

Error Recovery for Interactive Video Transmission over the Internet*

Injong Rhee[†]

Srinath R. Joshi

Department of Computer Science

North Carolina State University

Raleigh, NC 27695-7534, USA

{rhee, srjoshi}@csc.ncsu.edu

April 1999

Abstract

Real-time interactive video transmission in the current Internet has mediocre quality because of high packet loss rates. Loss of packets in a video frame manifests itself not only in the reduced quality of that frame but also in the propagation of that distortion to successive frames. This error propagation problem is inherent in any motion compensation-based video codec. In this paper, we present a new error recovery scheme, called *Recovery from Error Spread using Continuous Updates* (RESCU), that effectively alleviates error propagation in the transmission of interactive video. The main benefit of the RESCU scheme is that it allows more time for transport-level recovery such as retransmission and forward error correction to succeed while effectively masking out delays in recovering lost packets without introducing any playout delays, thus making it suitable for interactive video communication. Through simulation and real Internet experiments, we study the effectiveness and limitations of our proposed techniques and compare their performance with that of existing video error recovery techniques including H.263+ (NEWPRED). The study indicates that RESCU is effective in alleviating the error spread problem and can sustain much better video quality with less bit overhead than existing video error recovery techniques under various network environments.

Key words: Computer Network Protocols, Packet Loss Recovery, Interactive Video Transmission, Error Propagation, Forward Error Correction, Retransmission.

1 Introduction

Transmitting high-quality, real-time interactive video over lossy 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

*This work is supported in part by NSF CAREER ANI-9875651.

[†]contact author: email: rhee@csc.ncsu.edu, Phone: 919-515-3305, Fax: 919-515-7925

transmission requires extremely high compression efficiency. However, state-of-the-art video compression standards (MPEG I and II, 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 by encoding only pixel value differences between a currently encoded image and a motion-predicted image created from a previously encoded image (or 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 loss recovery focuses on repairing packet losses before the scheduled display times of those video frames contained in the lost packets (e.g., [34, 30, 23, 42, 1, 39, 22, 25]). 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 interactive communication.

Many researchers [34, 30, 23, 42] have proposed using retransmission of lost packets by delaying frame playout times to allow arrival of retransmitted packets *before* the display times of their video frames. 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 interactive communication under the current Internet environment.

Forward error correction is also commonly proposed for error recovery of continuous media transmission [22, 1, 19, 6, 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 losses. Moreover, since FEC is applied to a block of packets, before FEC packets are computed and transmitted, a large delay must transpire.

Contribution In this paper, we propose an entirely complementary approach to the above-mentioned by focusing on eliminating error propagation when distortion on displayed images occurs. Our point of departure from existing approaches is that packets do not have to arrive in time for them to be “useful” for the display of that video frame. Of course, if packets can arrive before the display time of their frame, 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 succeeding frames. In our approach, frames are simply displayed at their normal playout times without any delay, as they are decoded. Thus, if a repair packet arrives after the playout time of its frame, the frame will be displayed with errors. However, if we can buffer the displayed frame in a buffer, and use the “late” repair packet to restore its buffered frame, we can stop error propagation; because the frame is used as a reference frame for its succeeding frames, restoring the reference frame stops error propagation. We call this approach *Recovery from Error Spread using Continuous Updates* (RESCU).

When combined with RESCU, the conventional packet loss recovery techniques (such as retransmission and FEC),

which have been known ineffective for *interactive* video transmission, can be used since RESCU can effectively mask out the delays involved in recovering lost packets. For instance, when FEC is used, RESCU uses FEC packets to restore buffered reference frames. Thus, FEC packets can be transmitted over a relatively longer period, interleaving with the packets of other frames to help reduce the effect of bursty losses. This FEC scheme clearly differs from the conventional ones in that FEC packets can be transmitted over a longer period than a single frame interval without introducing any delay in frame playout times. Note also that our interleaving is different from link-level symbol interleaving [44, 14] where symbols from multiple codewords are interleaved. The granularity of the proposed interleaving is much larger and thus, more effective.

In this paper, we explore the effectiveness of RESCU in enhancing the error resilience of interactive video transmission. We use both simulation and actual Internet experiments to compare the performance of RESCU and other error recovery techniques such as NEWPRED [21]. We study the strengths and weaknesses of RESCU over various network situations. In particular, we investigate to find network environments where a particular transport recovery technique combined with RESCU can or cannot be effective.

Organization Section 2 describes the RESCU scheme and two recovery techniques each combined with retransmission and FEC respectively, Section 3 presents our simulation and experimental results, and Section 4 contains the description of related work. Section 5 summarize our work in this paper and discusses the impact and limitations of our work and future directions.

2 Error Recovery Techniques

2.1 Recovery from Error Spread using Continuous Updates (RESCU)

Although RESCU can be applicable to any video codec that employs motion compensation, we base our discussion on H.261, an International Telecommunication Union (ITU) video standard, for convenience. 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. If you lose any packet(s) belonging to a video frame, not only is that frame shown with distortion but the error also propagates to the succeeding frames until the next synchronization point (an I-frame). However, I-frames cannot be sent often since they have a large number of packets, which would increase bandwidth consumption.

RESCU 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 reconstruction of a currently displayed frame depends on the successful reconstruction of its reference frames. By using the late packets to restore reference frames, we can stop error propagation.

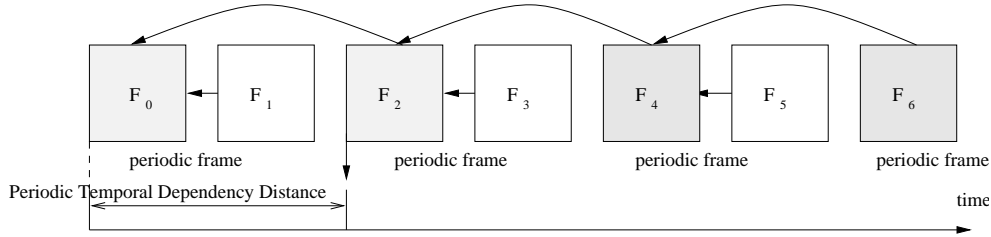


Figure 1: RESCU with PTDD 2

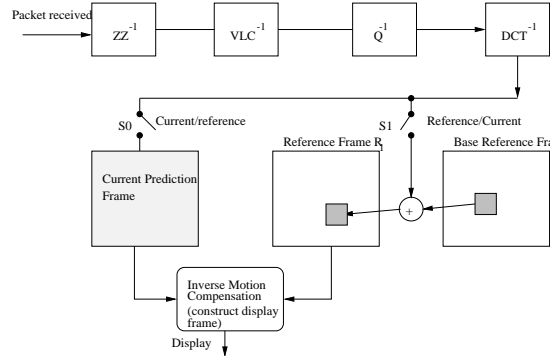


Figure 2: H.261 Decoder modified to handle the recovery of reference frames.

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 which is the number of frame intervals between that frame and the frame it temporally depends on. By extending TDD, we can arrange a frame to be referenced much later than its display time. This adjustment essentially masks out the delay in repairing lost packets. For instance, we can make every p -th frame (which we call *periodic frame*) reference another periodic frame p frame intervals away. The TDD of periodic frame is called *periodic TDD* (PTDD). Every non-periodic frame (frames between two consecutive periodic frames) depends only on its immediately preceding periodic frame. Thus the TDD of the non-periodic frames is between 1 and PTDD-1. Although a periodic frame may be displayed with errors because of some loss of its packets, when these losses can be recovered within a PTDD period, the errors will stop propagating beyond the next periodic frame. Also, errors in non-periodic frames do not propagate at all because all non-periodic frames temporally depend only on periodic frames. Note that extending TDD does not affect frame playout times because all frames are still displayed at their scheduled display times.

Figure 2 shows a H.261 decoder modified to handle the recovery of reference frames using retransmitted packets. The only difference from the original H.261 decoder is one additional frame buffer added to handle the recovery. In the figure, the current frame CP contains only the prediction error and motion vectors of the current frame while the reference frame buffer R_1 contains the fully motion compensated image of the reference frame (i.e., periodic frame) of CP . of the current frame, and the base reference frame buffer R_0 contains the reference frame of R_1 . When a packet is received and decoded into an image block, the decoder determines whether the block belongs to the current frame being decoded or its reference frame. If it is for the current frame, then the block is stored into frame buffer CP



Figure 3: Error propagation



Figure 4: RESCU stops error propagation

along with its motion vector. If it is for the reference frame, the block is added with its temporally dependent block in frame buffer R_0 and stored into R_1 . At each display time, the current frame is constructed using the information in CP and R_1 . If the current frame is a periodic frame, after displaying the frame, R_1 is copied to R_0 and the displayed image are copied to R_1 . In this scheme, as long as the packets belonging to R_1 arrive before the construction of the current frame, the packet can be used to prevent errors in the reference frame from propagating to the current frame.

Figures 3 and 4 show video clips from a proof-of-concept experiment. The distortion in the second picture of Figure 3 is due to packet losses, which propagates even though the rest of frames are correctly received in time. However, in Figure 4, when RESCU is used, the quality of the third picture immediately bounces back when the packets for the second frame are recovered before the decoding of the third frame.

