

Signal Processing for Internet Video Streaming: A Review

Jian Lu
Apple Computer, Inc.

ABSTRACT

Despite the commercial success, video streaming remains a black art owing to its root in proprietary commercial development. As such, many challenging technological issues that need to be addressed are not even well understood. The purpose of this paper is to review several important signal processing issues related to video streaming, and put them in the context of a client-server based media streaming architecture on the Internet. Such a context is critical, as we shall see that a number of solutions proposed by signal processing researchers are simply unrealistic for real-world video streaming on the Internet. We identify a family of viable solutions and evaluate their pros and cons. We further identify areas of research that have received less attention and point to the problems to which a better solution is eagerly sought by the industry.

Keywords: video streaming, video compression, Internet video, rate adaptation, packet loss, error control, error recovery.

1. INTRODUCTION

Video streaming is one of the most exciting applications on the Internet. It already created a new business known as Internet broadcast, or Intericast/Webcast. Although Internet broadcast is still in its infancy, we get a glimpse of the future of broadcasting where people can enjoy not only scheduled live or pre-recorded broadcast but also on-demand and personalized programs. Stirring more excitement is the idea that anyone will be able to set up an Internet TV station and broadcast his or her video programs on the Internet just like we publish our "homepage" on the Web today. While video streaming as a core technology behind a new business is a success story, it remains very much a black art owing to its root in proprietary commercial development. Indeed, to the author's knowledge, there has been no published research literature to date that offers a comprehensive review of video streaming architecture and system, and related design issues.

Even the definition of streaming is sketchy; it is often described as a process. Generally speaking, streaming involves sending media (e.g., audio and video) from a server to a client over a packet-based network such as the Internet. The server breaks the media into packets that are routed and delivered over the network. At the receiving end, the packets are reassembled by the client and the media is played as it comes in. A series of time-stamped packets is called a *stream*. From a user's perspective, streaming differs from simple file transfer in that the client plays the media as it comes in over the network, rather than waiting for the entire media to be downloaded before it can be played. In fact, a client may never actually download a streaming video; it may simply reassemble the video's packets and play the video as the packets come in, and then discard them. To a user the essence of streaming video may lie in an expectation that the video stream the user has requested should come in continuously without interruption. This is indeed one of the most important goals for which a video streaming system is designed.

Over the last decade, IP multicast and particularly the MBone have received considerable attention from the research community. IP multicast provides a simple and elegant architecture and service model that are particularly suitable for real-time multimedia communications. The emergence of the MBone, an experimental virtual multicast backbone, has offered a testbed for researchers around the world to develop and test a suite of applications known as MBone tools for real-time multimedia communications and collaborations. See, for example, a recent article by McCanne¹ for the evolution of the MBone and its applications. Many of these applications have been designed for audio and video conferencing and broadcasting using IP multicast. On November 18, 1994, the MBone reached a festive milestone amid great fanfare when Rolling Stones broadcast the first major rock band concert on the MBone.

Despite the promise demonstrated by the MBone and a logical reasoning that IP multicast would serve as a foundation for video streaming on the Internet, commercial deployment of multicast applications and services on the public Internet has been painfully slow by the Internet standard. The bulk of the streaming media traffic generated by three major commercial

Correspondence: One Infinite Loop, MS: 302-3MT, Cupertino, CA 95014, USA; Email: jian@computer.org.

streaming systems (i.e., QuickTime, RealSystem, and Windows Media) on today's Internet consists of largely unicast streams. It is out of this paper's scope to discuss the unresolved issues that have slowed down the deployment of multicast applications and services; interested readers are referred to a recent white paper published by the IP Multicast Initiatives.² However, we want to note that even when multicast is fully deployed, a very large volume of streaming media traffic will remain to be unicast streams. This is because a large portion of the streaming media content is archived and played on-demand by users on the Internet. Multicast is of little help to on-demand streaming unless a large number of users from the same neighborhood of the Internet would request the same content simultaneously.

The purpose of this paper is to review several important issues in Internet video streaming from a signal processing perspective. For the reasons we have just mentioned above, our review will focus on unicast streaming while considering multicast as a possible extension. Drawing from the author's experience in developing one of the major commercial media streaming systems, we attempt to identify the most challenging problems in video streaming and evaluate a number of signal processing solutions that have been proposed; some of these solutions have already been implemented in experimental and/or commercial video streaming systems. To make the review useful to both researchers and application developers, we would like to clearly identify among the proposed solutions: what works in real world, what doesn't, and why? We want to note that although to a Web surfer streaming video commonly implies synchronized audio as well, video and audio are actually delivered in independent streams in a streaming session. We will not discuss streaming audio in this paper.

The rest of this paper is organized as follows. Section 2 introduces streaming system and architecture and describes what we believe to be the most challenging problems in video streaming. We also examine some important characteristics of the streaming system and the constraints that are imposed on any prospective signal processing solutions. Section 3 reviews a number of signal processing solutions to the video streaming problems in great detail. We first comment on some proposed solutions that are simply unrealistic for real-world video streaming on the Internet. Then, we evaluate a family of viable solutions, identify areas of research that have received less attention, and point to the problems to which a better solution is eagerly sought by the industry. Section 4 concludes the paper.

2. VIDEO STREAMING SYSTEM AND ARCHITECTURE

2.1. Video Streaming Model and Architecture

A schematic system diagram of Internet video streaming is shown in Figure 1 for both unicast and multicast scenarios. Figure 1(a) represents the unicast model for two types of video streaming services, *on-demand* or *live*. In on-demand streaming, pre-compressed, streamable video content is archived on a storage system such as a hard disk drive and served to a client on demand by the streaming server. In live streaming, a video stream coming out of a live source such as a video camera is compressed in real-time by the encoder. The compressed bitstream is packetized by the packetizer, and sent over the Internet by the streaming server. On the client side, the video bitstream is reassembled from the received packets by the reassembler, then decoded by the decoder, and finally rendered on a display. Noting that each client connecting to the server receives an individual stream, the disadvantages of the unicast streaming model can be explained. Clearly, the server load will increase proportionally to the number of clients connected to and served by the server. When a large number of clients try to connect to a server at the same time, the server gets overloaded, creating a "hot spot" on the Internet. Additionally, sending copies of the same data in individual streams to a large number of clients is obviously inefficient; it causes congestion on the network and deteriorates quality of services.

The multicast model of Figure 1(b) has many advantages over unicast in live video streaming (broadcasting) where a large audience could tune in for the broadcast at the same time. Instead of sending a stream to each client individually, the server sends out only one stream that is routed to one or more "group addresses". A client tunes in to a multicast group in its neighborhood to receive the stream. Since the server is decoupled from connecting to each client directly, server load does not increase with the number of receiving clients. In the case of delivering a stream to multiple groups, the stream on its route gets replicated and branched only at the fan-out points within the network. When combined with layered coding that will be discussed in Section 3, multicast provides a decentralized and highly efficient video streaming architecture.

