

# Error Recovery for Interactive Video Transmission over the Internet

Injong Rhee and Srinath R. Joshi

**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 to 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.

**Index Terms**—Computer network protocols, error propagation, forward error correction, interactive video transmission, packet loss recovery, retransmission.

## I. INTRODUCTION

**T**RANSMITTING 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 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 the 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., [35], [31], [24], [45], [1], [41], [23], [26]). 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 [35], [31], [24], [45] 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 [23], [1], [20], [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.

In this paper, we propose an entirely complementary approach to the above-mentioned work 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. Clearly, 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

Manuscript received May 10, 1999; revised November 2, 1999. This work was supported in part by NSF CAREER ANI-9875651.

The authors are with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695-7534 USA (e-mail: {rhee, srjoshi}@csc.ncsu.edu).

Publisher Item Identifier S 0733-8716(00)04334-1.

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—retransmission and FEC—can also be used for interactive video transmission since RESCU can effectively mask out the delays involved in recovering lost packets. When combined with retransmission, RESCU use retransmitted packets arriving after the display time of their frames to stop error propagation, thus enhancing the end video quality. When FEC is used, 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. These schemes are clearly different from the conventional ones in that they do not introduce any playout delays.

The purpose of this paper is to show potential benefits of RESCU in enhancing the error resilience of interactive video transmission so that we can stimulate more research into developing better RESCU schemes. Using both simulation and actual Internet experiments, we compare the performance of RESCU, H.261, and H.263, and other error recovery techniques such as NEWPRED [22], H.261, and H.263. We also study the strengths and weaknesses of RESCU over various network situations. In particular, we investigate network environments where a particular transport recovery technique combined with RESCU can or cannot be effective.

This paper is organized as follows. In Section II, we first describe related work. In Section III, we describe the RESCU scheme and two recovery techniques each combined with retransmission and FEC, respectively. In Section IV, we present our simulation and experimental results. Finally, in Section V, we summarize our work in this paper and discuss the impact and limitations of our work.

## II. RELATED WORK

Error concealment is one of the widely proposed error control techniques (e.g., [14], [19], [27], [46]). 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 focus only on recovery techniques for video transmission.

### A. Feedback-Based Recovery

Recently, H.263+ incorporated two feedback-based techniques: *error tracking* and *reference picture selection*. Error tracking (ET) utilizes the intra-coding of blocks to stop error propagation, but limits its use to severely impaired image regions only [40]. ET 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 [22]. 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. Wada [44] also proposed a scheme similar to ET. As in ET, the encoder computes the extent of error propagation by lost packets. However, unlike ET where affected macroblocks are intra-coded, the Wada's scheme simply excludes them from next motion prediction. Thus, error propagation can be stopped when new frames are coded using only those regions that are not affected by lost packets. In contrast, cascaded RESCU does not involve the encoder, and error tracking is performed only at the decoder rather than at the encoder. Thus, it does not require any change in the encoded bit stream.

Retransmission has recently attracted much attention for packet loss recovery in video transmission. Ramamurthy and Raychaudhuri [35] 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 [31] 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.* [24] 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 noninteractive 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.* [45] 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.

ET 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 macroblocks 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. 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 [32] and [12], can be adopted.

In addition, all of existing retransmission techniques (except RESCU) use frame playout delays to compensate for retransmission delays.

### B. 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* [13], *vic* [29], and *CU-SeeMe* [11], 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], [33], [34]. There are only a few studies on applying FEC to video transmission. *Priority encoding transmission* (PET) [1], [23], [41], 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* [29], and is reported to give a good performance [43]. This good performance is partially due to *vic*'s intra-coding 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) [21]. 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 [42]. These include resynchronizations strategies, data partitioning, reversible VLC's, 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. These 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 because 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.

### C. Hybrid Recovery

Hybrid techniques combine ARQ (retransmission) and FEC for better error resilience. There are two types of hybrid techniques: *type-I hybrid ARQ* [10] and *type-II hybrid ARQ* [25]. 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 [26] 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 [15], [17], 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 [39], [30]. 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, a hybrid ARQ scheme still falls short of handling the retransmission delays and burst losses without introducing delays in frame playout delays.

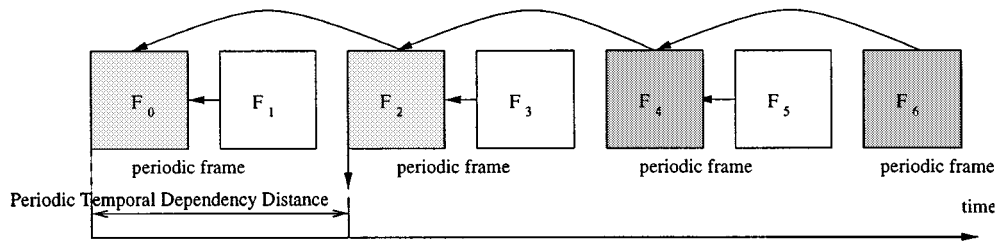


Fig. 1. RESCU with PTDD 2.

### III. ERROR RECOVERY TECHNIQUES

#### A. Recovery from Error Spread Using Continuous Updates (RESCU)

We base our discussion on H.261, an International Telecommunication Union (ITU) video standard (although RESCU can be applicable to any video codec that employs motion compensation such as H.263 and MPEG). 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 this codec, because of the interdependency among frames, if we 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.

1) *RESCU*: Using RESCU, we can increase the error resilience of H.261. In RESCU, packets arriving after their display times are not discarded but instead used to reduce error propagation. We define the *deadline of a packet* 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) (see Fig. 1). Every nonperiodic frame (frames between two consecutive periodic frames) depends only on its immediately preceding periodic frame. Thus, the TDD of the nonperiodic 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 nonperiodic frames do not propagate at all because all

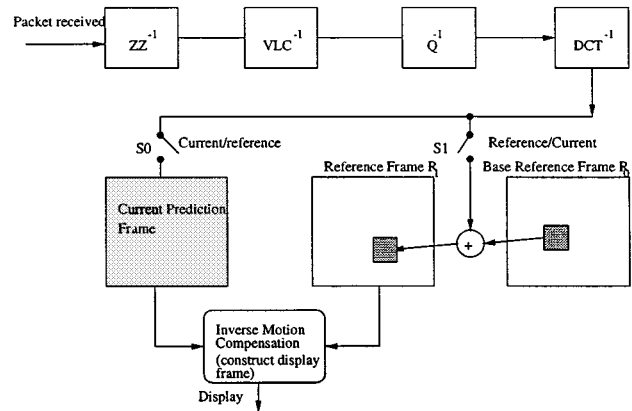


Fig. 2. H.261 decoder modified to handle the recovery of reference frames.

nonperiodic 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.

Fig. 2 shows an 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  $CP$  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 is 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.

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



Fig. 3. Error propagation.



Fig. 4. RESCU stops error propagation.

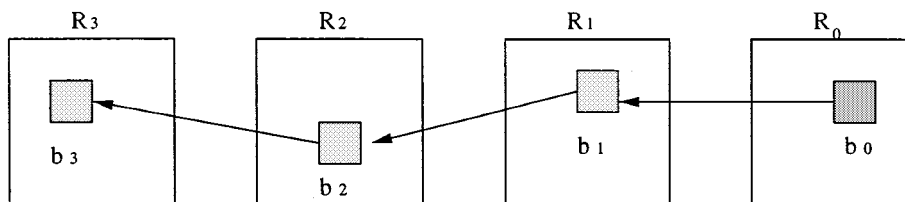


Fig. 5. Cascaded RESCU recovery.

back when the packets for the second frame are recovered before the decoding of the third frame.