Supporting RESCU in H.263+ does not require any change in the current standard of H.263+. The International Telecommunication Union (ITU) adopted a technique called *reference picture selection* [15] which allows the encoder to select any previously decoded frames as a reference frame for prediction. Since RPS allows the reference frame address of a frame to be encoded with that frame, PTDD can be adjusted by simply modifying this address.

Cascaded RESCU The encoder can determine appropriate PTDD based on the current network 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 a PTDD for the changed environment. This could cause the periodic frame to miss its deadline, resulting in error propagation. Since the frame has been already encoded and transmitted, there is nothing that the encoder can do to save the frame. *Cascaded RESCU recovery* alleviates this problem without involving the encoder (or sender). In RESCU, each periodic frame temporally depends on the previous periodic frames. Thus, by employing more reference frame buffers for periodic frames in the decoder, more late packets can be used to restore a sequence of erroneous periodic frames. Note that cascaded RESCU recovery allows packet deadlines to

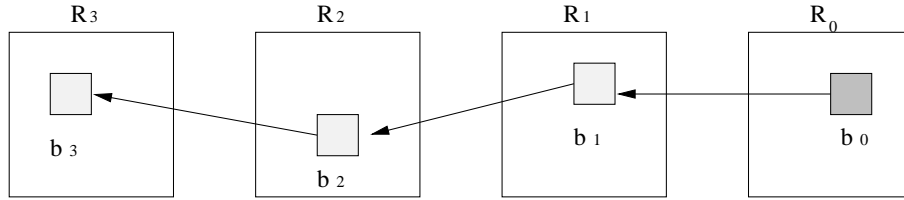


Figure 5: Cascaded RESCU recovery

be extended *at the receiving times*, but not at the encoding times. Figure 5 illustrates this scheme when PTDD is one (i.e., all frames are periodic). The shaded squares represent image blocks and the arrows represent the temporal dependency among blocks. For example, block b_3 depends on block b_2 and so on. Suppose that the current frame's sequence number is f . R_0 is the base reference frame and contains the completely reconstructed picture of frame $f - 4$ while R_i -frame ($i \geq 1$) contains decoded prediction error and motion vectors of its frame ($f - 4 + i$). The image block that corresponds to b_3 can be constructed by adding b_0 , b_1 , b_2 , and b_3 . Cascaded recovery trades additional buffers and computation for increased recovery times.

Replenishment It is also possible that RESCU can fail. When buffers are not available, or PTDD is too short for incurred repair delays, periodic frames cannot be recovered before the decoding of their dependent frame. This also leads to error propagation. To prevent this type of error propagation, we use a commonly adopted technique called *replenishment*. We use replenishment when the receiver detect losses in periodic frames not recovered even after a PTDD period. The receiver notifies the sender about those irrecoverable losses, and the notification triggers the sender to code the next frame as an intra-frame. The intra-frame stops error propagation due to the earlier losses because the intra-frame does not have temporal signal dependency with any of frames transmitted earlier. Since it significantly increases bandwidth consumption, a recovery scheme has to strive to minimize the number of replenishments.

2.2 RESCU + Retransmission

Retransmission is the most commonly used error recovery technique for reliable data transport. The sender (or another receiver in a multicast environment) simply retransmits the packets reported missing by a receiver. Since repair packets are retransmitted only when some indications exist that the packets are lost, retransmission incurs very low bit overhead. In addition, retransmission is less susceptible to burst losses. This is because the time distance between the time when the initial losses occur and the time when the corresponding retransmission effects is large enough for the initial losses to not affect the loss of retransmitted packets.

However, for interactive video transmission, conventional retransmission techniques do not work well. Conventional techniques require retransmitted packets to arrive within a single frame interval after the time that they are first lost, but the associated delays in detecting and retransmitting the lost packets are often larger than one frame interval.

In contrast, RESCU effectively masks out repair delays since retransmitted packets need to be received only within a PTDD period. Figure 6 illustrates error recovery using RESCU and retransmission in a video stream containing two

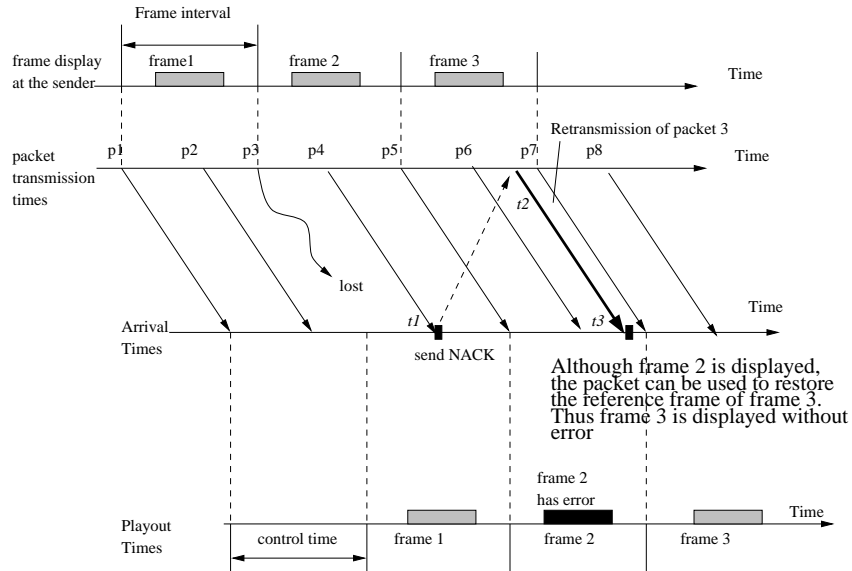


Figure 6: The recovery of frames using retransmission.

packets per frame and PTDD 2. Packet 3 is lost, and the receiver receives packet 4 at time t_1 and recognizing that 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 reference frame of frame 3 (frame 1), so frame 3 can be decoded and displayed without an error. This retransmission technique is fundamentally different from other retransmission techniques [34, 30, 23, 42] in that it does not introduce any delay in frame playout times.

ARQ The automatic repeat request (ARQ) scheme adopted for RESCU is very simple. Suppose that the receiver can buffer up to C periodic frames which can be used for Cascaded RESCU (in our experiments, we set it to 2). The sender assigns a unique integer as a sequence number for each packet being transmitted, and the sequence number is consecutively incremented for each packet. The sender keeps all the packets of the last C periodic frames.

The receiver detects packet losses through gaps in the sequence numbers of received packets. At every time that the first packet of a new frame is received, the receiver sends feedback to the sender notifying all the packets it has not received that belong to the last C periodic frames. When the sender receives the feedback, if the packets reported missing are in the sender's buffers, then it retransmits those packets. If the sender does not have the missing packets (which means that the sender removes them from its buffers), then it simply ignores the feedback.

2.3 Forward error correction (FEC) + RESCU

One main disadvantage of retransmission-based error recovery is that its performance is too sensitive to transmission delays. Although RESCU can accommodate larger transmission delays than conventional retransmission schemes, a larger transmission delay requires larger PTDD. As PTDD increases, compression efficiency gets lower because

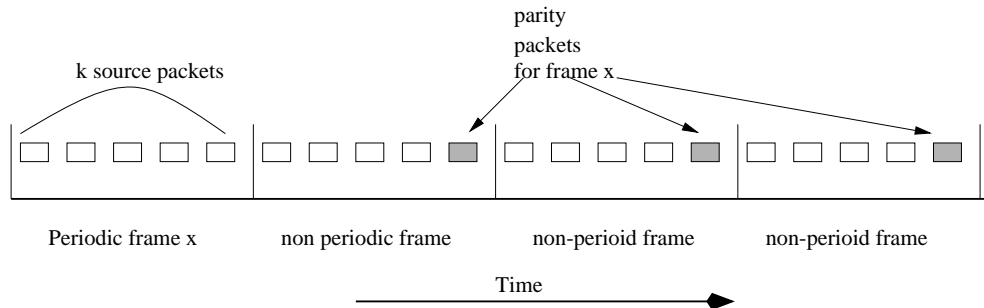


Figure 7: Interleaving of packets in RESCU of PTDD 4 with (5,8) LBC encoding

two consecutive periodic frames may not have much temporal redundancy, and the TDD of non-periodic frames also increase. In addition, packet losses in periodic frames can be restored only after one round trip time. Thus, during the time, non-periodic frames can have errors propagation.

Furthermore, in some networks, sending feedback to the sender can be costly. 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 send feedback frequently to the sender. In video multicast, it is also not desirable to have direct feedback from each receiver to the sender because of the known ramification of the acknowledgment implosion problem. In these circumstances, using feedback is very limiting.

FEC is a very compelling alternative for all these environments. A Reed Solomon Erasure correcting code (RSE code), such as the one described by McAuley [27], is a commonly used FEC encoder where k source packets of P bits are encoded into n ($> k$) packets of P bits (i.e., k data packets plus $n - k$ parity packets). This group of 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, 17, 37, 29]. For instance, the throughput of the software coder by Rizzo [37] can achieve 11 MB/s on a 133 MHz Pentium.

