Error Recovery using FEC and Retransmission for Interactive Video Transmission*

Injong Rhee † *Srinath R. Joshi*
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7534

July 1998

Abstract

This paper proposes a new error recovery technique for interactive video transmission over error-prone, slow, and bandwidth constrained networks. The technique effectively combines forward error correction (FEC) and retransmission to solve the error propagation problem which is inherent in any motion-based video codecs. Prior work has mainly taken an approach that prevents display errors from occurring from the first place. This approach is not effective for interactive video transmission over the target network environments because data losses inevitably occur in such networks and repairing them tends to introduce the delays in video playout. Our approach is to focus on eliminating error propagation and isolating errors when they occur. This approach allows more time for packet repairs and can effectively mask out delays in repairing lost packets. A simulation study shows that the proposed techniques can be effectively used to transmit high-frame-rate interactive video over slow, error-prone, and bandwidth constrained networks. We name this approach *Recovery from Error Spread using Continuous Updates* (RESCU). A simulation study shows that under 10% data loss, RESCU-FEC can give a PSNR 1 dB higher than H.261 with 10% less bandwidth. For H.261 to achieve a similar video quality, an experiment shows that it needs about 40% more bandwidth than RESCU-FEC. Further improvement can be achieved when retransmission is combined with RESCU-FEC. It is shown that this hybrid scheme can be used for interactive video transmission over error-prone, high latency networks with higher bit efficiency than RESCU with either FEC or retransmission alone.

*This work is supported by in part by Faculty Research and Professional Development Fund of North Carolina State University.
†contact author: email: rhee@csc.ncsu.edu, Phone: 919-515-3305, Fax: 919-515-7925

1 Introduction

Transmitting high-quality, real-time video over error prone networks such as the Internet and wireless networks is very challenging. Because of limited bandwidth on networks and the bandwidth-hungry nature of video, video transmission requires extremely high compression efficiency. However, state-of-the-art video compression standards (MPEG, H.261) are not designed for transmission over a lossy channel. Although they can achieve very impressive compression efficiency, even small data losses can severely degrade video quality. A few bit errors in encoded data can cause the decoder to lose synchronization in the encoded stream, and can render useless all the data received until the next synchronization point. Furthermore, motion estimation and compensation in these codecs pose an even more severe problem, namely *error propagation* (or *error spread*). Motion estimation removes temporal redundancy in successive video frames (inter frame) by encoding only pixel value differences (prediction error) between a currently encoded image and a motion-predicted image created from a previously encoded image (reference frame). Image distortion in a reference frame can propagate to its succeeding frames and becomes amplified as more bits are lost.

Most of earlier work on error recovery focuses on repairing packet losses before the scheduled display times of those video frames contained in lost packets (e.g., [31, 28, 20, 39, 1, 36, 19, 22]). However, this approach is ineffective for interactive video because data losses inevitably occur in packet-switched communication, and detecting and repairing losses incur latency. To handle this latency, existing techniques introduce additional delays in frame display times. However, delaying frame playout times greatly impairs the interactiveness of video communication.

Many researchers [31, 28, 20, 39] have proposed using retransmission of lost packets by delaying frame playout times to allow arrival of retransmitted packets *before* their display times. Any packets received after their display times will be discarded. In these schemes, the display time of a frame is delayed by at least three one-way trip times after its initial transmission (two for frame transmission and one for a retransmission request). This latency can significantly impair the interactiveness of any video applications under the current Internet.

Forward error correction is also commonly proposed for error recovery of continuous media transmission [19, 1, 16, 5, 3]. However, conventional FEC schemes do not work well for interactive video. This is because unless the playout time of a frame is delayed, both the original packets and their parity packets must be transmitted within the same frame interval, rendering the schemes very susceptible to burst errors. Moreover, since FEC is applied to a block of packets, before FEC-encoded redundant packets is computed and transmitted, large delay must transpire.

In our earlier work [32, 33], we proposed an entirely complementary approach to these earlier ap-

proaches by focusing on eliminating error propagation when distortion on displayed images happens. Our point of departure from previously proposed approaches is that packets do not have to arrive in time for display to be "useful". Obviously, if packets can arrive before their display times, that is optimal. However, due to packet losses and high latency, repair packets inevitably arrive "late", causing distortion in displayed images which starts to propagate to following frames. We believe that these late repair packets can be used to stop error propagation. In motion-compensated codecs, the correct display of a frame depends on successful reception of all of its reference frames. If we buffer displayed frames and use late packets to restore errors in the buffered frames, error propagation can be stopped because these frames will be used as reference frames for later frames. We named this approach *Recovery from Error Spread using Continuous Updates* (RESCU).

RESCU has been shown effective for interactive video transmission when retransmission is used to recover lost packets [32, 33]. However, retransmission tends to prolong error propagation. because of the delay involved in detecting and repairing lost packets. Moreover, in some networks such as wireless cable modems and direct satellites, feedback channels are contentious, and bandwidth-limited. Thus, in these networks, frequent transmission of feedback to the sender to notifying packet losses is too expensive.

In this paper, we show that RESCU can perform effectively with little or no feedback to the sender by incorporating FEC. Our FEC scheme differs from the conventional ones in that FEC can be transmitted over a longer period than a single frame interval without introducing a delay in frame playout times. Since in RESCU, late packets are used to restore buffered reference frames, parity packets can be transmitted over a relatively longer period, interleaving with the packets of other frames. The interleaving helps reduce bursty losses of the periodic frame. Note that this interleaving is different from link-level symbol interleaving [41, 12] where symbols from multiple codewords are interleaved. The granularity of our proposed interleaving is much larger and thus, more effective. Unlike retransmission, FEC involves no feedback delay, incurring shorter recovery delays and accordingly shorter error propagation. Thus, this technique can be effective for high frame rate transmission over high latency networks.

One drawback of FEC is that during error-free transmission, it wastes bandwidth. To reduce wasted bandwidth during error-free transmission, we combine FEC with retransmission. By incorporating retransmission, we can safely reduce the amount of redundant bits. In this paper, we show that under short burst losses, FEC with a little redundancy can be effective in quickly recovering lost packets while under long burst losses, retransmission can recover those packets irrecoverable by FEC. Since retransmission occurs only when some indication of packet losses exists, this hybrid scheme contributes to reducing the overall bit overhead. In addition, since FEC recovers most of random and short burst losses which happens more frequently, it expedites packet recovery and effectively confines error propagation.