Implementing RESCU in other standard codecs is straightforward. For instance, 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* [16] 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.

2) *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 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.

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, larger delays in receiving packets can be accommodated. In cascaded RESCU, every macroblock in a buffered periodic frame has a data structure called *dependent block list* (DBL). The DBL contains all the macroblocks of its immediately dependent periodic frame

that use at least part of that macroblock for motion prediction. This information is recorded when a periodic frame is decoded. Suppose that a periodic frame  $R_i$  is decoded using a frame  $R_{i-1}$  as a reference frame. When a macroblock  $M_i$  in  $R_i$  is decoded, the decoder can find all the macroblocks of  $R_{i-1}$  that contain at least part of the motion-prediction matching area of  $M_i$  (determined by its motion vector) and records the i.d. of  $M_i$  in the DBL's of those macroblocks. When a lost packet in  $R_{i-1}$  is recovered, the decoder finds the macroblocks stored in the DBL of each macroblock contained in the recovered packets, and recomputes motion compensation for those blocks. Fig. 5 illustrates this scheme when PTDD is one (i.e., all frames are periodic). The shaded squares represent image blocks, and the arrows represent temporal dependency among blocks. For example, block  $b_3$  uses as a reference for motion prediction at least part of block  $b_2$  (i.e.,  $b_2$ 's DBL contains  $b_3$ ), and so on. Suppose that the current frame's sequence number is  $f$ , and  $R_0$  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 ( $f - 4 + i$ ). Suppose again that  $b_0$  is lost. Then all the blocks  $b_1$ ,  $b_2$ , and  $b_3$  contain errors. When  $b_0$  is finally restored,  $b_3$  can be restored by restoring  $b_1$  using  $b_0$ , then  $b_2$  using the restored  $b_1$ , and then  $b_3$  using the restored  $b_2$ .

Cascaded recovery trades additional buffers and computation for increased recovery times. Note that cascaded RESCU recovery allows packet deadlines to be extended *at the receiving times*, but not at the encoding times. This scheme is somewhat

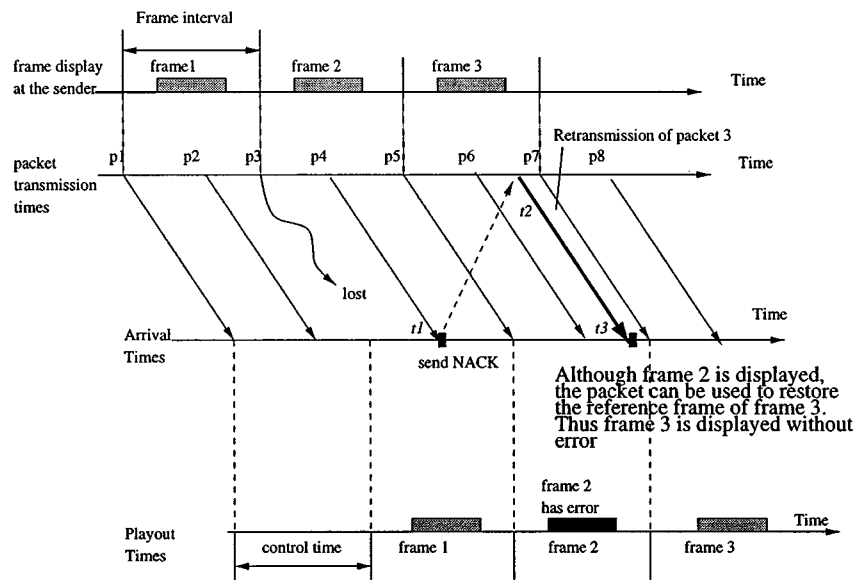


Fig. 6. The recovery of frames using retransmission.

similar to error tracking [40] or error concealment as done in [44]. However, they differ from cascaded RESCU since they require the encoder to be involved in recovery.

3) *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 detects 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.

### B. 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 is effected 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. Fig. 6 illustrates error recovery using RESCU and retransmission in a video stream containing two 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 [35], [31], [24], [45] in that it does not introduce any delay in frame playout times.

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. Every time 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.

### C. 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

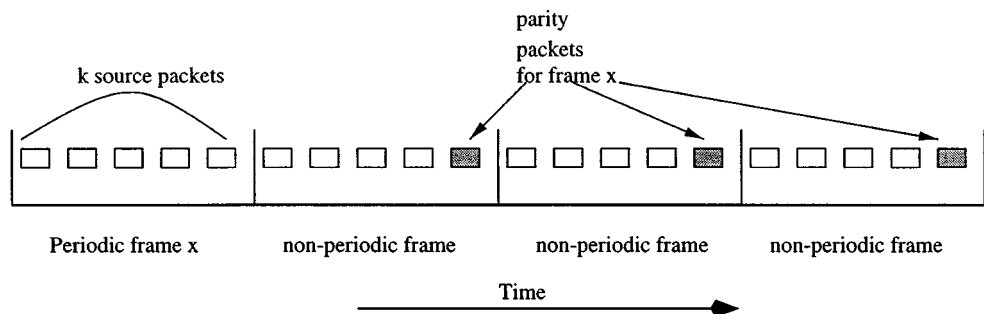


Fig. 7. Interleaving of packets in RESCU of PTDD 4 with (5, 8) LBC encoding.

transmission delay requires larger PTDD. As PTDD increases, compression efficiency gets lower because two consecutive periodic frames may not have much temporal redundancy, and the TDD of nonperiodic frames also increase. In addition, packet losses in periodic frames can be restored only after one round trip time. Thus, during the time, nonperiodic frames can have error 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 [28], 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 an *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], [18], [38], [30]. For instance, the throughput of the software coder by Rizzo [38] 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. Fig. 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 nonperiodic frames will be displayed with errors. However, as ensuing parity packets can be received to recover the periodic frame, this will cause the remaining nonperiodic 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.

In general, the bit overhead of FEC schemes is sensitive to packet size. This is because each packet contains a set of protocol headers which incur almost a constant bit overhead no matter how small a packet is. Over the Internet, the header size could be at least 40 octets (IPv4) or 60 octets (IPv6) when RTP/UDP/IP is used for video transmission. Thus, it is more advantageous to use larger packets to reduce the bit overhead due to protocol headers. However, larger packets increase the granularity of forward error correction, potentially also increasing bit overhead. Recognizing this header overhead, researchers are proposing new Internet standards for header compression [8], [9] which reduce the header size up to 2 octets. It is expected that the sensitivity of an FEC scheme to packet size will not be critical in the future.

#### IV. EXPERIMENTAL RESULT

The objective of our experimental work is to study the potential effectiveness of the two transport recovery techniques—retransmission and FEC—when integrated with RESCU for enhancing the error resilience of 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 interest is the examination of its behavior under various loss rates, transmission delays, and loss burstiness.