In 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 the PTDD period. The transmission time of each parity packet is evenly spaced over the period, interleaving with the packets of other frames. Figure 7 shows a transmission sequence of data and parity packets in RESCU. When several data packets are lost, the corresponding periodic frame and its dependent non-periodic frames will be displayed with errors. However, as ensuing parity packets can be received to recover the periodic frame, this will cause the remaining non-periodic frames within the PTDD period and the next periodic frame to be displayed without error propagation.

Conventional FEC schemes can be categorized into two kinds. One type 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 and since FEC is applied to a block of packets, before FEC packets are computed and transmitted, large delay must transpire. 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.

3 Experimental Result

The objective of our experimental work is to study the effectiveness of the two transport recovery techniques, retransmission and FEC, each integrated with RESCU for real-time interactive video transmission over the Internet. To achieve this objective, we first show the superior performance of the RESCU techniques to other existing recovery techniques in terms of error resilience and bit overhead. We then investigate the behavior (strength and weakness) of RESCU under various network environments. Of particular interests, is the examination of its behavior under various loss rates, transmission delays, and loss burstiness.

Below, we first describe our experimental methodology, and then in what follows, we discuss the results of the experiments. For convenience, we call the FEC technique integrated with RESCU as *RESCU-FEC* and the retransmission technique combined with RESCU as *RESCU-REC*.

3.1 Testing Methodology

We modified H.261 to incorporate RESCU. Full-search motion estimation, and a default quantization step size 8 are used for all experiments. We use test image sequences that are obtained from the MPEG-4 test sequences encoded by Telenor H.263 encoder. An MPEG-4 class A test video sequence called *container* is used. For every experiment, the frame rate is set to 10 frames per second. The image size of CIF (352×288 color) is used for experiments. The test video sequence is first compressed using each codec and the encoded video frame is packetized into approximately 256-byte packets such that the individual packets contain an integral number of macro blocks.

Cascaded RESCU recovery with one additional buffer is adopted for retransmission only. This effectively doubles the deadline of packets of periodic frames for RESCU-REC. In RESCU-FEC we also space parity packets evenly over a PTDD period, interleaving with the packets of other (non-periodic) frames. Most experiments set the number of parity packets equal to PTDD, thus having one parity packet per frame as shown in Figure 7. Recall that RESCU-FEC generates parity packets only for periodic frames.

We generate a packetized sequence corresponding to 190 frames. This sequence is replayed several times for about 2 minutes (1200 frames). The replay does not reduce the integrity of the experiment because the first frame is always intra-coded in all the tested schemes.

Performance study is conducted in two ways: simulation and actual Internet experiments. In most cases, we use Internet experiment traces to compare the performance of different techniques. However, when studying the effect of particular network parameters, we resort to simulation because it is difficult to control a certain parameter of real network environments. Unless indicated as simulation in performance figures presented in the next section, figures are taken from the results of Internet trace-driven experiments.

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 two-state Markov model is commonly used to model the loss behavior of the Internet [4, 38].

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

Given a packetized sequence, we determine whether each packet is lost or not through the Markov loss model. 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. 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.

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.

Internet-Transmission Test We also conducted actual video transmission tests over the Internet from Korea to US. These testing sites are chosen because transmission delays between two sites vary over a wide range of delays between 100ms and 1 second. The transmission tests were conducted every 45 minutes between Oct.10 and Oct. 13 to obtain traces. Each packet of a frame is transmitted at a regular interval by the given frame rate and the number of packets within that frame. One intra-frame is sent at every 95-th frame in addition to intra-frames sent for replenishment.

We first obtain a packetized sequence of the test video sequence and transmit that sequence by using a recovery scheme. In our transmission tests, we transmit only the packetized sequences of RESCU-REC. For each transmission test, we obtain a 2-minute trace that records the packet sequence numbers, the arrival times of all received packets and the number of retransmission attempts for each packet if any.

For fair comparison between any two schemes, we invent a technique called *mapping* which works as follows. The actual transmission traces are fed into a trace-driven simulator, shown in Figure 8, which employs a well-known UCB/VINT network simulator *ns*. The encoder/packetizer generates a packetized sequence of each frame at a rate of 10 frames per second, and passes it to *ns*. We modified the error model of *ns* so that it maps the state of each packet p in the input trace to that of a received packet q from encoder. If packet p is marked as lost in the trace, then packet q is dropped, and not delivered to the receiver. This process is continued until the trace runs out of packets to map. The

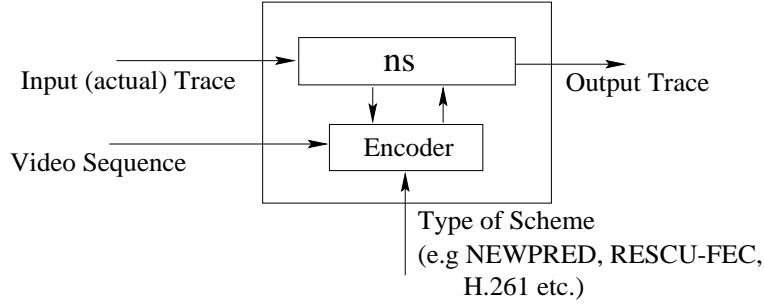


Figure 8: Mapping.

packet number	frame number	packet # in a frame	received Y/N	received time (ms)
0	1	0	Y	201
1	1	1	Y	450
2	1	2	Y	600
3	1	3	N	*
4	1	4	Y	980
5	2	0	Y	1100
6	2	1	Y	1305
7	3	0	N	*
8	3	1	Y	1790
9	3	2	N	*
10	3	3	Y	2100
11	4	0	N	*
12	4	1	N	*
13	5	0	Y	2710
14	5	1	Y	3000

(a)

packet number	frame number	packet # in a frame
0	1	0
1	1	1
2	1	2
3	1	3
4	1	4
5	2	0
6	2	1
7	2	2
8	2	3
9	3	0
10	3	1
11	3	2
12	3	3
13	4	0
14	4	1

(b)

packet number	frame number	packet # in a frame	received Y/N	received time (ms)
0	1	0	Y	201
1	1	1	Y	450
2	1	2	Y	600
3	1	3	N	*
4	1	4	Y	980
5	2	0	Y	1100
6	2	1	Y	1305
7	2	2	N	*
8	2	3	Y	1790
9	3	0	N	*
10	3	1	Y	2100
11	3	2	N	*
12	3	3	N	*
13	4	0	Y	2710
14	4	1	Y	3000

(c)

Figure 9: (c) is the result of mapping an actual trace (a) to a packet sequence (b).

transmission delays within ns are also modeled by taking an exponential distribution over the mean of transmission delays recorded in the trace over a short period (less than 100 ms) around the transmission time of the current packet in the trace. The receiver might also send a feedback message to the sender depending on a recovery scheme. Feedback messages are assumed to be received reliably in the mapping. In the actual trace taken with RESCU-REC, the feedback message can be lost. Thus, this assumption gives RESCU-REC a slight disadvantage for comparison. Based on the feedback received from the receiver, the encoder may vary its coding pattern if it is required by the recovery scheme.

Figure 9 illustrates the mapping process. Figure 9(a) shows a sample trace ('*' indicates the packet is not received). Those packets received (indicated by Y) show received times. Figure 9 (b) shows a sample of a packetized H.261 sequence. Figure 9 (c) shows the result of mapping (a) to (b). The mapped received times are ignored and replaced with the actual received time of packets observed at the receiver in the simulator.

Through the actual transmission tests, we obtained 66 traces of RESCU-REC for PTDD 3, 68 traces for PTDD 6, and 68 traces for PTDD 9. We mapped the traces of RESCU-REC with PTDD 6 to other schemes. On these transmission traces of the video sequence and the mapped traces, we run the off-line decoder to measure the distortion

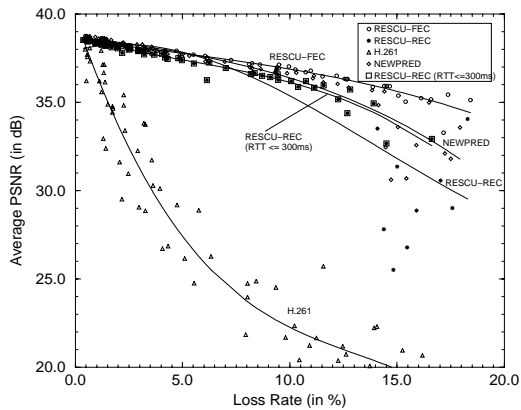


Figure 10: The average PSNR of RESCU-FEC, RESCU-REC and NEWPRED

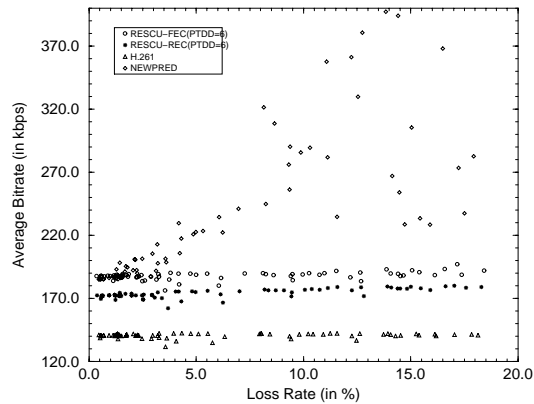


Figure 11: The average bit rate of RESCU-FEC, RESCU-REC and NEWPRED

in the video frame 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 Performance comparison of the RESCU techniques with others

3.2.1 Comparison with NEWPRED

The reference picture selection (RPS) mode adopted in H.263+ allows the encoder to select any previously transmitted frame as a reference frame for motion compensation. It is designed to support a coding technique called NEWPRED [21]. In NEWPRED, using feedback from the receiver, the encoder uses as reference frames for motion 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 eliminated.

