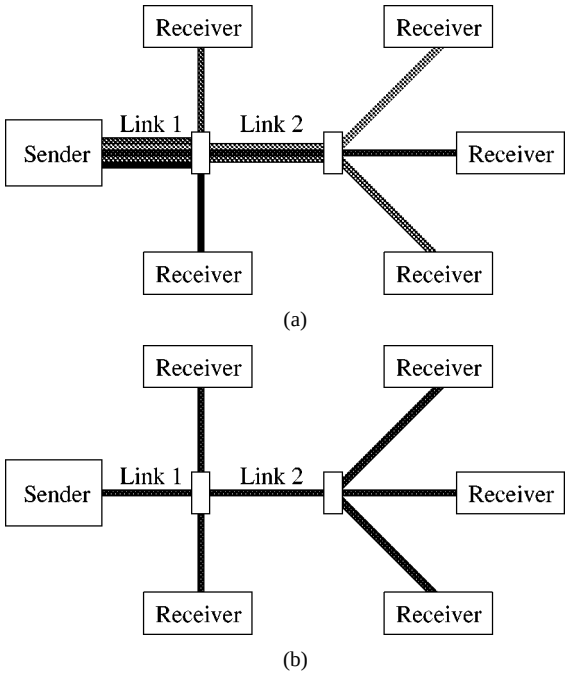

Fig. 7. (a) Unicast video distribution using multiple point-to-point connections. (b) Multicast video distribution using point-to-multipoint transmission.

Typically, for streaming video, congestion control takes the form of rate control [71]. Rate control attempts to minimize the possibility of network congestion by matching the rate of the video stream to the available network bandwidth. Next, we

present various approaches for rate control in Section III-A-1 and describe an associated technique called rate shaping in Section III-A-2.

1) Rate Control: Rate control is a technique used to determine the sending rate of video traffic based on the estimated available bandwidth in the network. Existing rate-control schemes can be classified into three categories: source-based, receiver-based, and hybrid rate control, which are presented as follows.

Source-Based Rate Control: Under the source-based rate control, the sender is responsible for adapting the video transmission rate. Typically, feedback is employed by source-based rate-control mechanisms. Based upon the feedback information about the network, the sender could regulate the rate of the video stream. The source-based rate control can be applied to both unicast [see Fig. 7(a)] [70] and multicast [see Fig. 7(b)] [4].

For unicast video, existing source-based rate-control mechanisms follow two approaches: probe-based and model-based approach [71].

The probe-based approach is based on probing experiments. Specifically, the source probes for the available network bandwidth by adjusting the sending rate in a way that could maintain the packet loss ratio p below a certain threshold $P_{\rm th}$ [70]. There are two ways to adjust the sending rate: 1) additive increase and multiplicative decrease [70] and 2) multiplicative increase and multiplicative decrease [64].

The model-based approach is based on a throughput model of a transmission control protocol (TCP) connection. Specifically, the throughput of a TCP connection can be characterized by the following formula [20]:

$$\lambda = \frac{1.22 \times MTU}{RTT \times \sqrt{p}} \tag{1}$$

where

- λ throughput of a TCP connection;
- *MTU* (maximum transit unit) is the packet size used by the connection;
- *RTT* round-trip time for the connection;
- p packet loss ratio experienced by the connection.

Under the model-based rate control, (1) is used to determine the sending rate of the video stream. Thus, the video connection could avoid congestion in a similar way to that of TCP and it can compete fairly with TCP flows. For this reason, the model-based rate control is also called "TCP-friendly" rate control [20].

For multicast under the source-based rate control, the sender uses a single channel to transport video to the receivers [see Fig. 7(b)]. Such multicast is called "single-channel multicast". For single-channel multicast, only the probe-based rate control can be employed [4].

Single-channel multicast is efficient since all the receivers share one channel. However, single-channel multicast is unable to provide flexible services to meet the different demands from receivers with various access link bandwidth. In contrast, if multicast video were to be delivered through individual unicast streams, the bandwidth efficiency is low, but the services could be differentiated since each receiver can negotiate the parameters of the services with the source. Unicast and single-channel multicast are two extreme cases shown in Fig. 8. To achieve good tradeoff between bandwidth efficiency and service flexibility for multicast video, receiver-based and hybrid rate-control were proposed.

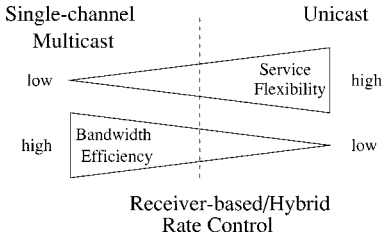

Fig. 8. Tradeoff between efficiency and flexibility.

Receiver-Based Rate-Control: Under the receiver-based rate control, the receivers regulate the receiving rate of video streams by adding/dropping channels while the sender does not participate in rate control [71]. Typically, receiver-based rate control is used in multicasting scalable video, where there are several layers in the scalable video and each layer corresponds to one channel in the multicast tree.

Similar to the source-based rate control, the existing receiver-based rate-control mechanisms follow two approaches: probe-based and model-based approach. The basic probe-based rate control consists of two parts [43].

1) When no congestion is detected, a receiver probes for the available bandwidth by joining a layer/channel, resulting in an increase of its receiving rate. If no congestion is detected after the joining, the join-experiment is successful. Otherwise, the receiver drops the newly added layer.
2) When congestion is detected, a receiver drops a layer (i.e., leaves a channel), resulting in a reduction of its receiving rate.

Unlike the probe-based approach, which implicitly estimates the available network bandwidth through probing experiments, the model-based approach uses explicit estimation for the available network bandwidth. The model-based approach is also based on (1).

Hybrid Rate-Control: Under the hybrid rate-control, the receivers regulate the receiving rate of video streams by adding/dropping channels, while the sender also adjusts the transmission rate of each channel based on feedback from the receivers. Examples of hybrid rate control include the destination set grouping [10] and a layered multicast scheme [29].

2) Rate Shaping: The objective of rate shaping is to match the rate of a pre-compressed video bitstream to the target rate constraint [17]. A rate shaper (or filter), which performs rate shaping, is required for the source-based rate control (see Fig. 9). This is because the stored video may be pre-compressed at a certain rate, which may not match the available bandwidth in the network.

There are many types of filters, such as codec filter, frame-dropping filter, layer-dropping filter, frequency filter, and re-quantization filter [75], which are described as follows.

A *codec filter* is to decompress and compress a video stream. It is commonly used to perform transcoding between different compression schemes. Depending on the compression scheme

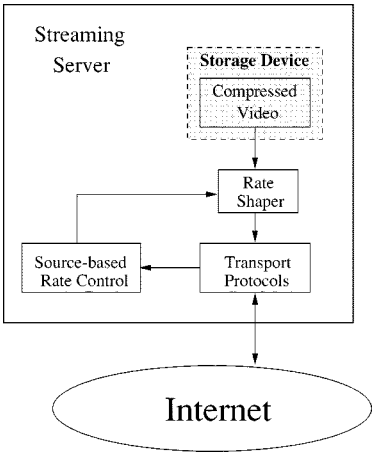

Fig. 9. Architecture for source-based rate control.

used, transcoding could be simplified without full decompression and recompression.

A *frame-dropping filter* can distinguish the frame types (e.g., I-, P-, and B-frame in MPEG) and drop frames according to importance. For example, the dropping order would be first B-frames, then P-frames, and finally I-frames. The frame-dropping filter is used to reduce the data rate of a video stream by discarding a number of frames and transmitting the remaining frames at a lower rate. The frame-dropping filter could be used at the source [80] or used in the network (see Section IV-A).

A *layer-dropping filter* can distinguish the layers and drop layers according to importance. The dropping order is from the highest enhancement layer down to the base layer.

A *frequency filter* performs operations on the compression layer. Specifically, it operates in the frequency domain (i.e., DCT coefficients). Frequency filtering mechanisms include low-pass filtering, color reduction filtering and color-to-monochrome filtering. Low-pass filtering is to discard the DCT coefficients of the higher frequencies. A color-reduction filter performs the same operation as a low-pass filter except that it only operates on the chrominance information in the video stream. A color-to-monochrome filter removes all color information from the video stream. In MPEG, this is done by replacing each chrominance block with an empty block. Unlike the frame-dropping filter, the frequency filter reduces the bandwidth without affecting the frame rate. Its cost is reduction in presentation quality of the resulting frame.

A *re-quantization filter* performs operations on the compression layer (i.e., DCT coefficients). The filter first extracts the DCT coefficients from the compressed video stream through techniques like dequantization, then it re-quantizes the DCT coefficients with a larger quantization step, resulting in rate reduction.

In sum, the purpose of congestion control is to avoid congestion. On the other hand, packet loss is inevitable in the Internet and may have significant impact on perceptual quality. This prompts the need to design mechanisms to maximize video presentation quality in the presence of packet loss. Error control is such a mechanism, which will be presented next.

B. Error Control

Error control mechanisms include FEC, retransmission, error-resilient encoding and error concealment, which are described in Sections III-B-1to III-B-4, respectively.

1) FEC: The principle of FEC is to add redundant information so that original message can be reconstructed in the presence of packet loss. Based on the kind of redundant information to be added, we classify existing FEC schemes into three categories: channel coding, source coding-based FEC, and joint source/channel coding [71].

For Internet applications, channel coding is typically used in terms of block codes. Specifically, a video stream is first chopped into segments, each of which is packetized into k packets; then, for each segment, a block code (e.g., Tornado code [1]) is applied to the k packets to generate an n-packet block, where $n > k$. To perfectly recover a segment, a user only needs to receive any k packets in the n-packet block.

Source coding-based FEC (SFEC) is a recently devised variant of FEC for Internet video [5]. Like channel coding, SFEC also adds redundant information to recover from loss. For example, the nth packet contains the nth group-of-blocks (GOB) and redundant information about the $(n-1)$th GOB, which is a compressed version of the $(n-1)$th GOB with larger quantizer.

Joint source/channel coding is an approach to optimal rate allocation between source coding and channel coding [71].

2) Delay-Constrained Retransmission: Retransmission is usually dismissed as a method to recover lost packets in real-time video since a retransmitted packet may miss its play-out time. However, if the one-way trip time is short with respect to the maximum allowable delay, a retransmission-based approach (called delay-constrained retransmission) is a viable option for error control.

For unicast, the receiver can perform the following delay-constrained retransmission scheme.