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

TABLE I  
TEST VIDEO SEQUENCES

Video Sequence	Class	Description
<i>container</i>	A	low spatial detail and low amount of motion
<i>news</i>	B	medium spatial detail and medium amount of motion
<i>children</i>	E	hybrid natural and synthetic movements

### A. Testing Methodology

We modified H.261 to incorporate RESCU, and use that implementation for our testing throughout this paper. Note that RESCU can be incorporated into any motion compensation-based codecs such as H.263. We expect that the performance of RESCU combined with H.263 would be better because of its better motion estimation than H.261.

Full-search motion estimation and a default quantization step size 8 are used for all experiments. We use three MPEG-4 test sequences as test video sequences for our experiments. The three video sequences are described in Table I. For every experiment, the frame rate is set to 10 frames per second. The image size of CIF ( $352 \times 288$  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 macroblocks.

Cascaded RESCU recovery with one additional buffer is adopted for RESCU-REC. 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 (nonperiodic) frames.

We generate a packetized sequence corresponding to 190 frames. This sequence is replayed several times for about 2 min. 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 all the figures presented in the next section are taken from the results of Internet trace-driven experiments.

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 two-state Markov model is commonly used to model the loss behavior of the Internet [4], [39].

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  $\tau$ . Network conditions are characterized by the packet transmission rate  $\lambda$ , 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.

2) *Internet-Transmission Test*: We conducted actual video transmission tests over the Internet from Korea to the U.S. These testing sites are chosen because transmission delays between two sites vary over a wide range of delays between 100 ms and 1 s. The transmission tests were conducted every 45 min between October 10 and October 13, 1998, 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 95th frame in addition to intra-frames sent for replenishment. We use only the ‘‘container’’ sequence for the actual transmission tests. The tests with other video sequences are conducted using the traces obtained from the actual transmission tests.

We first obtain a packetized sequence of a 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-min 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 apply a technique called *mapping* which works as follows. An actual transmission trace is fed into a trace-driven simulator 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 loss 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. Note  $p$  or  $q$  can also be repair packets (i.e., retransmission or FEC packets). After  $p$  is mapped, then its next packet in the trace is mapped to the next packet arriving from the encoder. This process is continued until the trace runs out of packets to map. The transmission delays within *ns* are also varied according to the exponential distribution over the mean of the network delays of packets transmitted around the same transmission time of the current packet being mapped in the trace. Feedback messages are assumed to be received reliably in the mapping. Based on

TABLE II  
STATISTICS FOR TEN DIFFERENT TRANSMISSION TRACES

No.	Avg. Packet Loss Rate (%)	Avg. RTT (ms)
1	0.5	245
2	5.03	268
3	8.20	255
4	8.41	261
5	8.91	355
6	12.23	313
7	13.84	276
8	14.04	800
9	15.35	572
10	17.45	976

the feedback received from the receiver, the encoder may vary its coding patterns if it is required by the recovery scheme.

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. On these transmission traces and the mapped traces, we run the off-line decoder to measure the distortion 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.

### B. Comparison of RESCU to Other Recovery Schemes

For clarity of presentation, we categorize traces into groups with a similar loss rate and round trip time (RTT). We selected ten groups, each of which shows a different loss rate and RTT, and consists of six traces of a similar loss rate and RTT (in most cases, the traces in the same group are taken at the same 45 min period). Table II summarizes the average traffic characteristics of the ten groups.

Five different schemes are being compared: RESCU-REC, RESCU-FEC, NEWPRED, H.261, and H.263. We run the two RESCU schemes over the 60 traces summarized in Table II. At each run, we vary PTDD and the number of FEC packets (for RESCU-FEC only) used to protect a periodic frame to find the best performance of RESCU, and obtain the average PSNR and bit rate. For each of the 60 traces, we find the run with the best ratio of the average PSNR over the average bit rate, and record its average PSNR and bit rate. We then take the average of the recorded PSNR and bit rate for each group. The results of RESCU-REC and RESCU-FEC on the three test video sequences are plotted in Figs. 8–13.

### C. Comparison to H.261 and H.263

We first provide a strawman's comparison of RESCU, H.261, and H.263 to determine the effect of error propagation. There are many techniques that can improve the performance of H.261 and H.263, such as error concealment [14], [19], [46], [27] or error tracking [40]. However, the comparison discussed in this subsection is to find the baseline performance when no recovery mechanism (other than *I*-frame refresh) is provided for H.261 and H.263. This will give an insight on how much improvement RESCU can achieve from the baseline.

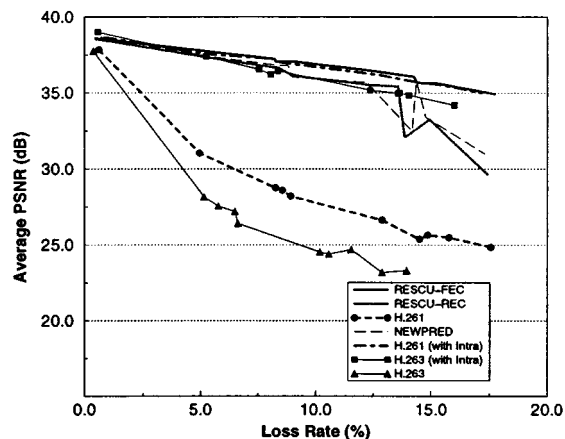


Fig. 8. The average PSNR for *container*.

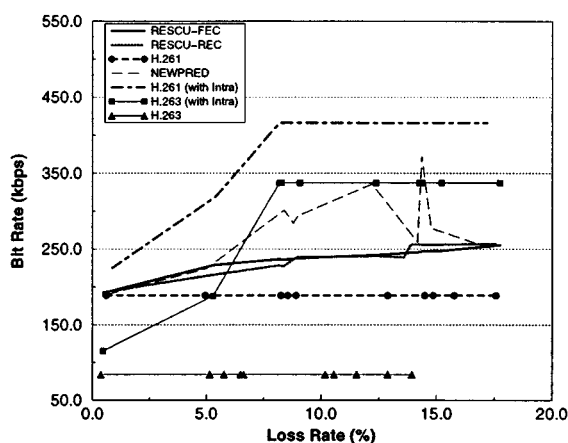


Fig. 9. The average bit rate for *container*.

