Packet Loss Recovery for Streaming Video

Nick Feamster and Hari Balakrishnan
M.I.T. Laboratory for Computer Science
Cambridge, MA 02139
{feamster,hari}@lcs.mit.edu
http://nms.lcs.mit.edu/projects/videocm/

ABSTRACT

While there is an increasing demand for streaming video applications on the Internet, various network characteristics make the deployment of these applications more challenging than traditional TCP-based applications like email and the Web. Packet loss can be detrimental to compressed video with interdependent frames because errors potentially propagate across many frames. While latency requirements do not permit retransmission of all lost data, we leverage the characteristics of MPEG-4 to *selectively* retransmit only the most important data in the bitstream. When latency constraints do not permit retransmission, we propose a mechanism for recovering this data using postprocessing techniques at the receiver. We quantify the effects of packet loss on the quality of MPEG-4 video, develop an analytical model to explain these effects, present a system to *adaptively* deliver MPEG-4 video in the face of packet loss and variable Internet conditions, and evaluate the effectiveness of the system under various network conditions.

1. INTRODUCTION

Streaming media is becoming increasingly prominent on the Internet. Although some progress has been made in media delivery, today's solutions (e.g., RealPlayer and Windows Media Player) [35, 43] are proprietary, inflexible, and do not provide the user with a pleasant viewing experience. The lack of an open framework hampers innovative research, particularly in the area of video delivery that *adapts* to changing network conditions. While today's streaming applications are closed and proprietary, the emerging MPEG-4 standard is gaining increasing acceptance and appears to be a promising open standard for Internet video [13, 17, 20, 28, 31, 37].

In this paper, we describe a system that enables the adaptive unicast delivery of streaming MPEG-4 video by responding to varying network conditions. This paper primarily focuses on techniques to deal with packet losses, which are common on the Internet.

Inter-frame video compression algorithms such as MPEG-4 exploit temporal correlation between frames to achieve high levels of compression by independently coding reference frames, and representing the majority of the frames as the difference from each frame and one or more reference frames. However, these algorithms suffer from the well-known *propagation of errors* effect, because errors due to packet loss in a reference frame propagate to all of the dependent difference frames. The resulting stream is not even resilient to small amounts of packet loss. There is a fundamental tradeoff between bandwidth efficiency (obtained by compression) and error resilience (obtained by coding or retransmission). Inter-frame compression schemes (such as MPEG-4) achieve significant compression of bits in comparison to other schemes that do not exploit temporal correlation (such as motion JPEG [21]), but they are also less resilient to packet loss because of the dependencies that exist between data from different frames. While many methods have been proposed to add redundancy to the bitstream to allow for more effective error correction [9, 10, 51, 54], they also reduce much of the gains garnered from compression.

Errors in reference frames are more detrimental than those in derived frames due to propagation of errors and should therefore be given a higher level of protection than other data in the bitstream. One solution is to add redundancy to more important portions of the bitstream, or to code more important portions of the stream at a relatively higher bitrate [1, 25, 39, 50]; however, this approach reduces compression gains and in many cases does not adequately handle packet losses that occur in bursts.

Prior work has gathered experimental results that describe packet loss characteristics for MPEG video and suggest the need for better error recovery and concealment techniques [11]. Motivated by prior analysis, as well a general model we have developed to explain the effects of packet loss on MPEG video, we have developed a system that uses *receiver-driven selective reliability* in conjunction with receiver postprocessing to efficiently recover from packet losses in reference frames.

Some researchers have argued that retransmission-based error resilience is infeasible for Internet streaming because retransmission of lost data takes at least one additional round-trip time, which may be too much latency to allow for adequate interactivity [10, 51, 54]. However, because of the nature of inter-frame compression, certain types of packet loss can be excessively detrimental to the quality of the received bitstream. We show that such losses can be corrected via retransmission without significantly increasing delay, using only a few frames' worth of extra buffering. In a streaming system that transports video bitstreams with inter-dependent frames, *careful* retransmission of lost packets provides significant benefits by alleviating the propagation of errors.

While our system primarily focuses on the use of selective retransmission for packet loss recovery, we also show how our system allows selective retransmission to be used in *conjunction* with other error control and concealment techniques. When delay or transient loss is prohibitively high, retransmission of lost packets may not always be feasible. In these circumstances, we propose a mechanism for recovering data in reference frames using postprocessing at the receiver. Specifically, we propose a scheme for reconstructing important missing data in reference frames using texture and motion information from surrounding frames. Our motion-compensated recovery techniques allow partial recovery of important data, limiting propagation of errors without imposing the buffering constraints required for selective retransmissions.

To recover from packet losses, our system uses application-level framing (ALF) [16]. Because dealing with data loss is application-dependent, the application, rather than the transport layer, is most capable of handling these losses appropriately. Moreover, in the case of video, the receiver is best-equipped to make decisions with regard to packet loss recovery (e.g., whether to request a retransmission, to use postprocessing and error concealment, or simply to drop the frame). The ALF principle states that data must be presented to the application in units that are both meaningful to that application and independently processible. These units, called application data units (ADUs), are also the unit of error recovery. We have used this philosophy in our design of a backwards-compatible *receiver-driven* selective retransmission extension to RTP [48] called SR-RTP. This extension provides semantics for requesting the retransmission of independently-processible portions of the bitstream and a means for reassembling fragmented portions of independently processible units. ALF allows the application to be notified when incomplete frames arrive and control error concealment decisions.

In addition to providing a means for recovering from packet loss, a video streaming system for the Internet should adapt its sending rate and the quality of the video stream it sends in accordance with the available bandwidth. It is widely believed that the stability of the modern Internet is in large part due to the cooperative behavior of the end hosts implementing the window increase/decrease algorithms described in [2, 29]. A video streaming system should deliver video at the highest possible quality for the available bandwidth and share bandwidth fairly with TCP flows. To accomplish this, our video server uses information in RTCP receiver reports to discover lost packets and round-trip time variations and adapt its sending rate according to a certain congestion control algorithm using the Congestion Manager (CM) [3, 5] framework. Rapid oscillations in the instantaneous sending rate often degrade the quality of the received video by increasing the required buffering and inducing layer oscillations. To achieve smoothing of video quality, our system exploits binomial congestion control algorithms [6, 19], a family of TCP-friendly congestion control algorithms that reduce rate oscillation.

This paper focuses on packet loss recovery and describes our implementation that enables this framework. We describe:

- A system employing SR-RTP, receiver postprocessing, and the CM to enable the adaptive transmission of MPEG-4 video in the face of packet loss, bandwidth variation, and delay variation.
- An analytical model to explain the effects of packet loss on the overall quality of an MPEG-4 bitstream and an evaluation of our system based on this model.

Section 2 presents an overview of the MPEG-4 video compression standard, derives a model for propagation of error due to packet loss based on empirical observations, and quantifies the effects of packet losses in reference frames. Section 3 presents our framework and implementation for streaming multimedia data in a manner that is resilient to packet loss and adaptive to varying network conditions. Section 4 discusses experiments that we performed with our streaming system that demonstrate situations in which selective reliability can provide considerable benefit. We discuss related projects in Section 5 and conclude in Section 6.

2. MODEL

In this section, we develop the case for *selective reliability*, whereby certain portions of an MPEG-4 bitstream can be transmitted reliably.

Prior work has proposed protocols that use selective retransmission for recovering from bit errors [36]. Others have gathered empirical data on the effect of transmitting MPEG video over the Internet [11]. In our work, we derive a *general packet loss model* that explains the quality degradation of MPEG-4 in the face of packet loss as seen on the Internet, validate our packet loss model with experiments, and show through analysis and experiments how our system provides performance benefits. This section presents our packet loss model and quantifies the benefits of selective retransmission for packet loss recovery.