Despite the clear advantages of multicast-based video streaming, it is employed in only a small fraction of today's Internet, primarily in Intranets within an institution or corporation. There are complex reasons for the slow deployment of multicast-based services in general.² Beyond deployment hurdles we note that a large volume of streaming video content is archived and streamed to users on demand. Multicast does not help very much in on-demand streaming where the service requests from users are random and highly desynchronized in terms of requested content and timing of these requests. Interestingly,

recent development in the so-called “content delivery network” appears to have achieved some improvements in unicast-based on-demand streaming that are reminiscent of those we have seen with multicast-based live streaming. A content delivery network such as the one developed by Akamai Technologies³ is essentially a load-balanced network of servers. The servers are deployed on the “edge” of the Internet, making them close to the end users. These “edge servers” communicate with a “source server” to update their content. A client requesting a streaming video is redirected dynamically to the closest edge server that is not overloaded. Like multicast-based video streaming, this decentralized service model can reduce “hot spots”. In principle, a content delivery network can also help unicast-based live streaming with a possibly increased latency.

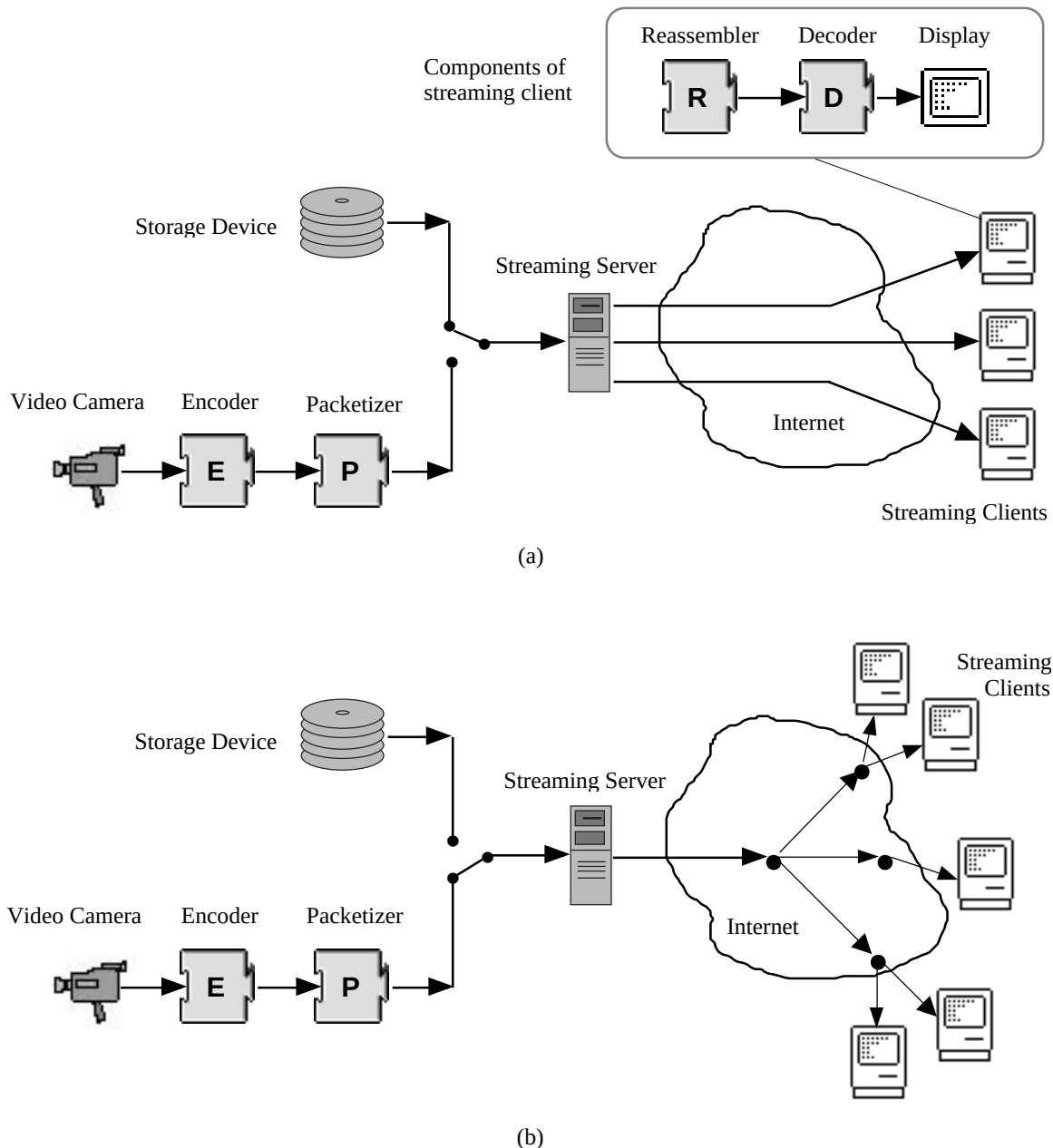


Figure 1. Video streaming system models. (a) Unicast video streaming, and (b) Multicast video streaming.

2.2. The Internet Challenge to Video Streaming

The Internet provides a great challenge to video streaming. From a communication point of view the Internet is a shared channel with a massive number of heterogeneous components. This channel is extremely volatile over space and time in terms of communication capacity and quality. To see what it means to video streaming let us look at some data that we have collected from a real-life video streaming session.