When the receiver detects the loss of packet N

$$\text{if } (T_c + RTT + D_s < T_d(N))$$
$$\text{send the request for packet } N \text{ to the sender}$$

where

T_c	current time;
RTT	estimated round-trip time;
D_s	a slack term;
$T_d(N)$	time when packet N is scheduled for display.

The slack term D_s may include tolerance of error in estimating *RTT*, the sender's response time, and the receiver's decoding delay. The timing diagram for receiver-based control is shown in Fig. 10, where D_s is only the receiver's decoding delay. It is clear that the objective of the delay-constrained retransmission is to suppress requests of retransmissions that will not arrive in time for display.

3) Error-Resilient Encoding: The objective of error-resilient encoding is to enhance robustness of compressed video to packet loss. The standardized error-resilient encoding schemes include resynchronization marking, data partitioning,

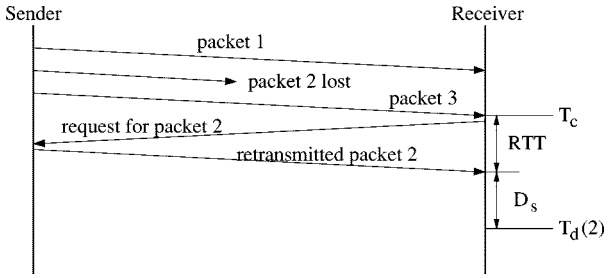

Fig. 10. Timing diagram for receiver-based control.

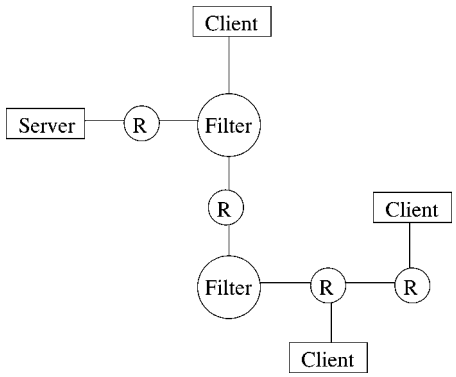

Fig. 11. Filters placed inside the network.

and data recovery [33]. However, resynchronization marking, data partitioning, and data recovery are targeted at error-prone environments like wireless channels and may not be applicable to the Internet environment. For video transmission over the Internet, the boundary of a packet already provides a synchronization point in the variable-length coded bit-stream at the receiver side. On the other hand, since a packet loss may cause the loss of all the motion data and its associated shape/texture data, mechanisms such as resynchronization marking, data partitioning, and data recovery may not be useful for Internet video applications. Therefore, we will not present the standardized error-resilient tools. Instead, we present multiple description coding (MDC) [47], [68], which is promising for robust Internet video transmission.

With MDC, a raw video sequence is compressed into multiple streams (referred to as descriptions) as follows: each description provides acceptable visual quality; more combined descriptions provide a better visual quality. The advantages of MDC are:

- *robustness to loss:* even if a receiver gets only one description (other descriptions being lost), it can still reconstruct video with acceptable quality;
- *enhanced quality:* if a receiver gets multiple descriptions, it can combine them together to produce a better reconstruction than that produced from any one of them.

However, the advantages come with a cost. To make each description provide acceptable visual quality, each description must carry sufficient information about the original video. This will reduce the compression efficiency compared to conventional single description coding (SDC). In addition, although more combined descriptions provide a better visual quality, a certain degree of correlation between the multiple descriptions has to be embedded in each description, resulting in further reduction of the compression efficiency. Further investigation is needed to find a good tradeoff between the compression efficiency and the reconstruction quality from one description.

4) Error Concealment: Error-resilient encoding is executed by the source to enhance robustness of compressed video before packet loss actually happens (this is called preventive approach). On the other hand, error concealment is performed by the receiver when packet loss has already occurred (this is called reactive approach). Specifically, error concealment is employed by the receiver to conceal the lost data and make the presentation less displeasing to human eyes.

There are two basic approaches for error concealment, namely, spatial and temporal interpolation. In spatial interpolation, missing pixel values are reconstructed using neighboring spatial information. In temporal interpolation, the lost data is reconstructed from data in the previous frames. Typically, spatial interpolation is used to reconstruct the missing data in intra-coded frames, while temporal interpolation is used to reconstruct the missing data in inter-coded frames.

In recent years, numerous error-concealment schemes have been proposed in the literature (refer to [69] for a good survey). Examples include maximally smooth recovery [67], projection onto convex sets [58], and various motion vector and coding mode recovery methods such as motion compensated temporal prediction [25].

So far, we have reviewed various application-layer QoS control techniques. These techniques are employed by the end systems and do not require any QoS support from the network. If the network is able to support QoS for video streaming, the performance can be further enhanced. In the next section, we present network QoS support mechanisms, which are based on the best-effort Internet.

IV. Continuous Media Distribution Services

In order to provide quality multimedia presentations, adequate support from the network is critical. This is because network support can reduce transport delay and packet loss ratio. Streaming video and audio are classified as continuous media because they consist of a sequence of media quanta (such as audio samples or video frames), which convey meaningful information only when presented in time. Built on top of the Internet (IP protocol), continuous media distribution services are designed with the aim of providing QoS and achieving efficiency for streaming video/audio over the best-effort Internet. Continuous media distribution services include network filtering, application-level multicast, and content replication, which are presented in Sections IV-A to IV-C, respectively.

A. Network Filtering

As a congestion-control technique, network filtering aims to maximize video quality during network congestion. As described in Section III-A-2, the filter at the video server

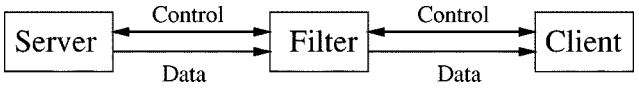

Fig. 12. System model of network filtering.

can adapt the rate of video streams according to the network congestion status. However, the video server may be too busy to handle the computation required to adapt each unicast video stream. Hence, the service providers may like to place filters in the network [32]. Fig. 11 illustrates an example of placing filters in the network. The nodes labeled "R" denote routers that have no knowledge of the format of the media streams and may randomly discard packets. The "Filter" nodes receive the client's requests and adapt the stream sent by the server accordingly. This solution allows the service provider to place filters on the nodes that connect to network bottlenecks. Furthermore, multiple filters can be placed along the path from a server to a client.

To illustrate the operations of filters, a system model is depicted in Fig. 12 [32]. The model consists of the server, the client, at least one filter, and two virtual channels between them. Of the two virtual channels, one is for control and the other is for data. The same channels exist between any pair of filters. The control channel is bi-directional, which can be realized by TCP connections. The model shown in Fig. 12 allows the client to communicate with only one host (the last filter), which will either forward the requests or act upon them. The operations of a filter on the data plane include: 1) receiving video stream from server or previous filter and 2) sending video to client or next filter at the target rate. The operations of a filter on the control plane include: 1) receiving requests from client or next filter; 2) acting upon requests; and 3) forwarding the requests to its previous filter.

Typically, frame-dropping filters (see Section III-A-2) are used as network filters. The receiver can change the bandwidth of the media stream by sending requests to the filter to increase or decrease the frame dropping rate. To facilitate decisions on whether the filter should increase or decrease the bandwidth, the receiver continuously measures the packet loss ratio p. Based on the packet loss ratio, a rate-control mechanism can be designed as follows [32]. If the packet loss ratio is higher than a threshold α, the client will ask the filter to increase the frame dropping rate. If the packet loss ratio is less than another threshold β ($\beta < \alpha$), the receiver will ask the filter to reduce the frame dropping rate.

The advantages of using frame-dropping filters inside the network include the following.

1) *Improved video quality:* For example, when a video stream flows from an upstream link with larger available bandwidth to a downstream link with smaller available bandwidth, use of a frame-dropping filter at the connection point (between the upstream link and the downstream link) could help improve the video quality. This is because the filter understands the format of the media stream and can drop packets in a way that gracefully degrades the stream's quality instead of corrupting the flow outright.
2) *Bandwidth efficiency:* This is because the filtering can help to save network resources by discarding those frames that are late.

B. Application-Level Multicast

The Internet's original design, while well suited for point-to-point applications like e-mail, file transfer, and Web browsing, fails to effectively support large-scale content delivery like streaming-media multicast. In an attempt to address this shortcoming, a technology called "IP multicast" was proposed [14]. As an extension to the IP layer, IP multicast is capable of providing efficient multipoint packet delivery. To be specific, the efficiency is achieved by having one and only one copy of the original IP packet (sent by the multicast source) be transported along any physical path in the IP multicast tree. However, with a decade of research and development, there are still many barriers in deploying IP multicast. These problems include scalability, network management, deployment, and support for higher layer functionality (e.g., error, flow, and congestion control). To address these issues, an application-level multicast mechanism was proposed [19].

The application-level multicast is aimed at building a multicast service on top of the Internet. It enables independent content delivery service providers (CSPs), Internet service providers (ISPs), or enterprises to build their own Internet multicast networks and interconnect them into larger, world-wide "media multicast networks." That is, the media multicast networks could support "peering relationships" at the application level or the streaming-media/content layer, where "content backbones" interconnect service providers. Hence, much as the Internet is built from an interconnection of networks enabled through IP-level peering relationships among ISPs, the media multicast networks can be built from an interconnection of content-distribution networks enabled through application-level peering relationships among various sorts of service providers, e.g., traditional ISPs, CSPs, and application service providers (ASPs).

We briefly describe the operation of the media multicast networks as follows. In the media multicast networks, each multicast-capable node (called MediaBridge [19]) performs routing at the application layer. In addition, each MediaBridge is interconnected with one or more neighboring MediaBridge through explicit configuration, which defines the application-level overlay topology. Collectively, the MediaBridges in a media multicast network employ a distributed application-level multicast routing algorithm to determine the optimal virtual paths for propagating content throughout the network. When the underlying network fails or becomes overly congested, the media multicast network automatically and dynamically re-routes content via alternate paths according to application-level routing policies. In addition, MediaBridges dynamically subscribe to multicast content when and only when a downstream client requests it. This capability ensures that one and only one copy of the multicast content flows across any physical or virtual path independent of the number of downstream clients, resulting in savings of network bandwidth.

The advantage of the application-level multicast is that it breaks the barriers such as scalability, network management,

support for congestion control, which have prevented ISPs from establishing "IP multicast" peering arrangements.