We describe the problem in detail, present an analysis of video quality in the presence of packet loss, make a quantitative case for selective reliability, and argue how selective reliability can be used in conjunction with other loss recovery techniques.

2.1 Problem Description

We start with a description of the MPEG-4 video compression standard, then analyze the quality degradation caused by packet loss. We focus on whole packet erasures, modeling congestion-related loss, rather than bit corruption.

2.1.1 MPEG-4 Background

The MPEG-4 compression standard achieves high compression ratios by exploiting both spatial and temporal redundancy in video sequences. While spatial redundancy can be exploited by simply coding each frame separately (just as it is exploited in still images), many video sequences exhibit temporal redundancy, as two consecutive frames are often very similar. An MPEG bitstream takes advantage of this by using three types of frames.[1]

"I-VOPs" or "I-frames" are *intra-coded* images, coded independently of other frames in a manner similar to a JPEG image. These are *reference frames* and do not exploit temporal redundancy. MPEG uses two types of dependent frames: predictively coded frames ("P-VOPs" or "P-frames"), and bi-directionally coded frames ("B-VOPs" or "B-frames"). P-frames are coded predictively from the closest previous reference frame (either an I-frame or a preceding P-frame), and B-frames are coded bi-directionally from the preceding and succeeding reference frames.

2.1.2 Error Propagation

The ability to successfully decode a compressed bitstream with inter-frame dependencies depends heavily on the receipt of reference frames (i.e., I-frames, and to a lesser degree P-frames). While the loss of one or more packets in a frame can degrade its quality, the more problematic situation is the propagation of errors to dependent frames. An example of error propagation is shown in Figures 1 and 2; the rectangular patch near the bottom of Figure 1 is the result of a single loss in an I-frame (no local error concealment is done in this example). This error spreads to neighboring frames as well, as shown in Figure 2 which depends on several preceding differentially coded frames.

Figure 3 shows the evolution of frame-by-frame PSNR[2] for the luminance (i.e., "Y") component as a function of the original raw frame number for various packet loss rates. The evolution for a

[1] In fact, MPEG-4 codes each independent object within a frame as a "VOP", or "video object plane", but for simplicity and without loss of generality, we will use the terms *frame* and *VOP* interchangeably.

[2] Peak Signal to Noise Ratio (PSNR) is a coarse and controversial indicator of picture quality that is derived from the root mean squared error (RMSE). The PSNR for a degraded $N_1 \times N_2$ 8-bit image f' from the original image f is computed by the formula $20 \log_{10} \frac{255}{\left(\frac{1}{N_1 N_2} \sum_{x=0}^{N_1-1} \sum_{y=0}^{N_2-1} [f(x,y) - f'(x,y)]^2\right)^{1/2}}$.

Figure 1: I-Frame #48 from the "coastguard" stream with packet loss. Y PSNR: 21.995697

Figure 2: B-Frame #65 from "coastguard" stream showing propagation of errors. Y PSNR: 17.538345

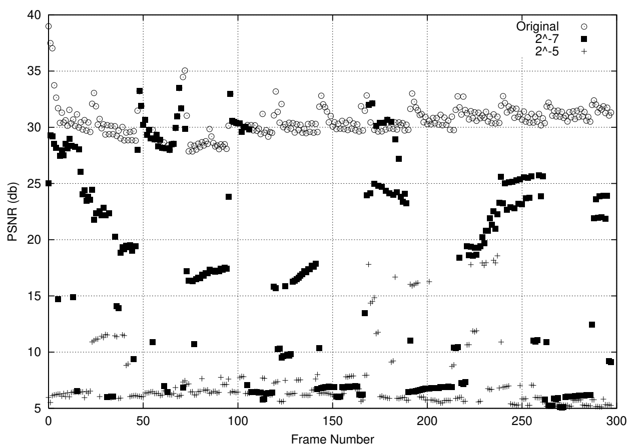

Figure 3: Y PSNR values measured against the original frames with varying degrees of packet loss. As the packet loss rate increases, the frame-by-frame PSNR drops dramatically, thus lowering the fraction of frames received which are suitable for display.

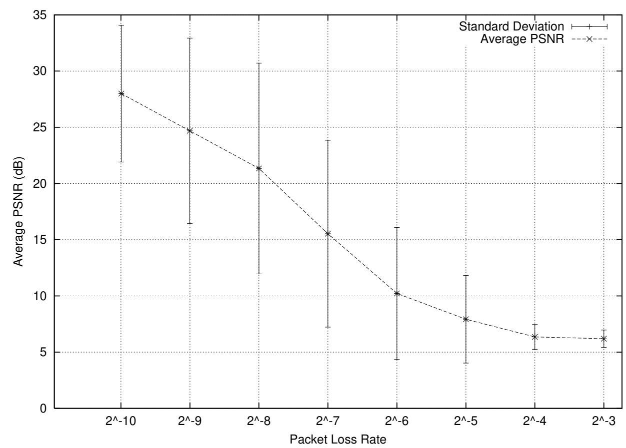

Figure 4: Average PSNR over 100 seconds of 30 fps video as a function of the packet loss rate. As the packet loss rate increases, the overall quality of the bitstream severely degrades.

decoded bitstream with no packet loss is also included as a baseline. As the packet loss rate increases, the quality (in PSNR) of an increasing number of the decoded frames becomes too poor for viewing. We generalize this in Figure 4 by averaging the observed frame rate over time for a given video sequence. If we assume that the viewer can only tolerate frames that are at least a certain PSNR quality, and that frames below such a quality are not pleasing to view, we can show how packet losses degrade frame rate. We model this in Section 2.2.

2.2 Packet Loss Model

We now analyze the effects of packet loss on the observed frame rate at the receiver. Using these results, we argue that under certain circumstances selective reliability can improve the quality of the received bitstream.

Our model is based on two premises. The first is that packet loss will result in degradation of quality of the video stream at the receiver; i.e., that there is some signal loss caused by the loss of a packet. This is true in general, unless packet-level FEC or erasure codes are extensively used (but notice that such mechanisms do reduce the effective bitrate of the transmission). The second premise is that, below a certain PSNR level, frames are not viewable by users. While it is true that PSNR does not necessarily accurately model perceptual quality, it has been extensively used in the literature. Because the viewability threshold varies from sequence to sequence, we perform our analysis for several PSNR thresholds. We note that this general method can be used for any quality metric, not just PSNR.

2.2.1 Experimental Results

To better understand how packet loss affects the quality of video received by a client, we will first look at how packet loss affects the average PSNR of a video sequence. In addition to PSNR, the quality of delivered video depends on the *frame rate*, which is the rate

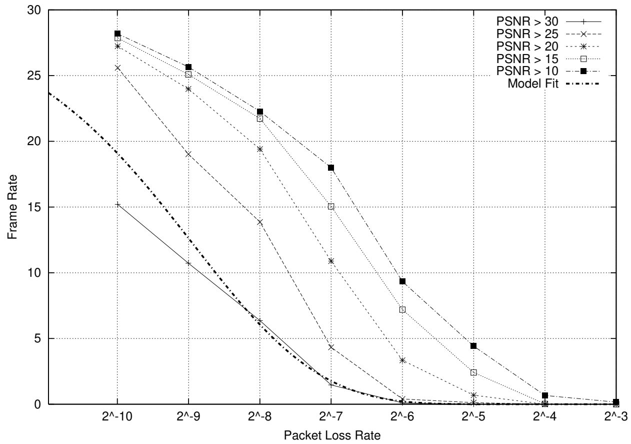

Figure 5: The effects of packet loss on frame rate.

at which frames whose PSNR is above some threshold is played at the receiver.

Figure 4 shows how increasing packet loss rates greatly degrade the overall quality of the received pictures. In this experiment run on the "coastguard" stream, packets were artificially dropped according to a Bernoulli process at each loss rate, ranging from 2^{-10} to 2^{-3} in powers of two. The vertical error bars plot the standard deviation of received PSNR across 100 seconds of a 30 fps video sequence. PSNR values of smaller than 20 dB are generally unviewable, which means that even individual frames are not particularly useful without a correction mechanism at packet loss rates larger than 2^{-8}.