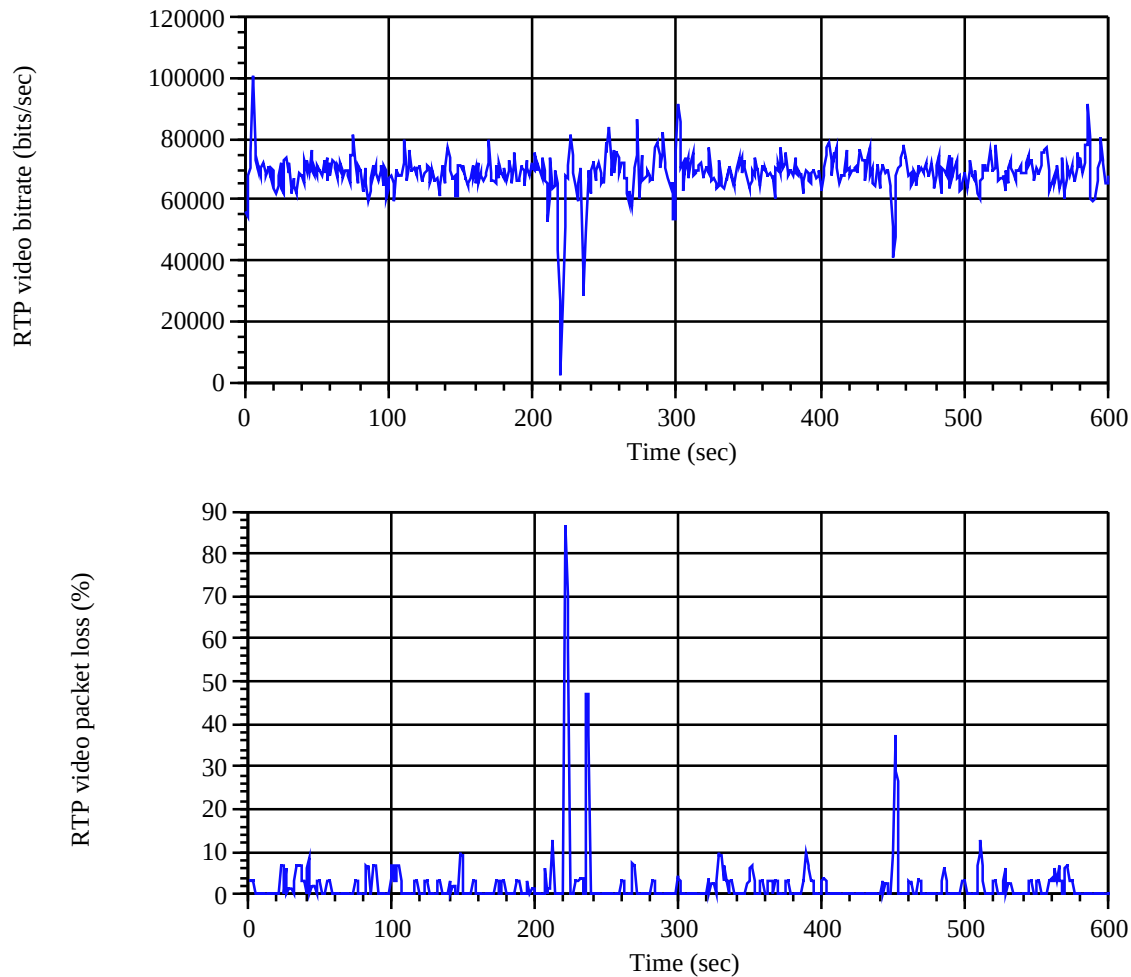


Figure 2. RTP video bitrate (top) and packet loss rate (bottom). The data was recorded at 10:08-10:18am on December 23, 1999, while watching BBC World live broadcast on QuickTimeTV from Sunnyvale, CA.

Figure 2 shows a 10-minute data log by a streaming client in Sunnyvale, California, during a session watching the BBC World channel on QuickTimeTV on December 23, 1999. The data shows the variations in bitrate and percentage of packet loss of the incoming RTP (Real-time Transport Protocol)⁴ video stream. The client has signaled to the server that it had a dual ISDN connection with up to 112kbps in bandwidth. Knowing that the client's bandwidth was also shared by streaming audio, we observed that the bitrate of the incoming video fluctuated between 60 and 80kbps in general and the packet loss ranged from 0 to 10%. There are, however, a few exceptional instances where the bitrate dropped to as low as 2kbps and packet loss rate shot up as high as 85%. It is also interesting to observe from Figure 2 that there is a strong correlation between a sudden drop in bitrate and a burst of packet loss. This appears to agree with a common-sense logic about the relationship between congestion and packet loss.

Figure 2 shows us all but a typical video streaming session when everything runs normally. There are also extreme cases. Many of us may have experience of watching streaming video that stalled completely for several times during a streaming session. In such cases, the bitrate of the incoming video is nil and the packet loss rate is 100%, making it totally worthless to watch. Generally speaking, the experience of streaming video isn't always as bad (or good) as that in our memory or in Figure 2. A person who was watching the same streaming video from another location might have had a totally different experience. The point is that the Internet channel variations are so volatile and space- and time-dependent that the observation can be different by every client on the network at every second. The data shown in Figure 2 and our experience about video streaming challenge us with two fundamental signal processing problems whose solutions can improve the performance of Internet video streaming dramatically. The first problem is concerned with how to adapt to the constantly changing bandwidth and congestion condition on the Internet. The second calls for solutions to coping with packet loss due to the best-effort service model of the current Internet. We will, for short, refer to these two problems as "rate adaptation", and "error control and recovery", respectively. Before we attempt to address these problems and evaluate a number of proposed solutions, we need to examine the constraints that have been imposed on any prospective solutions. We go back to the video streaming model and architecture of Section 2.1 and make a few important observations.

2.3. Video Streaming System Characteristics and Constraints

Many signal processing researchers are familiar with the video conferencing model thanks to the popularity of ITU H.261 and H.263 standards. In the following we review the characteristics of a video streaming system and point out its difference from video conferencing. For the comparison purpose we provide in Figure 3 a functional diagram of an IP-based peer-to-peer video conferencing system. As we have mentioned, we will focus on analyzing the unicast video streaming system.

- (a) *Latency.* One difference between streaming and conferencing is that the former does not require two-way user interaction. As such the requirement on latency is more relaxed in streaming than in conferencing. Latency requirement commonly translates into a decoder/client buffer of certain size; which is often simulated by the encoder/server for rate control. When a video streaming session is initiated, the client waits for a few seconds for its buffer to be filled by the server. This process is called pre-roll. After the pre-roll the client starts playing the video. Obviously, the client's buffer has to be kept from underflow in order for the video to continue to play without interruption. Generally speaking, streaming can afford to use a much larger buffer than that can be considered in conferencing to absorb network jitters. However, latency and smooth playback must be balanced to create a good user experience, which becomes more important in streaming live video. It is common in commercial video streaming systems to pre-roll a few seconds of data into the decoder/client buffer.
- (b) *Client-Server Model.* This makes a major distinction between the video streaming system of Figure 1(a) and the peer-to-peer conferencing system of Figure 3. In the video conferencing system of Figure 3, there is no server involved. Additionally, most components including encoder, decoder, packetizer, and reassembler at either end often reside on one piece of hardware, which is a desktop computer in the so-called desktop conferencing environment. The video streaming system in Figure 1(a), however, operates in a client-server environment. As such, a streaming server may be required to serve thousands of clients simultaneously. It is therefore necessary for the encoder to be separate from the server hardware so that the server hardware can be dedicated to serving the clients. For archived, on-demand video streaming, encoder and server are separate by default. In live streaming, encoder/server separation enables a more efficient "encode once, serve multiple times" service model. This model is almost always adopted in practice. A common implementation for live video streaming is to combine the encoder and packetizer to reside on one piece of hardware that broadcasts a stream to the server. The server simply "reflects" the stream to its clients. The client-server model and the nature of encoder/server separation make it unsuitable for many signal processing techniques that have been developed for video conferencing to be used in video streaming. We will come back to this observation later.
- (c) *Server Load Restrictions.* As we have pointed out that a streaming video server may be required to serve thousands of clients simultaneously. When the number of clients connected to a server increases, the server experiences an elevated load due to increased processing and I/O demands. Eventually the server can get overloaded and the quality of service drops. While it is relatively well understood that a good signal processing solution should not require the server to do much processing, the requirement for I/O is often overlooked. In archived, on-demand video streaming, when the number of clients increases, heavy I/O activity, particularly disk access, often becomes the bottleneck in the streaming system. This ought to be kept in mind in designing a signal processing solution for video streaming.

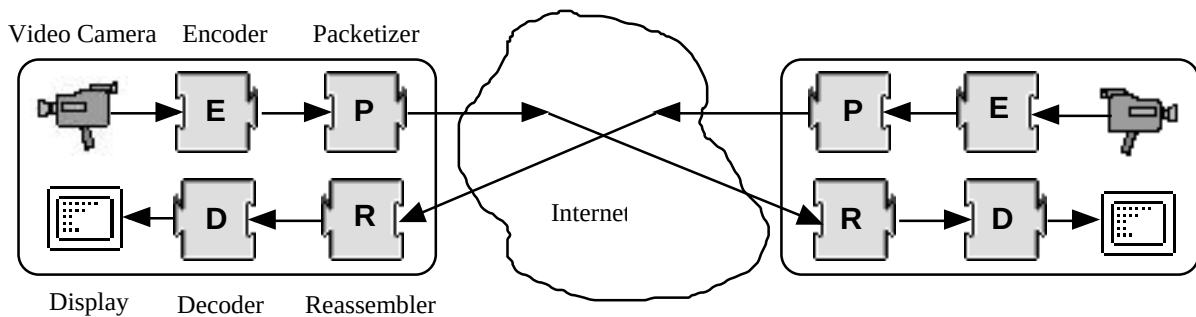


Figure 3. An IP-based peer-to-peer video conferencing system.