C. Content Replication

An important technique for improving scalability of the media delivery system is content/media replication. The content replication takes two forms, namely, caching and mirroring, which are deployed by publishers, CSPs and ISPs. Both caching and mirroring seek to place content closer to the clients and both share the following advantages:

1) reduced bandwidth consumption on network links;
2) reduced load on streaming servers;
3) reduced latency for clients;
4) increased availability.

Mirroring is to place copies of the original multimedia files on other machines scattered around the Internet. That is, the original multimedia files are stored on the main server while copies of the original multimedia files are placed on the duplicate servers. In this way, clients can retrieve multimedia data from the nearest duplicate server, which gives the clients the best performance (e.g., lowest latency). Mirroring has some disadvantages. Currently, mechanisms for establishing dedicated mirrors are expensive, *ad hoc*, and slow. In addition, establishing a mirror on an existing server, while cheaper, is still an *ad hoc* and administratively complex process. Finally, there is no standard way to make scripts and server setup easily transferable from one server to another.

Caching, which is based on the belief that different clients will load many of the same contents, makes local copies of contents that the clients retrieve. Typically, clients in a single organization retrieve all contents from a single local machine, called a cache. The cache retrieves a video file from the origin server, storing a copy locally and then passing it on to the client who requests it. If a client asks for a video file which the cache has already stored, the cache will return the local copy rather than going all the way to the origin server where the video file resides. In addition, cache sharing and cache hierarchies allow each cache to access files stored at other caches so that the load on the origin server can be reduced and network bottlenecks can be alleviated [8], [18].

Most of the techniques for caching are targeted at generic web objects. Some recent work demonstrated that caching strategies that are specific to particular types of objects can help improve the overall performance [44]. For this reason, many efforts have been contributed along this direction [49], [54], [76], [81]. A trivial extension of caching techniques to video is to store complete video sequences in the cache. However, such an approach may not be applicable due to the large volume of video data and possibly limited cache space on a proxy server. Instead, it was shown that even a few cached frames can contribute to significant improvement in performance [44]. Miao and Ortega [44] proposed two video-caching strategies, initial caching and selective caching, which store part of the video stream on the cache. They demonstrated that selective caching can maximize the robustness of the video stream against network congestion while not violating the limited decoder buffer size.

To increase the cache hit rate and reduce latency experienced by the clients, Kermode [35] proposed to use "hints" to assist the cache in scheduling data retrieval (e.g., prefetch and replacement of the cache content). The hints can be classified into two categories: content hints and application hints. Content hints are provided by the content's sender while application hints are provided by the receiving application. Content hints describe the data and the way it is delivered. For example, the content hints can inform receiving caches about when an object's data can be deleted from the cache. Application hints describe the receiving application's needs for the object's data. An instance of application hints is to describe the application's needs for data in the immediate future so that the cache can prefetch data for the application. In addition, for the multicast scenario, content and application hints can be used by the cache to determine which multicast channels should be joined, when the join should occur, and for how long the cache should listen.

V. Streaming Servers

Streaming servers play a key role in providing streaming services. To offer quality streaming services, streaming servers are required to process multimedia data under timing constraints in order to prevent artifacts (e.g., jerkiness in video motion and pops in audio) during playback at the clients. In addition, streaming servers also need to support VCR-like control operations, such as stop, pause/resume, fast forward, and fast backward. Furthermore, streaming servers have to retrieve media components in a synchronous fashion. For example, retrieving a lecture presentation requires synchronizing video and audio with lecture slides.

A streaming server typically consists of the following three subsystems.

1) *Communicator:* A communicator involves the application layer and transport protocols implemented on the server (shown in Fig. 1). Through a communicator, the clients can communicate with a server and retrieve multimedia contents in a continuous and synchronous manner. We have addressed the application layer in Section III and will address transport protocols in Section VII.
2) *Operating system:* Different from traditional operating systems, an operating system for streaming services needs to satisfy real-time requirements for streaming applications.
3) *Storage system:* A storage system for streaming services has to support continuous media storage and retrieval.

In this section, we are primarily concerned with operating system support and storage systems for streaming media, which will be presented in Sections V-A and V-B, respectively.

A. Real-Time Operating System

The operating system shields the computer hardware from all other software. The operating system offers various services related to the essential resources, such as the CPU, main memory, storage, and all input and output devices. In the following sections, we discuss the unique issues of real-time operating systems and review the associated approaches to the problems introduced by streaming services. Specifically, Sec-

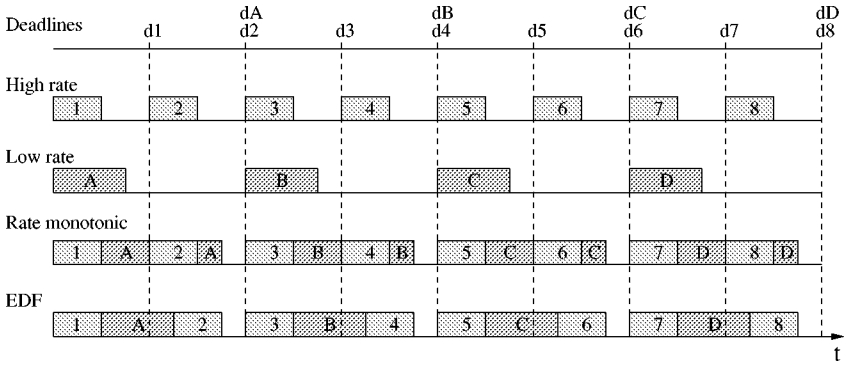

Fig. 13. EDF versus rate-monotonic scheduler.

tion V-A1) shows how process management takes into account the timing requirements imposed by streaming media and apply appropriate scheduling methods; Section V-A2) describes how to manage resources to accommodate timing requirements; Section V-A3) discusses the issues on file management.

1) Process Management: Process management deals with the main processor resource [57]. The process manager maps each single process onto the CPU resource according to a specified scheduling policy such that all processes can meet their requirements.

To fulfill the timing requirements of continuous media, the operating system must use real-time scheduling techniques. Most attempts to solve real-time scheduling problems are variations of two basic algorithms for multimedia systems: earliest deadline first (EDF) [42] and rate-monotonic scheduling [12]. In EDF scheduling, each task is assigned a deadline and the tasks are processed in the order of increasing deadlines. In rate-monotonic scheduling, each task is assigned a static priority according to its request rate.[3] Specifically, the task with the shortest period (or the highest rate) gets the highest priority, and the task with the longest period (or the lowest rate) gets the lowest priority. Then the tasks are processed in the order of priorities.

Both EDF and rate-monotonic scheduling are preemptive; that is, the schedulers can preempt the running task and schedule the new task for the processor based on its deadline/priority. The execution of the interrupted task will resume at a later time. The difference between EDF and rate-monotonic scheduling is as follows. EDF scheduler is based on one-priority task queue and the processor runs the task with the earliest deadline. On the other hand, a rate-monotonic scheduler is a static-priority scheduler with multiple-priority task queues. That is, the tasks in the lower priority queue cannot be executed until all the tasks in the higher priority queues are served. In the example of Fig. 13, there are two task sequences. The high rate sequence is Task 1 to Task 8; the low rate sequence is Task A to Task D. As shown in Fig. 13, in rate-monotonic scheduling, Task 2 preempts Task A since Task 2 has a higher priority; on the other hand, in EDF, Task 2 does not preempt Task A since Task A and Task 2 have the same deadlines $(dA = d2)$. It is clear that a rate-monotonic scheduler is more prone to task switching than EDF. In general, the rate-monotonic algorithm ensures that all deadlines will be met if the processor utilization is under 69% [12]; the EDF algorithm can achieve 100% utilization of processor, but may not guarantee the processing of some tasks during overload periods.

2) Resource Management: Resources in a multimedia server include CPUs, memories, and storage devices. Since resources are limited, a multimedia server can only serve a limited number of clients with requested QoS. Therefore, resource management is required to manage resources so as to accommodate timing requirements. Resource management involves admission control and resource allocation. Specifically, before admitting a new client, a multimedia server must perform admission control test to decide whether a new connection can be admitted without violating performance guarantees already committed to existing connections. If a connection is accepted, the resource manager allocates resources required to meet the QoS for the new connection.

Admission control algorithms can be classified into two categories: deterministic admission control [22] and statistical admission control [65]. Deterministic admission control algorithms provide hard guarantees to clients while statistical admission control algorithms provide statistical guarantees to clients (i.e., the continuity requirements of at least a fixed percentage of media units are ensured to be met). The advantages of deterministic admission control are simplicity and strict assurance of quality; its limitation is lower utilization of server resources. In contrast to this, statistical admission control improves the utilization of server resources by exploiting the human perceptual tolerances as well as the differences between the average and the worst-case performance characteristics of a multimedia server [65].

Corresponding to admission control algorithms, resource allocation schemes can be either deterministic or statistical. Deterministic resource allocation schemes make reservations for the worst case, e.g., reserving bandwidth for the longest processing time and the highest rate that a task might ever need. On the other hand, statistical resource allocation schemes achieve higher utilization by allowing temporary overload and a small percentage of QoS violations.

3) File Management: The file system provides access and control functions for file storage and retrieval [23]. There are two basic approaches to supporting continuous media in file systems. In the first approach, the organization of files on disks remains as it is for discrete data (i.e., a file is not scattered across several disks), with the necessary real-time support provided

[3]Assume that each task is periodic.

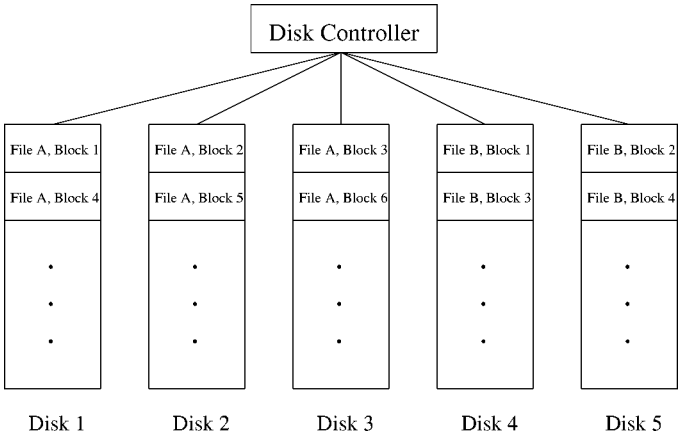

Fig. 14. Data striped in multiple disks and accessed in parallel.