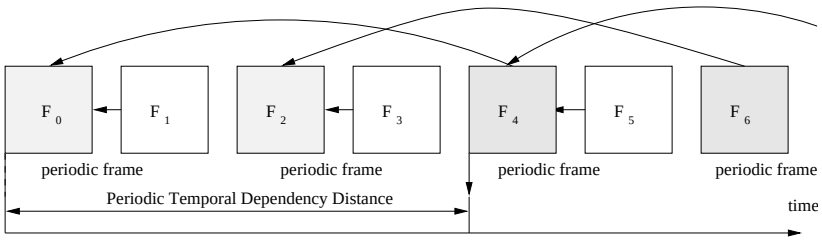

Figure 1: Periodic RESCU with PTDD 4 and NPF 2

A simulation study shows that under 10% data loss, RESCU-FEC can give a PSNR 1 dB higher than H.261 with 10% less bandwidth. H.261 can achieve high error resilience by more frequently sending intra-frames. For H.261 to achieve a similar video quality, an experiment shows that it needs about 40% more bandwidth than RESCU-FEC. Further improvement can be achieved when retransmission is combined with RESCU-FEC. It is shown that this hybrid scheme can be used for interactive video transmission over error-prone, high latency networks with much higher bit efficiency than RESCU with either FEC or retransmission alone.

Organization Section 2 describes the RESCU schemes, Section 3 presents the simulation results, and Section 4 discusses most related work.

2 Error Recovery Techniques

2.1 Recovery from Error Spread using Continuous Updates (RESCU)

We base our discussion on H.261, an International Telecommunication Union (ITU) video standard. In H.261, a video sequence consists of two types of video frames: *intra-frame* (I-frame) and *inter-frame* (P-frame). I-frame removes only spatial redundancy present in the frame. P-frame is encoded through motion estimation using another P-frame or I-frame as a reference frame (R-frame). For each image block in a P-frame, motion estimation finds a closely matching block within its R-frame, and generates the displacement between the two matching blocks as a motion vector. The pixel value differences between the original P-frame and a motion-predicted image of the P-frame obtained by simply cut-and-pasting the matching image blocks from its R-frame are encoded along with the motion vectors.

In RESCU, packets arriving after their display times are not discarded but instead used to reduce error propagation. In motion compensation-based codecs, the correct image reconstruction of a currently displayed image depends on a successful reconstruction of its R-frames. By using the late packets to restore R-frames, we can stop errors due to the lost packet to spread to following frames.

The *deadline of a packet* is defined to be the time by which the packet must arrive at the receiver to be useful. RESCU allows this deadline to be arbitrarily adjusted through the *temporal dependency distance* (TDD) of a frame that is the minimum number of frame intervals between that frame and its temporally dependent one. By extending TDD, we can arrange a frame to be referenced much later than its display time. This adjustment effectively masks out the delay in repairing lost packets. For instance, we can make every p-th frame (which we call *periodic frame*) reference the one in p frame intervals away as shown in Figure 1. This scheme is called *periodic-RESCU* (P-RESCU). The TDD of periodic frames are called *periodic TDD* (PTDD). Every non-periodic frame depends on its immediately preceding periodic frame. Although a periodic frame is displayed with errors, if the errors can be repaired within a PTDD period through transport-level recovery such as retransmission or FEC, the errors will stop propagating beyond the next periodic frame. Also, errors in non-periodic frames do not propagate because non-periodic frames temporally depend only on periodic frames. The picture pattern is very similar to that of MPEG, but while MPEG uses the pattern to increase compression efficiency, P-RESCU uses it to increase error resilience. Note that extending TDD does not affect frame playout times because frames are displayed at their scheduled display times.

More flexibility and error resilience can be obtained by increasing the number of periodic frames (NPF) within a PTDD period. This technique effectively reduces the duration of error propagation due to loss in a periodic frame without reducing PTDD. Figure 1 shows an example of P-RESCU with NPF 2. However, the increased error resilience comes at the expense of additional buffers because for each periodic frame within a PTDD period, it requires another frame buffer and the extension of the frame's TDD.

The encoder determines PTDD based on the current traffic conditions. However, if network conditions change (e.g., latency becomes longer) after a periodic frame is sent, that frame on the way to the destination might have too short PTDD for the changed environment. This causes the periodic frame to miss its deadline, resulting in error propagation. Since the frame is already encoded and transmitted, there is nothing that the encoder can do to save the frame. A scheme called *cascaded-RESCU* (C-RESCU) alleviates this problem without involving the encoder. In H.263, each reference frame temporally depends on the previous reference frames. Thus, by employing more reference frame buffers in the decoder, more late packets can be used to restore a sequence of erroneous (periodic) reference frames. This allows packet deadlines to be extended *at the receiving times*, but not at the encoding times.

When buffers are not available, PTDD is too short, or more data packets are lost than parity packets, periodic frames may not be recoverable. Error propagation starts. To prevent this type of error propagation, the receiver can detect losses in periodic frames not recovered even after a PTDD period, and can notify

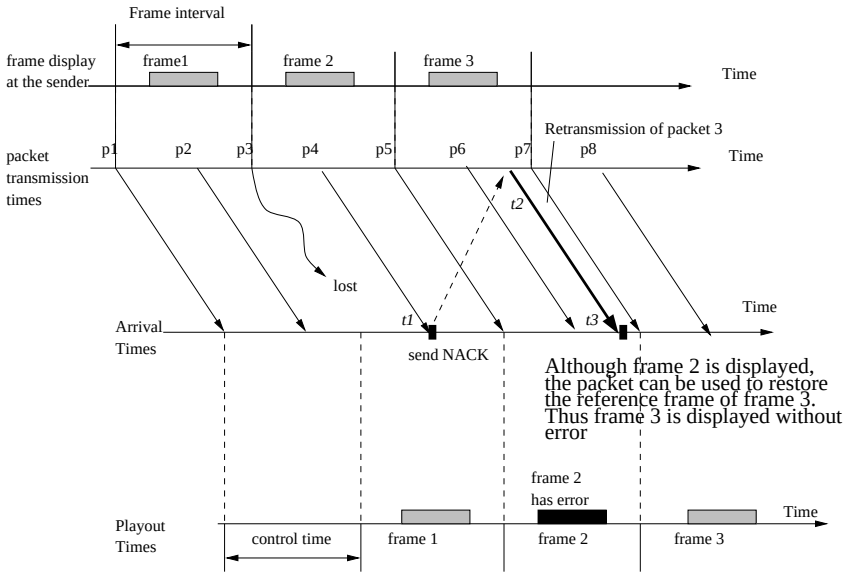

Figure 2: The recovery of frames using retransmission

the sender about this irrecoverable losses. This triggers the sender to code the next frame as an intra-frame which stops error propagation due to the earlier losses because the intra-frame does not have temporal signal dependency with any of frames transmitted earlier. This is called *replenishment*. This technique can be adopted in any scheme (including H.261). However, this may increase bandwidth consumption too much. The main motivation for RESCU is to reduce the number of replenishments.

2.2 Transport-level repair techniques

We show how two of the most common transport-level recovery techniques for packet losses (retransmission and FEC) can be integrated with RESCU.

Retransmission Retransmission is the most commonly used error recovery technique in reliable transport. The sender (or another receiver) simply retransmits the packets reported missing by a receiver. For interactive continuous media transmission, conventional schemes require retransmitted packets to arrive within a single frame interval after the time that they are first lost. However, its associated delays in detecting and retransmitting the lost packets are often larger than one frame interval. In contrast, RESCU allows these retransmission delays to be masked out since retransmitted packets need to be received only within a PTDD period.