- (d) *The Best-Effort Service Model.* The Internet is a best-effort service network. As such, each request to send a packet is honored by the Internet at best it can. Congestion on the Internet causes packet delay and loss. Normally, a packet is routed through multiple hops on its way to the destination. At each switching point linking the hops, if there are other packets that have arrived earlier, the incoming packet is queued before it can be processed by the router, resulting a queuing delay. If the router's queue is full, the incoming packet will be dropped, resulting a packet loss. Statistics have shown that packet delay and loss often occur in bursts because an overloaded router and the affected route are likely to stay congested for some time. In video streaming, each packet is scheduled to be played at certain time; a packet that has arrived later than the scheduled time is useless and will be discarded. For this reason, we don't distinguish packets that are dropped by a router due to congestion or by the streaming client because they come too late to be used. More generally, taking a signal processing approach, we can view the Internet as a "black box". A streaming client from outside the black box can observe an effective bandwidth and packet loss rate without having to know what is going on inside the black box.
- (e) *Packetization.* A packetizer takes an encoded video bitstream and packetizes it for transmission over a transport protocol such as RTP. One of the most important aspects in designing a packetizer is to determine how to fragment the video bitstream. A well-designed fragmentation scheme with associated header information can help contain error in case of a packet loss. To date many video payload formats have been specified or proposed for RTP including those for the popular H.261, H.263 and the MPEG family. These payload formats offer fragmentation schemes of different granularities, such as Group-Of-Block(GOB) based or Macro-Block (MB) based. Besides fragmentation scheme it is also important to choose the size of a packet. On the one hand, small packets could control the adverse effect of a packet loss; on the other hand, it is desirable to send large packet to reduce packet header overhead. When streaming video over RTP, it is common to packetize the data for the maximum transmission unit (MTU) of a local area network (LAN). For the Ethernet, the MTU is about 1500 bytes, or 12000 bits. However, other restrictions may apply. For example, a RTP video packet is required to contain at most a single video frame. In other words, a new frame cannot start in the middle of a packet, though a frame can be split into multiple packets.
- (f) *Back-channel.* A back-channel in a video streaming system is a communication channel established for a streaming client to report its state to and/or request services from the server. In RTP streaming, implementation of back-channels is facilitated by the Real-time Transport Control Protocol (RTCP) that is part of the RTP specifications.⁴ The RTCP defines a type of packet for receiver report (RR). A RR packet includes such information as fraction of packets lost since the previous RR packet, cumulative number of packets lost, and interarrival jitter. It also supports profile- or application-specific extensions for including more information from the receiver. In addition to using the RR packets, it is also possible to define customized RTCP packet to carry feedback information from a streaming client. While the information about effective bandwidth and the state of packet loss can be fed back and useful to the streaming server, it is a common mistake to assume that the video encoder can always take advantage of the back-channel. We will discuss more on this in next Section.

3. SIGNAL PROCESSING SOLUTIONS FOR VIDEO STREAMING

3.1. Solutions That Do Not Fit

After reviewing the architecture and characteristics of a video streaming system in the previous Section, it becomes clear that some of the methods that have been developed for video conferencing and other applications such as wireless video are not quite applicable to video streaming. We comment on these “unfit” solutions before moving onto evaluating a family of viable solutions.

3.1.1. Feedback-Based Encoder Adaptation

Many signal processing researchers have explored the idea of having a video encoder adapt to the changing conditions in bandwidth and packet loss. It is assumed that a back-channel exists and will feed the necessary information back to the encoder which in turn adjusts its behavior in response to the bandwidth and packet loss conditions. This idea has motivated the design of the RTCP channel along with RTP.⁴⁻⁶ To date, many ways of encoder adaptation have been proposed and studied. Bolot, Turetletti, and Wakeman⁵ proposed an adaptive rate control mechanism by which a video encoder adjusts its output bitrate based on the feedback information from the receivers. There has been continued work along this direction.⁷⁻¹⁰ In the same spirit as feedback-based rate control, Steinbach, Färber, and Girod¹¹ proposed to adapt the coding mode selection at the encoder to the error conditions at the receiver for wireless video. The idea is that the error damaged MBs may be tracked by the sender through a receiver’s feedback, and a rapid error recovery can be achieved by INTRA refreshing corresponding MBs in the next frame. More recently, the scheme of adaptive coding mode selection was proposed for Internet video based on a packet loss probability model that must be estimated at the receiver and fed back to the sender¹⁰. Similar to adaptive INTRA MB refreshing and placement, it was proposed that a damaged reference picture could be dynamically replaced at the sender according to the back-channel signaling.¹² This mechanism is now supported in ITU H.263+ standard.¹³ There are still more ways of encoder adaptation. For example, unequal error protection can be applied according to packet loss rate,¹⁴ provided that a current estimate of the packet loss rate at a specific receiver is available.

Although the feedback-based encoder adaptation techniques that we have just reviewed can be effective in peer-to-peer video conferencing and wireless video, they don’t fit in a real-world video streaming system. This is due to the encoder/server separation that we discussed in the previous Section. As we mentioned there (see Figure 1(a)), in live video streaming, for efficiency purpose the encoder will encode once, producing one bitstream for the server. Of course, the system can be configured to enable the encoder to generate a bitstream adapted to each client. However, such a design will not scale up to an industry-strength video streaming system. It is well-known that real-time video compression is computationally expensive; this is one reason why the encoder is separate from the server. Considering the server may be serving thousands of clients simultaneously, it is simply unrealistic to expect the encoder to compress and output thousands of different bitstreams that are adapted to each client. In on-demand video streaming, it is obvious that encoder adaptation can not be applied because the video content is compressed off-line.

3.1.2. Complex Server Involvement

Now that we know the encoder in a video streaming system cannot afford to adapt to each client’s condition. One may wonder if the server can adapt. Generally speaking, the server does have some flexibility and capability to manipulate the video bitstream to adapt to a client’s needs. In fact, all viable solutions to rate adaptation to be reviewed in Section 3.2 rely on the server to do some kind of processing. The exact requirements for the server involvement depend on a specific adaptation technique, but for the obvious reason the server should not be tasked with complex processing on a per client basis.

An example of tasks that are too complex is for the server to convert the bitrate of a pre-encoded video bitstream dynamically by transcoding. Although a transcoder is often much less complex than a full encoder, a considerable amount of computing such as bit re-allocation, and motion vector search or refinement may be required.¹⁵⁻¹⁷ A server is unlikely to scale up to serving a large number of clients if it performs transcoding for each client. A simplified version of transcoding, called dynamic rate shaping (DRS).^{18,19} requires even less processing because it achieves bitrate reduction by selectively dropping quantized transform coefficients or blocks, or by re-quantization of the blocks.²⁰ Despite the significant reduction in computational complexity at the expense of introducing a quality problem known as “drift”, we still feel that DRS is too complex for a server to do on a per client basis. This is because DRS requires the server to have knowledge about a specific codec to process the bitstream.