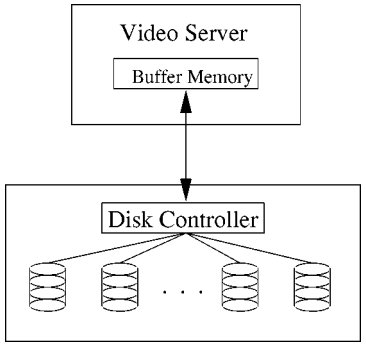

Fig. 15. Disk-based video storage.

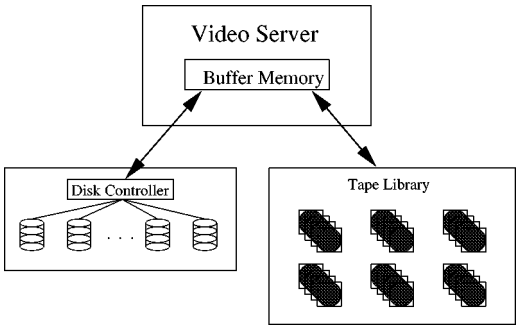

Fig. 16. Hierarchical storage.

through special disk-scheduling algorithms and enough buffer capacity to avoid jitter. The second approach is to organize audio and video files on distributed storage like disk arrays. Under the second approach, the disk throughput can be improved by scattering/striping each audio/video file across several disks (described later in Section V-B) and disk seek-times can be reduced by disk-scheduling algorithms.

Traditional disk-scheduling algorithms such as first-come-first-serve and SCAN [15], [61] do not provide real-time guarantees. Hence, many disk-scheduling algorithms have been proposed to address this issue. These include SCAN-EDF [48], grouped sweeping scheduling (GSS) [77], and dynamic circular SCAN (DC-SCAN) [30], which are described as follows.

a) The SCAN-EDF combines the seek optimization of the traditional disk-scheduling method SCAN [15] and the real-time guarantees of the EDF mechanism. Note that the EDF mechanism in disk scheduling is nonpreemptive, which is different from the preemptive EDF scheme used in process management.
b) The grouped sweeping scheduling divides the set of n streams into g groups; groups can be formed in such a way that all streams belonging to the same group have similar deadlines. Individual streams within a group are served according to SCAN.
c) DC-SCAN employs a circular SCAN [56] service order so as to minimize disk seek overhead and variations in inter-service time, resulting in high throughput. It reduces start-up delay by dynamically adapting the circular SCAN service order.

As a result, the three algorithms, SCAN-EDF, GSS and DC-SCAN, can improve continuous media data throughput and meet real-time requirements imposed by continuous media.

Another function that needs to be supported by file management is interactive control, such as pause/resume, fast forward, and fast backward. The pause/resume operations pose a significant challenge to the design of efficient buffer management schemes because they interfere with the sharing of a multimedia stream among different viewers. This issue is still under study. The fast-forward and fast-backward operations can be implemented either by playing back media at a higher rate than normal or by continuing playback at the normal rate while skipping some data. Since the former approach can significantly increase the data rate, its direct implementation is impractical. The latter approach, on the other hand, needs to be carefully designed if inter-data dependencies are present (for example, P frames and B frames depend on I frames in MPEG) [9]. As a result, for streaming MPEG video, entire group of pictures (GOPs) have to be skipped during fast-forward operations, and the viewer sees normal resolution video with gaps, which is acceptable.

B. Storage System

There are several challenging issues on designing storage systems for multimedia, such as high throughput, large capacity and fault-tolerance [23], which we discuss as follows.

Increase throughput with data striping. If an entire video file is stored on one disk, the number of concurrent accesses to that file are limited by the throughput of that disk. This dictates the number of clients that are viewing the same video file. To overcome this limitation, data striping was proposed [55]. Under data striping schemes, a multimedia file is scattered across multiple disks and the disk array can be accessed in parallel. An example of data striping is shown in Fig. 14, where Block 1, 2 and 3 of File A can be read in parallel, resulting in increased throughput. An important issue in design of a data-striping scheme is to balance the load of most heavily loaded disks to avoid overload situations while keeping latency small. The designers have to trade off load balance with low latency since load balance and low latency are two conflicting objectives [55]. Note that data striping is different from file replication (an expensive way to increase throughput) in that data striping allows only one copy of a video file stored on disks while file replication allows multiple copies of a video file stored on disks.

Increase capacity with tertiary and hierarchical storage. The introduction of multiple disks can increase the storage capacity as shown in Fig. 15. However, the cost for large archives (e.g., with 40 Tbyte storage requirement) is prohibitively high if a large number of disks are used for storage. To keep the storage cost down, tertiary storage (e.g., an automated tape library or CD-ROM jukebox) must be added.

To reduce the overall cost, a hierarchical storage architecture (shown in Fig. 16) is typically used. Under the hierarchical storage architecture, only a fraction of the total storage is kept on disks while the major remaining portion is kept on a tertiary tape system. Specifically, frequently requested video files are kept on disks for quick access; the remainder resides in the automated tape library.

To deploy streaming services at a large scale, a storage area network (SAN) architecture was proposed (shown in Fig. 17) [16], [28]. An SAN can provide high-speed data pipes between storage devices and hosts at far greater distances than conventional host-attached small-computer-systems-interface (SCSI). The connections in an SAN can be direct links between specific storage devices and individual hosts, through fiber-channel arbitrated loop (FC-AL) connections; or the connections in an SAN can form a matrix through a fiber channel switch. With these high-speed connections, an SAN is able to provide a many-to-many relationship between heterogeneous storage devices (e.g., disk arrays, tape libraries, and optical storage arrays), and multiple servers and storage clients.

Another approach to deploying large-scale storage is network attached storage (NAS) (shown in Fig. 18) [26]. Different from SAN, an NAS equipment can attach to a local area network (LAN) or a wide area network (WAN) directly. This is because an NAS equipment includes a file system such as network file system (NFS) and can run on Ethernet, asynchronous transfer mode (ATM), and fiber distributed data interface (FDDI). The protocols that NAS uses include hypertext transfer protocol (HTTP), NFS, TCP, UDP, and IP. The main differences between NAS and SAN are summarized in Table I. On the other hand, both NAS and SAN achieve data separation from the application server so that storage management can be simplified. Specifically, both NAS and SAN have the following advantages over the traditional storage: 1) simplification of storage management by centralizing storage; 2) scalability; and 3) fault tolerance.

Fault tolerance. In order to ensure uninterrupted service even in the presence of disk failures, a server must be able to reconstruct lost information. This can be achieved by using redundant infoffrmation. The redundant information could be either parity data generated by error-correcting codes like FEC or duplicate data on separate disks. That is, there are two fault-tolerant techniques: error-correcting (i.e., parity-encoding) [2], [46], [63] and mirroring [45]. Parity data adds a small storage overhead but requires synchronization of reads and additional processing time to decode lost information. In contrast, mirroring does not require synchronization of reads or additional processing time to decode lost information, which significantly simplifies design and implementation of video servers. However, mirroring incurs at least twice as much storage volume as in the nonfault-tolerant case. As a result, there is a tradeoff between reliability and

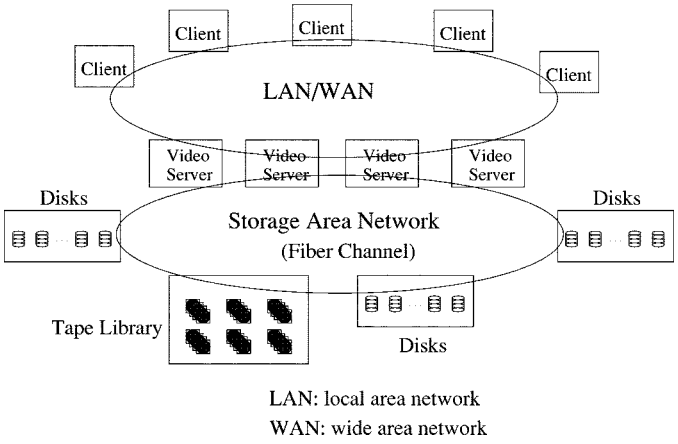

Fig. 17. SAN-based server and storage architecture for large-scale deployment.

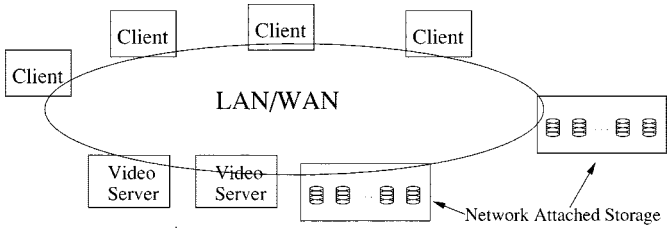

Fig. 18. Network-attached storage architecture for large-scale deployment.

TABLE I
DIFFERENCES BETWEEN SAN AND NAS

	Networking technologies	Protocols
SAN	Fiber channel	Encapsulated SCSI
NAS	Ethernet, ATM, FDDI	HTTP, NFS, TCP, UDP, IP

complexity (cost). A recent study [21] shows that, for the same degree of reliability, mirroring-based schemes always outperform parity-based schemes in terms of per-stream cost, as well as restart latency after disk failure.

To summarize, we have addressed various issues in streaming server design and presented important techniques for efficient, scalable and reliable storage and retrieval of multimedia files. In the next section, we discuss synchronization mechanisms for streaming media.

VI. MEDIA SYNCHRONIZATION

A major feature that distinguishes multimedia applications from other traditional data applications is the integration of various media streams that must be presented in a synchronized fashion. For example, in distance learning, the presentation of slides should be synchronized with the commenting audio stream (see Fig. 19). Otherwise, the current slide being displayed on the screen may not correspond to the lecturer's explanation heard by the students, which is annoying. With media synchronization, the application at the receiver side can

present the media in the same way as they were originally captured.

Media synchronization refers to maintaining the temporal relationships within one data stream and between various media streams. There are three levels of synchronization, namely, intra-stream, inter-stream, and inter-object synchronization. The three levels of synchronization correspond to three semantic layers of multimedia data as follows [57].