Figure 2 illustrates the error recovery using retransmission in a video stream containing two packets per frame and PTDD=1. Packet 3 is lost, and the receiver receives packet 4 at time t_1 and recognizing that

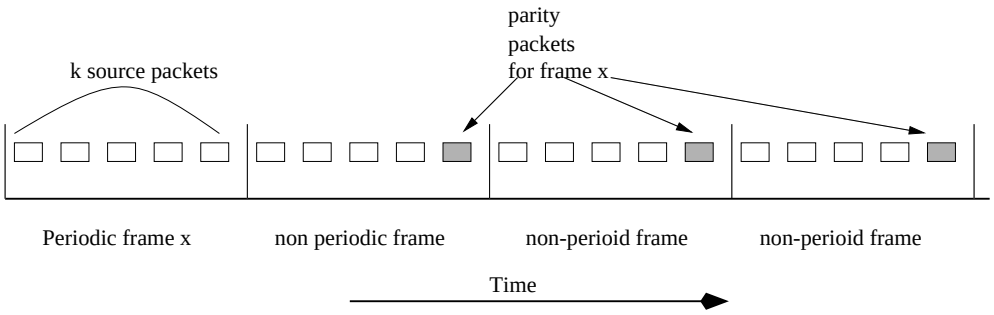

Figure 3: Interleaving of packets in P-RESCU of PTDD 4 and NPF 1 with (5,7) RSE encoding

packet 3 is not received, sends a retransmission request (NACK) to the sender. The sender gets the NACK at time t_2 and retransmits packet 3. The retransmitted packet arrives at time t_3 which is before frame 3 is displayed. Packet 3 is now used to restore the R-frame of frame 3, so frame 3 can be decoded and displayed without an error.

Our retransmission technique is fundamentally different from other retransmission schemes [31, 28, 20, 39] in that it does not introduce any delay in frame playout times. In interactive video conferencing, introducing a delay in frame playout times severely impairs interactive communication. Thus, the previously proposed schemes do not work well in this environment.

Forward error correction (FEC) Over direct broadcast satellite links or cable modems, feedback channels are highly bandwidth limited and contention-based. Some mobile wireless hosts simply do not have extra capacity to frequently send feedback to the sender. Furthermore, in video multicast, it is not desirable to have direct feedback from each receiver to the sender because of the known ramification of the acknowledgment implosion problem. In all of these circumstances, using feedback is very limiting.

FEC is a very compelling alternative for these environments. A Reed Solomon Erasure correcting code (RSE code), such as the one described by McAuley [24], is a commonly used FEC encoder where k source packets of P bits, $d_1, d_2, \ldots, d_k$, are encoded into $n > k$ packets of P bits (i.e., $n - k$ parity packets, $p_1, p_2, \ldots p_{n-k}$). This n packets is called as a *FEC block*. The RSE decoder at the receiver side can reconstruct the source data packets using any k packets out of its FEC block. Efficient (n, k) RSE encoding and decoding algorithms have been developed and implemented to achieve real-time performance [2, 14, 34, 27]. For instance, the throughput of the software coder by Rizzo [34] can achieve 11 MB/s on a 133 MHz Pentium.

In P-RESCU, the original data of a periodic frame packetized into k source packets are transmitted at the frame interval of the periodic frame and then its $n - k$ parity packets are transmitted over its PTDD

period. The transmission time of each parity packet is evenly spaced over the period, interleaving with the packets of other frames. Figure 3 shows a transmission sequence of data and parity packets in P-RESCU. When several data packets are lost, the corresponding periodic frame and accordingly, its dependent non-periodic frames will be displayed with errors. However, as following parity packets are received to recover the original data packets, the periodic frame can be restored. This will cause the remaining non-periodic frames within the PTDD period and the next periodic frame to be displayed without errors if these frames are correctly received.

Our FEC scheme differs from conventional FEC schemes. Conventional FEC schemes can be categorized into two schemes. One scheme is to transmit both data and their parity packets within the same frame interval. The other scheme is to transmit the parity packets in later frame intervals than the interval in which data packets are sent. The former scheme is susceptible to burst packet losses. While this scheme can be effective under random packet losses, under burst losses, data and parity packets can be lost together. The latter scheme has to introduce additional delays in frame playout times to allow enough time for the receiver to receive parity packets and restore the currently displayed images. Although these schemes can be effective for a one-way, near-real-time video transmission, it seriously impairs interactive video communication. In contrast, our scheme does not have these drawbacks.

Hybrid ARQ schemes One serious drawback of FEC is that it wastes bandwidth during error-free transmission. In order to guard against long burst errors, many parity packets have to be transmitted at all times since it is unpredictable when such errors occur. However, when recoverable packet losses occur, FEC can quickly repair the losses. In contrast, retransmission does not incur much bandwidth overhead because packets are retransmitted only when some indication exists they are lost. However, since packet losses have to be detected and notified to the sender before retransmission occurs, it introduces delays in recovery which prolong error propagation.

A good balance among the advantages and disadvantages of FEC and retransmission can be achieved if the two schemes can be combined. A hybrid ARQ technique of Type-I [7] can be used as follows. First, the original data packets of a periodic frame are sent within its frame interval. Then a small number of their parity packets are transmitted over a fixed interval to guard against random and short burst losses. When irrecoverable long burst losses occur (i.e., as soon as the receiver detects that incoming or received parity packets cannot recover all the lost packets of the original data packets), the receiver notifies the source which immediately schedules the retransmission of missing packets. The sender retransmits only the additional number of parity packets sufficient to recovery fully the original periodic frame when they are combined with outgoing or received parity packets. For instance, when $(4, 6)$ RES encoding is used, if three

data packets are lost in a burst, and the receiver detects the loss before receiving the two parity packets on the way to the receiver, the receiver sends a request for retransmission of only one additional parity packet. Since long burst errors happen relatively rarely, the initial transmission of parity packets can be effective in quickly repairing most of losses. Since retransmission can recovers long burst losses, many parity packets do not have to be sent a priori, which reduces wasted bandwidth.

3 Simulation

We showed the efficacy of RESCU with retransmission in [33, 32]. In this section, we show through simulation that the performance of RESCU can be enhanced by incorporating FEC. Under various network conditions characterized by packet loss rates, burst loss lengths, and transmission delays, we measure the effectiveness of FEC and retransmission in terms of the end video quality and bandwidth overhead.

In the remainder of this section, we show the followings.

1. We show the bit efficiency advantage of RESCU-FEC over H.261 in achieving the same level of error resilience. H.261 can improve error resilience by transmitting intra-frames more frequently.
2. We show the impact of burst losses to RESCU-FEC to the final image quality and bit efficiency.
3. We show the impact of network delays to RESCU-REC to the final image quality and bit efficiency.
4. We show that there is a hybrid RESCU scheme that combines the benefits of FEC and REC so that the scheme is less sensitive to both burst losses and network delays.