3.1.3. Intraframe Error Control and Concealment at Low Bitrate

There has been much work on designing error resilient bitstream for error control. For example, the emerging MPEG-4 video standard supports such mechanisms as resynchronization marker, data partition, and reversible VLC for intraframe error control. When these features are implemented in a compressed bitstream, the decoder can regain synchronization in case of bit errors, and continue to decode a video frame even if a portion of the frame is damaged or lost. Furthermore, error concealment techniques can be used to recover a damaged or lost region within a video frame to certain degree.

Obviously, any intraframe error control and concealment technique would become useless if a whole video frame is lost completely. Unfortunately, this is often the case at low bitrate when we attempt to packetize a video bitstream for the MTU of LAN. As we mentioned in Section 2.3, the MTU for the Ethernet is 12000 bits. Consider that we compress video for a target bitrate of 36kbps at a frame rate of 4fps. This would be suitable for users with a 56kbps modem connection because we need to leave some room for audio and bandwidth fluctuation. At this target bitrate and frame rate, the average size of a video frame is 9000 bits, which is less than the MTU of 12000 bits. In practice, a motion compensated frame is often much smaller in size than 9000 bits in low bitrate video streaming. This implies that when a packet is lost, it is most likely that a whole video frame is lost. Therefore, no matter how well intraframe error control and concealment techniques are implemented in the video bitstream and at the decoder, they are of little help in low bitrate video streaming.

3.2. Solutions for Rate Adaptation

As we discussed in Section 3.1, the video encoder cannot afford to adapt to individual clients, and the streaming server is constrained to doing only light processing on a per client basis. The solutions to be reviewed below for rate adaptation fit these requirements. Additionally, these solutions do not require the server to understand a compressed video bitstream. From a server design point of view, it is advantageous to make the server codec-independent, or even media-independent. A good example of media-independent RTP streaming server is Apple's Darwin QuickTime Streaming Server (QTSS). The Darwin QTSS is open-sourced and available from Apple.²¹

3.2.1. Dynamic Stream Switching

Video streaming using multiple bitstreams encoded at different bitrates has been supported in commercial streaming systems such as Apple's QuickTime, RealNetworks' RealSystem G2, and Microsoft Windows Media. RealSystem G2 was the first to support dynamic stream switching under the name of SureStream.^{22,23} The idea is that the server can switch among multiple streams to serve a client with one that best matches the client's available bandwidth. The switching is dynamic, depending on the estimated available bandwidth reported by a client and the server's decision on whether it will respond. We want to note that in some research literature, video streaming using dynamic stream switching is called simulcast.²⁴ We prefer not to use this term because what we are describing here is different from the same word originated and used in the broadcasting industry.

In the SureStream implementation, three streams are prepared for a single target bitrate. It is worth noting that at the low end is a keyframe only stream encoded at approximately 50% of the target bitrate.²² The bitrate of this keyframe only stream can be further reduced by dropping additional keyframes since there is no dependency among the keyframes. Furthermore, the keyframe only stream is a good fallback choice when the packet loss rate is high and there is a need to resynchronize.

Although being effective and requiring only a little server involvement, dynamic stream switching makes the encoding process more complex. The mechanism of making a switchable SureStream is not published, but one can think of some feasible approaches. For example, we can build some synchronization points to facilitate switching among the multiple streams. A keyframe shared by multiple streams would provide a convenient synchronization point, provided that the buffers of these streams can be synchronized. Besides making encoding process complicated, we note that dynamic stream switching requires redundant data storage but still has limited rate adaptability because the server can switch among only a small number of available streams.

3.2.2. Scalable Video Coding

Research in scalable video coding dates back to almost a decade ago.²⁵ A similar term, namely layered coding, is also used frequently to refer scalability, and has been popular in multicast community.^{26,27} To date, support for scalable video coding

can be found in modern video compression standards such as H.263+ and MPEG-4. There has also been continued research in this direction. See, for example, a few representative papers.²⁸⁻³⁰ There are three types of scalability that can be built into a compressed video bitstream. They are *temporal scalability*, *spatial (resolution) scalability*, and *(spatial) quality scalability*. The last one is also called *SNR scalability* because the quality of a compressed picture is often measured in signal-to-noise ratio (SNR). Let's now look at what these different types of scalability can do for video streaming.

Temporal scalability is commonly achieved by including B-frames in a compressed video bitstream. B-frames are bidirectionally predicted from the forward and backward I- or P-frames, and not used as reference pictures for motion compensation in reconstructing other B- or P-frames. For this reason, a B-frame can be dropped without affecting the picture quality of other frames. However, dropping a B-frame will lower the frame rate or temporal quality of a video playback, giving the meaning of temporal scalability. Temporal scalability through B-frame placement is supported in all MPEG family video coding standards and H.263+. It should be noted that P-frames can also be dropped for rate reduction, but doing so will cause a decoder drift unless all P-frames are dropped. Use of temporal scalability for rate adaptation by dropping B- and P-frames at the server has been experimented in video streaming^{23,31}. Although effective to some degree, it is found that temporal scalability alone is often inadequate to achieve desired rate reduction.

Spatial scalability is obtained by creating a multiresolution representation of each frame in a video bitstream. In spatial scalable coding, a base layer at a reduced resolution is first encoded, and an enhancement layer representing the difference between the original picture and the base layer is also encoded. It is worth noting that an enhancement layer can be split into multiple enhancement layers so that each one is built upon the previous ones at a lower resolution. Spatial scalability is supported in MPEG-4 and H.263+. There is a limited application of spatial scalability in video streaming. This is because that for good user experience the displayed frame resolution will not be changed during a video playback. So, even if the server could drop an enhancement layer and send only the base layer to a client in rate adaptation, the client would have to interpolate the resolution-reduced base layer to the full frame size for display. This makes spatial scalability a special case of SNR scalability that we are going to discuss below.

SNR scalability is similar to spatial scalability in that a base layer and one or more enhancement layers are encoded. The difference is that in SNR scalable coding the base layer is not subsampled to a lower resolution. Instead, the base layer is coarsely quantized and the difference between the base layer and the original picture is encoded into one or more enhancement layers. Spatial scalability can be viewed as a special case of SNR scalability if we consider an interpolated base layer in spatial scalable coding as the base layer in SNR scalable coding. The SNR or spatial scalability that we have just described assumes that an enhancement layer is either available and decoded as a whole or not available at all. It is not very effective for rate adaptation in video streaming for several reasons. First, if only one or two enhancement layers are used, we have very limited rate adaptability that is similar to the situation in dynamic stream switching. Second, if more enhancement layers are used, overhead will be up and coding efficiency will drop significantly. In fact, a recent study³² on H.263+ scalability performance suggested that even with only two enhancement layers, the loss in coding efficiency of SNR scalable coding in comparison to multiple bitstream coding becomes unacceptable (about 1-1.5dB loss per layer) at low bitrate. This explains why the industry chose to use dynamic stream switching in commercial video streaming systems instead of scalable video coding options that are available in the modern video compression standards.