Figures 10 and 11 show the final video quality, and transmission bit rate of NEWPRED, RESCU-REC, and RESCU-FEC for actual transmission traces. Each point in the figures indicates the performance of a given technique for one trace. They are plotted over the mean loss rates versus the mean peak signal-to-noise ratio or bit rate. The bit rate takes into account all the bits being transmitted by the sender including retransmitted packets. The lines are the results of cubic-order regression based on resulting experimental data points. The performance of H.261 is shown for a base-line comparison. For RESCU, PTDD 6 is chosen and in RESCU-FEC, six parity packets of a periodic frame are transmitted over one PTDD period.

The PSNR of H.261 rapidly drops as the loss rate increases; for loss rates over 5%, its PSNR reduces below 26 dB. The PSNR of the other techniques remains relatively high even for high loss rates. The quality of RESCU-FEC is the best (suffering about 3dB drop from the best achievable quality), and that of NEWPRED is second (suffering

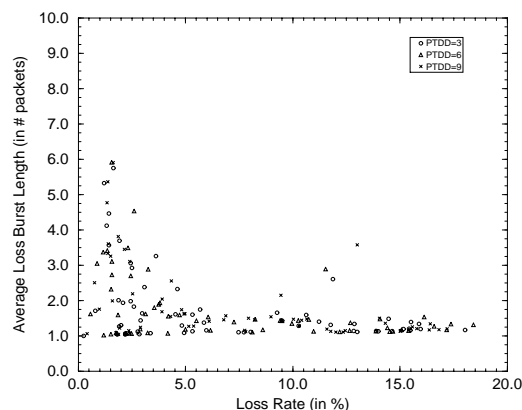


Figure 12: The average loss burst lengths observed in traces obtained through real Internet transmissions.

about 5-6dB drop). The quality degradation of RESCU-REC is due to longer transmission delays caused by heavy congestion. PTDD 6 is not enough to recover lost packet by retransmission under some of the high loss cases. Later, we will explore the impact of transmission delays on the performance of REC.

The good quality of NEWPRED is not without cost. The transmission bit rate of NEWPRED, shown in Figure 11, is highly unpredictable, for some cases, consuming over 300% more bandwidth than H.261. Even for the low loss rate traces (around 5% loss rates), it shows high bit overhead.

There are two reasons for this high bit overhead. First, generally, the loss behavior of the current Internet is only moderately bursty, and most of the times, the loss burst length is less than 2. This is true for all the traces we have obtained as shown in Figure 12 which shows the mean loss burst length of all 200 traces we have obtained. This fact is also confirmed by other studies [4]. When packet losses are well spread out, it is likely that almost all the frames experience some packet losses. Since only those frames that do not contain packet losses can be used as reference frames, NEWPRED might have to reference a frame transmitted many frames before. Since there is little redundancy between a reference frame and the encoded frame if they are far apart, compression efficiency is greatly reduced, causing high bit overhead. Second, when sending an intra-frame, the packets of that intra-frame can also be lost. Thus, when a NACK is received indicating losses in the earlier intra-frame, a new intra-frame has to be sent since no frame can be used as a reference frame. Sending intra-frames more frequently causes high bit overhead.

For a little over 20% bit overhead over that of H.261 (including the retransmission bit, replenishment, and compression overhead), RESCU-REC shows reasonably good video quality under low to medium loss environments. Even under high losses cases, when the delays are below 300ms, Figure 10 shows high video quality for RESCU-REC. RESCU-REC also shows very low bit overhead. This is because bit overhead is incurred only when losses occur and only those packets reported lost are retransmitted.

RESCU-FEC incurs about 35% bit overhead although its quality remains highest even for loss rates higher than 15%. However, unlike NEWPRED's, its bit overhead is almost constant. This is because repair packets are always

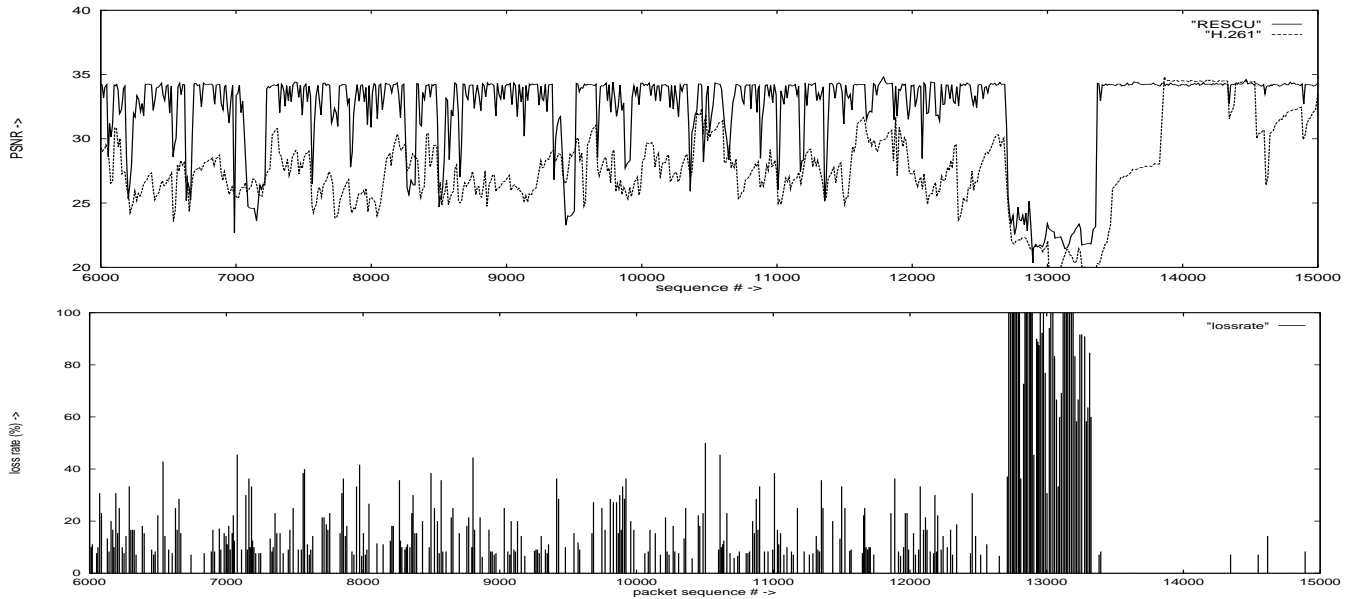


Figure 13: The performance of RESCU and H.261 in a 10% mean packet loss trace

proactively added during transmission times regardless of actual packet losses.

In summary, although NEWPRED shows good video quality by solving the error propagation problem, it tends to incur very high bit overhead when losses are not very bursty. In contrast, the bit overhead of RESCU seems much lower than NEWPRED, and also immune to the degree of loss burstiness.

3.2.2 Comparison with H.261

In Figure 10, we get a glimpse of the impact of error propagation on the video quality of H.261. We now study it in more detail by comparing the performance of H.261 and RESCU on two fronts. First, we will compare the video quality of H.261 and RESCU when both consumes the same amount of bandwidth. Second, we will study the bit overhead of H.261 when it achieves the same video quality as RESCU.

The impact of error propagation on video quality We can reduce the bit rate of a RESCU technique to that of H.261 by either using conditional replenishment [28] or increasing quantization steps. We show only the case with conditional replenishment as the result with the other case is similar. Figure 13 shows the result of a technique combining FEC and RESCU, and H.261 for a single trace with about 10% loss rates. The bit rate of both H.261 and RESCU is set to a similar rate (around 140 kbps). The top picture compares the PSNR of H.261 and RESCU for each frame while the bottom picture indicates loss rates for each frame. They are plotted against packet sequence numbers (i.e., time). H.261 shown in the figure sends one intra-frame at every 95-th frame.