3.1 Simulation Method

We model burst packet losses using a two state continuous Markov chain $\{X_t\}$ where $X_t \in \{0, 1\}$. A packet transferred at time t is lost if $X_t = 1$ and not lost if $X_t = 0$. The infinitesimal generator of this Markov chain is

$$Q = \begin{pmatrix} -\mu_0 & \mu_0 \\ \mu_1 & -\mu_1 \end{pmatrix}$$

The stationary distribution associated with this chain is $\pi = (\pi_0, \pi_1)$ where $\pi_0 = \mu_1/(\mu_0 + \mu_1)$ and $\pi_1 = \mu_0/(\mu_0 + \mu_1)$. Let $p_{i,j}(t)$ be the probability that the process is in state j at time $t + \tau$ given that it was in state i at time τ. These probabilities are given in [26] (chap. 6). Let λ be the packet transmission

rate, b the expected number of consecutively lost packets, and p the packet loss probability. Then $\mu_0 = -\pi_1 \lambda \log(1 - 1/b)$ and $mu_1 = \mu_0(1-p)/p$. The network delay is modeled by the exponential distribution with the mean delay D. Network conditions are characterized by the loss probability p, the mean burst loss length b, and the mean network delay D.

RESCU is implemented based on a H.261 codec. We use the full-search motion estimation for all experiments and PTDD is varied for different experiments. C-RESCU with one additional buffer is used for retransmission. We use the test image sequences that are obtained from the MPEG-4 test sequences encoded by Telenor H.263 encoder. In this paper, we show the result from a MPEG-4 class B test video sequence (with a medium level of motion) called *container*. For every experiment, the frame rate is set to 10 frames per second. The image size of CIF (352×288 color) are used for experiments. Each encoded image sequence is packetized into approximately 256-byte packets.

Given a packetized sequence, we obtain transmission traces of the sequence which contain information about the delivery time of each packet, and the number of retransmission attempts. Using the above network model, we calculate the delivery times. When retransmission is used for recovery, for each lost packet that belongs to a periodic frame, we find out whether the packet is received by retransmission before its deadline. The deadline is determined by the time period between the initial deadline of the packet and the time when the retransmission is made. Each retransmission attempt costs one round trip time which is calculated from the network model. A packet can be retransmitted as many times as it is allowed by its deadline. When the packet is received by retransmission, we record the time that the packet is received.

After obtaining a transmission trace of a video sequence, we run the decoder on the trace to measure the image distortion due to packet losses. The image distortion is computed using the peak signal-to-noise ratio (PSNR) of decoded images over the original images.

3.2 Simulation Results

3.3 RESCU with forward error correction (RESCU-FEC)

We first measure the bandwidth overhead of H.261 and RESCU-FEC. H.261 can improve its error resilience by transmitting intra-frames more frequently. RESCU-FEC can increase its error resilience by transmitting more parity packets or spreading a given number of parity packets over a longer PTDD period to guard against burst losses. However, these techniques have different impact on bandwidth overhead and video quality. Figures 4 and 5 show the video quality and bit rates of the two scheme over different PTDD periods under 10% losses, burst loss length 2, and network delay 100ms. In H.261, one intra-frame is transmitted

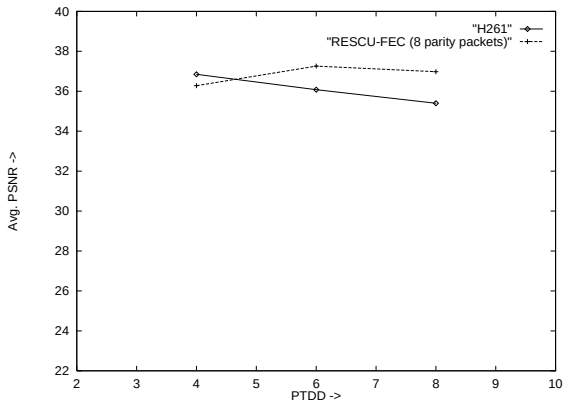

Figure 4: Comparison in average PSNR of H.261 and RESCU-FEC (with 8 parity packet)

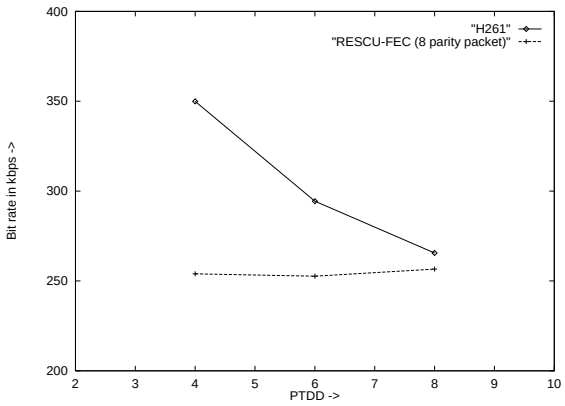

Figure 5: Comparison in mean bit rate of H.261 and RESCU-FEC (with 8 parity packet)

within one PTDD period[1], and in RESCU-FEC, 8 parity packets are transmitted over one PTDD period. Under PTDD 4, H.261 shows a slightly better PSNR (0.4dB) than RESCU-FEC, but with about a 40% higher bit rate. This higher PSNR is because intra-frames in H.261 tend to have better image quality than periodic frames in RESCU-FEC.

The PSNR of RESCU-FEC improves as PTDD increases. This is because parity packets can be spread over a longer period, allowing more effective packet interleaving. Thus, the transmission becomes less susceptible to burst losses. In contrast, H.261 loses PSNR more than 1dB as PTDD increases. This is because of error propagation between two intra-frames, and as this interval increases, error propagation persists longer. The PSNR of H.261 falls 1dB below that of RESCU-FEC under PTDD 8. This is because RESCU-FEC can recover losses in periodic frames more quickly, and this allows some of non-periodic frames within a PTDD to be displayed without error propagation. Recall that in RESCU, non-periodic frames uses only periodic frames as reference frames. Thus, continuous updates in a periodic frame subsequently improves the quality of non-periodic frames depending on that periodic frame.

The bit rate of H.261 decreases as intra-frames are added less frequently. In the testing sequence, intra-frame uses about 6.5 times more bits than inter-frames. However, the bit rate of RESCU-FEC does not vary much as PTDD increases. This is because as PTDD increases, compression efficiency of RESCU decreases. Thus, less frequent transmission of parity packets is compensated by the decrease in compression efficiency.

Under PTDD 6, RESCU-FEC gives the approximately same PSNR as H.261 under PTDD 4 with 40% less bit consumption.

[1]Note that H.261 does not have any periodic temporal dependency, and we call the frame intervals between two consecutive intra-frames PTDD just for convenience.