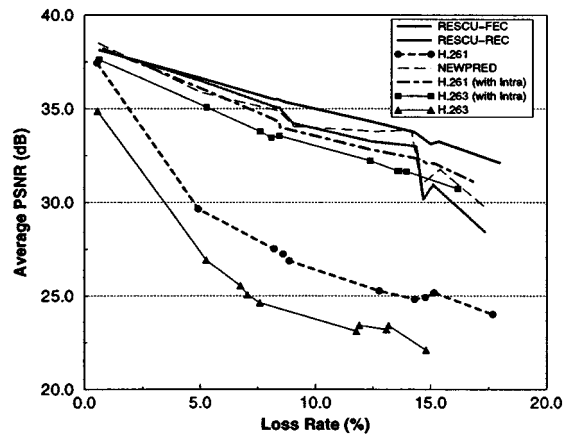
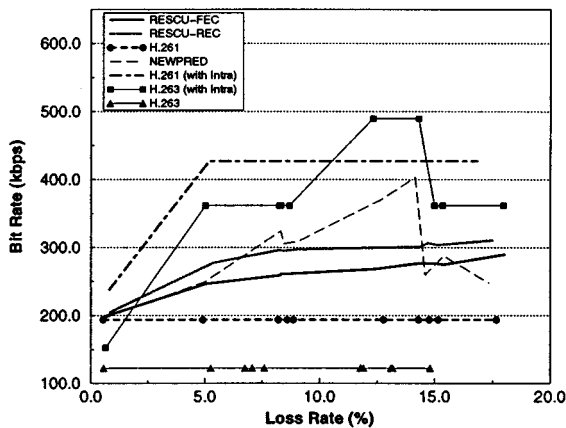
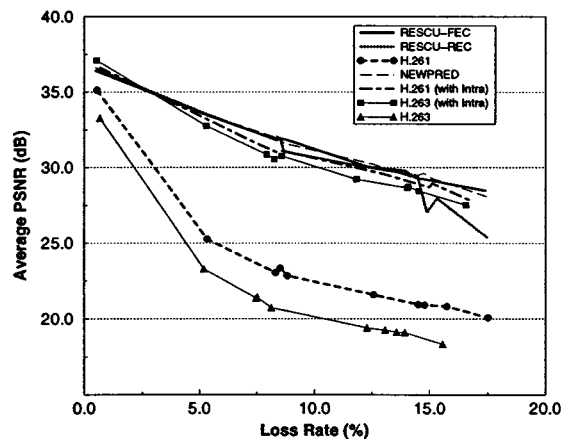
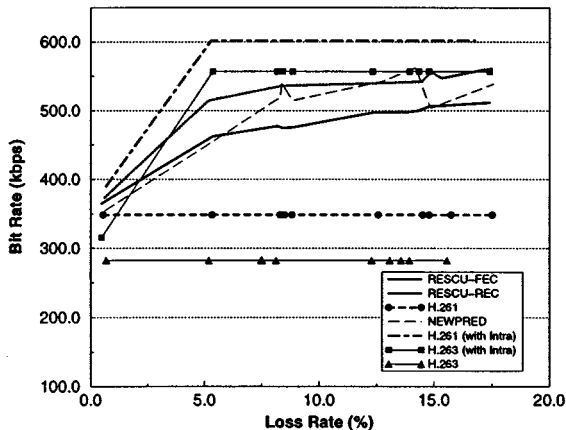


Fig. 10. The average PSNR for *news*.

Tables III–V show the percentage increase of average bit rate per frame for each video sequence as PTDD increases when RESCU is combined with H.261 (RESCU+H.261) and with H.263 (RESCU+H.263). When PTDD is set to one, the average bit rates of RESCU+H.261 and RESCU+H.263 are the same as the average bit rates of H.261 and H.263. The tables also show the average bit rate increase of periodic frames in RESCU+H.261 (denoted *Periodic frame* in the tables), and the average bit rate when we intra-code every frame (denoted *Intra frame* in the tables). The results indicate that for each increment

Fig. 11. The average bit rate for *news*.Fig. 12. The average PSNR for *children*.Fig. 13. The average bit rate for *children*.

of PTDD, the compression efficiency of RESCU drops about 3% to 5% in *container*, and about 2% to 12% in *news* and *children*. From the tables, we can see that as more motion becomes present, the bit overhead of RESCU increases. We will see in Figs. 8–13 that RESCU achieves the best quality and bit rate tradeoffs compared to all the techniques we tested. The compression efficiency sacrifice of H.263 for increased PTDD is larger than of H.261, although H.263 gives much better compression efficiency. This is because H.263 relies more on motion compensation than H.261 (using a technique such as half-pixel

motion estimation). As PTDD increases, the temporal redundancy between two adjacent periodic frames and between a periodic frame and its dependent nonperiodic frames get reduced. Thus, motion compensation becomes less effective. However, as we see in Figs. 8–13, this bit overhead is much less than what H.261 or H.263 has to pay to be as error resilient as RESCU. In addition, since the bit overhead of periodic frames is much less than I-frames, there exists much advantage in exploiting temporal redundancy between two periodic frames than coding them as I-frames.

In all video sequences tested, the average PSNR's of H.261 and H.263 fall rapidly (to as much as 12 dB lower) as loss rates increase because of error propagation. Even at low loss rates, quality degradation is quite significant in all video sequences. The PSNR degradation of H.263 is more severe than that of H.261. This is because H.263 relies heavily on motion estimation to get the bit rate as low as 50% of what H.261 gives. Thus, the impact of error propagation is larger.

We observe that RESCU shows the best ratio of PSNR over the bit rate. The quality degradation of RESCU is much slower as loss rates increase than the other recovery schemes. For instance, in *container*, under low loss rates below 5%, with only less than 12% bit overhead, it can achieve more than 5 dB higher PSNR than H.261. Even under very high loss rates, RESCU can show about 10 dB higher PSNR than H.261 with only 25% bit overhead in *container*. We expect that the bit overhead of RESCU will be further reduced when it is combined with H.263 instead of H.261. The effectiveness of RESCU reduces as there is more motion in the input video. While it can sustain over 30 dB under low and medium motion sequences even under the highest loss rate, its quality drops below 30 dB under high loss rates in high motion. While still achieving much higher PSNR than H.261 under high loss rates, the bit overhead of RESCU has increased to 60% in *children*. This is because motion prediction using only periodic frames as references becomes less effective as rapid motion becomes more present.

From the figures, we also notice the sensitivity of RESCU-REC to network delays. Under less than 300 ms round trip time (RTT), RESCU-REC gives slightly less bit overhead than RESCU-FEC, while both give similar PSNR. However, under high delays traces (8, 9, 10), its PSNR degrades substantially. We will study further the performance impact of network delays to RESCU in Section IV-F2.

We also estimate the bit overhead of H.261 and H.263 to achieve similar video quality as RESCU over the tested traces. To show this, we run various instances of H.261 and H.263, each with a different intra-frame refresh rate. For each trace, we find all the instances that give a similar average PSNR value as RESCU under that same trace. Among them, we select the instance that gives the highest ratio of the average PSNR over the average bit rate, and record its PSNR and bit rate. At the end, we average PSNR and bit rate values for each group, and plot them along with the values of RESCU. The results are labeled as *H.261(with intra)* and *H.263(with intra)* in the figures. The results indicate the following. To maintain similar video quality as RESCU, H.261 needs about 70% more bit overhead in *container*, 55% in *news*, and 15% in *children*, and H.263 needs about 40% in *container*, about 15% to 35% in *news*, and 5% to

TABLE III  
BIT RATES PER FRAME OF VARIOUS IMPLEMENTATIONS OF RESCU AS PTDD INCREASES USING *CONTAINER*

PTDD	1 (bits/f)	2 (%)	3 (%)	4 (%)	5 (%)	6 (%)	7 (%)	8 (%)	9 (%)	10 (%)
RESCU+H.261	18880	6	10	14	18	20	23	26	29	32
RESCU+H.263	9037	6	26	35	48	53	66	69	76	87
Periodic frame	18880	12	28	32	49	50	64	67	79	81
Intra frame	88360	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE IV  
BIT RATES PER FRAME OF VARIOUS IMPLEMENTATIONS OF RESCU AS PTDD INCREASES USING *NEWS*

PTDD	1 (bits/f)	2 (%)	3 (%)	4 (%)	5 (%)	6 (%)	7 (%)	8 (%)	9 (%)	10 (%)
RESCU+H.261	18880	6	10	14	18	20	23	26	29	32
RESCU+H.263	9037	6	26	35	48	53	66	69	76	87
Periodic frame	18880	12	28	32	49	50	64	67	79	81
Intra frame	88360	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE V  
BIT RATES PER FRAME OF VARIOUS IMPLEMENTATIONS OF RESCU AS PTDD INCREASES USING *CHILDREN*