The overall mean PSNR of RESCU is very good although its quality has to be reduced due to conditional replenishment. The PSNR of RESCU drops quite substantially when a frame undergoes high loss. However, in most cases, its quality quickly bounces back when the loss rate of the subsequent frames reduces. This is because repair packets

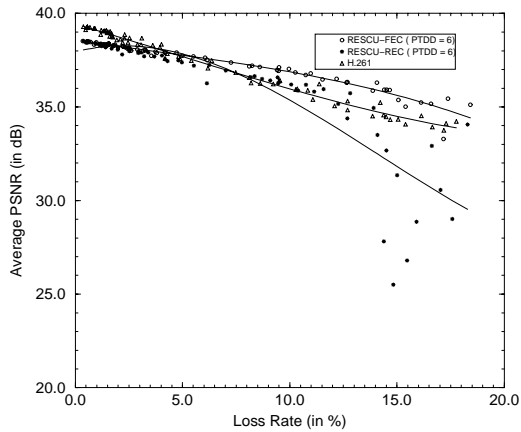


Figure 14: The average PSNR of H.261 with an Intra-frame every 5 frame

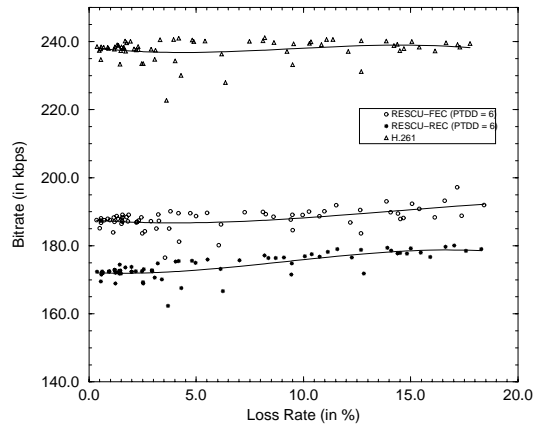


Figure 15: The average bit rate of H.261 with an Intra-frame every 5 frame

sent later restore damaged periodic frames and subsequent frames will be displayed without error propagation. Generally, we can also observe many plateaus for RESCU in Figure 13. This is because losses in non-periodic frames do not affect the quality of next frames as non-periodic frames are not used as reference frames. Thus, when subsequent frames do not have losses, they can be displayed without errors as long as their referenced periodic frames are restored in time.

On the other hand, error propagation takes a heavy toll on the video quality of H.261. Clearly when losses occur the quality of frames containing lost packets degrades. However, even when loss rates get better, its quality does not improve because of error propagation. For instance, around sequence number 13,000, the trace experiences almost 100% losses for a period longer than 10 seconds. Congestion collapse on the network path allows no packets to be delivered to the receiver. After that period when the loss rate becomes low, both technique perform replenishment at the same time (around sequence number 13,300). Both schemes suffer losses in the intra-frames sent at that time, and display damaged frames. H.261 cannot recover from these losses until near sequence number 14,000 when a new intra-frame is received. On the other hand, RESCU recovers the lost packets before the reception of the next periodic frame, and restores the video quality immediately. This result shows that eliminating error propagation tremendously increases the error resilience and video quality of interactive video transmission.

Impact of error propagation on bit overhead We now compare the bandwidth overhead of H.261 over the RESCU techniques in achieving comparable video quality. H.261 can improve its error resilience by transmitting intra-frames more frequently. By increasing the frequency of intra-frame transmission, we can raise the video quality to the level of RESCU's. Figures 14 and 15 show the PSNR and bit overhead of H.261 when an intra-frame is sent at every five frame interval. The final video quality of H.261 seems comparable to that of RESCU-FEC and RESCU-REC.

For loss rates up to 5%, H.261 gives a slightly better video quality than RESCU-FEC. This is because at low

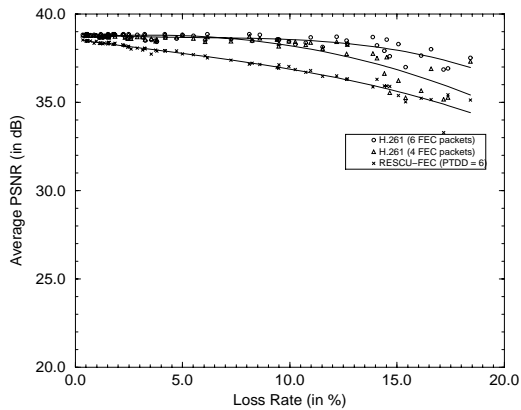


Figure 16: The average PSNR of RESCU-FEC and conventional FEC schemes over various loss rates

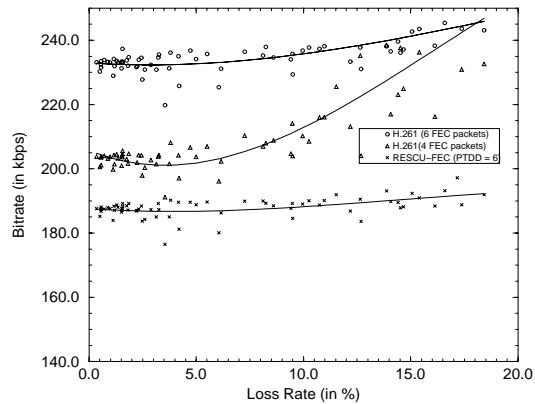


Figure 17: The average bit rate of RESCU-FEC and conventional FEC schemes over various loss rates

loss rates, very few frames are likely to lose packets. Thus the effect of error propagation is less pronounced. Since the intra-frame has much better quality than other frames, transmitting more intra-frames increases video quality. However, at all other loss rates, the video quality of RESCU-FEC is slightly better than H.261. The quality of RESCU-FEC drops under high loss rates. However, the bit overhead of H.261 in achieving the same video quality as RESCU-FEC is very high: over 40% more than that of RESCU-FEC, and over 25% more than that of RESCU-FEC.

3.2.3 Comparison of RESCU-FEC with a conventional FEC scheme

In this section, we study the improvement of RESCU-FEC over a conventional FEC scheme combined with H.261 where parity packets are transmitted immediately after the transmission of their protecting data packets in a back-to-back fashion. Since in H.261, every frame temporally depends on its immediately preceding frame, we add parity packets for every frame to prevent error propagation. In addition, we allow H.261 to perform replenishment when a frame suffers from packet losses that cannot be recovered by parity packets (i.e., when losing more packets than parity packets) to prevent error propagation. We ran experiments to measure the bandwidth required to achieve the video quality of H.261 comparable to that of RESCU-FEC when H.261 is protected by FEC. The results for experiments with 4 and 6 parity packets per frame are shown in Figures 16 and 17.

The results show that under low loss rates, four parity packets are enough to sustain good video quality. However, as loss rate increases, we notice that with four parity packets, the receiver often requests for replenishment. This is why we see high increase in bandwidth usage. As the time of arrival of the replenishment depends on network delays, there can be error propagation between the time of replenishment request and the arrival of intra-frame at the receiver. We see the effect of this error propagation on video quality under high loss rates where the quality gradually degrades. In contrast, the bit overhead is somewhat steady when 6 parity packets per frame are added although it also shows

slight turn upwards near high loss rates. In conclusion, we see that at low loss rates the conventional scheme with 4 parity packets gives good video quality and at moderate to high loss rates, at least 6 parity packets are needed for adequate protection.

In RESCU-FEC, non-periodic frames are not protected. Thus, if they lose any packets, the frames are displayed with distortion. Moreover, periodic frames are recovered relatively later than in the conventional FEC scheme since the parity packets are dispersed over a PTDD period. This is the reason why we see that the conventional FEC scheme gives a PSNR about 1dB or 2 dB higher than RESCU-FEC. In fact, no matter how large PTDD is, RESCU-FEC cannot perform better than the conventional FEC scheme due to distortion in non-periodic frames since only periodic frames are protected in RESCU. However, the better video quality of the conventional FEC scheme comes only at the expense of bit overhead caused by replenishment and parity packets. Our experiment indicates that conventional FEC scheme needs about 15 to 20% more bit overhead to maintain similar or better video quality than RESCU-FEC.

In summary, the experimental result implies that transmitting data packets and their corresponding parity packets all within the same frame interval (as in the conventional FEC schemes) leaves the video stream vulnerable to burst losses and does not effectively use bandwidth. RESCU-FEC allows a more effective use of bandwidth while providing a reasonably good error resilience.

3.3 Evaluation of RESCU over various network environments

Several network parameters are critical to the performance of RESCU: particularly, loss rates, transmission delays, and loss burstiness. In this section, we explore these parametric spaces, and investigate the impact they have on the error resilience, final video quality and bit overhead of interactive video transmission over the Internet.

3.3.1 Impact of loss rates

Figures 18 and 19 show the video quality of RESCU-FEC and RESCU-REC over various loss rates. Both techniques show high error resilience till under 10% loss rate. However, when the loss rate becomes larger than 12%, they suffer degradation in their quality. Especially, the video quality of REC becomes unpredictable. This behavior under heavy packet loss is due to high transmission delays. The effect of transmission delays can be seen from the good performance it shows when we consider only the traces with low round trip delays (less than 300ms). Figure 20 shows the results. The advantage of RESCU-REC over RESCU-FEC becomes even clearer when we look at their bit overhead.