In recent years, research in scalable coding has centered on developing coders that can generate an "embedded bitstream". Much of the work was inspired by the design and excellent coding efficiency of the EZW³³ and SPIHT³⁴ coders for still images. An embedded bitstream is such that bits are packed in the order of "significance", a measure that is often tied to the magnitude of image coefficients in the transform domain. An intuitive way of embedding would be to order image coefficients by their magnitude and pack them in a bitstream in descending order of bit-planes. An embedded bitstream can be truncated from the end to match a reduced target bitrate. In EZW³³ and SPIHT,³⁴ the embedding is so fine-granular that one can scale the size of a bitstream in byte accuracy. For this reason, an embedded bitstream is said to have *fine-granularity scalability* (FGS). Embedding a video bitstream can be a straightforward extension to embedded image coding if a 3-D video transform is used.^{35,36} It will be more complicated if a motion compensated residual coding is employed. This is because that any truncation or scaling performed on a reference picture will result in a decoder drift. A common solution to making a drift-free embedded video bitstream is to encode two layers as we discussed before in spatial and SNR scalable coding. The base layer, which is not embedded, will be used as a reference for motion compensation, while the enhancement layer will be embedded in fine granularity. This is a compromise, for the resulting bitstream is not fully embedded, but it still offers a higher rate adaptability than non-embedded coding using multiple layers.

More recently, applications of FGS to video streaming have been reported.^{35,37,38} In October 1998, MPEG started a new work item on fine granular video scalability.³⁹ The work has since moved forward under the name of MPEG-4 FGS and converged to a solution based on bit-plane coding of DCT coefficients for enhancement layers.^{40,41} There is also work on defining an MPEG-4 streaming video application profile based on MPEG-4 FGS.^{42,43} The advantages of FGS for rate adaptive video streaming is obvious. It has excellent rate adaptability yet the requirement for server involvement is reasonably small. There remain issues to be solved, such as how to minimize overhead in layered FGS coding, but we feel that FGS, particularly when combined with temporal scalability, provides a promising solution to rate adaptation.

3.3. Solutions for Error Control and Recovery

Error resilience can be built into streaming video to control the spread of error in case of a packet loss. A large number of error resilience tools have been developed in the last decade and now included in the latest video coding standards, such as H.263+ and MPEG-4. However, many of these tools were developed with mobile and wireless video communications in mind. See, for example, a few recent reviews on this topic.⁴⁴⁻⁴⁶ In the following we evaluate the effectiveness of these tools for error control in video streaming. We look at solutions for error control in source and channel coding on the encoder/server side, and error recovery and concealment on the decoder/client side.

3.3.1. Error Control in Source Coding

One approach to error resilient video coding is to add some error control mechanisms during source coding and make them part of a compression codec design. Most modern image and video compression standards provide one or more error resilient modes that are supported by a special bitstream syntax and some coding tools. For example, H.263+ has several error resilience modes at source coding level, including Slice-Structured Mode, Independent Segment Decoding Mode, and Reference Picture Selection (RPS) Mode.¹³ MPEG-4 on the other hand provides several error resilience tools, including resynchronization markers, data partitioning, and reversible variable length codes (RVLC).⁴⁷ The error resilience performance of H.263+ and MPEG-4 has been reported in several papers.^{13,48,49} Here we want to note that all of these tools except the RPS mode in H.263+ are designed for intraframe error control. They are most suitable for controlling errors that result from loss of small random data blocks within a video frame. On the other hand, they are not very effective in controlling errors that result from a packet loss in video streaming. This is especially true for low bitrate RTP video streaming as we discussed in Section 3.1.3.

An interesting exception in the above error resilience tools is the RPS mode supported in H.263+.¹³ The RPS mode may be implemented in different ways and they don't necessarily provide a good solution to error control for video streaming. For example, one possible implementation of the RPS mode is reviewed in Section 3.1.1 and concluded to be unfit to the video streaming architecture. Nevertheless, a technique known as "video redundancy coding" (VRC)¹³ can be implemented in RPS mode and provides a good interframe error control. In VRC, multiple threads of predicted video frames are synchronized at a common keyframe, but otherwise the threads are independent of each other. For example, we can choose all odd-numbered frames to form one thread and even-numbered frames to form another thread. Both threads start at a common keyframe, say frame #0, and ends at another keyframe, say #20. During video streaming, if one packet is lost and a video frame is completely lost, we would have one damaged thread. The decoder might choose to discard the damaged thread and the video playback could continue at half of the original frame rate. On the other hand, if all packets are received error-free at the decoder, the two threads of frames can be combined to give the original sequence.

Similar in spirit to VRC, multiple description coding (MDC) has received some attention recently in image coding research community.⁵⁰⁻⁵⁵ The idea behind MDC is that if an image is a superposition of several independent descriptions, losing one or more descriptions would only degrade the quality of the reconstructed image at the decoder, but not destroy the decoding process. A simple, intuitive example of MDC is to regroup the pixels in an image into four sub-images by subsampling, and compress them independently. Clearly, if one, two, or even three sub-images are lost, we would still be able to reconstruct an image, albeit in lower quality. Although conceptually appealing, MDC has not been demonstrated to be practically useful to video streaming. There remain issues in complexity and coding efficiency that need to be addressed for MDC.

3.3.2. Error Control in Channel Coding

To enhance the error resilience of an already compressed video bitstream, error control codes can be added to the bitstream. This is commonly referred to as error control coding or channel coding. In video streaming, packetization can also be

considered part of channel coding because a well designed packetization scheme can isolate errors caused by a packet loss so that the video bitstream can be reassembled from an incomplete set of packets.

Forward Error Correction (FEC) is a classic channel coding technique that has been widely used. It can be either media independent or dependent, and implemented at different levels. In media independent FEC for packet networks, the protection is exercised at the packet level. The implementation and operation of FEC do not depend on the content of the packets, and the error recovery means an exact replacement of a lost packet. Media dependent FEC is often designed for a compressed video bitstream before packetization. For example, H.263+ specifies a FEC mode for its own bitstream.¹³ Traditional FEC, either media independent or dependent, is not very effective for video streaming for two reasons. First, FEC adds redundant information to achieve protection. Recalling that our observation from the data in Figure 2 suggests a strong correlation between congestion and packet loss, it seems to be counter-productive for us to send more data to combat packet loss. Of course, we could maintain the same output bitrate as without FEC by reducing the bandwidth for the video payload; the result would be lower video quality. A more important reason for the ineffectiveness of FEC is due to its open-looped nature. It is difficult to determine *a priori* the right amount of protection for a constantly changing packet loss pattern. In practice, the amount of protection exercised is often not enough for occasions of high packet loss rate and excessive for other time of low packet loss rate.

It is possible to assign unequal amounts of FEC to different segments of video data by a technique called Priority Encoding Transmission (PET).⁵⁶ The PET algorithm requires a user to specify the priorities of video segments which are also user defined. In Section 3.1.1, we commented on one method¹⁴ that assigns unequal FEC according to an estimated packet loss rate. Another way of specifying unequal FEC is by the importance of data segments.⁵⁷ Unequal FEC based on the content of data can be considered a case of joint source-channel coding which has been identified as an important area for continued research.⁵⁸ We feel that this is a good approach that can be combined with the FGS coding described in Section 3.2.2.