Figure 5 shows the measured results of the resulting frame rates as a function of the packet loss rate, with one curve per PSNR threshold. For the "coastguard" stream, we measure the number of frames per second, on average, that are above a set PSNR threshold and plot it as a function of the Bernoulli packet loss rate. As the picture quality threshold increases for a given packet loss rate, the number of acceptable frames in the sequence (the frame rate) decreases. For a given picture quality threshold, an increase in the packet loss rate results in a considerable reduction in the frame rate. This graph shows that as the packet loss rate p increases, the frame rate degrades roughly as $f(p) = \alpha(1-p)^c$ for some constants α and c.

To understand better how packet loss affects frame rate, we have developed a simple analytic model to explain these results. We find that the analytic model matches the experimental results rather well.

2.2.2 Analytic Model

Our goal is to derive a relationship between the packet loss rate p and the observed frame rate f. When calculating the frame rate, f, we assume that if the quality of a frame (i.e., PSNR) falls beneath a certain threshold, then the frame is "dropped." We express the observed frame rate f as $f_o(1-\phi)$, where ϕ is the "frame drop rate", the fraction of frames dropped, and f_o is the frame rate of the original bitstream in frames per second (e.g., 30 fps).

The frame drop rate ϕ is a sum of conditional probabilities:

$$\phi = \sum_i P(f_i) \cdot P(\overline{F}|f_i) \tag{1}$$

where i runs over the three possible frame types (I, P, and B), and $\overline{F}$ represents the event that a frame is "useless" because it falls below

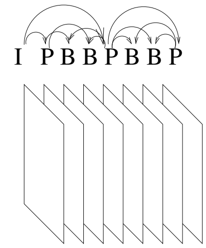

Figure 6: Frame dependencies in an MPEG bitstream.

a certain quality threshold. f_i is the event that the corresponding frame is of type i. The *a priori* probabilities $P(f_i)$ can be determined directly from the fractions of bitstream data of each frame type.

Next, we express the conditional probabilities $P(\overline{F}|f_i)$ for each frame type f_i. We do this under the simplifying assumption that if even one packet within a frame is lost (or the effects of one packet loss from a reference frame are seen), that the frame is rendered useless (relaxing this assumption makes the analysis more complicated, although the general form of the result does not change). In this case, determining $P(\overline{F}|I)$ is simply a Bernoulli random variable, expressible as one minus the probability that no packets are lost within the I-frame. Thus,

$$P(\overline{F}|I) = 1-(1-p)^{S_I} \tag{2}$$

where S_I is the number of packets on average in an I-frame, and p is the packet loss rate.

The conditional probabilities for P and B frames are somewhat more involved, and require an understanding of the inter-frame dependencies in MPEG video. These dependencies are shown in Figure 6. Every P-frame depends on the preceding I or P frame in the "group of video object planes" (GOV), and every B-frame depends on the surrounding two reference frames (the closest two I or P frames that surround it). Thus, the successful decoding of a P-frame depends on *all* I and P frames that precede it in the GOV, and the successful decoding of a B-frame depends on the successful decoding of the surrounding reference frames, which implies the successful decoding of all preceding I and P frames *and* the succeeding I or P frame. These dependencies can be expressed in the following relationships:

$$P(\overline{F}|P) = \frac{1}{N_P}\sum_{k=1}^{N_P}\left(1-(1-p)^{S_I+kS_P}\right) \tag{3}$$

$$P(\overline{F}|B) \leq \frac{1}{N_P}\sum_{k=1}^{N_P}\left(1-(1-p)^{S_I+(k+1)S_P+S_B}\right) \tag{4}$$

Here, S_P is the average number of packets in a P-frame, N_P is the number of P-frames in a GOV, and S_B the number of packets in a B-frame. These simplify to the following closed form expressions:

$$P(\overline{F}|P) = 1-\frac{(1-p)^{S_I}}{N_P\left(1-(1-p)^{S_P}\right)}\left(1-(1-p)^{S_P N_P}\right) \tag{5}$$

$$P(\overline{F}|B) \leq 1-\frac{(1-p)^{S_I+S_P+S_B}}{N_P\left(1-(1-p)^{S_P}\right)}\left(1-(1-p)^{S_P N_P}\right) \tag{6}$$

We can then obtain an expression for ϕ using equations 1, 2, 5, and 6. Given this expression for ϕ, we can determine $f = f_o(1-\phi)$, given values of N_P, S_I, S_P, S_B, and f_o. We have graphed this curve in Figure 5 using the parameters of the "coastguard" bitstream we used in our experiments; our model matches

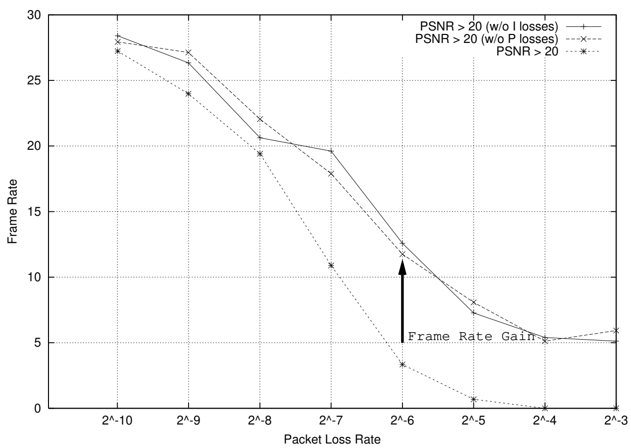

Figure 7: The effects of recovering reference frame data on frame rate. By recovering packet losses in I-frames, the frame rate for a given acceptable quality can be increased up to 3 times. This graph also shows that recovering all P-frames is roughly as effective as recovering only I-frame data.

the experimental results rather closely.

This result can by extended to derive analytical results for lower PSNR thresholds, assuming that there is a relationship between the number of packets lost in a particular frame and PSNR degradation. Instead of performing the calculations so that one packet loss results in a "useless" packet, we can generalize to allow for n losses, with a larger value of n corresponding to a lower threshold PSNR.

2.3 Selective Reliability is Beneficial

We have established that packet loss substantially affects the frame rate of a received video sequence and would like to somehow recover some of these packet losses. While it would be preferable to be able to recover all packets, the latency of the network, as well as bandwidth constraints, limit this possibility. Fortunately, the structure of an MPEG-4 bitstream allows us to capitalize on the notion that some data is more important that others. By judiciously recovering *some* of the more important data in the bitstream, we can substantially increase the observed frame rate.

Figure 7 shows the effects of recovering lost I-frame packets via retransmissions on the effective frame rate for a given PSNR threshold of 20 dB. Recovering I-frame data can increase the effective frame rate significantly, in some cases by up to three times the frame rate without recovery. One upper curve shows the effects of recovering only I-frame data, whereas the other curve shows the effects of recovering only P-frame data. In both cases, the frame rate is significantly increased; this shows that recovering only the I-frame packets in a group of pictures results in comparable gains to recovering all P-frame data across a group of pictures. Therefore, by recovering either the I-frame data or the P-frame data via selective reliability, it is possible to significantly improve the quality of received video—there is no real need to recover all missing packets. Furthermore, recovering missing B-frame packets is not particularly useful.

3. SYSTEM ARCHITECTURE

In this section, we describe our architecture for unicast streaming of MPEG-4 video that implements techniques for selective reliability and bandwidth adaptation. This architecture has been developed as the basis for next-generation streaming systems such as the DivX

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

V	P	X	CC	M	Payload Type	Sequence Number
Timestamp						
Synchronization Source (SSRC) Identifier						
Zeros						length of extension (words)
ADU sequence number						
ADU length (bytes)						
ADU offset (bytes)						
Priority						Layer Number

Figure 9: SR-RTP header for selective reliability.