The good performance of FEC comes at the expense of higher bit rates. As shown in Figures 21 and 22, the bit rate of FEC is generally 5 to 8% higher than that of REC. There are two reasons why REC gives lower bit rates. First, retransmission occurs only when packet losses occur while FEC redundant packets are continually sent regardless of packet losses. The good video quality observed in the experiments involving low delays with less bit overhead than RESCU-FEC (in Figures 20, 21 and 22) attests to this. Second, REC is less sensitive to burst losses. Since retransmitted packets take more than one round trip time to arrive; if a loss burst starts at the time of the first loss

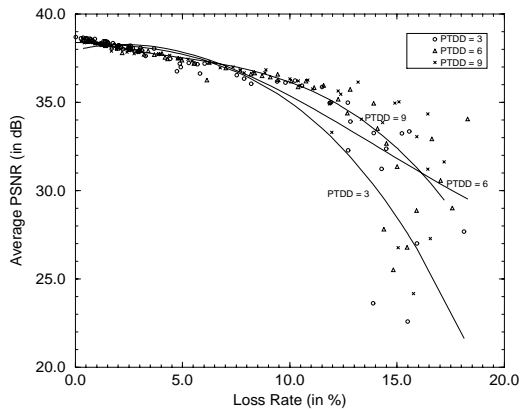


Figure 18: The average PSNR of RESCU-REC with various loss rates

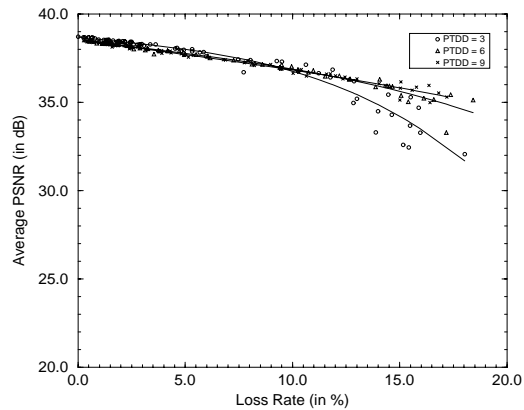


Figure 19: The average PSNR of RESCU-FEC with various loss rates

that triggers retransmission, the same burst is most likely to be ended by the time when retransmitted packets arrive at the receiver. However, this is not the case for FEC. Although the effect of burst losses is much less critical for RESCU-FEC (due to its coarse-grained interleaving) than for conventional FEC techniques, the performance of our FEC is still affected by burst losses. This effect is not clearly visible from Figure 22. Later, we investigate this issue in more detail.

3.3.2 Impact of transmission delays

The disadvantage of RESCU-REC is its sensitivity to transmission delays. When transmission delays are too long, retransmitted packets do not arrive before their deadlines, causing error propagation. RESCU-FEC does not have this problem. because the inter-arrival times of repair packets after a packet loss are independent of transmission delays, and are determined by the sender mostly based on the number of parity packets needed to protect a periodic frame. Figures 23 and 24 clearly show the effect of transmission delays on the performance of RESCU-REC.

In Figure 23, RESCU-REC shows good video quality under low network latency (less than 250ms) even with PTDD 3. However in all other cases, RESCU-REC is highly sensitive to network latency. RESCU-REC shows total ineffectiveness under high RTTs. As PTDD becomes larger, the video quality generally improves, but larger PTDD increases bit overhead due to lower compression efficiency (see Figure 21). In Figure 24, RESCU-FEC is clearly much less sensitive to RTTs. As RTTs increase, its performance degrades a little because high latency usually occurs at the time of congestion (under heavy packet losses). However, its sustained performance is much higher than that of RESCU-REC.

To further study the effect of PTDD and transmission delays on the performance of interactive video transmission, we run a series of simulation experiments under varying mean transmission delays and PTDD periods. Figures 25, 26 and 27 show the effect of network delays on the video quality of RESCU-REC with various PTDD periods. The

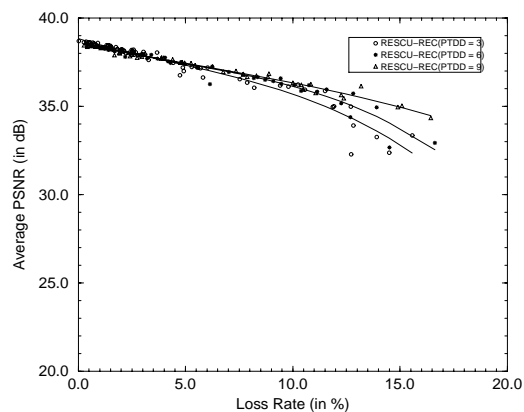


Figure 20: The average PSNR of RESCU-REC for traces with less than 300ms round trip delays

simulation experiments are conducted with 10% mean packet loss rate. We observed that the performance of RESCU-REC is not sensitive to the mean loss burst length. Therefore, we show only the results of experiments done for a mean loss burst length of 1.5

The replenishment count shown in Figure 26 indicates whether a certain PTDD period is long enough to recover losses in periodic frames or not. Recall that RESCU requests for replenishment whenever a periodic frame is not recovered. The higher the count is, the less inadequate a PTDD period is. The higher count of replenishments also results in higher bit overhead as shown in Figure 27. According to Figure 27, about 8 to 9 replenishments per minute does not seem to have much impact on the bit overhead (note that simulation experiments are conducted for a two minute simulated playout period).

The results shown in Figure 26 indicate that for 10% loss rate which is moderately high (a majority of real Internet traces fall less than 10%), PTDD must be at least as large as five to six times one way delay to incur low replenishment counts. For instance, at PTDD 8, the PTDD time period is 800ms since one frame interval is 100ms. Then 150ms mean one way delay seems to give acceptable performance while 200ms seems to be too much a delay for PTDD 8 to handle. Recall that REC requires at the minimum one round trip time for a lost packet to be repaired by retransmission. Because retransmitted packets can also be lost under a high loss environment, it requires several retransmission attempts to recover a lost packet. Therefore, under high loss environments, this is why REC incur longer recovery delays than one round trip delay.

From Figure 26, we can observe the impact of PTDD on compression efficiency. Under 50ms one way delays, all PTDD periods perform fairly well (i.e., little impact on the replenishment count), and hence, the bit overhead of RESCU is mostly due to reduced compression efficiency and retransmission overhead. Since retransmission overhead is fairly similar for all PTDD periods, the increase in bit rates under 50ms delays as PTDD increases is mainly due to lower compression efficiency. As PTDD increases, the increased temporal dependency distance of frames reduces temporal redundancy between a reference frame and its dependent frames. For our test video sequence bandwidth

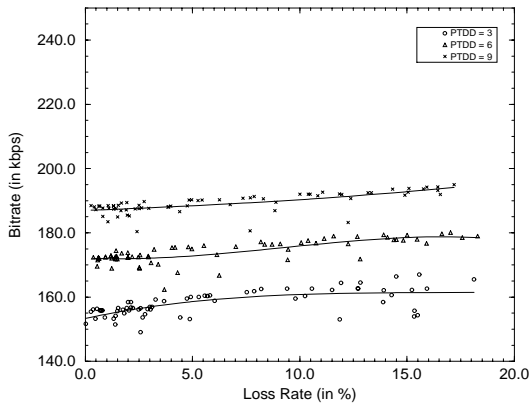


Figure 21: The average bit rate of RESCU-REC with various loss rates

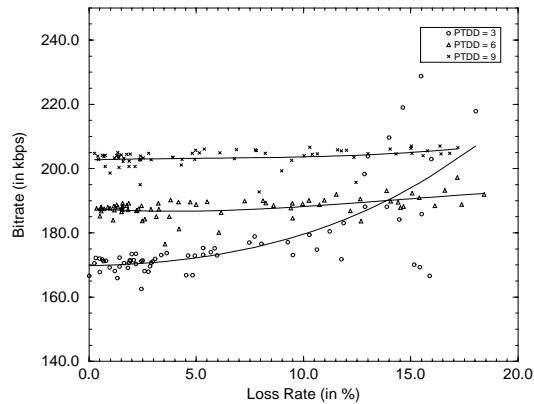


Figure 22: The average bit rate of RESCU-FEC with various loss rates

increases about 5 kbps/PTDD. For other video sequences which contain more rapid motion, compression efficiency gets reduced more noticeably. As RESCU is designed to exploit temporal redundancy in video sequences, it becomes less effective for input videos with high motion. This is one of limitations of RESCU.

3.3.3 Impact of burst losses

To study the impact of burst losses to the image quality of RESCU-FEC, we look at the PSNRs of RESCU-FEC over the mean burst lengths of obtained traces (Figure 28). In the Internet traces we obtained, most of long loss bursts happen under relatively low loss rates. These traces have long loss burst lengths because they include one or two occurrences of very long burst lengths (larger than 100 packets). When loss rates are low, these traces result in long mean burst lengths. We have very few occurrences of long loss burst lengths for high loss rates. Most of those traces have mean loss burst lengths between 1 and 3. and RESCU-FEC with PTDD 6 or 9 shows very good performance for these burst lengths. Note that consecutive FEC packets within a PTDD period are very unlikely to lose in the same loss burst since they are spaced by one frame interval (about 10 packets). Thus, burst losses happen mostly for the original periodic data packets. Since high loss rate traces have mean burst lengths around 1 to 3, 6 FEC packets in PTDD 6 can effectively recover these losses. The short burst lengths we observed from most traces confirms the result of earlier Internet study [4] indicating that long loss bursts are rare in the Internet, while short mean burst lengths less than 3 are common.