PTDD	1 (bits/f)	2 (%)	3 (%)	4 (%)	5 (%)	6 (%)	7 (%)	8 (%)	9 (%)	10 (%)
RESCU+H.261	34856	12	20	26	30	38	37	42	50	52
RESCU+H.263	25058	19	31	42	47	58	62	67	78	78
Periodic frame	34856	25	41	51	58	70	75	81	90	93
Intra frame	111600	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

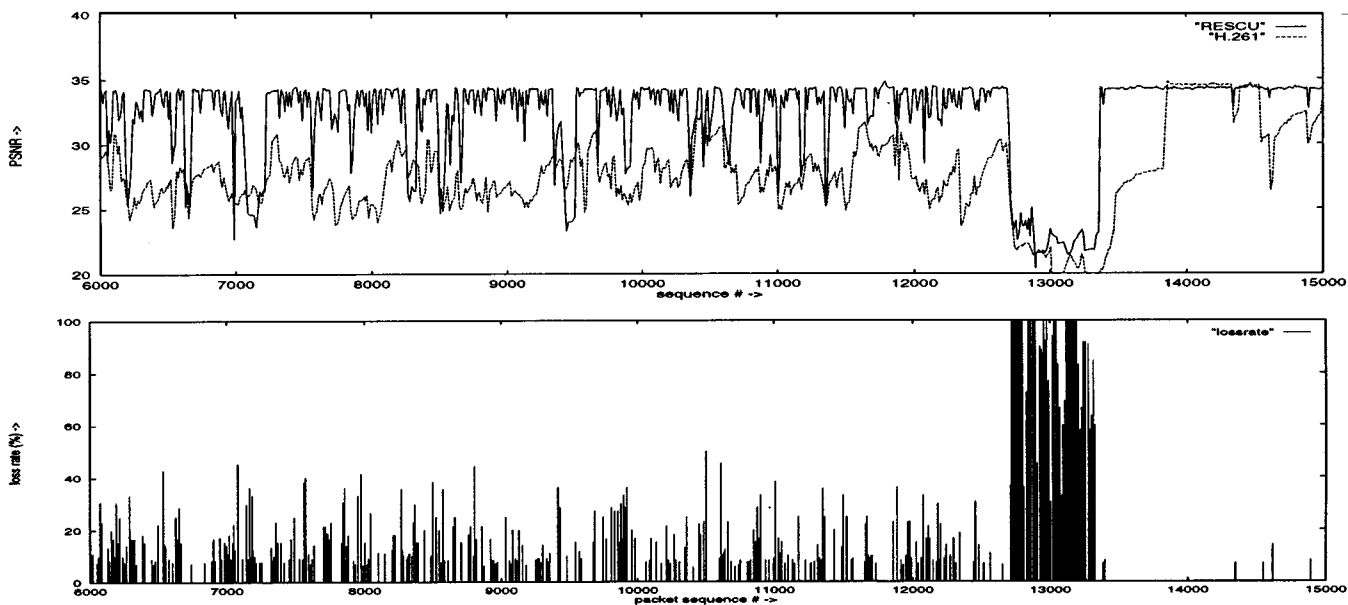


Fig. 14. The performance of RESCU and H.261 in a 10% mean packet loss trace.

20% in *children*. This high bit overhead indicates that RESCU achieves very good tradeoff of video quality over bit rates.

We can also study the video quality of H.261 and RESCU when they both use the same bit rate. The bit rate of a RESCU technique can be matched to that of H.261 by either using conditional replenishment [29] or increasing quantization steps. We show only the case with conditional replenishment, as the result with the other case is similar. Fig. 14 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 180 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).

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 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 Fig. 14. This is because losses in nonperiodic frames do not affect the quality of next frames as nonperiodic 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 s. 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 techniques 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.

#### D. Comparison to NEWPRED

The reference picture selection (RPS) mode adopted in H.263+ allows the encoder to select any previously transmitted frame as reference frame for motion compensation. It is designed to support a coding technique called NEWPRED [22]. 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.

We tested the performance of NEWPRED in NACK mode over the ten trace groups shown in Table II. Figs. 8–13 show the performance comparison of RESCU and NEWPRED. The average PSNR of NEWPRED seems similar to that of RESCU-REC. However, its bit rate overhead is much more than RESCU, especially under high loss rates. This can be explained as follows. Under high loss rates, only a small number of frames are received without loss. 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 the reference frame and the encoded frame, compression efficiency is greatly reduced. Further, when NEWPRED sends 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

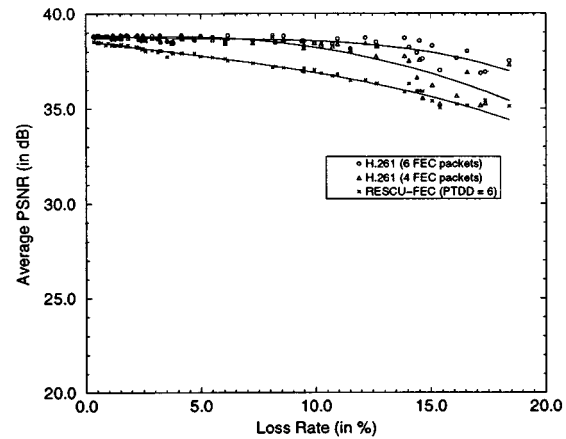


Fig. 15. The average PSNR of RESCU-FEC and conventional FEC schemes over various loss rates (*container*).

has to be sent since no frame can be used as a reference frame. Sending intra-frames more frequently causes high bit overhead.

The performance of NEWPRED is also sensitive to RTT because error propagation will continue until feedback is received indicating whether a certain frame is received (in *ACK mode*) or not received (in *NACK mode*). This is the reason why it shows similar PSNR as RESCU-REC.

RESCU is more effective under low and medium motion sequences such as *container* and *news*. The bit rate difference between NEWPRED and RESCU is much smaller in the *children* sequence. NEWPRED incurs about 30–40% more overhead than RESCU under *news* and *container*, while only about 10% in *children*.

#### E. Comparison of RESCU-FEC to a Conventional FEC Scheme

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 Figs. 15 and 16.

The results show that under low loss rates, four parity packets are enough for H.261 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 a high increase in bandwidth usage. Before a replenishment request is received and serviced by the encoder, error propagation occurs. 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

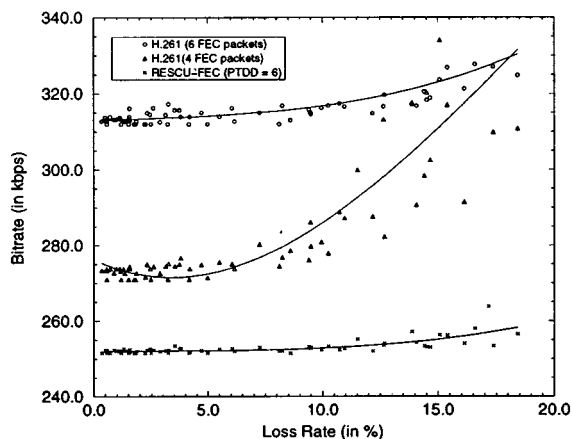


Fig. 16. The average bit rate of RESCU-FEC and conventional FEC schemes over various loss rates (container).