Networks platform [17]. We have recently integrated our library to enable video streaming using SR-RTP in `mplayer` [38] with minimal changes to the distribution.

3.1 Overview

Figure 8 shows the components of our system. The server listens for requests on an RTSP [49] port, establishes session parameters via SDP [24], and streams requested data to the client via RTP (over UDP) [48] that has been extended to support application-level framing (ALF) and selective reliability. Feedback is provided to the server at the RTP layer via RTCP receiver reports, and the server adjusts the congestion window size using the Congestion Manager (CM) [4]. The CM implements a TCP-friendly congestion control algorithm for the MPEG-4 streams and provides an API by which the server adapts to prevailing network conditions.

Our system supports backwards-compatible extensions to RTP/RTCP that allow for the application-level framing of the data with Application Data Units (ADUs) [16]. ADUs enable fragmentation and reassembly of independently processible units of data and also make selective recovery of application-specific data units possible at the receiver. For MPEG-4, one frame of the compressed video bitstream corresponds to one ADU. The SR-RTP layer divides the data to be sent into ADUs according to frame boundaries. The sender packetizes these ADUs, which are reassembled by the receiver and passed to the application layer for decoding once the complete frame has been received. An ADU may involve multiple packets (i.e., ADU fragments). Each is uniquely named by its ADU sequence number and byte offset within that ADU in order to efficiently request and perform selective retransmissions.

3.2 Loss-resilience

We have extended RTP to provide selective reliability. Each video frame is an ADU; we specify information such as the sequence number of that ADU in the header. Additionally, should an ADU not fit within one packet (e.g., for I-frames), we provide a mechanism for specifying the byte offset within an ADU. The length of the ADU is also contained within the extension header.

The server packetizes the video (in the case of MPEG-4, the bitstream is packetized on resynchronization marker boundaries), labels the packets with ADU sequence numbers and offsets, and sends the RTP packets over UDP. These semantics allow the client to reassemble the packet and to determine if any data is missing. On receiving a packet, the client sends back an ACK to the receiver, acknowledging the successful receipt of an ADU (or portion thereof). Alternatively, the client can send a retransmission request requesting retransmission of a specific portion of the bitstream.

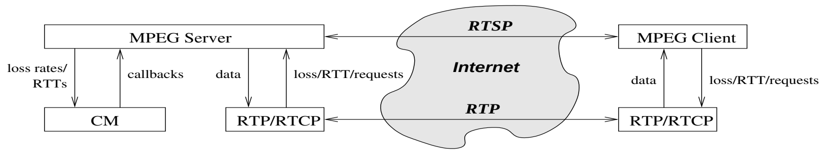

Figure 8: System architecture. Feedback is sent to the streaming application via RTCP, which is used to appropriately adjust the transmission rate.

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 (bits 0–31)

V	P	RC	Payload Type=RR	Length
SSRC of packet sender (Receiver ID)				
SSRC_1 (SSRC of first source)				
fraction lost			cumulative number of lost packets	
extended highest sequence number received				
Interarrival Jitter				
Timestamp Echo (LSR)				
Processing Time (DLSR)				
Window Size (kB)				Padding
ADU Sequence Number				
ADU Fragment Length (bytes)				
ADU Offet (bytes)				

Figure 10: SR-RTCP Receiver Report for selective reliability.

3.2.1 Header Formats

Figure 9 shows the extended RTP header with extension to enable selective retransmission of lost packets. The first 12 bytes of the packet are identical to the RTP header format specification [48]. Additionally, we have provided a generic selectively reliable RTP (SR-RTP) extension that provides for application-level framing (ALF) [16] as well as selective reliability. The *Zeros* field is a field that is required by the standard to allow multiple interoperating implementations to operate independently with different header extensions; we set this to all zeros.

The *ADU sequence number* field uniquely identifies the ADU; in the case of MPEG-4 video, one frame corresponds to one ADU, so this is equivalent to a frame number. The *ADU length* field indicates the number of bytes contained within that particular ADU; this allows the transport layer to detect missing packets at the end of an ADU. The *ADU offset* uniquely identifies one packet within an ADU and allows for reassembly of a packet when reordering occurs. The header provides a *Priority* field that allows the transport layer to specify the relative importance of packets. In particular, for the purposes of our experiments, we mark MPEG-4 I-frames with a high priority so that a retransmission request is sent on an I-frame loss but not a P-frame or B-frame loss. The *Layer* field is used when transmitting layered video to specify the layer of video to which the packet corresponds; this feature can be used to calculate playout times, in decoding, or for caching purposes.

Figure 10 shows an SR-RTCP receiver report, largely the same as an RTCP receiver report, but with profile-specific extensions added to the end of the header. The *Length* field indicates how many requests to expect at the end of the header. The first 16 bytes of the extension serve as an ACK to the sender acknowledging the receipt of a particular ADU fragment and report the current window size of the receiver for flow control purposes. Optionally, the report can include one or more *ADU request*s, of 12 bytes each; these requests uniquely identify the ADU fragment that is to be retransmitted.

3.2.2 Loss Detection and Recovery Decisions

Using SR-RTP, the receiver detects packet loss by finding gaps in packet arrivals, which can be determined given information about the length of each ADU and the offset of each packet within an ADU. We assume that I-frames consist solely of intra-coded macroblocks, and predicted frames consist primarily of predicted macroblocks. In such a situation, the priority for retransmission of missing blocks is generally determined from surrounding blocks. Specifically, there are four cases of packet loss that must be detected:

- *Mid-frame loss.* A mid-frame loss is detected by detecting a gap in the reconstructed ADU. The priorities of the missing packets are equal to the priority of the surrounding packets. In the event that surrounding packets in the same ADU have differing priorities, the highest priority is assumed for the missing portion.

- *Start-of-frame loss.* A start of frame loss is detected in a similar fashion to a mid-frame loss. If the first packet received for a particular ADU has a nonzero offset, a loss will be detected at the start of the frame with priority for those packets equal to those that *follow* the gap.

- *End-of-frame loss.* If the number of bytes in a particular ADU is less than the reported length for that ADU, and no gaps exist in the received data, a loss will be detected at the end of the frame with priority for those packets equal to those that *precede* the gap.

- *Complete frame loss.* A complete frame loss can be detected by a gap in the ADU sequence number space. In this case, our system foregoes any retransmissions of that frame's data. The likelihood that a lost frame is an I-frame is very low since I-frames are large; because of the size of an I-frame, complete retransmission is also expensive and should be avoided in general.

Using this logic, SR-RTP detects packet loss and optionally requests retransmission of ADU gaps based on the determined priority of the lost region. As a simple scenario, priorities could be assigned such that missing I-frame packets are retransmitted, while other gaps in data are ignored or corrected by postprocessing techniques. A more complex retransmission policy could assign different priorities to different frames. For example, missing packets from P-frames might be retransmitted with varying priorities, since P-frames that are closer to the preceding I-frame are more valuable for preserving picture quality than later P-frames in the GOV.

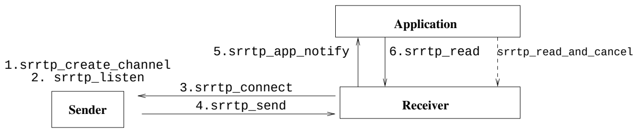

Figure 11: Summary of SR-RTP API. The receiver's API is callback-based.