To further study the behavior of RESCU-FEC under high burst loss environments, we ran simulation experiments under various burst lengths while fixing the mean loss rate to 10%. In the experiments, we also fix PTDD to 6, and vary the number of parity packets from two to six. Parity packets are evenly spaced over each PTDD period. Figures 29 and 30 illustrate the impact of burst losses in the performance of RESCU-FEC. The loss rate of 10% is applied in all simulation experiments. While the PSNR remains relatively the same over different burst lengths, the replenishment

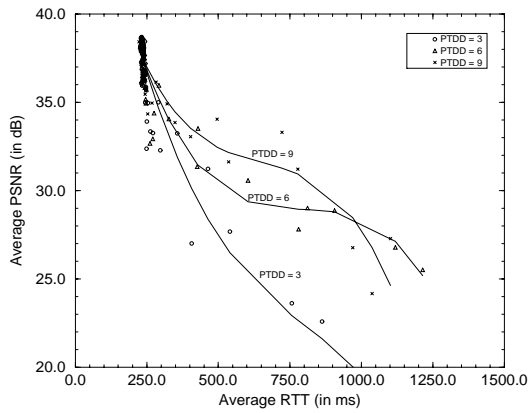


Figure 23: The average PSNR of RESCU-REC with various round trip delays

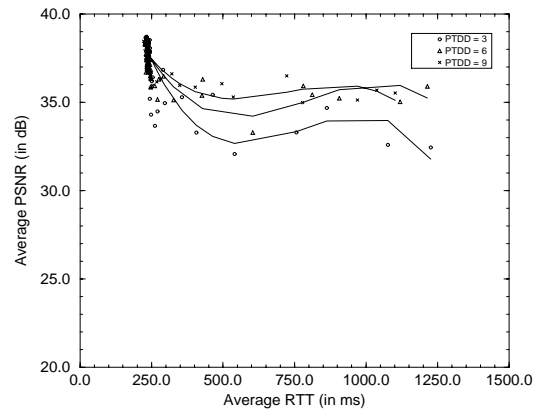


Figure 24: The average PSNR of RESCU-FEC with various round trip delays

count clearly 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. For each additional redundant packet with a PTDD period, the bandwidth increases by a factor of about 1.6% since each frame consists of about 10 packets. Clearly from Figure 30, 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 mean burst length 1, four parity packets are enough while under burst lengths 1.5 and 2, five and six parity packets are needed which is only 6-8% bit overhead. This indicates that RESCU-FEC can perform very well with only a small amount of redundancy even under high burst losses. Since replenishment confines error propagation very well, the video quality of RESCU-FEC does not show many variations under varying burst lengths.

3.4 The summary of experimental results

In comparison with NEWPRED, NEWPRED shows relatively good video quality through its solution for the error propagation problem. However, it tends to incur very high bit overhead when losses are not very bursty. As (a small number of) packet losses spread over many frames, NEWPRED cannot find a reference frame that is not damaged, and incurs high compression overhead. In contrast, the bit overhead of RESCU seems much lower than NEWPRED and immune to the degree of burstiness as it generally shows steady low bit overhead while maintaining comparable video quality to that of NEWPRED.

In comparison with various protection mechanisms applied to H.261 such as frequent intra-frame replenishments, and more forward error correction per frame, the result shows that eliminating error propagation in H.261 tremendously increases error resilience and video quality of interactive video transmission. However, the bit overhead of H.261 in achieving the same video quality as RESCU is very high: over 40% more than that of RESCU-REC, and over

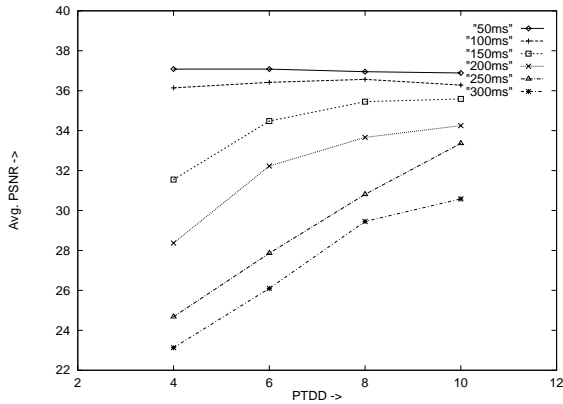


Figure 25: The average PSNR of RESCU-REC with various one-way trip delays (simulation)

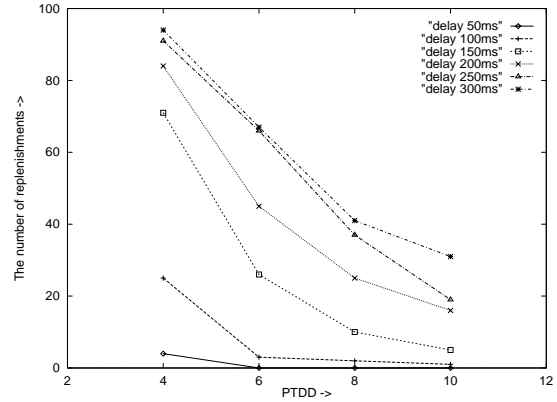


Figure 26: The number of replenishments for RESCU-REC with various one-way trip delays (within a 2 minute simulated playout period)

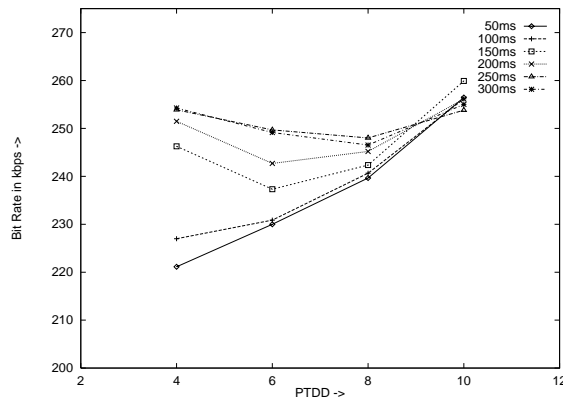


Figure 27: The average bit rate of RESCU-REC with various one-way trip delays (simulation)

25% more than RESCU-FEC. The conventional FEC scheme which transmits data packets and their corresponding parity packets all within the same frame interval leaves the video stream still vulnerable to burst losses and does not effectively use bandwidth. RESCU-FEC allows a more effective use of bandwidth while providing a reasonably good error resilience.

Through the study of RESCU under various network conditions, we learned that RESCU-REC can give very good error resilience with low bit overhead under low transmission delays. However, under long delays, the quality of RESCU-REC suffers a great setback. The advantage of RESCU-FEC 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. RESCU-FEC does not have these problems and is the reason for its good performance. Our experimental results indicates that with a small amount of bit overhead (6-8%) for parity packets, RESCU-FEC can

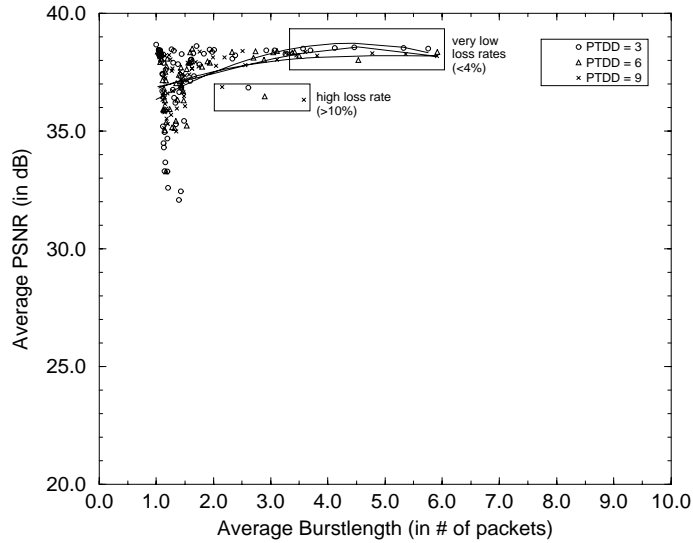


Figure 28: The average PSNR of RESCU-FEC over three different average burst lengths

achieve good error resilience under significantly high loss rate and loss burst length.

4 Related Work

Error concealment is one of widely proposed error control techniques (e.g., [12, 18, 43, 26]). Although error concealment techniques can be combined with error recovery techniques, they are not strictly error recovery techniques [7], 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 intra-coding 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 intra-codes 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 [21]. 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.