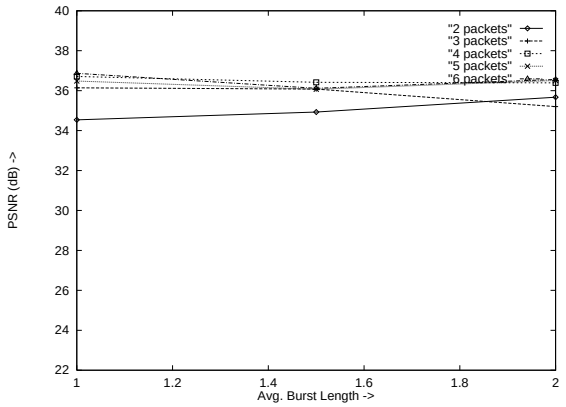

Figure 6: Average PSNR of RESCU-FEC with various numbers of parity packets within PTDD 6 over different loss burst lengths

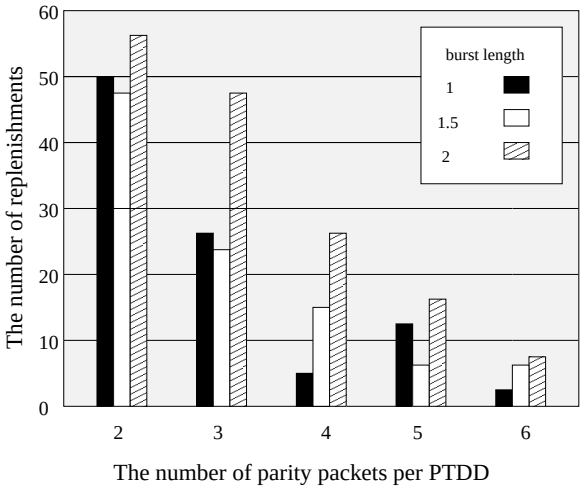

Figure 7: The number of replenishments for RESCU-FEC with various numbers of parity packets within PTDD 6 over different loss burst lengths (within a 2 minute play period)

Figures 6, 7, and 8 illustrate the impact of burst losses in the performance of RESCU-FEC. Only the loss rate of 10% is shown. While the PSNR and bit rates remains relatively the same over different burst lengths, the replenishment count shows the effect of bursts. Replenishment with an intra-frame occurs when RESCU-FEC fails. Thus, the count represents how effective RESCU-FEC with a given amount of redundancy can be. Clearly from Figure 7, we can see that two parity packets within PTDD 6 are not enough in all loss burst lengths tested – more than 48 replenishments within the 2 minute playout time were made in all cases. The count reduces as more redundant packets are added. It is also shown that a long loss burst length causes more replenishments. Under burst length 1, four parity packets are enough while under burst lengths 1.5 and 2, five and six parity packets are needed. Since replenishment confines error propagation very well, the video quality of RESCU-FEC does not show many variations under different burst lengths. However, when only two parity packet are used, the quality degrades quite severely. The bit rate does not change much either because less redundancy causes more replenishments.

3.4 Retransmission Based Error Control (REC)

Figures 9, 10, and 11 show the effect of network delays on the video quality of RESCU-REC with different PTDD periods. The figures show the result of experiments with 10 % loss rate. The performance of RESCU-REC was fairly insensitive to loss burst lengths. So we show only the result of loss burst length 1.5.

The simulation result shows that under low network latency, RESCU-REC performs very well even with

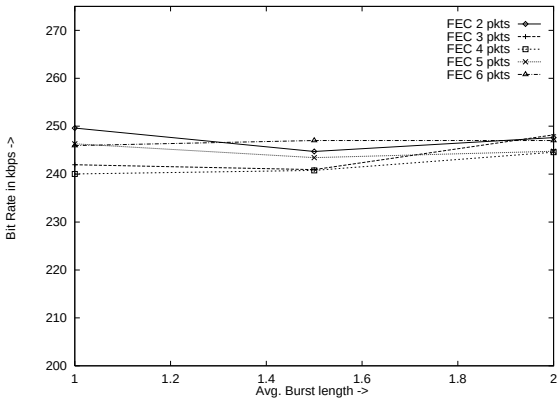

Figure 8: Average bit rate of RESCU-FEC for various numbers of parity packets within PTDD 6 over different loss burst lengths

a short PTDD period. Under network delay 50ms, the PSNR of RESCU-REC is around 37dB. However as network delays become longer, video quality dramatically reduces. This is even more severe for the case under a short PTDD period where most retransmission attempts fail. This is shown by the high replenishment count. Under long delays, as PTDD increases, video quality generally improves. This is because PTDD determines the deadline of packets – the shorter the PTDD is, the shorter the deadline is. Thus, when network delays are long and PTDD is not sufficiently large, most retransmitted packets are not received before their deadlines. Thus video quality degrades.

The bit overhead incurred by retransmission is very small when PTDD is sufficiently large. The simulation results showed that retransmission uses a little bit over 10% of the original bit rate under loss rate 10%. This is less than 15 to 20% overhead of FEC when 2 to 6 parity packets are added. However, as delays get longer under a fixed PTDD period, the bit rate increases because retransmission fails to recover periodic frames, which causes replenishment. However, setting PTDD arbitrarily high has an adverse effect on compression efficiency. This is shown in Figure 11 where under 50ms, the bit rate increases proportionally as PTDD increases. These effects of long delays and long PTDD on the bit rate are shown in the U-shape of the bit rate graph under 150ms. It indicates that the effectiveness of RESCU-REC highly depends on finding the minimal PTDD period that gives the best video quality under a given network environment.

One advantage of FEC over retransmission is that FEC repairs lost packets more quickly than retransmission since no loss detection and feedback delays are incurred in FEC. In RESCU-REC, lost packets are detected only by a gap in received packet sequence numbers, and furthermore feedback has to travel to the sender to trigger retransmission. This effect is shown in Figures 6 and 9 where under a short loss burst length, FEC seems to perform better than retransmission under a larger network delay even with a

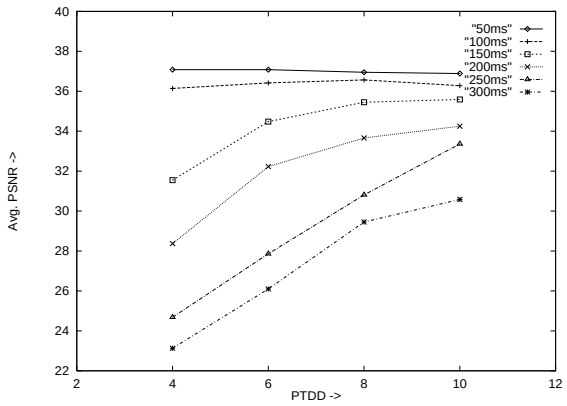

Figure 9: Average PSNR of RESCU-Retransmission with different one-way trip network delay

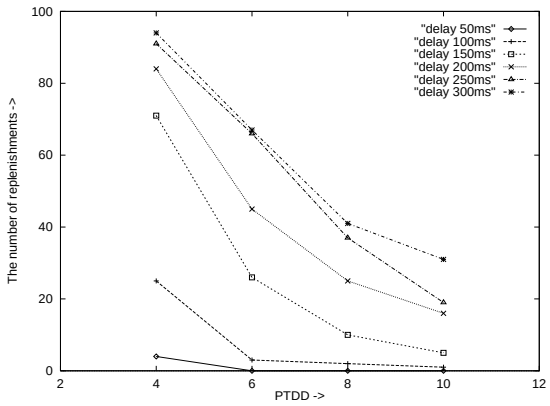

Figure 10: The number of replenishments for RESCU-Retransmission with different one-way trip network delay (within a 2 minute play period)