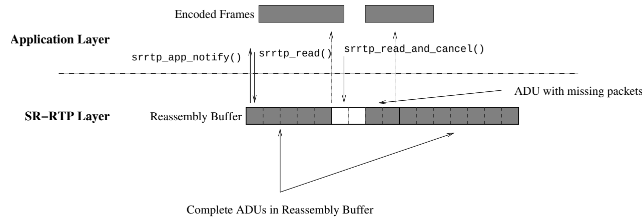

Figure 12: Reassembly of ADUs at the SR-RTP enabled receiver. When a complete ADU arrives, SR-RTP makes an `srrtp_app_notify()` callback to the application, which subsequently calls `srrtp_read()` to read the complete ADU into the application's memory. If playout time for an ADU occurs before a complete ADU arrives, the application makes an `srrtp_read_and_cancel()` call to SR-RTP, which cancels further retransmit requests for that ADU.

3.2.3 Implementation

We have implemented the SR-RTP library, which enables selective reliability and bandwidth adaptation using CM, as well as an accompanying RTSP [49] client/server library which allows an application developer to easily layer RTSP on top of SR-RTP. Software is available from the OpenDivX project Web site [42], as well as from our project Web site [53]. In this section, we describe how an application developer can extend an existing application to support SR-RTP functionality, as we have done with `mplayer` [38].

While the SR-RTP library can be used independently of the RTSP library, streaming applications commonly use RTSP at the application layer. Our auxiliary library thus makes it easier to incorporate RTSP/SR-RTP functionality into a video playback application that does not support streaming.

The SR-RTP library is designed as a backwards-compatible extension to RTP [48], but has included additional functionality for permitting receiver-driven retransmission requests from the server, appropriately packetizing data by frames, and setting the appropriate priorities. The library is configurable to work with or without the Congestion Manager (CM) extensions to the Linux kernel.

If the library is configured to interoperate with CM, the library will use feedback from SR-RTCP receiver reports to appropriately adjust the bandwidth and layer video quality appropriately. Otherwise, the library will support only the selective retransmission functionality, which does not require CM to operate. All functions discussed below have a corresponding function call which performs the equivalent functionality plus additional functions required to support bandwidth adaptation with CM. Figure 11 summarizes the calls that are described in further detail below.

The sender and receiver initialize the channel by invoking `srrtp_create_channel()`. An SR-RTP listener can bind to a listening socket by calling `srrtp_listen()`, and a client connects to this listening socket using `srrtp_connect()`. These functions establish connections on two ports, one for data, and one for the receiver feedback channel, over which SR-RTCP receiver reports are sent from the receiver back to the sender to provide loss and round-trip time information.

When the sender wishes to send data to the receiver, it calls the `srrtp_send()` function, which expects a buffer of data that is independently processible by the application (i.e., an ADU). In the case of MPEG video transmission, for example, this function is called once per MPEG frame. This function subsequently fragments the frame into packets and labels each packet in a fashion that the application can understand. That is, the transport layer attaches header information such as the ADU sequence number, as well as the offset of that particular packet within an ADU, thus enabling reassembly of fragmented ADUs at the receiver and detection of lost packets. The sender can also optionally give a particular packet priority value. Typically, all packets within one frame will receive the same priority so that the priority of a missing packet can be determined from surrounding packets.

After the sender has completely sent an ADU, it keeps recently sent packets in a *retransmission buffer*. In the event that one or more packets must be retransmitted, the sender need only find the packet in its retransmission buffer and send the packet again.

As packets arrive at the receiver, they are reassembled in a *reassembly buffer*. At this point, one of two things can happen:

- the entire ADU will arrive, or
- sections of the ADU will be missing.

Figure 12 shows how the receiver reassembles packets into ADUs for processing by the application layer. As soon as the entire ADU arrives at the receiver, the `srrtp_app_notify()` callback is made to the application layer. The application then handles each ADU in an independent fashion. In the case of MPEG video, an `srrtp_app_notify()` callback implies the arrival of a complete MPEG video frame. At this point the application will first call the `srrtp_read()` function to read the given ADU from the reassembly buffer into the application memory, and will subsequently decode this frame and place it into a playout buffer for future display. Generally, when any application receives the `srrtp_app_notify()` callback, it will call `srrtp_read()` and subsequently perform any application-specific processing.

If sections of the ADU are missing, the receiver will send a data-driven request to the sender asking for the retransmission of lost packets. This operation is *completely transparent to the application*. Alternatively, the application can ask SR-RTP to cancel any further retransmission requests by calling the `srrtp_read_and_cancel()` function. This function reads the specified ADU from the retransmission buffer and informs the SR-RTP layer at the receiver that it should no longer make any retransmission requests for that particular ADU. This may cause some late retransmissions to be sent because of round-trip time delay; thus, retransmissions should be canceled approximately 1 RTT before the frame is read to minimize futile retransmissions.

Bandwidth adaptation is performed using the Congestion Manager (CM) extensions to the Linux kernel [3, 5]. Bandwidth adaptation is performed by the sender if it supports CM functionality – note that the receiver does not need to support CM to enable bandwidth adaptation for streaming video.

3.3 Receiver Postprocessing

This section focuses on the benefits that SR-RTP provides for performing error concealment via decoder postprocessing and argues that

- SR-RTP is a complementary scheme that can be used in combination with other techniques, and

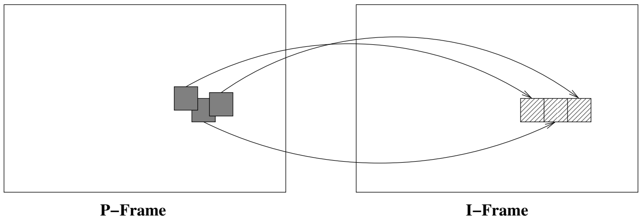

Figure 13: In scenes with high motion, replace missing texture with more appropriate pixels from the preceding P-frame.

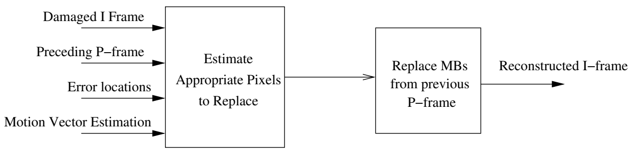

Figure 14: Block Diagram. Exploit temporal dependencies to achieve more accurate texture replacement in I-frames.

- SR-RTP helps provide information about packet loss to the application that can be used in performing other error concealment techniques.

We propose a postprocessing routine for performing temporal error concealment of I-frames at the receiver.

First, we describe a conventional macroblock replacement approach, where missing macroblocks are replaced. We find that this approach, while simple, does not work well in high-motion scenes.

Because motion in video sequences tends to be highly correlated, however, we can make use of motion information that may be present in previous portions of the bitstream to reconstruct the lost information in the current I-frame. For scenes with higher motion, we present an algorithm for recovering I-frames that exploits both temporal correlation and the motion vector information in the bitstream.

We assume that the location of a packet loss within a given picture can be detected since this information is provided by SR-RTP. Mechanisms of this flavor allow the transport layer to detect when a certain segment of data has been lost and inform the application of these losses.

A simple approach replaces the missing macroblocks with the same blocks from a previous frame. This works well in low-motion sequences, or when the loss occurs in a uniform background region. In the event of high motion, however, simple macroblock replacement is not acceptable, because the missing macroblock data will not correspond well with the same blocks in a previous frame. In this case, we must search for appropriate corresponding pixel values for the missing macroblocks. Fortunately, with high probability the bitstream will contain motion vectors for surrounding pictures, from which we can estimate the motion that has occurred in the region where packet loss has occurred.

MPEG-4 uses motion vectors to estimate the B-frame(s) immediately prior to the given I-frame; motion vectors also exist from the preceding P-frame for these B-frames. Using the available motion vector information, we can estimate the motion that has occurred between this P-frame (which we will use to obtain the texture data) and the given I-frame. For our experiments, we used the motion vectors from the preceding P-frame to the preceding B-frame for the corresponding macroblock and adjust its value accordingly to reconstruct data in the I-frame. We can think of this as essentially *performing motion compensation on the I-frame*. The conceptual illustration of this method is shown in Figure 13, and the corresponding block diagram is shown in Figure 14 (a similar idea to recover errors in dependent frames was proposed in [22]).