1) *Intra-stream synchronization:* The lowest layer of continuous media or time-dependent data (such as video and audio) is the media layer. The unit of the media layer is logical data unit such as a video/audio frame, which adheres to strict temporal constraints to ensure acceptable user perception at playback. Synchronization at this layer is referred to as intra-stream synchronization, which maintains the continuity of logical data units. Without intra-stream synchronization, the presentation of the stream may be interrupted by pauses or gaps.
2) *Inter-stream synchronization:* The second layer of time-dependent data is the stream layer. The unit of the stream layer is a whole stream. Synchronization at this layer is referred to as inter-stream synchronization, which maintains temporal relationships among different continuous media. Without inter-stream synchronization, skew between the streams may become intolerable. For example, users could be annoyed if they notice that the movements of the lips of a speaker do not correspond to the presented audio.
3) *Inter-object synchronization:* The highest layer of a multimedia document is the object layer, which integrates streams and time-independent data such as text and still images. Synchronization at this layer is referred to as inter-object synchronization. The objective of inter-object synchronization is to start and stop the presentation of the time-independent data within a tolerable time interval, if some previously defined points of the presentation of a time-dependent media object are reached. Without inter-object synchronization, for example, the audience of a slide show could be annoyed if the audio is commenting one slide while another slide is being presented.

Media streams may lose synchronization after moving from the server to the client. As shown in Fig. 1, there are many components along the path which transports data from its storage site to the user. Specifically, the server retrieves data from the storage device and sends that data into the network; the network transports the data to the client; the client reads the data from its network interface and presents it to the user; operating systems and protocols allow these systems to run and do their work. Each of these components on the transport path performs a certain task and affects the data in a different way. They all inevitably introduce delays and delay variations in either predictable or unpredictable manners. In particular, the delay introduced in the network is typically unpredictable due to the best-effort nature of the Internet. The incurred delays and delay variations could disrupt intra-media, inter-media, and inter-object synchronization. Therefore, media synchronization mechanisms are required to ensure proper rendering of the multimedia presentation at the client.

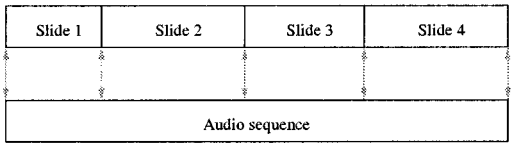

Fig. 19. Synchronization between the slides and the commenting audio stream.

The essential part of any media synchronization mechanism is the specifications of the temporal relations within a medium and between the media. The temporal relations can be specified either automatically or manually. In the case of audio/video recording and playback, the relations are specified automatically by the recording device. In the case of presentations that are composed of independently captured or otherwise created media, the temporal relations have to be specified manually (with human's support). The manual specification can be illustrated by the design of a slide show: the designer selects the appropriate slides, creates an audio object and defines the units of the audio stream where the slides have to be presented (see Fig. 19).

The methods that are used to specify the temporal relations include interval-based, axes-based, control flow-based, and event-based specifications [3]. A widely used specification method for continuous media is axes-based specifications or time-stamping: at the source, a stream is time-stamped to keep temporal information within the stream and with respect to other streams; at the destination, the application presents the streams according to their temporal relation.

Besides specifying the temporal relations, it is desirable that synchronization be supported by each component on the transport path. For example, the servers store large amount of data in such a way that retrieval is quick and efficient to reduce delay; the network provides sufficient bandwidth, and delay and jitter introduced by the network are tolerable to the multimedia applications; the operating systems and the applications provide real-time data processing (e.g., retrieval, re-synchronization, and display). However, real-time support from the network is not available in the current Internet. Hence, most synchronization mechanisms are implemented on the end systems. These synchronization mechanisms can be either preventive or corrective [34].

Preventive mechanisms are designed to minimize synchronization errors as data is transported from the server to the user. In other words, preventive mechanisms attempt to minimize latencies and jitters. These mechanisms involve disk-reading scheduling algorithms, network transport protocols, operating systems, and synchronization schedulers. Disk-reading scheduling is the process of organizing and coordinating the retrieval of data from the storage devices. Network transport protocols provide means for maintaining synchronization during data transmission over the Internet. Operating systems achieve the precise control of timing constraints by using EDF or rate monotonic scheduling. A synchronization scheduler can use the synchronization specifications for a presentation to create a schedule for the delivery of the media streams to the client by the servers (delivery schedule) and the presentation of these media streams to the user by the client application (presentation

schedule). This scheduler can be centralized (entirely located at the client) or distributed (the delivery scheduling functionalities are shared among the servers and the client).

Corrective mechanisms are designed to recover synchronization in the presence of synchronization errors. Synchronization errors are unavoidable, since the Internet introduces random delay, which destroys the continuity of the media stream by incurring gaps and jitters during data transmission. Therefore, certain compensations (i.e., corrective mechanisms) at the receiver are necessary when synchronization errors occur. An example of corrective mechanisms is the stream synchronization protocol (SSP) [24]. In SSP, the concept of an "intentional delay" is used by the various streams in order to adjust their presentation time to recover from network delay variations. The operations of SSP are described as follows. At the client side, units that control and monitor the client-end of the data connections compare the real arrival times of data with the ones predicted by the presentation schedule and notify the scheduler of any discrepancies. These discrepancies are then compensated by the scheduler, which delays the display of data that are "ahead" of other data, allowing the late data to "catch up".

In sum, media synchronization is one of the key issues in the design of media streaming services. A great deal of effort has been contributed to the synchronization area. So far, we have described the synchronization concepts, requirements and approaches. For more information on media synchronization, please refer to [3], [57] and references therein.

VII. Protocols for Streaming Video

Quite a few protocols have been designed and standardized for communication between clients and streaming servers. According to their functionalities, the protocols directly related to Internet streaming video can be classified into the following three categories.

1) *Network-layer protocol* provides basic network service support such as network addressing. The IP serves as the network-layer protocol for Internet video streaming.
2) *Transport protocol* provides end-to-end network transport functions for streaming applications. Transport protocols include UDP, TCP, real-time transport protocol (RTP), and real-time control protocol (RTCP). UDP and TCP are lower-layer transport protocols while RTP and RTCP [51] are upper-layer transport protocols, which are implemented on top of UDP/TCP (see Fig. 20).
3) *Session control protocol* defines the messages and procedures to control the delivery of the multimedia data during an established session. The RTSP [53] and the session initiation protocol (SIP) [31] are such session control protocols.

To illustrate the relationship among the three types of protocols, we depict the protocol stacks for media streaming in Fig. 20. For the data plane, at the sending side, the compressed video/audio data is retrieved and packetized at the RTP layer. The RTP-packetized streams provide timing and synchronization information, as well as sequence numbers. The RTP-packetized streams are then passed to the UDP/TCP layer and the IP layer. The resulting IP packets are transported over the Internet. At the receiver side, the media streams are processed in the reversed manner before their presentations. For the control plane, RTCP packets and RTSP packets are multiplexed at the UDP/TCP layer and move to the IP layer for transmission over the Internet.

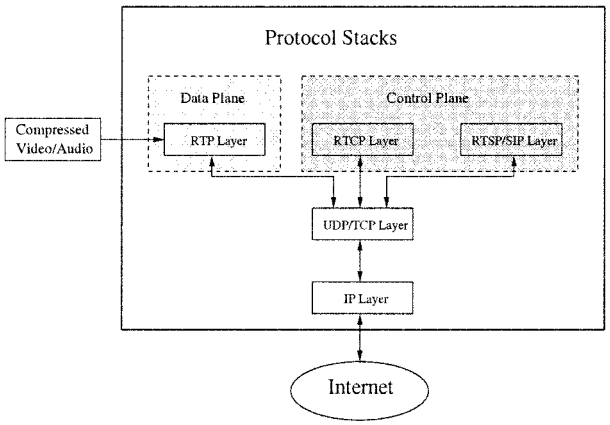

Fig. 20. Protocol stacks for media streaming.

The rest of this section is organized as follows. In Section VII-A, we discuss transport protocols for streaming media. Section VII-B describes the session control protocols, i.e., RTSP and SIP.

A. Transport Protocols

The transport protocol family for media streaming includes UDP, TCP, RTP, and RTCP protocols [41]. UDP and TCP provide basic transport functions while RTP and RTCP run on top of UDP/TCP.

UDP and TCP protocols support such functions as multiplexing, error control, congestion control, or flow control. These functions can be briefly described as follows. First, UDP and TCP can multiplex data streams from different applications running on the same machine with the same IP address. Second, for the purpose of error control, TCP and most UDP implementations employ the checksum to detect bit errors. If a single or multiple bit-errors are detected in the incoming packet, the TCP/UDP layer discards the packet so that the upper layer (e.g., RTP) will not receive the corrupted packet. On the other hand, different from UDP, TCP uses retransmission to recover lost packets. Therefore, TCP provides reliable transmission while UDP does not. Third, TCP employs congestion control to avoid sending too much traffic, which may cause network congestion. This is another feature that distinguishes TCP from UDP. Lastly, TCP employs flow control to prevent the receiver buffer from overflowing while UDP does not have any flow control mechanism.

Since TCP retransmission introduces delays that are not acceptable for streaming applications with stringent delay requirements, UDP is typically employed as the transport protocol for video streams. In addition, since UDP does not guarantee packet delivery, the receiver needs to rely on upper layer (i.e., RTP) to detect packet loss.

RTP is an Internet standard protocol designed to provide end-to-end transport functions for supporting real-time appli-

cations [51]. RTCP is a companion protocol with RTP and is designed to provide QoS feedback to the participants of an RTP session. In other words, RTP is a data transfer protocol while RTCP is a control protocol.

RTP does not guarantee QoS or reliable delivery, but rather, provides the following functions in support of media streaming:

1) *Time-stamping:* RTP provides time-stamping to synchronize different media streams. Note that RTP itself is not responsible for the synchronization, which is left to the applications.
2) *Sequence numbering:* Since packets arriving at the receiver may be out of sequence (UDP does not deliver packets in sequence), RTP employs sequence numbering to place the incoming RTP packets in the correct order. The sequence number is also used for packet loss detection.
3) *Payload type identification:* The type of the payload contained in an RTP packet is indicated by an RTP-header field called payload type identifier. The receiver interprets the content of the packet based on the payload type identifier. Certain common payload types such as MPEG-$1/2$ audio and video have been assigned payload type numbers [52]. For other payloads, this assignment can be done with session control protocols.
4) *Source identification:* The source of each RTP packet is identified by an RTP-header field called Synchronization SouRCe identifier (SSRC), which provides a means for the receiver to distinguish different sources.