3.3.3. Error Concealment

Error concealment by postprocessing can be performed at the decoder to restore or conceal the portion of the data that is damaged or lost. Many error concealment techniques have been developed, ranging from simple operations that just copy data from the previous frame to complex processes that compute regularized image restoration. A good review on these techniques is presented in a recent paper.⁴⁴

Most error concealment techniques are based on an assumption that a video image and its motion field are smooth. For block-based coders, this assumption implies that the pixel values and the motion vector associated each coding block vary smoothly across the image. It has been demonstrated that when data corruption occurs in randomly distributed blocks or macroblocks, error concealment based on the smoothness assumption works well. Unfortunately, errors resulting from a packet loss rarely appear in randomly distributed blocks unless the blocks are interleaved randomly. It is more likely that many rows of blocks will be lost when a packet loss happens. In low bitrate video streaming, this is almost always the case, and very often all blocks in a frame are lost as we discussed in Section 3.1.3. When a large number of consecutive rows of blocks are lost, spatial or intraframe error concealment techniques do not work well. This is because the smoothness assumption does not hold for the block loss pattern we have. There is only a weak correlation between blocks that are far apart. Indeed, our experience has been that the simplest error concealment technique (i.e., copying data from the previous frame) provides the best tradeoff between enhanced visual quality and decoder complexity.

4. CONCLUSION

Research in scalable video coding appears to be maturing by the evidence that scalability options are now supported in standards such as H.263+ and MPEG-4. Nevertheless, the industry opt to take a brute-force approach to rate adaptation by using multiple bitstreams encoded at different bitrates. One wonders why. We tried in this paper to explain the reasons behind this kind of observations. The approach we take is to first identify the most challenging problems in Internet video streaming from a signal processing perspective. We see on top of our list are two types of problems, namely rate adaptation and error control and recovery, which demand good signal processing solutions. There are already a large number of signal processing techniques developed for video conferencing, an application that many signal processing researchers are familiar with. We have an opportunity to extend some of the work in video conferencing to video streaming.

However, although video streaming and conferencing share some core technologies such as video compression, the difference between the two systems is often overlooked. We reviewed carefully the characteristics of a video streaming

system and architecture, and the constraints they impose on prospective signal processing solutions. Such a review is helpful because it makes us realize that some of the existing and proposed solutions, such as those employing feedback-based encoder adaptation or requiring complex server involvement, are simply unfit in the video streaming system and architecture. Among the viable solutions, we commented on the pros and cons of different approaches, and concluded that FGS combined with temporal scalability coding could provide a promising solution for rate adaptation. As for error control and recovery, we found that the existing solutions are inadequate. Particularly, we found that there exist few good solutions that are designed for error control and concealment in such circumstances that a large number of, or all, blocks in a video frame are lost. Unfortunately, this is often the case in video streaming when a packet loss happens, especially at low bitrate.

It must be pointed out that we have looked at only signal processing solutions for video streaming. There are other solutions that do not fall into the scope of signal processing. One example is the deployment of content delivery networks such as Akamai's. Even within the scope of signal processing, the coverage of our review is almost certain to be incomplete. Nonetheless, we hope this paper serves two purposes: for application developers to learn what signal processing solutions are available for video streaming, and for signal processing researchers to better understand video streaming systems and architecture, and their associated constraints on signal processing solutions.

REFERENCES

1. S. McCanne, "Scalable multimedia communication using IP multicast and lightweight sessions," *IEEE Internet Computing*, Vol. 2, No.2, pp. 33-45, Mar. 1999.
2. Stardust.com, Inc, "The evolution of multicast," IP Multicast Initiatives White Paper, December 1999.
<http://www.ipmi.org/>
3. Akamai Technologies, Inc. <http://www.akamai.com/>
4. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," RFC 1889, Internet Engineering Task Force (IETF), Jan. 1996.
5. J.-C. Bolot, T. Turetti, and I. Wakeman, "Scalable feedback control for multicast video distribution in the internet," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (London, England), pp. 58-67, Aug. 1994.
6. I. Busse, B. Deffner, and H. Schulzrinne, "Dynamic QoS control of multimedia applications based on RTP," *Computer Communications*, Jan. 1996.
7. J-C. Bolot, T. Turetti, "Experience with rate control mechanisms for packet video in the Internet," *Computer Communication Review*, Vol. 28, No. 1, Jan. 1998.
8. Y.J. Chung, J.W. Kim, and C.-C.J. Kuo, "Network friendly video streaming via adaptive LMS bandwidth control," in *Applications of Digital Image Processing XXI, Proc. SPIE*, vol. 3460, pp. 448-456, July 1998.
9. Y.J. Chung, J. Kim, and C.-C.J. Kuo, "Real-time streaming video with adaptive bandwidth control and DCT-based error concealment," *IEEE Trans. CAS II: Analog and Digital Signal Processing*, Vol. 46, No. 7, pp. 951-956, July 1999.
10. F. Le Léanec, F. Toutain, C. Guillemot, "Packet loss resilient MPEG-4 compliant video coding for the Internet," *Signal Processing: Image Communications*, Vol. 15, pp. 35-56, Sept. 1999.
11. E. Steinbach, N. Färber, and B. Girod, "Standard compatible extension to H.263 for robust video transmission in mobile environments," *IEEE Trans. CAS for Video Technol.*, Vol. 7, No. 6, pp. 872-881, Dec. 1997.
12. S. Fukunaga, Y. Matsumura, and T. Nakai, "Error resilient video coding controlled by backward channel signaling," *Signal Processing: Image Communications*, Vol. 14, pp. 531-540, May 1999.
13. S. Wenger, G. Knorr, J. Ott, and F. Kossentini, "Error resilience support in H.263+," *IEEE Trans. Circuits and Sys. for Video Tech.*, Vol. 8, No. 7, pp. 867-877, Nov. 1998.
14. U. Horn, K. Stuhlmüller, M. Link, and B. Girod, "Robust Internet video transmission based on scalable coding and unequal error protection," *Signal Processing: Image Communications*, Vol. 15, pp. 77-94, Sept., 1999.
15. H. Sun, W. Kowk, and J.W. Zdepski, "Architectures for MPEG compressed bitstream scaling," *IEEE Trans. CAS for Video Technol.*, Vol. 6, No. 2, pp. 191-199, April 1996.
16. G. Keesman, R. Hellinghuizen, F. Hoeksema, and G. Heideman, "Transcoding of MPEG bitstreams," *Signal Processing: Image Commun.*, Vol. 8, pp. 481-500, Sept. 1996.
17. P.A.A. Assuncao and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams," *IEEE Trans. CAS for Video Technol.*, Vol. 8, No. 8, pp. 953-967, Dec. 1998.
18. S., Jacobs and A. Eleftheriadis, "Streaming video on the Web using dynamic rate shaping," *J. Visual Commun. Image Representation*, Vol. 9, No. 3, pp. 211-222, Sept. 1998.
19. W. Zeng and B. Liu, "Rate shaping by block dropping for transmission of MPEG-precoded video over channels of dynamic bandwidth," in *Proc. ACM Multimedia '96*, pp. 385-393, Nov. 1996.