In addition to knowledge about the location of the error and texture data from the preceding P-frame, the decoder must also have access to relevant motion information. If this data is somehow lost, it can be estimated using spatial techniques, such as an averaging of motion vectors for surrounding macroblocks. Figure 14 assumes that motion information is not lost and we can perform temporal error concealment, using the motion vectors from P_{n-2} to B_{n-1} to estimate the motion vectors MV' for P_{n-2} to I_n. The motion vectors MV' are then used to locate the relevant texture data in P_{n-2} to use for replacement in I_n. This algorithm can be generalized depending on the number of B-frames that exist between I and P-frames.

If too much information is lost and latency permits, it may be better to recover from errors via retransmission of the missing data using the selective retransmission features of SR-RTP.

4. PERFORMANCE EVALUATION

We conducted experiments to show that selective reliability is both a feasible and beneficial means of improving the quality of received video. The video server (running on a Pentium 4 1.5 GHz Linux 2.2.18 box) streamed data to the receiver (a Pentium II 233 MHz Linux 2.2.9 box) across a 1.5 Mbps link, configured using Dummynet [18].

We performed two loss recovery experiments using 300 frames of a 20 Kbps 30 fps sequence. To examine the gains of selective reliability for various packet loss rates, we streamed 300 frames of a 20 Kbps 30 fps sequence across a 1.5 Mbps 50 ms link with varying packet loss rates. We studied the performance of SR-RTP for various bandwidths by transmitting this bitstream across a 50 ms link with a 2^{-5} packet loss rate. To emulate actual Internet conditions, we used a 200 ms round-trip time (RTT) link and introduced background Web cross-traffic using the SURGE toolkit [7] to emulate the varying network conditions that an actual streaming server might see in the face of competing Web traffic on the Internet.

4.1 Selective Reliability

Our experiments show that selective retransmission of I-frame data can result in significant performance gains. We present our findings from experiments on an emulated network that show considerable performance improvement for only a small amount of buffering, and discuss the tradeoff between reliability, interactivity, and general buffering requirements.

4.1.1 Benefits of Selective Reliability

Using the 200 ms of initial buffering and the buffering required to combat round-trip time jitter (actually, in the case of these experiments, buffering for retransmission was dwarfed by the amount of buffering required to combat round-trip time jitter), we were able to achieve significant gains in resulting video quality by performing selective retransmissions of I-frame data.

Performing selective recovery on important data within an MPEG-4 bitstream results in significant improvements in perceptual quality. As mentioned in the previous section, different amounts of buffering will allow for a variable amount of selective retransmission. Figure 15 shows two curves: the bottom curve is the resulting picture quality for various packet loss rates without performing selective retransmission, and the upper curve shows the corresponding picture quality that can be achieved by using selective retransmission. This graph shows the potential for quality gain that exists under certain conditions. For other quality thresholds,

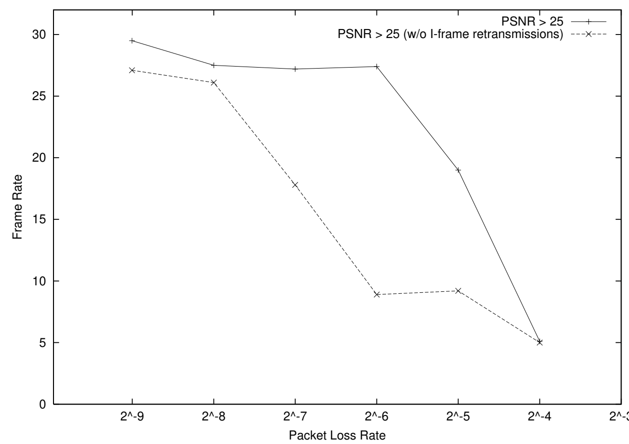

Figure 15: The benefits of selective reliability. Substantial quality improvement can be achieved by performing selective retransmission of I-frames.

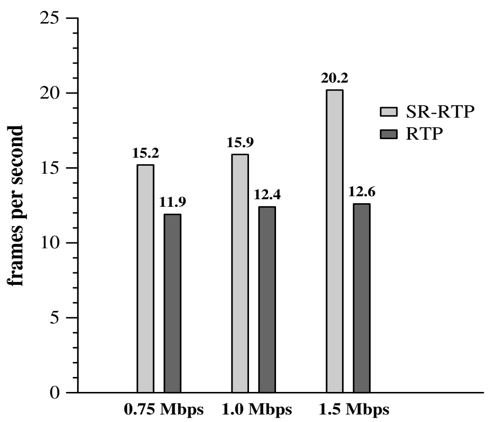

Figure 16: SR-RTP can provide benefits for channels of various bandwidths. This graph shows the effective frame rate for various bandwidths using a fixed PSNR threshold of 25 dB and packet loss rate of 2^{-5}.

bitrates, etc., the benefits will vary; nevertheless, it is clear that selective reliability can be a boon in certain circumstances. These results generally correspond to our expected results in Figure 7.

In a second experiment, we fixed the acceptable picture quality at 25 dB and the packet loss rate at 2^{-5} and examined the benefits of selective reliability for various bandwidths. The results in Figure 16 show that selective reliability can provide significant frame rate improvements at different bandwidths.

4.1.2 Buffer Requirements

There is a fundamental tradeoff between the amount of reliability obtained via retransmission and the degree of interactivity possible. For instance, one extreme is simply to transmit the bitstream over TCP; while this provides complete reliability, the degree of interactivity is small because of the delays incurred while achieving complete reliability [34]. Smooth quality of a received video signal depends on appropriate buffering. In particular, receiver buffering must be large enough to (1) account for network jitter, (2) allow time for retransmission of lost packets, and (3) enable quality adaptation [45]. The buffering required to counteract network jitter is a function of the variance in network delay, where the instantaneous

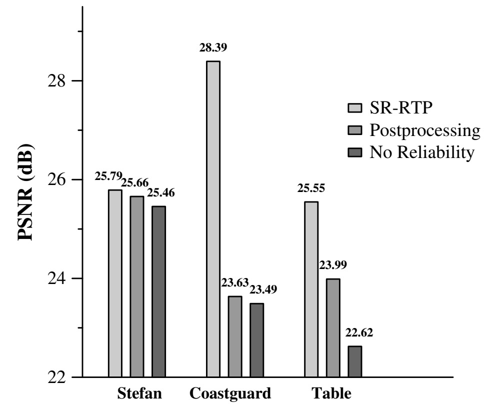

Figure 17: The benefits of receiver postprocessing. Given knowledge about the location of packet losses in an I-frame, which SR-RTP can provide, the receiver can perform temporal concealment on I-frames to recover from packet loss and thus alleviate propagation of errors, even if selective retransmission is not possible. This graph shows the gains that receiver postprocessing can provide for I-frames from 3 different sequences: *coastguard*, *stefan*, and *table*.

jitter j_i can be expressed as $|(A_i - A_{i-1}) - (S_i - S_{i-1})|$ [34, 48]. Using this, the required buffering to counteract network jitter is $\beta\delta_i$, where δ_i is smoothed jitter; smaller values of β reduce overall delay, and larger values decrease the likelihood of late (hence, effectively lost) packets. Buffering for retransmission of lost packets also depends on the absolute network round-trip time. Buffering for quality adaptation depends on the absolute transmission rate. A larger rate results in a larger backoff in the event of a packet loss, and thus requires more buffering to sustain playout at the current layer. We have shown that required QA buffering is $O(R)$ for SQRT congestion control and $O(R^2)$ for AIMD [19].

The dominant factor depends on the relation of the absolute round-trip time to the RTT variance, as well as the absolute transmission rate. As the absolute RTT becomes large with respect to RTT variance, buffering due to retransmission requests will dominate buffering required to counteract jitter, and vice versa. As the absolute bitrate grows large, the amount of buffering required for QA will increase; using more aggressive congestion control algorithms such as AIMD also result in more buffering required for QA.