RTCP is the control protocol designed to work in conjunction with RTP [51]. In an RTP session, participants periodically send RTCP packets to convey feedback on quality of data delivery and information of membership. Basically, RTCP provides the following services:

1) *QoS feedback:* This is the primary function of RTCP. RTCP provides feedback to an application regarding the quality of data distribution. The feedback is in the form of sender reports (sent by the source) and receiver reports (sent by the receiver). The reports can contain information on the quality of reception such as: 1) fraction of the lost RTP packets, since the last report; 2) cumulative number of lost packets, since the beginning of reception; 3) packet interarrival jitter; and 4) delay since receiving the last sender's report. The control information is useful to the senders, the receivers, and third-party monitors. Based on the feedback, the sender can adjust its transmission rate (see Section III-A-1); the receivers can determine whether congestion is local, regional, or global; network managers can evaluate the network performance for multicast distribution.
2) *Participant identification:* A source can be identified by the SSRC field in the RTP header. Unfortunately, the SSRC identifier is not convenient for human users. To remedy this problem, the RTCP provides a human-friendly mechanism for source identification. Specifically, RTCP SDES (source description) packets contain textual information called canonical names as globally unique identifiers of the session participants. It may include a user's name, telephone number, email address, and other information.
3) *Control packets scaling:* To scale the RTCP control packet transmission with the number of participants, a control mechanism is designed as follows. The control mechanism keeps the total control packets to 5% of the total session bandwidth. Among the control packets, 25% are allocated to the sender reports and 75% to the receiver reports. To prevent control packet starvation, at least one control packet is sent within 5 s at the sender or receiver.
4) *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.
5) *Minimal session control information.* This optional functionality can be used for transporting session information such as names of the participants.

B. Session Control Protocols: RTSP and SIP

The RTSP is a session control protocol for streaming media over the Internet [53]. One of the main functions of RTSP is to support VCR-like control operations such as stop, pause/resume, fast forward, and fast backward. In addition, RTSP also provides means for choosing delivery channels (e.g., UDP, multicast UDP, or TCP), and delivery mechanisms based upon RTP. RTSP works for multicast as well as unicast.

Another main function of RTSP is to establish and control streams of continuous audio and video media between the media servers and the clients. Specifically, RTSP provides the following operations.

1) *Media retrieval:* The client can request a presentation description, and ask the server to setup a session to send the requested media data;
2) *Adding media to an existing session:* The server or the client can notify each other about any additional media becoming available to the established session.

In RTSP, each presentation and media stream is identified by an RTSP universal resource locator (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, email, or other means.

SIP [31] is another session control protocol. Similar to RTSP, SIP can also create and terminate sessions with one or more participants. Unlike RTSP, SIP supports user mobility by proxying and redirecting requests to the user's current location.

To summarize, RTSP and SIP are designed to initiate and direct delivery of streaming media data from media servers. RTP is a transport protocol for streaming media data while RTCP is a protocol for monitoring delivery of RTP packets. UDP and TCP are lower-layer transport protocols for RTP/RTCP/RTSP/SIP packets and IP provides a common platform for delivering UDP/TCP packets over the Internet. The combination of these protocols provides a complete streaming service over the Internet.

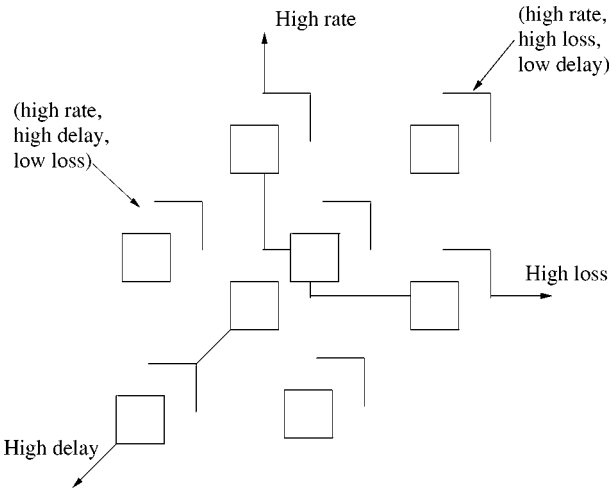

Fig. 21. A 3-D view of the set of video layers/levels.

VIII. Summary

Video streaming is an important component of many Internet multimedia applications, such as distance learning, digital libraries, home shopping, and video-on-demand. The best-effort nature of the current Internet poses many challenges to the design of streaming video systems. In this paper, we have surveyed major approaches and mechanisms for Internet video streaming. The objective is not to provide an exhaustive review of existing approaches and mechanisms, but instead to give the reader a perspective on the range of options available, and the associated tradeoffs among performance, functionality, and complexity.

To provide insights on design of streaming video systems, we summarize the pros and cons of the approaches discussed in the paper and point out future directions as follows.

1) *Video compression:* Most recent efforts on video compression for streaming video have been focused on scalable video coding. The primary objectives of on-going research on scalable video coding are to achieve high compression efficiency, high flexibility (bandwidth scalability), and/or low complexity. Due to the conflicting nature of efficiency, flexibility, and complexity, each scalable video coding scheme seeks a tradeoff among the three factors. Designers of video-streaming service need to choose an appropriate scalable video coding scheme, which meets the target efficiency and flexibility at an affordable cost/complexity.

   A promising direction on scalable video coding is to integrate several video coding techniques to deal with QoS fluctuations in the networks. Scalable video coding is capable of coping with bandwidth variations; error-resilient encoding (discussed in Section III-B-3) can deal with packet loss; delay cognizant video coding [6], [7] was shown to be effective in dealing with delay variations. Hence, it is foreseen that combination of the three techniques could provide a range of solutions to the problem of QoS fluctuations. Fig. 21 illustrates the video layers/levels encoded by such an integrated encoder [7]. Specifically, Fig. 21 shows a 3-D view of the whole set of video layers/levels when a two-level decomposition is applied to each dimension. The rate dimension corresponds to a partition of layers by bit rates; the delay dimension to a partition by delay tolerance; and the loss dimension to a partition by error resilience. A cube represents a video layer/level and characterizes its QoS triplet (rate, delay, loss) by its location. The cube nearest to the origin represents the core video information, i.e., the most visually significant data requiring the least bandwidth, least error resilient and lowest delay. This core layer may only carry key frames with the most aggressive compression and has no adaptability to any fluctuation. Adaptability is increased by adding layers/levels along one or more dimensions.

2) *Application-layer QoS control* includes congestion control and error control.

   Congestion control takes the form of rate control. There are three kinds of rate control: source-based, receiver-based, and hybrid rate-control. The source-based rate control is suitable for unicast; the receiver-based and hybrid rate-control are suitable for multicast since both can achieve good tradeoff between bandwidth efficiency and service flexibility for multicast video. Probing and TCP-friendly modeling are two approaches for rate control. Most recent studies on source-based rate control have been focused on TCP-friendly adaptation [60], [78]. A number of TCP-friendly adaptation schemes have been proposed and demonstrated to achieve certain degree of fairness among competing connections, including TCP connections. However, strictly TCP-like rate control may result in sharp reductions in the transmission rate, and possibly unpleasant visual quality [66]. Therefore, for TCP-like rate control, it needs further investigation on how to trade off responsiveness in detecting and reacting to congestion with smooth fluctuation in visual quality.

   Error-control mechanisms include FEC, retransmission, error-resilient encoding, and error concealment.

   There are three kinds of FEC: channel coding, source coding-based FEC, and joint source/channel coding. The advantage of all FEC schemes over retransmission-based schemes is reduction in video transmission latency. Source coding-based FEC can achieve lower delay than channel coding while joint source/channel coding could achieve optimal performance in rate-distortion sense. The disadvantages of all FEC schemes are: increase in the transmission rate, and inflexibility to varying loss characteristics.

   Unlike FEC, which adds redundancy regardless of correct receipt or loss, a retransmission-based scheme only resends the packets that are lost. Thus, a retransmission-based scheme is adaptive to varying loss characteristics, resulting in efficient use of network resources. The limitation of delay-constrained retransmission-based schemes is that their effectiveness diminishes when the round trip time is too large.

   Currently, an important direction is to combine FEC with retransmission [11], [29], [50]. In addition, FEC can be used in layered video multicast so that each client can

individually trade off latency for quality based on its requirements. Examples of such an FEC-protected multicast include hierarchical FEC [59] and a receiver-driven layered multicast [11].

Multiple description coding is a recently proposed mechanism for error-resilient encoding. The advantage of MDC is its robustness to loss. The cost of MDC is reduction in compression efficiency. Current research effort gears toward finding a good tradeoff between the compression efficiency and the reconstruction quality from one description.

Error concealment is performed by the receiver (when packet loss occurs) and can be used in conjunction with any other techniques (e.g., congestion control and other error control mechanisms).

3) *Continuous media distribution services:* Network support is important to provide quality multimedia presentations. Continuous media distribution services are built on top of the best-effort Internet with the aim of achieving QoS and efficiency for streaming video. A major topic of active research is how to build a scalable, efficient, cost-effective and incremental deployable infrastructure for continuous media distribution.
4) *Streaming servers* are essential in providing streaming services. We have addressed the unique issues that concern designers of streaming servers and have reviewed the major approaches to the issues. Current research efforts include: 1) how to efficiently support VCR-like interactive control; 2) how to design efficient and reliable storage and retrieval of multimedia objects on disk arrays; 3) how to design highly scalable multimedia servers in a varity of environments ranging from video-on-demand servers to integrated multimedia file systems; and 4) how to design fault-tolerant storage systems with desirable features of both parity and mirroring (i.e., trade off the parity group size with the number of disks across which original data of a single disk is replicated for mirroring).
5) *Media synchronization* is a unique feature of multimedia applications. A great deal of effort has been contributed to the media synchronization area. However, how to achieve synchronization in multicast video while efficiently supporting VCR-like interactive functions have not been adequately addressed and remains a topic for future research.
6) *Protocols for streaming media:* Several protocols have been standardized for communication between clients and streaming servers. Future research topics on design of protocols include:1) how to take caches into account (e.g., how to communicate with continuous media caches and how to control continuous media caches); 2) how to efficiently support pause/resume operations in caches (since the pause/resume operations interfere with the sharing of a multimedia stream among different viewers); and 3) how to provide security in the protocols.