20. Y. Nakajima, H. Hori, and T. Kanoh, "Rate conversion of MPEG coded video by re-quantization process," in *Proceedings of IEEE Intl. Conf. Image Proc.*, Washington, DC, Oct. 1995, pp. 408-411.
21. Darwin Streaming Server Project, <http://www.publicsource.apple.com/projects/streaming/>
22. A. Lippman, "Video coding for multiple target audiences," in *Visual Communications and Image Processing'99, Proc. SPIE*, Vol. 3653, pp. 780-782, Jan. 1999.
23. "SureStream™—Delivering superior quality and reliability," RealNetworks White Papers, <http://www.realnetworks.com/devzone/library/whitepapers/surestrm.html>
24. B. Girod, N. Färber, and K. Stuhlmüller, "Internet video streaming," in: F. De Natale and S. Pupolin (eds.), *Multimedia Communications*, Springer, 1999, pp. 547-556.
25. K.M. Uz, M. Vetterli, and D. LeGall, "Interpolative multiresolution coding of advanced television with compatible subchannels," *IEEE Trans. CAS for Video Technol.*, Vol. 1, pp. 86-99, Mar. 1991.
26. S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," *ACM SIGCOMM'96*, Stanford, CA, Aug. 1996.
27. S. McCanne, M. Vetterli, and V. Jacobson, "Low complexity video coding for receiver driven layered multicast," *IEEE J. Selected Areas in Communications*, Vol. 15, No. 6, pp. 983-1001, Aug. 1997.
28. D. Taubman and A. Zakhor, "Multi-rate 3-dimensional subband coding of video," *IEEE Trans. Image Proc.*, Vol. 3, pp. 572-588, Sept. 1994.
29. J.Y. Tham, S. Ranganath, and A.A. Kassim, "Highly scalable wavelet-based video codec for very low bit-rate environment," *IEEE J. Selected Areas in Communications*, vol. 16, No. 1, pp. 12-27, Jan. 1998.
30. K. Chen and E.J. Delp, "Wavelet based rate scalable video compression," *IEEE Trans. CAS for Video Technol.*, Vol. 9, No. 1, pp. 109-22, Feb. 1999.
31. K. Sripanidkulchai and T. Chen, "Network-adaptive video coding and transmission," in *Visual Communications and Image Processing'99, Proc. SPIE*, Vol. 3653, pp. 854-861, Jan. 1999.
32. L. Yang, F.C.M. Marins, and T.R. Gardos, "Improving H.263+ scalability performance for very low bit rate applications," in *Visual Communications and Image Processing'99, Proc. SPIE*, Vol. 3653, pp. 768-779, Jan. 1999.
33. J.M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing*, Vol. 41, No. 12, pp. 3445-3462, Dec. 1993.
34. A. Said and W.A. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. CAS for Video Technol.*, Vol. 6, pp. 243-250, June 1996.
35. W.-T. Tan and A. Zakhor, "Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol," *IEEE Trans. Multimedia*, Vol. 1, No. 2, pp. 172-186, June 1999.
36. B.-J. Kim, Z. Xiong, W.A. Pearlman, and Y.S. Kim, "Progressive video coding for noisy channels," *Journal of Visual Communication and Image Representation*, Vol.10, No. 2, pp.173-185, June 1999.
37. W. Li, "Bit-plane coding of DCT coefficients for fine granularity scalability," MPEG contributions, m3989, Atlantic City, NJ, Oct. 1998.
38. H. Radha, Y. Chen, K. Parthasarathy, and R. Cohen, "Scalable Internet video using MPEG-4," *Signal Processing: Image Communications*, Vol. 15, No. 1-2, pp. 95-126, Sept. 1999.
39. Y. Chen, C. Dufour, J.-F. Vial, A. Zakhor, W. Li, J. Liang, and B. Schuster, "Description of mini-experiment of fine granular video scalability," MPEG contributions, m3881, Atlantic City, NJ, Oct. 1998.
40. ISO/IEC JTC1/SC29/WG11, "Text of ISO/IEC 14496-2 FGS Amendment Working Draft 3.0," MPEG Document N3095, Maui, USA, Dec. 1999.
41. W. Li, F. Ling, and X. Chen, "Fine granularity scalability in MPEG-4 for streaming video," to appear in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS'2000)*, Geneva, Switzerland, May 28-31, 2000.
42. W. Li, et al, "Streaming video profile," MPEG Contributions, m4990, Melbourne, Australia, Oct. 1999.
43. W. Li, et al, "Streaming video profiles based on fine-granularity-scalability," MPEG Contributions, m5582, Maui, USA, Dec. 1999.
44. Y. Wang, and Q. Zhu, "Error control and concealment for video communications: a review," *Proc. IEEE*, Vol. 86, No. 5, pp. 974-997, May 1998.
45. J. Villasenor, Y.-Q. Zhang, and J. When, "Robust video coding algorithms and systems," *Proc. IEEE*, Vol. 87, No. 10, pp. 1724-1733, Oct. 1999.
46. B. Girod and N. Färber, "Feedback-Based Error Control for Mobile Video Transmission", *Proc. IEEE*, Vol. 87, No. 10, pp. 1707-1723, Oct. 1999.
47. R. Talluri, "Error resilient video coding in the MPEG-4 standard," *IEEE Commun. Mag.*, Vol. 36, pp. 112-119, June 1998.
48. R. Talluri, I. Moccagatta, Y. Nag, and G. Cheung, "Error concealment by data partitioning," *Signal Processing: Image Communication*, vol.14, No.6-8, pp. 505-518, May 1999.

49. L. Ducla-Soares, and F. Pereira, "Error resilience and concealment performance for MPEG-4 frame-based video coding," *Signal Processing: Image Communication*, vol.14, No.6-8, pp. 447-472, May 1999.
50. Y. Wang, M.T. Orchard, and A.R. Reibman, "Multiple description image coding for noisy channels by pairing transform coefficients," in *Proc. First IEEE Workshop on Multimedia Signal Processing*, New York, NY, June 1997. Pp. 419-424.
51. D.M. Chung and Y. Wang, "Multiple description image coding using signal decomposition and reconstruction based on lapped orthogonal transforms," *IEEE Trans. CAS for Video Technol.*, vol. 9, No. 6, pp. 895-908, Sept. 1999.
52. S.D. Servetto, K. Ramchandran, V.A. Vaishampayan, and K. Nahrstedt, "Multiple description wavelet based image coding," submitted to *IEEE Trans. Image Proc.*, Nov. 1998.
53. V.K. Goyal, J. Kovacevic, R. Arian, and M. Vetterli, "Multiple, description transform coding of images," in *Proceedings of IEEE Conference on Image Processing*, Chicago, IL, Oct. 1998, pp. 674-678.
54. X. Yang and K. Ramchandran, "Optimal multiple description subband coding," in *Proceedings of IEEE Conference on Image Processing*, Chicago, IL, Oct. 1998, pp. 654-658.
55. R. Chellappa, and M. Srinivasan, "Multiple description subband coding," in *Proceedings of IEEE Conference on Image Processing*, Chicago, IL, Oct. 1998, pp. 684-688.
56. A. Albanese, J. Bloemer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Trans. Inform. Theory*, Vol. 42, No. 6, pp. 1737-1747, Nov. 1996.
57. A.E. Mohr, E.A. Riskin, and R.E. Ladner, "Graceful degradation over packet erasure channels through forward error correction," in *Proc. Data Compression Conference*, Snowbird, UT, Mar. 1999, pp. 92-101.
58. Workshop Report: 1999 NSF Sponsored Workshop on Joint Source-Channel Research, Oct. 1999.