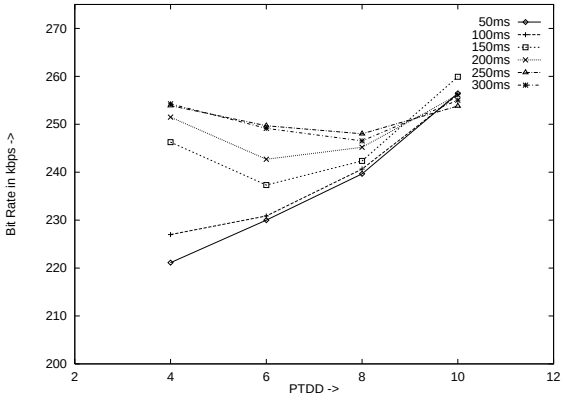

Figure 11: Average bit rate of RESCU-Retransmission with different one-way trip network delay

longer PTDD period. For instance, while most FEC schemes show above 36dB under loss burst length 1, retransmission maintains below 36dB under any delay larger than 150ms.

A high replenishment count under high network latency does not improve video quality very much. This is because replenishment is also affected by network delays. After the receiver recognizes irrecoverable losses in a periodic frame, it takes at least one round trip delay before the receiver sees an intra-frame. The impact of this delay on quality is is shown in Figure 9 where under PTDD 4, higher delays give a much lower PSNR even with much a higher replenishment count. In contrast, RESCU-FEC experiments show in Figure 6 that when network delays are fixed to 100ms, replenishment made video quality appear similar for all loss burst lengths.

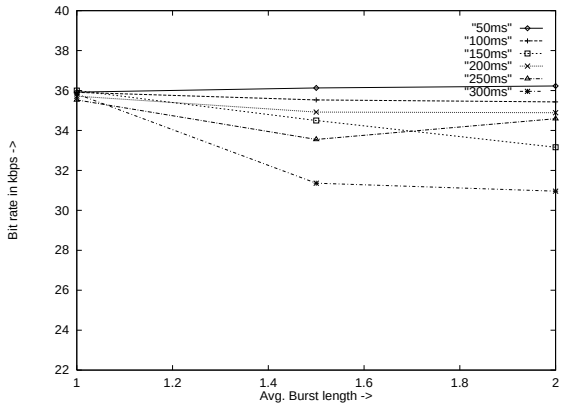

Figure 12: Average PSNR of RESCU-hybrid with different one-way trip network delay

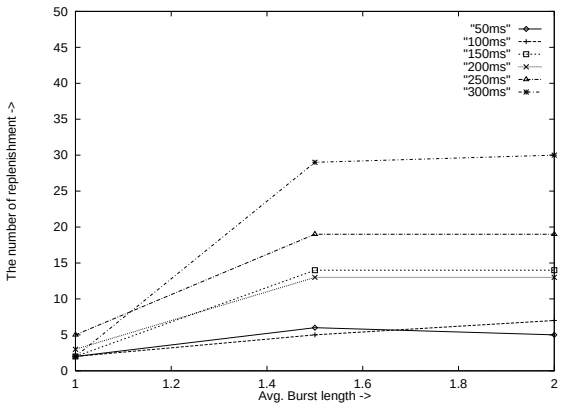

Figure 13: The number of replenishments for RESCU-hybrid with different one-way trip network delay (within a 2 minite play period)

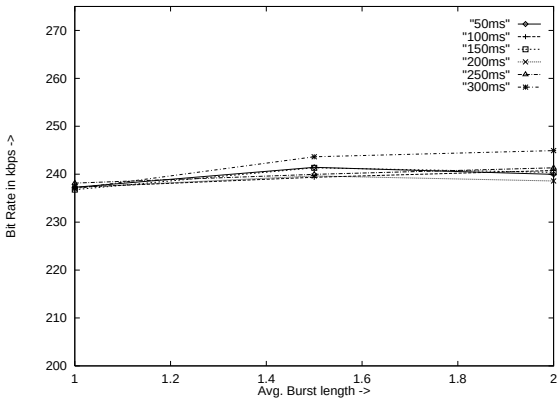

Figure 14: Average bit rate of RESCU-hybrid with different one-way trip network delay

3.5 Hybrid ARQ

Figures 12, 13, and 14 show the performance of RESCU-hybrid under PTDD 6 with 3 parity packets. The figures show the result of simulation experiments with loss rate 10%.

Under a short loss burst length, the performance of RESCU-hybrid shows similarity with that of RESCU-FEC, and under a long loss burst length, with that of RESCU-REC. For instance, under loss burst length 1, the PSNR of RESCU-hybrid is around 36dB for all network delays. This behavior is similarly observed in the experiments of RESCU-FEC. Under burst lengths 1.5 and 2, the PSNR of RESCU-hybrid shows a spectrum between 31dB and 36dB. This spectrum is similarly observed in the experiments of RESCU-REC.

The improvement of RESCU-hybrid over RESCU-FEC and RESCU-REC is more clearly shown in

Figure 13. The replenishment count of RESCU-FEC with three parity packets is about 30% to 500% more than that of RESCU-hybrid, and the count of RESCU-REC for PTDD 6 is about more than 250% higher than that of RESCU-hybrid. This reduction in the replenishment count results in the decrease of the bit rate shown in Figure 14. The replenishment count is very small at burst length 1 while between loss burst lengths 1.5 and 2, the count remains almost constant. This is because when the loss burst length is short, three parity packets are sufficient to recover lost packets and thus, its performance becomes less sensitive to network delays. When the loss burst length increases, retransmission becomes more effective in recovering packet losses. Since retransmission is less sensitive to the loss burst length, the replenishment count does not chance much between loss burst lengths 1.5 and 2.

The bit rate of RESCU-hybrid is in between those of RESCU-FEC and RESCU-REC. To achieve comparable video quality, RESCU-FEC needs a bit rate between 240 and 250 kbits/s, and RESCU-REC needs 220 to 260 kbits/s. The overhead is much more than that of RESCU-REC when network latency is low and PTDD is short, but much less than that of RESCU-REC when network latency is high and PTDD is long. It is much more than that of RESCU-FEC under short burst lengths with fewer parity packets while much less under long loss burst lengths with more parity packets.

In summary, the simulation results indicate that RESCU-hybrid can achieve good balance between the advantages and disadvantages of RESCU-FEC and RESCU-REC. Under low loss burst, it shows less sensitivity to network delays (an improvement over RESCU-REC), and when PTDD is sufficiently large, it shows less sensitivity to loss burst lengths (an improvement over RESCU-FEC). Thus video quality is reasonaly good (32dB to 36dB) with only about 30% bit overhead over H.261 with PTDD 100 (188kbits/s).