6 parity packets per frame are added to H.261, although it also shows slight turn upwards near high loss rates.

In RESCU-FEC, nonperiodic frames are not protected. Thus, if they lose any packets, they are displayed with distortion. Moreover, periodic frames are recovered relatively later than in the conventional FEC scheme since FEC packets are transmitted over a PTDD period after the transmission of the frame they are protecting. This is the reason why we see that the conventional FEC scheme gives a PSNR about 1 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 nonperiodic 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 15–20% 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.

#### F. 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 subsection, 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.

1) *Impact of Loss Rates:* Figs. 17 and 18 show the video quality of RESCU-FEC and RESCU-REC over various loss rates. To eliminate the impact of network delays, for RESCU-REC, we take the results from the traces with less than 300 ms round trip delays. Both techniques show high error resilience under 10% loss rate. However, when the loss rate becomes larger than 12%, they suffer degradation in their quality.

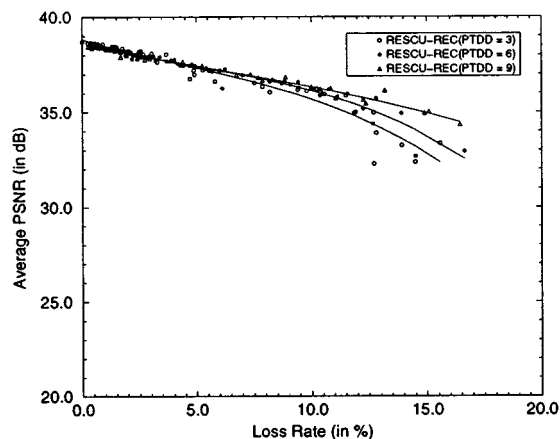


Fig. 17. The average PSNR of RESCU-REC for traces with less than 300 ms round trip delays.

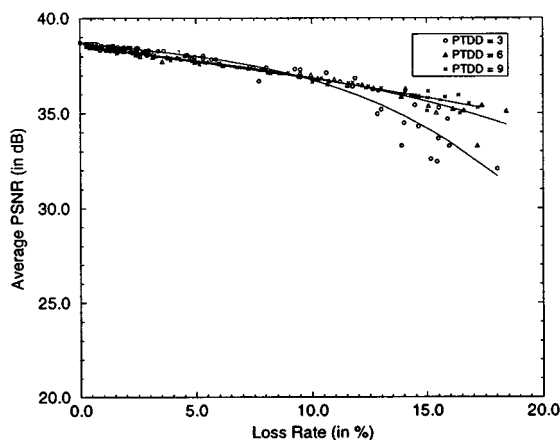


Fig. 18. The average PSNR of RESCU-FEC with various loss rates (container).

The advantage of RESCU-REC over RESCU-FEC becomes evident in their bit consumption. The good video quality of RESCU-FEC comes at the expense of higher bit rates. As shown in Figs. 19 and 20, the bit rate of FEC is generally 5–8% higher than that of REC. There are two reasons why RESCU-REC gives lower bit rates. First, retransmission occurs only when packet losses occur while FEC packets are continually sent regardless of packet losses. This effect can be seen from the good video quality with less bit overhead than RESCU-FEC under low transmission delays. 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 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 RESCU-FEC is still affected by burst losses. This effect is not clearly visible from Fig. 20. Later, we investigate this issue in more detail.

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.

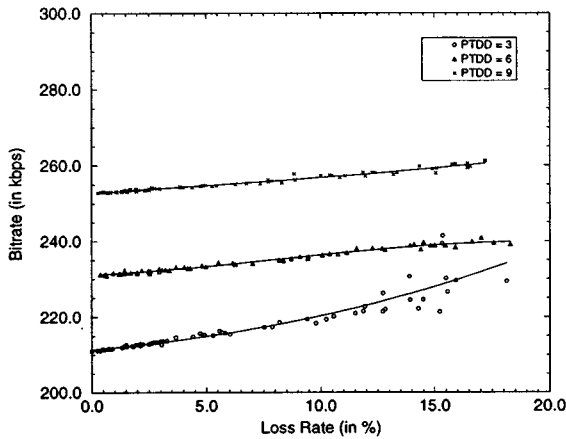


Fig. 19. The average bit rate of RESCU-REC with various loss rates (container).

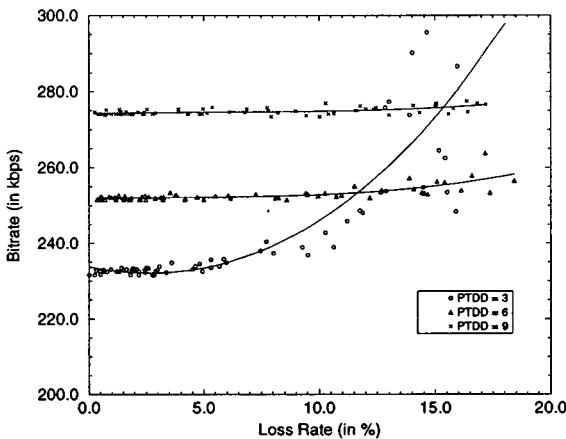


Fig. 20. The average bit rate of RESCU-FEC with various loss rates (container).

RESCU-FEC does not have this problem because the inter-arrival times of repair packets after a packet loss are independent of transmission delays. Figs. 21 and 22 clearly show the effect of transmission delays on the performance of RESCU-REC.

In Fig. 21, RESCU-REC shows good video quality under low network latency (less than 250 ms) even with PTDD 3. However, in all other cases, RESCU-REC is highly sensitive to network latency. RESCU-REC shows total ineffectiveness under high RTT's. As PTDD becomes larger, the video quality generally improves, but larger PTDD increases bit overhead due to lower compression efficiency (see Fig. 19). In Fig. 22, RESCU-FEC is clearly much less sensitive to RTT's. As RTT's 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.

3) *Impact of Burst Losses:* The Internet traces we obtained do not show a wide range of loss burst lengths. Most of the 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

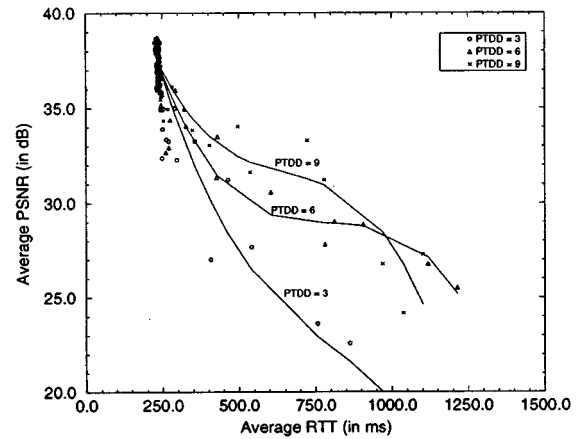


Fig. 21. The average PSNR of RESCU-REC with various round trip delays (container).