We would like to stress that the six areas (i.e., video compression, application-layer QoS control, continuous media distribution services, streaming servers, media synchronization, and protocols) are basic building blocks for a streaming video architecture. This architecture ties together a broad range of technologies from signal processing, networking and server design. A thorough understanding of the whole architecture is essential for developing the particular signal processing techniques (e.g., video compression) suitable for streaming video. Furthermore, an in-depth knowledge on both signal processing and networking technologies helps to make effective design and use of application-layer QoS control, continuous media distribution services, and protocols. Finally, a clear understanding of the overall architecture is instrumental in the design of efficient, scalable, and/or fault-tolerant streaming servers.

REFERENCES

[1] A. Albanese, J. Blömer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1737–1744, Nov. 1996.

[2] S. Berson, L. Golubchik, and R. R. Muntz, "Fault tolerant design of multimedia servers," in *Proc. ACM SIGMOD'95*, May 1995, pp. 364–375.

[3] G. Blakowski and R. Steinmetz, "A media synchronization survey: Reference model, specification, and case studies," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 5–35, Jan. 1996.

[4] J.-C. Bolot, T. Turletti, and I. Wakeman, "Scalable feedback control for multicast video distribution in the Internet," in *Proc. ACM SIGCOMM'94*, London, U.K., Sept. 1994, pp. 58–67.

[5] J.-C. Bolot and T. Turletti, "Adaptive error control for packet video in the Internet," *Proc. IEEE Int. Conf. Image Processing (ICIP'96)*, pp. 25–28, Sept. 1996.

[6] Y.-C. Chang, D. G. Messerschmitt, T. Carney, and S. A. Klein.. *Delay cognizant video coding: Architecture, applications, and quality evaluations.* [Online] http://divine.EECS.Berkeley.EDU/~messer/

[7] Y.-C. Chang and D. G. Messerschmitt.. *Adaptive layered video coding for multi-time scale bandwidth fluctuations* [Online] http://divine.EECS.Berkeley.EDU/~messer/

[8] A. Chankhunthod, P. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, "A hierarchical Internet object cache," in *Proc. USENIX 1996 Annu. Technical Conf.*, Jan. 1996.

[9] M. Chen, D. D. Kandlur, and P. S. Yu, "Support for fully interactive playout in a disk-array-based video server," in *Proc. ACM Multimedia'94*, New York, Oct. 1994.

[10] S. Y. Cheung, M. Ammar, and X. Li, "On the use of destination set grouping to improve fairness in multicast video distribution," *Proc. IEEE INFOCOM'96*, pp. 553–560, Mar. 1996.

[11] P. A. Chou, A. E. Mohr, A. Wang, and S. Mehrotra.. *Error control for receiver-driven layered multicast of audio and video* [Online] http://www.research.microsoft.com/~pachou/publications.htm

[12] J. Y. Chung, J. W. S. Liu, and K. J. Lin, "Scheduling periodic jobs that allows imprecise results," *IEEE Trans. Comput.*, vol. 19, pp. 1156–1173, Sept. 1990.

[13] G. J. Conklin, G. S. Greenbaum, K. O. Lillevold, A. F. Lippman, and Y. A. Reznik, "Video coding for streaming media delivery on the Internet," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, Mar. 2001.

[14] S. Deering, "Multicast routing in internetworks and extended LANs," in *Proc. ACM SIGCOMM'88*, Stanford, CA, Aug. 1988, pp. 55–64.

[15] P. J. Denning, "Effects of scheduling on file memory operations," in *Proc. AFIPS Spring Joint Computer Conf.*, Reston, VA, 1967, pp. 9–21.

[16] D. H. C. Du and Y.-J. Lee, "Scalable server and storage architectures for video streaming," *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, pp. 62–67, June 1999.

[17] A. Eleftheriadis and D. Anastassiou, "Meeting arbitrary QoS constraints using dynamic rate shaping of coded digital video," in *Proc. 5th Int. Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV'95)*, Apr. 1995, pp. 95–106.

[18] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: A scalable wide-area web cache sharing protocol," *IEEE Trans. Networking*, vol. 8, pp. 281–293, June 2000.

[19] FastForward Networks. (2000) FastForward networks' broadcast overlay architecture. [Online]. Available: http://www.ffnet.com/pdfs/boa-whitepaper.pdf.

[20] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE Trans. Networking*, vol. 7, pp. 458–472, Aug. 1999.

[21] J. Gafsi and E. W. Biersack, "Performance and reliability study for distributed video servers: Mirroring or parity?," in *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, June 1999, pp. 628–634.

[22] J. Gemmell and S. Christodoulakis, "Principles of delay sensitive multimedia data storage and retrieval," *ACM Trans. Inform. Syst.*, vol. 10, no. 1, pp. 51–90, Jan. 1992.

[23] J. Gemmell, H. M. Vin, D. D. Kandlur, P. V. Rangan, and L. A. Rowe, "Multimedia storage servers: A tutorial," *IEEE Comput. Mag.*, vol. 28, pp. 40–49, May 1995.

[24] N. D. Georganas, "Synchronization issues in multimedia presentational and conversational applications," in *Proc. Pacific Workshop on Distributed Multimedia Systems (DMS'96)*, June 1996.

[25] M. Ghanbari, "Cell-loss concealment in ATM video codes," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 238–247, June 1993.

[26] G. A. Gibson and R. V. Meter, "Network attached storage architecture," *Commun. ACM*, vol. 43, no. 11, pp. 37–45, Nov. 2000.

[27] B. Girod, U. Horn, and B. Belzer, "Scalable video coding with multiscale motion compensation and unequal error protection," in *Proc. Symp. Multimedia Communications and Video Coding*, New York, Oct. 1995, pp. 475–482.

[28] A. Guha, "The evolution to network storage architectures for multimedia applications," *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, pp. 68–73, June 1999.

[29] Q. Guo, Q. Zhang, W. Zhu, and Y.-Q. Zhang, "Sender-adaptive and receiver-driven video multicasting," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS 2001)*, Sydney, Australia, May 2001.

[30] B. Hamidzadeh and J. Tsun-Ping, "Dynamic scheduling techniques for interactive hypermedia servers," *IEEE Trans. Consumer Electron.*, vol. 45, pp. 46–56, Feb. 1999.

[31] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session Initiation Protocol," Internet Engineering Task Force, RFC 2543, Mar. 1999.

[32] M. Hemy, U. Hengartner, P. Steenkiste, and T. Gross, "MPEG system streams in best-effort networks," *Proc. IEEE Packet Video'99*, Apr. 1999.

[33] *Information Technology—Coding of audio-visual objects, part 1: Systems, part 2: Visual, part 3: Audio*, ISO/IEC JTC 1/SC 29/WG 11, FCD 14 496, Dec. 1998.

[34] J. P. Jarmasz and N. D. Georganas, "Designing a distributed multimedia synchronization scheduler," *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, pp. 451–457, June 1997.

[35] R. G. Kermode, "Smart Network Caches: Localized Content and Application Negotiated Recovery Mechanisms for Multicast Media Distribution," Ph.D. dissertation, MIT, Cambridge, MA, June 1998.

[36] B.-J. Kim, Z. Xiong, and W. A. Pearlman, "Low bit-rate scalable video coding with 3D set partitioning in hierarchical trees (3D SPIHT)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 1374–1387, Dec. 2000.

[37] S. Li, F. Wu, and Y.-Q. Zhang, "Study of a New Approach to Improve FGS Video Coding Efficiency," ISO/IEC JTC1/SC29/WG11, MPEG99/M5583, Dec. 1999.

[38] W. Li, "Bit-Plane Coding of DCT Coefficients for Fine Granularity Scalability," ISO/IEC JTC1/SC29/WG11, MPEG98/M3989, Oct. 1998.

[39] ——, "Streaming video profile in MPEG-4," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, Mar 2001.

[40] C.-W. Lin, J. Youn, J. Zhou, M.-T. Sun, and I. Sodagar, "MPEG video streaming with VCR functionality," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, . , Mar. 2001.

[41] *Handbook of Communication Technologies: The Next Decade—Multimedia over IP: RSVP, RTP, RTCP, RTSP*, R. Osso, Ed., CRC Press, Boca Raton, FL, 1999, pp. 29–46.

[42] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *J. Assoc. Comput. Mach.*, vol. 20, no. 1, pp. 46–61, Jan. 1973.

[43] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proc. ACM SIGCOMM'96*, Aug. 1996, pp. 117–130.

[44] Z. Miao and A. Ortega, "Proxy caching for efficient video services over the Internet," in *Proc. Packet Video'99*, New York, Apr. 1999.

[45] A. Mourad, "Doubly-striped disk mirroring: Reliable storage for video servers," *Multimedia, Tools and Applicat.*, vol. 2, pp. 253–272, May 1996.

[46] B. Ozden, R. Rastogi, P. Shenoy, and A. Silberschatz, "Fault-tolerant architectures for continuous media servers," in *Proc. ACM SIGMOD'96*, June 1996, pp. 79–90.

[47] R. Puri, K. Ramchandran, K. W. Lee, and V. Bharghavan, "Application of FEC based multiple description coding to Internet video streaming and multicast," in *Proc. Packet Video Workshop*, Cagliari, Sardinia, Italy, May 1–2, 2000.

[48] A. L. Reddy and J. Wyllie, "Disk scheduling in a multimedia I/O system," in *Proc. ACM Multimedia'93*, Anaheim, CA, Aug. 1–6, 1993, pp. 289–297.

[49] J. Rexford, S. Sen, and A. Basso, "A smoothing proxy service for variable-bit-rate streaming video," in *Proc. IEEE Global Internet Symp.*, Rio de Janeiro, Brazil, Dec. 1999.

[50] I. Rhee, "Error control techniques for interactive low-bit-rate video transmission over the Internet," in *Proc. ACM SIGCOMM'98*, Vancouver, Canada, Aug. 1998.

[51] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," Internet Engineering Task Force, RFC 1889, Jan. 1996.

[52] H. Schulzrinne, "RTP Profile for Audio and Video Conferences with Minimal Control," Internet Engineering Task Force, RFC 1890, Jan. 1996.

[53] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," Internet Engineering Task Force, RFC 2326, Apr. 1998.

[54] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," *Proc. IEEE INFOCOM'99*, Mar. 1999.

[55] P. Shenoy and H. M. Vin, "Efficient striping techniques for multimedia file servers," *Perform. Eval.*, vol. 38, no. 3–4, pp. 175–199, Dec. 1999.

[56] A. Silberschatz, J. Peterson, and P. Galvin, *Operating System Concepts*, 3rd ed. Reading, MA: Addison-Wesley, 1991.