4 Related Work

There are many different types of error recovery techniques for video. We divide them into three main categories: (1) feedback-based recovery which relies on feedback from a receiver to recover video quality from packet losses, (2) proactive error recovery which suppresses errors before they occur, and (3) hybrid recovery which combines the above two techniques. Error concealment is one of widely proposed error control techniques (e.g.,[10, 15, 40, 23]). Although error concealment techniques can be combined with error recovery techniques, they are not strictly error recovery techniques [6], so we do not discuss them. We also focus on recovery techniques for video transmission.

4.1 Feedback-based recovery

Recently H.263+ incorporated two feedback-based techniques: error tracking and reference picture selection. Error tracking (ER) utilizes the intracoding of blocks to stop error propagation, but limits its use to severely impaired image regions only. ER requires the encoder to know the location and extent of erroneous image regions in displayed images. This can be achieved by feedback from the receiver. The receiver sends information about missing packets, and the encoder estimates the region of error propagation in the displayed images, and intracodes those blocks contained the region. The reference picture selection (RPS) mode allows the encoder to select one of several previously decoded frames as a reference picture for motion estimation. It is designed To support a coding technique called NEWPRED [18]. In NEWPRED, using feedback from the receiver, the encoder uses for prediction only those pictures that are reported to be received (in the *ACK mode*), or not reported missing (in the *NACK mode*). Since motion prediction is always based on the frames that are received by the receiver, error propagation is effectively eliminated.

Retransmission has recently attracted much attention for packet loss recovery in video transmission. Ramamurthy and Raychaudhuri [31] applied a similar technique to video transmission over ATM. They analyzed the performance of video transmission over an ATM network when both retransmission and error concealment are used to repair errors occurring from cell loss. Padopoulos and Parulkar [28] proposed an implementation of an ARQ scheme for continuous media transmission. Various techniques including selective repeat, retransmission expiration, and conditional retransmission are implemented inside a kernel. Their experiment over an ATM connection showed the effectiveness of their scheme.

Retransmission is also used for video multicast. Li *et al.* [20] proposed an elegant scheme for multicasting MPEG-coded video. By transmitting different frame types (I, P and B frames) of MPEG to different multicast groups, they implemented a simple layering mechanism in which a receiver can adjust frame playout times during congestion by joining or leaving a multicast group. Retransmission is used for high priority streams. The scheme is shown to be effective for non-interactive real-time video applications. In a video conferencing involving a large number of participants, different participants may have different service requirements. While some participants may require real-time interactions with other participants, others may simply want to watch or record the conference. Xu *et al.* [39] contended that retransmission can be effectively used for the transmission of high quality video to the receivers that do not need a real-time transfer of video data. They designed a new protocol called *structure-oriented resilient multicast* (STORM) in which senders and receivers collaborate to recover lost packets using a dynamic hierarchical tree structure.

Remarks A drawback of feedback-based techniques is that when the networks do not support feedback channels, they are useless. If feedback channels are supported, but limited in bandwidth, then those tech-

niques requiring frequent feedback may not be effective. Another drawback of ER and NEWPRED is that they are not suitable for multicast. Since they compensate for packet losses by altering the coding pattern based on information about missing packets, if there is more than two receivers with widely different loss patterns, then their coding efficiency will drastically decrease. All of the mentioned retransmission techniques use frame playout delays to compensate for retransmission delays in high-latency networks.

4.2 Proactive recovery

Error propagation can be alleviated by intracoding more image blocks, but at the expense of compression efficiency. Several popular video conferencing tools, such as `nv`[9], `vic`[25] and `CU-SeeMe`[8], adopt this approach. Using a technique called *conditional replenishment*, these tools filter out the blocks that have not changed much from the previous frame and intra-code the remaining blocks. Since all the coded blocks are temporally independent, packet loss affects only those frames that are contained in lost packets.

FEC has been successfully applied to audio transmission [4, 3, 29, 30]. There are only a few studies on applying FEC to video transmission. *Priority encoding transmission* (PET) [1, 19, 36], encodes different segments of video data with different priority. Each packet contains relatively more redundant information about the higher priority segments of the data, so the information with a higher priority can have a higher chance of correct reception. The PET scheme is also incorporated into `vic` [25] and is reported a good performance [38]. This good performance is partially due to `vic`'s intracoding which limits error propagation caused by loss of lower priority segments.

Bolot and Turletti [5] proposed an interesting FEC technique for packet video where a packet contains the redundant information of some of previous packets. The redundant information is created by encoding the image blocks contained in the previous packets with a large quantization step. They claimed that if the video source is not bursty, long burst losses are rare, and the proposed scheme would work well for video.

H.263+ also includes a similar technique called *independent segment decoding* (ISD) [17]. In the ISD mode, each video slice is encoded as an individual picture (or subvideo) independent of other slices. In particular, each slice boundary is treated just like picture boundary. ISD does not eliminate error propagation, but limits the extent of error propagation to a slice.

MPEG-4 adopted several error resilient techniques for wireless video transmission [37]. These include resynchronizations strategies, data partitioning, reversible VLCs, and header extension codes. Most of these techniques focus on preventing lost data from affecting the decoding of received data.

Remarks A drawback of proactive techniques is that it wastes bandwidth under error-free transmission. Furthermore, in wireless mobile networks, the assumption made in [5] does not hold as long burst losses

can be common due to fading and channel interference. Thus, the FEC schemes mentioned above are susceptible to burst errors because both data and FEC-encoded packets have to be transmitted at the same frame interval. Also if FEC-encoded packets are transmitted over a longer period, additional playout delays may be incurred. MPEG-4's techniques can be used along with RESCU to further reduce the effect of data loss.

4.3 Hybrid recovery

Hybrid techniques combine ARQ (retransmission) and FEC for better error resilience. There are two types of hybrid techniques: *type-I hybrid ARQ* [7], and *type-II hybrid ARQ* [21]. Type-I hybrid ARQ transmits both error detection and correction data at the initial transmission of data. If the receiver cannot recover lost packets using the transmitted parity data, it requests retransmission of the same data from the sender. Type-II hybrid ARQ does not send any redundant data with the first transmission, but sends only parity data when retransmission is requested.

Liu and El Zarki [22] applied a hybrid ARQ technique for video transmission. They proposed a hybrid ARQ technique that combines the benefit of type-I and type-II techniques, and showed its efficacy for video transmission over wireless networks. They showed that using rate-compatible punctured convolutional (RCPC) codes [11, 13], one or two retransmission attempts achieve a low packet error rate under a perfectly interleaved Rayleigh fading channel.

Hybrid ARQ techniques were also studied in reliable multicast [35, 27]. While FEC helps reduce occurrences of independent losses, retransmission repairs correlated packet losses. They showed that hybrid ARQ reduces the bandwidth overhead of repairing packet losses in reliable multicast involving many receivers.

Remarks Although hybrid ARQ schemes work better than FEC or retransmission alone, they do not overcome the limitations of traditional recovery techniques. Since retransmission always incurs delays and FEC is still susceptible to burst losses, hybrid ARQ scheme still fall short of handling the retransmission' delays and burst losses without introducing delays in frame playout delays. Delaying frame playout reduces interactiveness and introduces annoying jitters.