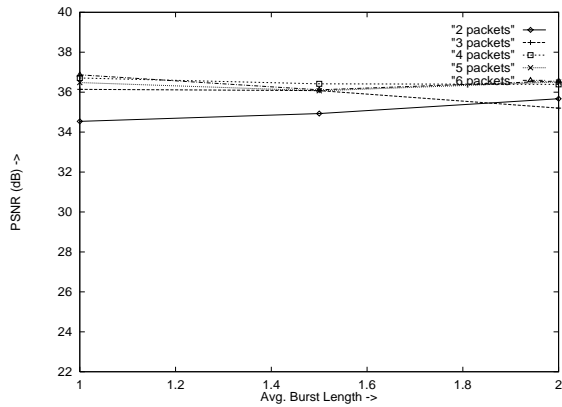


Figure 29: The average PSNR of RESCU-FEC with various numbers of parity packets within PTDD 6 over different loss burst lengths (simulation)

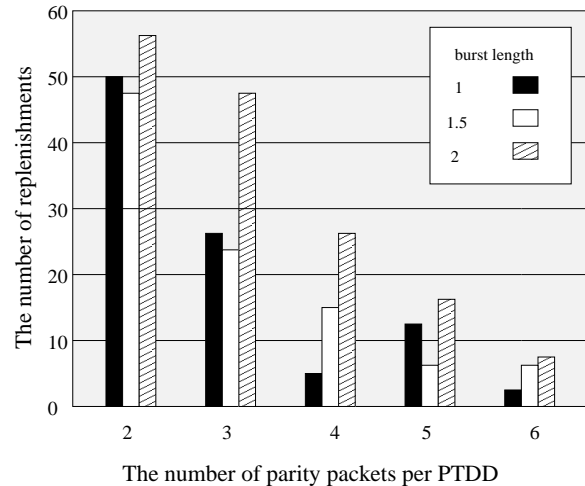


Figure 30: The number of replenishments for RESCU-FEC with various numbers of parity packets within PTDD 6 over different loss burst lengths (simulation)

Retransmission has recently attracted much attention for packet loss recovery in video transmission. Ramamurthy and Raychaudhuri [34] 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 [30] 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.* [23] 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 play-out 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.* [42] 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 techniques requiring frequent

feedback may not be effective.

ER and NEWPRED adopted in H.263+ have a serious limitation. First, they modify their picture coding patterns based on specific information about lost packets (such as which frame lost packets belong to or which macro blocks lost packets contain). Thus, continuous feedback from a receiver is essential in their performance. In some networks such as wireless cable modems and direct satellites, feedback channels are highly limited and contention-based, or even unavailable. Often mobile hosts are too low powered to transmit frequent feedback. It is also clear that these techniques are not for multicast. Furthermore, because their coding patterns depend on knowing exactly which packets are lost by a receiver, they are not useful in a multicast environment involving many receivers. As different receivers may lose different packets, adjusting picture coding or reference frame address based on specifics about lost packets does not scale well in a multicast environment.

In contrast, RESCU does not change its coding pattern based on the specifics on lost packets, but rather based on network characteristics such as loss rates, and burst length. It also relies only on transport-level recovery to recover lost packets which can adapt to given network environments. For instance, when feedback channels are limited, then more RESCU-FEC is used, and under a multicast environment, more scalable packet loss recovery using retransmission, as in [31, 10], can be adopted.

In addition, 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 intra-coding more image blocks, but at the expense of compression efficiency. Several popular video conferencing tools, such as *nv*[11], *vic*[28] and *CU-SeeMe*[9], 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 [5, 3, 32, 33]. There are only a few studies on applying FEC to video transmission. *Priority encoding transmission* (PET) [1, 22, 39], 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* [28] and is reported a good performance [41]. This good performance is partially due to *vic*'s intracoding which limits error propagation caused by loss of lower priority segments.

Bolot and Turletti [6] 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) [20]. 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 [40]. 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. For instance, loss of some header information affects all the data that tagged on the header although they are received correctly. The techniques minimize this effect.

Remarks A drawback of proactive techniques is that it wastes bandwidth under error-free transmission. Furthermore, in wireless mobile networks, the assumption made in [6] 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* [8], and *type-II hybrid ARQ* [24]. 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 [25] 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 [13, 16], 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 [38, 29]. 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 schemes still fall short of handling the retransmission's delays and burst losses without introducing delays in frame playout delays. Delaying frame playout reduces interactivity and introduces annoying jitters.

5 Conclusion and limitations of RESCU

In this paper, we show that retransmission and forward error correction could be made feasible alternatives in error-recovery schemes for interactive video applications without introducing any artificial extension of frame playout times.

The central idea is that correcting errors in a reference frame due to packet losses could be used to prevent error spread. Our performance comparison study based on real Internet traces and simulation experiments indicates that RESCU gives superior error resilience with lower bit overhead for interactive video transmission when compared to other existing recovery techniques. In this paper, we show the result of performance comparison only for variations of H.261 and NEWPRED. In our preliminary work reported in [36, 35], we also reported that RESCU gives much higher performance than Intra-H.261 (adopted in `vic` Mbone tool [28]), and layered coding techniques.

The main implication of our work is that for most of the practical situations on the current Internet, retransmission and FEC techniques can effectively alleviate the problem of error spread with only a small extra bandwidth. Especially, FEC has an advantage in making a minimal use of a feedback channel, and retransmission has an advantage in efficiently using bandwidth. The techniques have the potential to be very useful also in multicast scenarios and wireless and satellite based communications.

The two main limitations of RESCU are noteworthy. One is that it is effective only when neighboring frames in the video sequences contain significant temporal redundancy. Since RESCU buys recovery times by increasing temporal dependency distance, this point is essential. Therefore, under high motion scenes, RESCU would be less effective. Second, it exacts extra computation and buffer space at the receiver side because the receiver has to engage in restoring reference frames before decoding a dependent frame, and displayed reference frames have to be buffered. This problem becomes more severe as the technique employs cascaded recovery involving many buffers. However, as we limit the cascade to one or two buffers, this problem becomes manageable. As only the currently damaged part of reference frames has to be restored, computation overhead is not too much. The buffer requirement of RESCU is also much less than NEWPRED because in NEWPRED, the receiver does not know which frame will be selected to be referenced by the sender, and thus it has to buffer many reference frames.

The work presented here has also a limitation. Since network conditions vary over time, the PTDD period and the number of parity packets have to be adjusted to optimize error resilience and bandwidth usage. Such an adaptable technique is a future goal of our research.

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. End-to-end packet delay and loss behavior in the internet. In *Proceedings of the ACM SIGCOMM*, pages 289–298, San Francisco, CA, September 93.

- [5] 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)*.
- [6] 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.
- [7] G. Carle and E. Biersack. Survey of error recovery techniques for ip-based audio-visual multicast applications. *IEEE Network*, December 1997.
- [8] 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.
- [9] T. Dorcsey. Cu-seeme desktop videoconferencing software. *ConneXions*, 9(4), March 1995.
- [10] S. Floyd, V. Jacobson, S. McCanne, C. G. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. In *Proceedings of the ACM SIGCOMM Conference*, pages 342–356, October 1995.
- [11] R. Fredrick. Network video(nv). Technical report, Xerox Palo Alto Research Center.
- [12] 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.
- [13] J. Hagenauer. Rate-compatible punctured convolutional codes (rpc codes) and their applications. *IEEE Transactions on Communications*, 36(4):389–400, April 1988.
- [14] H. Hessenmuller. Video signal transmission in ATM-based broadband network-treatment of cell losses. In *3rd Int. Workshop on Packet Video*, March 1990.
- [15] ITU-T. Recommendation, h.263+: Video codec for low bit-rate communications, 1998.
- [16] 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.
- [17] S.K. Kaseta, J. Kurose, and D. Towsley. Scalable, reliable multicast using multiple multicast groups. In *Proceedings of ACM SIGMETRICS*, pages 64–74, Seattle WA, 1997.
- [18] 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.
- [19] 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.
- [20] LBC. Document, LBC-95-309 (ITU-T SG 15, WP 15/1), Sub-videos with retransmission and intra-refreshing in mobile/wireless environments, 1995.

- [21] LBC. Document, LBC-96-033 (ITU-T SG 15, WP 15/1), An error-resilience method based on back channel signalling and FEC, 1996.
- [22] C. Leicher. Hierarchical encoding of MPEG sequences using priority encoding transmission (pet). Technical Report 94-058, ICSI, November 1994.
- [23] 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.
- [24] S. Lin and D. Costello. *Error control coding: fundamentals and applications*. Prentice-Hall Inc., Englewood Cliffs, NJ, 1983.
- [25] 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.
- [26] 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.
- [27] J. McAuley. Reliable broadband communications using a burst erasure correcting code. In *Proceedings of the ACM SIGCOMM*, Philadelphia, PA, September 1990.
- [28] 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.
- [29] Jorg. Nonnenmacher, Ernst Biersack, and Don Towsley. Parity-based loss recovery for reliable multicast transmission. *ACM Transactions on Networking (to appear)*.
- [30] 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.
- [31] S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya. Reliable multicast transport protocol (RMTP). In *Proceedings of the IEEE INFOCOM*, San Francisco, CA, March 1996.
- [32] 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.
- [33] 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.
- [34] 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.
- [35] 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.

- [36] 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.
- [37] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *Computer Communication Review*, 27(2):24–36, Apr 1997a.
- [38] Dan Rubenstein, Jim Kurose, and Don Towsley. Real-time reliable multicast using proactive forward error correction. *NOSSDAV 98 (to appear)*.
- [39] R. Storn. Modeling and optimization of pet-redundancy assignment for MPEG sequences. Technical Report TR-95-018, ICSI, Berkeley, CA, May 1995.
- [40] R. Talluri. Error-resilient video coding in ISO MPEG-4 standard. *IEEE Communication Mag.*, 36(6):112–119, June 1998.
- [41] Priority Encoding Transmission. Web page: <http://www.icsi.berkeley.edu/pet/icsi-pet.html>.
- [42] 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.
- [43] 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.
- [44] M. Zorzi and R. Rao. Capture and retransmission control in mobile radio. *IEEE Journal on Selected Areas in Communications*, 12(8):1289–1298, 1994.