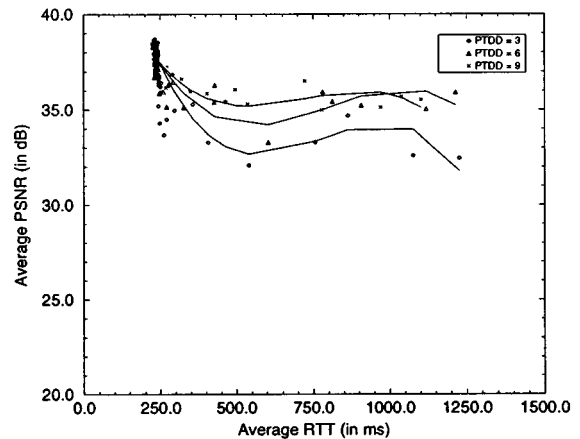


Fig. 22. The average PSNR of RESCU-FEC with various round trip delays (container).

lengths between 1 and 3. 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. The performance study using our Internet traces does not show the impact of burst length to video quality because low loss rate traces generally give very good video quality (even though with occasional long loss bursts). To show the performance impact of loss burst lengths on a variety of network environments, we resort to simulation.

We run simulation experiments which simulate network environments with a wide range of loss burstiness 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 2 to 6. Parity packets are evenly spaced over each PTDD period. Figs. 23 and 24 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 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

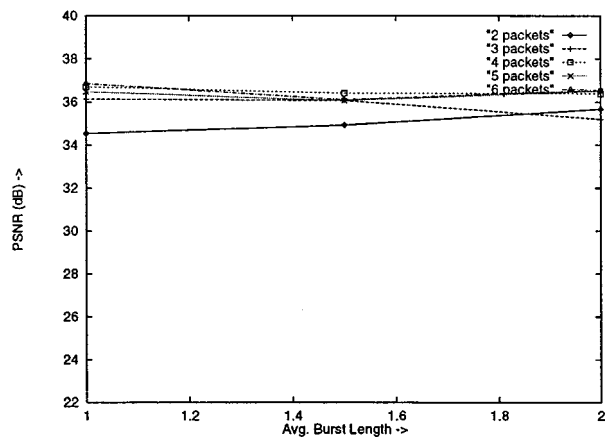


Fig. 23. The average PSNR of RESCU-FEC with various numbers of parity packets within PTDD 6 over different loss burst lengths (simulation on container).

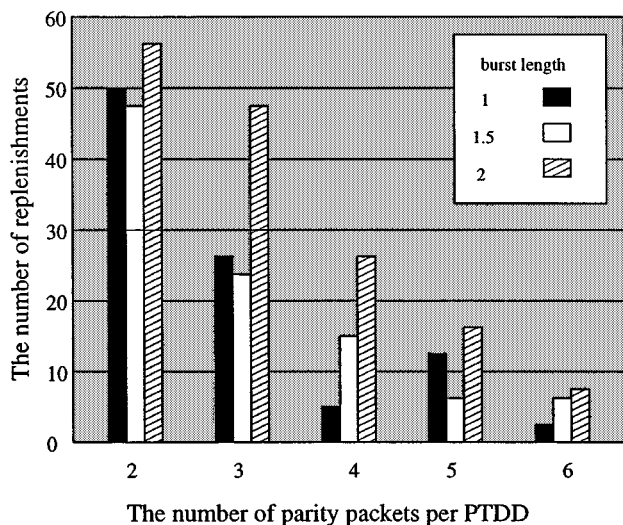


Fig. 24. The number of replenishments for RESCU-FEC with various numbers of parity packets within PTDD 6 over different loss burst lengths (simulation on container).

consists of about 10 packets. Clearly from Fig. 24, 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-min 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.

### G. The Summary of Experimental Results

In comparison to NEWPRED, NEWPRED shows relatively good video quality through its solution for the error propagation problem. However, it tends to incur very high bit overhead

under high loss rates, especially when losses are somewhat uniformly distributed. 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 over a wide range of loss rates while maintaining comparable video quality to that of NEWPRED.

In comparison to various protection mechanisms applied to H.261 and H.263, such as frequent intra-frame replenishments, and more forward error correction per frame, the result shows that eliminating error propagation in H.261 and H.263 tremendously increases error resilience and video quality of interactive video transmission. However, the bit overhead of these schemes that must be incurred to achieve the same video quality as RESCU is very high, requiring over 15–70% more bit rates over RESCU. Even when bit rates are fixed the same for RESCU and H.261, the video quality of RESCU shows more than 5 dB higher PSNR than H.261 in an Internet experiment. The conventional FEC scheme which transmits data packets and their corresponding parity packets all within the same frame interval leaves the video stream very 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.

We also notice that as more high motion becomes present in a test video sequence, RESCU's performance degrades significantly. This is because the effectiveness of RESCU relies heavily on motion prediction, and video scenes with high motion do not contain much temporal redundancy for motion prediction to be effective. We observed this behavior in the test sequences with high motion (*children*). However, RESCU generally gave higher performance than any other techniques we tested.

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 indicate that with a small amount of bit overhead (6–8%) for parity packets, RESCU-FEC can achieve good error resilience under significantly high loss rate and loss burst length.

## V. CONCLUSIONS AND LIMITATIONS

In this paper, we show that retransmission and forward error correction could be made effective error-recovery techniques for interactive video applications without introducing any artificial extension of frame playout times. The key 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, H.263, and NEWPRED. In our preliminary work reported in [37] and [36], we also reported that RESCU gives much higher performance than Intra-H.261 (adopted in vic Mbone tool [29]), 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 are proven to effectively alleviate the problem of error spread with only lower bandwidth overhead than other recovery schemes. 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 and wireless or 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. Since 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 also has 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. We leave the study of such an adaptive technique as future work.