References

[1] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan. Priority encoding transmission. *IEEE Transactions on Information Theory*, 42(6), November 1996.

[2] N. Alon and M. Luby. A linear time erasure-resilient code with nearly optimal recovery. *IEEE Transactions on Information Theory*, 42(6):1732–1736, Nov 1996.

[3] E. W. Biersack. Performance evaluation of FEC in ATM networks. In *Proceedings of the ACM SIGCOMM*, Baltimore, MD, August 1992.

[4] J. Bolot and A. Vega-Garcia. The case for FEC-based error control for packet audio in the internet. *ACM Multimedia Systems Journal (to appear)*.

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

[6] G. Carle and E. Biersack. Survey of error recovery techniques for ip-based audio-visual multicast applications. *IEEE Network*, December 1997.

[7] H. Deng and M. Lin. A type I hybrid ARQ system with adaptive code rates. *IEEE Transactions on Communications*, COM-46(2):733–737, Feb. 1995.

[8] T. Dorcey. Cu-seeme desktop videoconferencing software. *ConneXions*, 9(4), March 1995.

[9] R. Fredrick. Network video(nv). Technical report, Xerox Palo Alto Research Center.

[10] M. Ghanbari and V. Seferidis. Cell-loss concealment in ATM video codecs. *IEEE Transactions on Circuits and Sys. for Video Tech.*, 3(3):238–47, June 1993.

[11] J. Hagenauer. Rate-compatible punctured convolutional codes (rcpc codes) and their applications. *IEEE Transactions on Communications*, 36(4):389–400, April 1988.

[12] H. Hessenmuller. Video signal transmission in ATM-based broadband network-treatment of cell losses. In *3rd Int. Workshop on Packet Video*, March 1990.

[13] S. Kallel and D. Haccoun. Generalized type-II hybrid ARQ scheme using punctured convolutional coding. *IEEE Transactions on Communications*, 38(11):1938–1946, November 1990.

[14] S.K. Kaseta, J. Kurose, and D. Towsley. Scalable, reliable multicast using multiple multicast groups. In *Proceedins of ACM SIGMETRICS*, pages 64–74, Seattle WA, 1997.

[15] L. Kieu and K. Ngan. Cell-loss concealment techniques for layered video codecs in an ATM network. *IEEE Transactions on Image Processing*, 3(5):666–77, September 1994.

[16] T. Kinoshita, Nakahashi, and M. Takizawa. Variable bit-rate hdtv coding algorithm for ATM environments in B-ISDN. In *Proceedings of SPIE Conference on Visual Communications and Image Processing: Visual Communication*, pages 604–612, November 1991.

[17] LBC. Document, LBC-95-309 (ITU-T SG 15, WP 15/1), Sub-videos with retransmission and intra-refreshing in mobile/wireless environments, 1995.

[18] LBC. Document, LBC-96-033 (ITU-T SG 15, WP 15/1), An error-resilience method based on back channel signalling and FEC, 1996.

[19] C. Leicher. Hierarchical encoding of MPEG sequences using priority encoding transmission (pet). Technical Report 94-058, ICSI, November 1994.

[20] X. Li, S. Paul, P. Pancha, and M. Ammar. Layered video multicast with retransmission (lvmr):evaluation of error recovery schemes. In *Proceedings of the Sixth International Workshop on Network and Operating System Support for Digital Audio and Video, St Louis*, May 1997.

[21] S. Lin and D. Costello. *Error control coding: fundamentals and applications*. Prentice-Hall Inc., Englewood Cliffs, NJ, 1983.

[22] H. Liu and M. El Zarki. Performance of video transport over wireless networks using hybrid ARQ. In *Proceedings of Proceedings of IEEE International Conference on Universal Personal Communications (ICUPC)*, pages 567–571, Boston, MA, October 1996.

[23] W. Luo and M. El Zarki. Analysis of error concealment schemes for MPEG-2 video transmission over ATM networks. In *Proceedings of the SPIE/IEEE Visual Communications and Image Processing*, Taiwan, May, 1995.

[24] J. McAuley. Reliable broadband communications using a burst erasure correcting code. In *Proceedings of the ACM SIGCOMM*, Philadelphia, PA, September 1990.

[25] S. McCanne and V. Jacobson. vic: a flexible framework for packet video. In *Proceedings of ACM Multimedia'95, San Francisco, CA*, pages 511–522, November 1995.

[26] P. Morse. *Queues, Inventories, and Maintenance*. John Wiley, 1958.

[27] Jorg. Nonnenmacher, Ernst Biersack, and Don Towsley. Parity-based loss recovery for reliable multicast transmission. *ACM Transactions on Networking (to appear)*.

[28] 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.

[29] M. Podolsky. A study of speech/audio coding on packet switched networks. Technical report, MS Thesis, Dept. of Electrical Eng. and Computer Sciences, Univ of California, Berkeley, December 1996.

[30] M. Podolsky, C. Romer, and S. McCanne. Simulation of FEC-based error control for packet audio on the internet. In *Proceedings of the ACM SIGMETRIC/PERFORMANCE*, June 1998.

[31] G. Ramamurthy and D. Raychaudhuri. Performance of packet video with combined error recovery and concealment. In *Proceedings of the IEEE INFOCOM*, pages 753–761, April 1995.

[32] I. Rhee. Error control techniques for interactive low-bit rate video transmission over the internet. In *Proceeding of the ACM SIGCOMM'98 (to appear)*, Vancouver, Canada, Sept. 1998.

[33] I. Rhee. Retransmission-based error control for interactive video applications over the internet. In *Proceedings of International Conference on Multimedia Computing and Systems*, pages 118–127, Texas, Austin, June 1998.

[34] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *Computer Communication Review*, 27(2):24–36, Apr 1997a.

[35] Dan Rubenstein, Jim Kurose, and Don Towsley. Real-time reliable multicast using proactive forward error correction. *NOSSDAV 98 (to appear)*.

[36] R. Storn. Modeling and optimization of pet-redundancy assignment for MPEG sequences. Technical Report TR-95-018, ICSI, Berkeley, CA, May 1995.

[37] R. Talluri. Error-resilient video coding in ISO MPEG-4 standard. *IEEE Communication Mag.*, 36(6):112–119, June 1998.

[38] Priority Encoding Transmission. Web page: http://www.icsi.berkeley.edu/pet/icsi-pet.html.

[39] R. Xu, C. Myers, H. Zhang, and R. Yavatkar. Resilient multicast support for continuous-media applications. In *Proceedings of the Sixth International Workshop on Network and Operating System Support for Digital Audio and Video*, St. Louis, May 1997.

[40] J. Zdepski and H. Sun. Error concealment strategy for picture-header loss in MPEG compressed video. In *Proceedings of High-Speed Networking and Multimedia Computing*, pages 145–152, San Jose, CA, Feb 1994.

[41] M. Zorzi and R. Rao. Capture and retransmission control in mobile radio. *IEEE Journal on Selected Areas in Communications*, 12(8):1289–1298, 1994.