4.2 Receiver Postprocessing

Figure 17 summarizes how using temporal postprocessing at the receiver to recover I-frames can result in improved image quality. We examine three distinct instances of packet loss in I-frames in three independent video sequences: *coastguard*, *stefan*, and *table* (a table tennis scene). In certain cases recovery of I-frame data can improve the PSNR of that reference frame by more than 2 dB. While perfect recovery via a scheme like selective retransmission via SR-RTP allows for the highest possible quality at the decoder, in cases of high end-to-end latency, a scheme such as receiver postprocessing allows for reasonable I-frame recovery to take place. Even if selective retransmission is not possible, SR-RTP can provide information regarding losses to the decoder and thus aid in receiver postprocessing. Thus, receiver postprocessing can be used in combination with selective retransmission to improve the quality of important data in compressed video, thereby limiting the effects of error propagation.

5. RELATED WORK

MPEG-4 is the latest standard for inter-frame compression and storage of digital video from the Moving Picture Experts Group [13, 20, 28, 31, 37]. Much prior work has focused on the transmission of MPEG video over networks. In the following section, we outline how other contributions relate to our work.

5.1 Media Transport

The Real-time Transport Protocol (RTP) [48] is an end-to-end protocol that enables the transport of real-time streaming data. The Real Time Streaming Protocol (RTSP) [49] is an application-level protocol that controls the delivery of data with real-time properties. RFC 3016 defines an RTP payload format for MPEG-4 for the purpose of directly mapping MPEG-4 Audio and Visual bitstreams onto packets without using the MPEG-4 Systems standard [30]. This specification works in concert with SR-RTP, as it simply defines the manner in which an MPEG-4 bitstream can be mapped into packets. Previous RFCs define payload formats for MPEG-1/2 video and bundled MPEG-2 video and audio layers [15, 27]. Early work recognized some of the challenges of streaming compressed video on the Internet and explored the use of RTP as a transport protocol for video [8]. We extend this work by defining a framework for ALF within RTP and implementing a system that supports this principle.

Concurrent work proposes one mechanism for performing multiple selective retransmissions of generic media data [12, 36]. This work has a different goal from our work as it is designed primarily for packet loss resulting from *bit errors* and does not address recovery from congestion-related packet loss. Other work describes an extended RTP profile that allows for providing immediate feedback to the sender using RTCP receiver reports [40]. This work complements SR-RTP, which allows the receiver to provide RTT and loss information to the sender via frequent RTCP receiver reports.

Prior work uses filtering techniques within the network to adapt with congestion-related packet loss on a best effort network [26]. Others propose using a rate-based approach, similar to TFRC, to control the rate of video transmission [52]. Our system uses the Congestion Manager to adapt video transmission at the end host in accordance with changing network conditions, and uses a combination of retransmission requests and postprocessing at the receiver to recover from congestion-related packet loss.

Rejaie et al. propose a quality adaptation scheme using receiver buffering for AIMD-controlled transmission and playback for hierarchically-encoded video [44, 46]. We extend this work and have incorporated it into our system as described in [19].

5.2 Error and Loss Recovery

Previous work performed error concealment in dependent frames by replacing missing macroblocks with a motion-compensated block from a different frame [22]. We use a similar idea to recover I-frame data. Wah et al. have recently surveyed various error concealment schemes for real-time audio and video transmission and argue that error recovery via retransmission is not feasible due to the imposed delay [54]. However, other researchers have shown that retransmission can be a feasible option for error recovery. Rhee proposes a retransmission-based error control technique without incurring additional latency by rearranging the temporal dependency of frames so that a reference frame is referenced by its dependent frames much later than its display time, thereby masking the delay in recovering lost packets [47]. Another scheme uses playout buffering, conditional retransmission requests, and various other techniques to alleviate the effects of packet loss [41]. Our work shows how MPEG-4 delivery can be improved using selective retransmissions and receiver postprocessing.

Prior work has analyzed MPEG-4's built-in error resilience capabilities and examined propagation of errors on an inter-frame video bitstream when bit errors occur [23]. In contrast, we examine propagation of errors due to *packet loss* and develop a model to describe the effects of these errors.

Forward error correction (FEC) been proposed in several projects as a means for providing error recovery and packet reconstruction [9, 10]. An RTP payload format for packet-level FEC has been defined to achieve uneven error protection of RTP-encapsulated data [33]. However, these schemes rely on the fact that the FEC information contained in one packet itself is not lost. FEC-based schemes add redundant information, which can potentially worsen network traffic and aggravate existing packet loss.

Several schemes use prioritization ideas to protect data of high importance on lossy channels. Quality assurance layering (QAL) protects high priority data with FEC [39, 50]. One approach is to use the priorities associated with a bitstream to provide error protection at the time of encoding [25]. This approach allocates more bits to more important information, such as header and motion information, while allocating fewer bits to texture information, within a particular video packet. Priority encoding transmission (PET) [1] is an approach for sending messages over a lossy network based on a specified prioritization scheme. It has been used to provide a mechanism for using PET to create a hierarchical encoding of MPEG-1 video bitstreams [32].

Another approach to error concealment is multiple description coding (MDC) [14, 51], a joint sender-receiver approach for designing transforms. This scheme divides the bitstream into equally important "descriptions", so that each additional description is useful in enhancing the quality of the received video.

6. CONCLUSION

In order for video streaming to succeed on the Internet, systems must account for the anomalies of packet loss and changes in bandwidth and delay that make the delivery of real-time video on the Internet challenging. We have analyzed the effects of packet loss on the quality of MPEG-4 video and proposed a model to explain these effects. We have shown that, by recovery of only the most important data in the bitstream, significant performance gains can be achieved *without* much additional penalty in terms of latency. Finally, we have designed a system that employs backwards compatible extensions to RTP to enable selective retransmission of important data in conjunction with receiver postprocessing and uses the Congestion Manager to perform TCP-friendly congestion control that is more amenable to the transmission of video. Through the combination of these techniques, we have enabled a streaming system that is adaptive to changing conditions and delivers high-quality, highly-interactive video.

Acknowledgments

We thank David Andersen, John Apostolopoulos, Reza Rejaie, and Jennifer Rexford for many helpful comments and discussions.

7. REFERENCES

[1] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan. Priority encoding transmission. In *Proc. 35th Ann. IEEE Symp. on Foundations of Computer Science*, pages 604–612, November 1994.

[2] M. Allman and V. Paxson. *TCP Congestion Control*. Internet Engineering Task Force, April 1999. RFC 2581.

[3] D. Andersen, D. Bansal, D. Curtis, S. Seshan, and H. Balakrishnan. System support for bandwidth management and content adaptation in Internet applications. In *Proc. Symposium on Operating Systems Design and Implementation*, October 2000.

[4] H. Balakrishnan, H. S. Rahul, and S. Seshan. An Integrated Congestion Management Architecture for Internet Hosts. In *Proc. ACM SIGCOMM*, pages 175–187, Cambridge, MA, September 1999.

[5] H. Balakrishnan and S. Seshan. *The Congestion Manager*. Internet Engineering Task Force, June 2001. RFC 3124.

[6] D. Bansal and H. Balakrishnan. Binomial Congestion Control Algorithms. In *Proceedings INFOCOM 2001*, Anchorage, AK, April 2001. Also available as MIT-LCS-TR-806 from `http://nms.lcs.mit.edu/papers/cm-binomial.html`.

[7] P. Barford and M. Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. In *Proc. ACM SIGMETRICS '98*, Madison, WI, 1998.

[8] A. Basso, G.L. Cash, and M.R. Civanlar. Transmission of MPEG-2 streams over non-guaranteed quality of service networks. In *Picture Coding Symposium*, September 1997.