## REFERENCES

- [1] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Trans. Inform. Theory*, vol. 42, Nov. 1996.
- [2] N. Alon and M. Luby, "A linear time erasure-resilient code with nearly optimal recovery," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1732–1736, Nov. 1996.
- [3] E. W. Biersack, "Performance evaluation of FEC in ATM networks," in *Proc. ACM SIGCOMM*, Baltimore, MD, Aug. 1992.
- [4] J. Bolot, "End-to-end packet delay and loss behavior in the internet," in *Proc. ACM SIGCOMM*, San Francisco, CA, Sept. 1993, pp. 289–298.
- [5] J. Bolot and A. Vega-Garcia, "The case for FEC-based error control for packet audio in the internet," *ACM Multimedia Syst. J.*, to be published.
- [6] J.-C. Bolot and T. Turletti, "Adaptive error control for packet video in the internet," in *Proc. Int. Conf. Internet Protocols*, Lausanne, Sept. 1996.
- [7] G. Carle and E. Biersack, "Survey of error recovery techniques for ip-based audio-visual multicast applications," *IEEE Network*, Dec. 1997.
- [8] S. Casner and V. Jacobson, "Compressing IP/UDP/RTP headers for low-speed serial links (work in progress), ETF," RFC-2508, Feb. 1999.
- [9] M. Degermark, H. Hannu, L. Jonsson, and K. Svanbro, "Crtp over cellular radio links (work in progress), IETF," RFC-2509, June 1999.
- [10] H. Deng and M. Lin, "A type I hybrid ARQ system with adaptive code rates," *IEEE Trans. Commun.*, vol. 46, pp. 733–737, Feb. 1995.
- [11] T. Dorcay, "Cu-seeme desktop videoconferencing software," *ConneXions*, vol. 9, no. 4, Mar. 1995.
- [12] 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 *Proc. ACM SIGCOMM Conf.*, Oct. 1995, pp. 342–356.
- [13] R. Fredrick, "Network video(nv)," Xerox Palo Alto Res. Center, Tech. Rep.
- [14] M. Ghanbari and V. Seferidis, "Cell-loss concealment in ATM video codecs," *IEEE Trans. Circuits Syst. Video Tech.*, vol. 3, pp. 238–247, June 1993.
- [15] J. Hagenauer, "Rate-compatible punctured convolutional codes (rcpc codes) and their applications," *IEEE Trans. Commun.*, vol. 36, pp. 389–400, Apr. 1988.
- [16] ITU-T, Video codec for low bit-rate communications, 1998.
- [17] S. Kallel and D. Haccoun, "Generalized type-II hybrid ARQ scheme using punctured convolutional coding," *IEEE Trans. Commun.*, vol. 38, pp. 1938–1946, Nov. 1990.
- [18] S. K. Kaseta, J. Kurose, and D. Towsley, "Scalable, reliable multicast using multiple multicast groups," in *Proc. ACM SIGMETRICS*, Seattle, WA, 1997, pp. 64–74.
- [19] L. Kieu and K. Ngan, "Cell-loss concealment techniques for layered video codecs in an ATM network," *IEEE Trans. Image Processing*, vol. 3, pp. 666–677, Sept. 1994.
- [20] T. Kinoshita, T. Nakahashi, and M. Takizawa, "Variable bit-rate hdtv coding algorithm for ATM environments in B-ISDN," in *Proc. SPIE Conf. Visual Commun. Image Processing: Visual Commun.*, Nov. 1991, pp. 604–612.
- [21] LBC, "Sub-videos with retransmission and intra-refreshing in mobile/wireless environments," Document, LBC-95-309 (ITU-T SG 15, WP 15/1), 1995.
- [22] LBC, "An error-resilience method based on back channel signalling and FEC," Document, LBC-96-033 (ITU-T SG 15, WP 15/1), 1996.
- [23] C. Leicher, "Hierarchical encoding of MPEG sequences using priority encoding transmission (pet)," ICSI, Tech. Rep. 94-058, Nov. 1994.
- [24] X. Li, S. Paul, P. Pancha, and M. Ammar, "Layered video multicast with retransmission (Ivmr): Evaluation of error recovery schemes," in *Proc. 6th Int. Workshop Network Operating Syst. Support Digital Audio Video*, St. Louis, May 1997.
- [25] S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
- [26] H. Liu and M. El Zarki, "Performance of video transport over wireless networks using hybrid ARQ," in *Proc. IEEE Int. Conf. Universal Personal Commun. (ICUPC)*, Boston, MA, Oct. 1996, pp. 567–571.
- [27] W. Luo and M. El Zarki, "Analysis of error concealment schemes for MPEG-2 video transmission over ATM networks," in *Proc. SPIE/IEEE Visual Commun. Image Processing*, Taiwan, May 1995.
- [28] J. McAuley, "Reliable broadband communications using a burst erasure correcting code," in *Proc. ACM SIGCOMM*, Philadelphia, PA, Sept. 1990.
- [29] S. McCanne and V. Jacobson, "Vic: A flexible framework for packet video," in *Proc. ACM Multimedia'95*, San Francisco, CA, Nov. 1995, pp. 511–522.
- [30] J. Nonnenmacher, E. Biersack, and D. Towsley, "Parity-based loss recovery for reliable multi-cast transmission," *IEEE/ACM Trans. Networking*, to be published.
- [31] C. Papadopoulos and G. Parulkar, "Retransmission-based error control for continuous media applications," in *Proc. 6th Int. Workshop Network Operating Syst. Support Digital Audio Video*, 1996, pp. 5–12.
- [32] S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya, "Reliable multi-cast transport protocol (RMTP)," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1996.
- [33] M. Podolsky, "A study of speech/audio coding on packet switched networks," M.S. thesis, Dept. Elec. Eng. Computer Sci., Univ. Calif., Berkeley, Dec. 1996.
- [34] M. Podolsky, C. Romer, and S. McCanne, "Simulation of FEC-based error control for packet audio on the internet," in *Proc. ACM SIGMETRIC/PERFORMANCE*, June 1998.
- [35] G. Ramamurthy and D. Raychaudhuri, "Performance of packet video with combined error recovery and concealment," in *Proc. IEEE INFOCOM*, Apr. 1995, pp. 753–761.
- [36] I. Rhee, "Error control techniques for interactive low-bit rate video transmission over the internet," in *Proc. ACM SIGCOMM'98*, Vancouver, Canada, Sept. 1998.

- [37] —, “Retransmission-based error control for interactive video applications over the internet,” in *Proc. Int. Conf. Multimedia Computing Syst.*, TX, June 1998, pp. 118–127.
- [38] L. Rizzo, “Effective erasure codes for reliable computer communication protocols,” *Computer Commun. Rev.*, vol. 27, pp. 24–36, Apr. 1997.
- [39] D. Rubenstein, J. Kurose, and D. Towsley, “Real-time reliable multicast using proactive forward error correction,” in *NOSSDAV’98*, to be published.
- [40] E. Steinbach, N. Farber, and B. Girod, “Standard compatible extension of H.263 for robust video transmission in mobile environments,” *IEEE Trans. Circuits Syst. Video Tech.*, vol. 7, pp. 872–881, Dec. 1997.
- [41] R. Storn, “Modeling and optimization of pet-redundancy assignment for MPEG sequences,” ICSI, Berkeley, CA, Tech. Rep. TR-95-018, May 1995.
- [42] R. Talluri, “Error-resilient video coding in ISO MPEG-4 standard,” *IEEE Commun. Mag.*, vol. 36, pp. 112–119, June 1998.
- [43] Priority encoding transmission. [Online]. Available: <http://www.icsi.berkeley.edu/pet/icsi-pet.html>
- [44] M. Wada, Selective recovery of video packet loss using error concealment, vol. 7, no. 5, pp. 807–814, 1989.
- [45] R. Xu, C. Myers, H. Zhang, and R. Yavatkar, “Resilient multicast support for continuous-media applications,” in *Proc. 6th Int. Workshop Network Operating Syst. Support Digital Audio Video*, St. Louis, May 1997.
- [46] J. Zdepski and H. Sun, “Error concealment strategy for picture-header loss in MPEG compressed video,” in *Proc. High-Speed Networking Multimedia Computing*, San Jose, CA, Feb. 1994, pp. 145–152.



**Injong Rhee** received the B.E. degree in electrical engineering from Kyung-Pook National University, Korea in 1989, and the Ph.D. degree in computer science from the University of North Carolina, Chapel Hill in 1994.

After conducting postdoctoral research for one year at Warwick University, U.K., and one year at Emory University, he has been an Assistant Professor of Computer Science at North Carolina State University since 1997. He received the NSF Faculty Career Development Award in 1999. His research and teaching interests include computer networks, multimedia networking, distributed systems, and operating systems.



**Srinath R. Joshi** received the B.Tech. degree in computer science and engineering from the University of Calicut, India, in 1995. He worked with Wipro Infotech and later Aditi Technologies, Bangalore, India, before joining the graduate program in the Computer Science Department at North Carolina State University, Raleigh, in 1998.

His current research is in the area of error recovery schemes for video transmission in lossy environments.