[57] R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[58] H. Sun and W. Kwok, "Concealment of damaged block transform coded images using projection onto convex sets," *IEEE Trans. Image Processing*, vol. 4, pp. 470–477, Apr. 1995.

[59] W. Tan and A. Zakhor, "Multicast transmission of scalable video using receiver-driven hierarchical FEC," in *Proc. Packet Video'99*, New York, Apr. 1999.

[60] ——, "Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol," *IEEE Trans. Multimedia*, vol. 1, pp. 172–186, June 1999.

[61] A. S. Tanenbaum, *Modern Operating Systems*, 1st ed. Englewood Cliffs, NJ: Prentice-Hall, Feb. 1992.

[62] D. Taubman and A. Zakhor, "A common framework for rate and distortion based scaling of highly scalable compressed video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 329–354, Aug. 1996.

[63] R. Tewari, D. M. Dias, W. Kish, and H. Vin, "Design and performance tradeoffs in clustered video servers," *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, pp. 144–150, June 1996.

[64] T. Turletti and C. Huitema, "Videoconferencing on the Internet," *IEEE Trans. Networking*, vol. 4, pp. 340–351, June 1996.

[65] H. M. Vin, P. Goyal, A. Goyal, and A. Goyal, "A statistical admission control algorithm for multimedia servers," in *Proc. ACM Multimedia'94*, Oct. 1994, pp. 33–40.

[66] X. Wang and H. Schulzrinne, "Comparison of adaptive Internet multimedia applications," *IEICE Trans. Commun.*, vol. E82-B, no. 6, pp. 806–818, June 1999.

[67] Y. Wang, Q.-F. Zhu, and L. Shaw, "Maximally smooth image recovery in transform coding," *IEEE Trans. Commun.*, vol. 41, pp. 1544–1551, Oct. 1993.

[68] Y. Wang, M. T. Orchard, and A. R. Reibman, "Multiple description image coding for noisy channels by pairing transform coefficients," *Proc. IEEE Workshop on Multimedia Signal Processing*, pp. 419–424, June 1997.

[69] Y. Wang and Q.-F. Zhu, "Error control and concealment for video communication: A review," *Proc. IEEE*, vol. 86, pp. 974–997, May 1998.

[70] D. Wu, Y. T. Hou, W. Zhu, H.-J. Lee, T. Chiang, Y.-Q. Zhang, and H. J. Chao, "On end-to-end architecture for transporting MPEG-4 video over the Internet," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 923–941, Sept. 2000.

[71] D. Wu, Y. T. Hou, and Y.-Q. Zhang, "Transporting real-time video over the Internet: Challenges and approaches," *Proc. IEEE*, vol. 88, pp. 1855–1875, Dec. 2000.

[72] F. Wu, S. Li, and Y.-Q. Zhang, "A framework for efficient progressive fine granularity scalable video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 282–300, Mar. 2001.

[73] X. Wu, S. Cheng, and Z. Xiong, "On packetization of embedded multimedia bitstreams," *IEEE Trans. Multimedia, Special Issue on Multimedia over IP*, 2001, to be published.

[74] J. Xu, Z. Xiong, S. Li, and Y.-Q. Zhang. Three-dimensional embedded subband coding with optimized truncation (3-D ESCOT). [Online] http://lena.tamu.edu/~zx/

[75] N. Yeadon, F. Garcia, D. Hutchison, and D. Shepherd, "Filters: QoS support mechanisms for multipeer communications," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 1245–1262, Sept. 1996.
[76] F. Yu, Q. Zhang, W. Zhu, and Y.-Q. Zhang, "QoS-adaptive proxy caching for multimedia streaming over the Internet," in *Proc. 1st IEEE Pacific-Rim Conf. Multimedia*, Sydney, Australia, Dec. 13–15, 2000.
[77] P. S. Yu, M. S. Chen, and D. D. Kandlur, "Grouped sweeping scheduling for DASD-based multimedia storage management," *ACM/Springer Multimedia Syst.*, vol. 1, no. 3, pp. 99–109, 1993.
[78] Q. Zhang, Y.-Q. Zhang, and W. Zhu, "Resource allocation for audio and video streaming over the Internet," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS'2000)*, Geneva, Switzerland, May 28–31, 2000.
[79] Q. Zhang, G. Wang, W. Zhu, and Y.-Q. Zhang, "Robust scalable video streaming over Internet with network-adaptive congestion control and unequal loss protection," in *Proc. Packet Video Workshop*, Kyongju, Korea, Apr. 2001.
[80] Z.-L. Zhang, S. Nelakuditi, R. Aggarwa, and R. P. Tsang, "Efficient server selective frame discard algorithms for stored video delivery over resource constrained networks," *Proc. IEEE INFOCOM'99*, pp. 472–479, Mar. 1999.
[81] Z.-L. Zhang, Y. Wang, D. H. C. Du, and D. Su, "Video staging: A proxy-server-based approach to end-to-end video delivery over wide-area networks," *IEEE Trans. Networking*, vol. 8, pp. 429–442, Aug. 2000.

Dapeng Wu (S'98) received the B.E. degree from Huazhong University of Science and Technology, Wuhan, China, and the M.E. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 1990 and 1997, respectively, both in electrical engineering. Since January 2000, he has been working toward the Ph.D. degree in electrical and computer engineering at Carnegie Mellon University, Pittsburgh, PA.

From July 1997 to December 1999, he was involved in graduate research at Polytechnic University, Brooklyn, NY. During the summers of 1998–2000, he conducted research at Fujitsu Laboratories of America, Sunnyvale, CA, on architectures and traffic management algorithms in the Internet and wireless networks for multimedia applications. His current interests are in the areas of rate control and error control for video communications over the Internet and wireless networks, and next-generation Internet architecture, protocols, and implementations for integrated and differentiated services.

Mr. Wu is a student member of ACM.

Yiwei Thomas Hou (S'91–M'98) received the B.E. degree (*summa cum laude*) from the City College of New York in 1991, the M.S. degree from Columbia University, New York, in 1993, and the Ph.D. degree from Polytechnic University, Brooklyn, NY, in 1997, all in electrical engineering.

While a graduate student, during the summers of 1994 and 1995 he was with AT&T Bell Labs, Murray Hill, NJ, working on internetworking of IP and ATM networks. He conducted research at Bell Labs, Lucent Technologies, Holmdel, NJ, during the summer of 1996, on fundamental problems of network traffic management. Since September 1997, he has been a Research Scientist at Fujitsu Laboratories of America, Sunnyvale, CA, where he received several corporate awards for intellectual property contributions. His current research interests are in the areas of scalable architecture, protocols, and implementations for differentiated services Internet, optical networking, quality of service support for multimedia over wired and wireless Internet, and emerging service overlay infrastructure. He has authored or co-authored more than 50 referred papers in the above areas, including over 20 papers in major international journals.

Dr. Hou was awarded a National Science Foundation Graduate Research Traineeship for pursuing Ph.D. degree in high-speed networking, and was recipient of the Alexander Hessel award for outstanding Ph.D. dissertation (1997–1998 academic year) from Polytechnic University. He is a member of ACM, Sigma Xi, and the New York Academy of Sciences.

Wenwu Zhu (S'92–M'97) received the B.E. and M.E. degrees from National University of Science and Technology, Changsha, China, in 1985 and 1988, respectively. He received the M.S. degree from Illinois Institute of Technology, Chicago, and the Ph.D. degree from Polytechnic University, Brooklyn, NY, in 1993 and 1996, respectively, both in electrical engineering.

From August 1988 to December 1990, he was with Graduate School, University of Science and Technology of China (USTC), and Institute of Electronics, Academia Sinica (Chinese Academy of Sciences), Beijing, China. From July 1996 to October 1999, he was with Bell Labs, Lucent Technologies, Murray Hill, NJ. He joined Microsoft Research China, Beijing, in October 1999, where he is currently a Researcher. He has published over 50 papers in various referred conferences and journals. His current research interests are in the areas of video over IP and wireless networks, multimedia signal processing and multimedia applications.

Ya-Qin Zhang (S'87–M'90–SM'93–F'97) is currently the Managing Director of Microsoft Research China, Beijing. He was previously the Director of Multimedia Technology Laboratory at Sarnoff Corporation, Princeton, NJ (formerly David Sarnoff Research Center and RCA Laboratories). From 1989 to 1994, he was with GTE Laboratories Inc., Waltham, MA, and Contel Technology Center, Chantilly, VA. He has authored and co-authored over 200 referred papers and has 40 U.S. patents granted or pending in digital video, Internet multimedia, wireless, and satellite communications. Many of the technologies that he and his team developed have become the basis for start-up ventures, commercial products, and international standards. He has been an active contributor to the ISO/MPEG and ITU standardization efforts in digital video and multimedia.

Dr. Zhang was Editor-In-Chief for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY from July 1997 to July 1999. He was a Guest Editor for the special issue on Advances in Image and Video Compression for the PROCEEDINGS OF THE IEEE (February 1995). He serves on the editorial boards of seven other professional journals and over a dozen conference committees. He has received several industry technical achievement awards and IEEE awards, in addition to many others. He was nominated Research Engineer of the Year in 1998 by the Central Jersey Engineering Council for his "leadership and invention in communications technology, which has enabled dramatic advances in digital video compression and manipulation for broadcast and interactive television and networking applications."

Jon M. Peha (M'87-SM'96) received a Diploma from Jagiellonian University, the B.S. degree from Brown University, and the M.S. and Ph.D. degrees from Stanford University, Stanford, CA, all in electrical engineering with a computer science minor.

He has been a Professor at Carnegie Mellon University since 1991, jointly in the Department of Electrical and Computer Engineering, and the Department of Engineering and Public Policy. He also manages an active consulting practice, working for computer and telecommunications companies, consulting and patent law firms, and government agencies in the U.S. and abroad. His work spans technical and policy issues of computer and telecommunications networks, including wireless networks, and IP- or ATM-based broadband integrated-services networks which carry voice, video, and data traffic. He has been a member of Technical Staff at SRI International, AT&T Bell Laboratories, and Microsoft. He has held legislative positions in the House and Senate addressing telecommunications and electronic commerce, and has helped establish a U.S. Government program to assist developing countries with telecommunications and Internet infrastructure.

Dr. Peha was an IEEE Congressional Fellow and an AAAS Diplomacy Fellow.