[9] J.C Bolot and T. Turletti. Adaptive error control for packet video in the Internet. In *Proceedings of International Conference on Internet Protocols*, September 1996.

[10] J.C Bolot and T. Turletti. Experience with control mechanisms for packet video in the internet. *SIGCOMM Computer Communication Review*, 28(1), January 1998.

[11] J. Boyce and R. Gaglianello. Packet loss effects on MPEG video sent over the public internet. In *ACM Multimedia*, 1998.

[12] C. Burmeister, R. Hakenberg, J. Rey, A. Miyazaki, and K. Hata. Multiple selective retransmissions in RTP. In *11th International Packet Video Workshop*, Kyongju, Korea, May 2001.

[13] L. Chiariglione. MPEG-4 FAQs. Technical report, ISO/IEQ JTC1/SC29/WG11, July 1997.

[14] D. Chung and Y. Wang. Multiple description iamge coding using signal decomposition and reconstruction based on lapped orthogonal transforms. *IEEE Trans. Circuits and Systems for Video Technology*, 9(6):895–908, Sept 1999.

[15] M. Civanlar, G. Cash, and B. Haskell. *RTP Payload Format for Bundled MPEG*. Internet Engineering Task Force, May 1998. RFC 2343.

[16] D. Clark and D. Tennenhouse. Architectural Consideration for a New Generation of Protocols. In *Proc. ACM SIGCOMM*, pages 200–208, Philadelphia, PA, September 1990.

[17] Divx networks. `http://www.divxnetworks.com/`, 2001.

[18] Dummynet. `http://www.iet.unipi.it/~luigi/ip_dummynet`, September 1998.

[19] N. Feamster, D. Bansal, and H. Balakrishnan. On the interactions between congestion control and layered quality adaptation for streaming video. In *11th International Packet Video Workshop*, Kyongju, Korea, May 2001.

[20] International Organization for Standardization. *Overview of the MPEG-4 Standard*, December 1999.

[21] G. Wallace. The JPEG Still Picture Compression Standard. *Communications of the ACM*, April 1991.

[22] M. Ghanbari. Cell-loss concealment in ATM video codecs. *IEEE Trans. CAS for Video Tech.*, 3(3):238–247, June 1993.

[23] S. Gringeri, R. Egorov, K. Shuaib, A. Lewis, and B. Basch. Robust compression and transmission of MPEG-4 video. In *Proc. ACM Multimedia*, 1999.

[24] M. Handley and V. Jacobson. *Session Description Protocol*. Internet Engineering Task Force, April 1998. RFC 2327.

[25] W. Heinzelman. *Application-Specific Protocol Architectures for Wireless Networks*. PhD thesis, Massachusetts Institute of Technology, June 2000.

[26] M. Hemy, U. Hengartner, P. Steenkiste, and T. Gross. MPEG system streams in best-effort networks. In *Proc. Packet Video Workshop*, May 1999.

[27] D. Hoffman, G. Fernando, and V. Goyal. *RTP Payload Format for MPEG1/MPEG2 Video*. Internet Engineering Task Force, January 1998. RFC 2250.

[28] International Organization for Standardisation, Atlantic City. *Information Technology - Generic Coding of Audio-Visual Objects, Part 2: Visual*, October 1998. ISO/IEC JTC 1/SC 29/WG 11 N 2502 FDIS 14496-2.

[29] V. Jacobson. Congestion Avoidance and Control. In *Proc. ACM SIGCOMM*, pages 314–329, August 1988.

[30] Y. Kikuchi, T. Nomura, S. Fukunaga, Y. Matsui, and H. Kimata. *RTP Payload Format for MPEG-4 Audio/Visual Streams*. Internet Engineering Task Force, November 2000. RFC 3016.

[31] Rob Koenen. Overview of the MPEG-4 standard. Technical report, ISO/IEC JTC1/SC29/WG11, December 1999. `http://www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm`.

[32] C. Leicher. Hierarchical encoding of MPEG sequences using priority encoding transmission (PET). Technical Report TR-94-058, ICSI, Berkeley, CA, November 1994.

[33] A. Li, F. Liu, J. Villasenor, J. Park, D. Park, and Y. Lee. *An RTP Payload Format for Generic FEC with Uneven Level Protection*. Internet Engineering Task Force, October 2001. `http://www.ietf.org/internet-drafts/draft-ietf-avt-ulp-02.txt` Work in progress, expires April 2002.

[34] S. McCanne. Scalable multimedia communication with Internet multicast, light-weight sessions, and the MBone. Technical Report CSD-98-1002, August 1998.

[35] Microsoft Windows Media Player. `http://www.microsoft.com/windows/mediaplayer/`.

[36] A. Miyazaki et al. *RTP Payload Format to Enable Multiple Selective Retransmissions*. Internet Engineering Task Force, November 2001. `http://www.ietf.org/internet-drafts/draft-ietf-avt-rtp-selret-03.txt` Work in Progress, expires May 2002.

[37] MPEG home page. `http://www.cselt.it/mpeg/`.

[38] mplayer. `http://mplayer.sourceforge.net`, 2001.

[39] M. Normura, T. Fujii, and N. Ohta. Layered packet loss protection for variable rate coding using DCT. In *Proceedings of International Workshop on Packet Video*, September 1988.

[40] J. Ott et al. *Extended RTP Profile for RTCP-based Feedback (RTP/AVPF)*. Internet Engineering Task Force, November 2001. `http://www.ietf.org/internet-drafts/draft-ietf-avt-rtcp-feedback-01.txt` Work in Progress, expires May 2002.

[41] C. Papadopoulos and G. Parulkar. Retransmission-based error control for continuous media applications. In *Proceedings of the Sixth International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 5–12, 1996.

[42] Project mayo. `http://www.projectmayo.com/`, 2001.

[43] Real Networks. `http://www.real.com/`.

[44] R. Rejaie, M. Handley, and D. Estrin. Quality Adaptation for Unicast Audio and Video. In *Proc. ACM SIGCOMM*, Cambridge, MA, September 1999.

[45] R. Rejaie, M. Handley, and D. Estrin. RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. In *Proc. IEEE INFOCOM*, volume 3, pages 1337–1345, New York, NY, March 1999.

[46] R. Rejaie, M. Handley, and D. Estrin. Layered Quality Adaptation for Internet Video Streaming. *IEEE Journal on Selected Areas of Communications (JSAC)*, Winter 2000. Special issue on Internet QOS. Available from `http://www.research.att.com/~reza/Papers/jsac00.ps`.

[47] I. Rhee. Error control techniques for interactive low-bit rate video transmission over the Internet. In *Proc. ACM SIGCOMM*, September 1998.

[48] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. IETF, January 1996. RFC 1889.

[49] H. Schulzrinne, A. Rao, and R. Lanphier. *Real Time Streaming Protocol (RTSP)*. IETF, April 1998. RFC 2326.

[50] K. Shimamura, Y. Hayashi, and F. Kishino. Variable bitrate coding capable of compensating for packet loss. In *Proceedings of Visiual Communications and Image Processing*, pages 991–998, November 1988.

[51] X. Su and B. W. Wah. Multi-description video streaming with optimized reconstruction-based DCT and neural-network compensations. *IEEE Transactions on Multimedia*, 3(3), May 2001.

[52] W. Tan and A. Zakhor. Real-time Internet Video Using Error Resilient Scalable Compression and TCP-friendly Transport Protocol. *IEEE Trans. on Multimedia*, 1(2):172–186, May 1999.

[53] Adaptive Video Streaming Home Page. `http://nms.lcs.mit.edu/projects/videocm/`, 2001.

[54] B. W. Wah, X. Su, and D. Lin. A survey of error-concealment schemes for real-time audio and video transmissions over the Internet. In *Proc. Int'l Symposium on Multimedia Software Engineering*, pages 17–24, Taipei, Taiwan, December 2000. IEEE.