Unequal Loss Protection for H.263 Compressed Video

Justin Goshi, Alexander E. Mohr
Richard E. Ladner
Computer Science and Engr.
University of Washington
Seattle, WA 98195–2350
{goshi,amohr,ladner}@cs.washington.edu

Eve A. Riskin
Electrical Engineering
University of Washington
Seattle, WA 98195–2500
riskin@ee.washington.edu

Alan Lippman
Trusted Media Networks, Inc.
P.O. Box 3822
Seattle, WA 89124
alanl@trustedmedianetworks.com

***Abstract*— We study the application of Unequal Loss Protection (ULP) algorithms to motion-compensated video over lossy packet networks. In particular, we focus on streaming video applications over the Internet. The original ULP framework [14] applies unequal amounts of forward error correction (FEC) to embedded data to provide graceful degradation of quality in the presence of increasing packet loss. In this paper, we apply the ULP framework to baseline H.263, a video compression standard that targets low bit rates, by investigating re-orderings of the bitstream to make it embedded. The re-ordering process allows a receiver to display high quality video, even at the high loss rates encountered in wireless transmissions and the current Internet.**

I. Introduction

The rapid growth of the Internet and increasing bandwidth and computing power over the past few years, have heightened the interest in multimedia applications such as video conferencing and video-on-demand. A particular application, streaming media over the Internet, is rapidly increasing in popularity, with future applications to wireless devices expected. With streaming media, users can request a multimedia file and, after a small buffering delay, can start playing the file while it is still being downloaded. However, these time-sensitive applications do not fit well into the current best-effort nature of the Internet. When the network transmission capacity is exceeded, packets are discarded at random and reliability is commonly implemented by retransmitting lost packets. However, for multimedia applications, the delay introduced by retransmissions is often undesirable. Recent research has found that packet losses even on the wired Internet often occur at high rates [3], [15] and a study by Boyce [6] on the effects of packet loss on the quality of MPEG video shows that those high loss rates can severely damage video quality due to the temporal dependencies inherent in motion-compensated video coders.

This work was supported by RealNetworks, Inc., the Washington Technology Center, and NSF Grants No. EIA-9973531 and CCR-0104800.

The Unequal Loss Protection (ULP) framework [14] was developed to provide graceful degradation of quality in the presence of increasing packet loss. Graceful degradation provides a user with reduced but acceptable quality in proportion to the amount of data lost. The ULP framework accomplishes this by applying unequal amounts of forward error correction (FEC) to protect embedded data from packet losses. Embedded data have the property that the earlier parts of the compressed bit stream are most important to the overall quality of the reproduction. Furthermore, the later parts of the bit stream are meaningless without the beginning parts of the bit stream. Using this framework, it was shown that graceful degradation of image quality can be achieved without retransmissions and without any support from the network, such as priority dropping of packets. In this paper, we focus on streaming video applications for the Internet and adapt the ULP framework to achieve graceful degradation of video quality. We concentrate our efforts on motion-compensated video coders, which take advantage of temporal redundancy to achieve efficient compression. This approach is used by most current video coding standards, but it makes these coders susceptible to losses. We use the H.263 video coding standard [9], [18] as our test-bed because its techniques are typical of those used at the low bit rates found on both wireless networks and the current Internet.

A. Related Work

Many people have contributed effort towards making multimedia data more robust in the presence of bit errors and packet losses. For a good review of many of these error resilience techniques, see [7], [19], [20]. Here, we focus on techniques specific to combating packet loss, both within the MPEG and H.263 video coding standards and previously proposed schemes that use FEC.

Scalable video coding techniques encode the source into a base layer and one or more enhancement layers. The base layer

is coded so that a coarse video sequence can be decoded from the base layer, while the enhancement layers serve to increase quality should they be available. This approach is well-suited to networks that offer different priority levels or for adapting to channel bandwidth in multicast environments. In [2], the loss resilience of the four scalable video coding algorithms for MPEG-2 is examined and in [21] the error resilience schemes (including scalable video coding) in H.263+ are studied.

The latest version of H.263, commonly called H.263++, contains support for data partitioning and coding using reversible variable length codes (RVLC) [12]. This partitioning is aimed at protecting against bit errors in wireless networks by splitting the macroblock headers, motion vectors, and coefficients into separate segments, which allows each segment to be isolated from errors or erasures in other segments. The RVLCs allow a segment to be decoded from both directions and improve resilience to bit errors.

Leicher [11] applied Priority Encoding Transmission (PET) [1] to entire compressed MPEG video sequences using a coarse three-level system where the intra (I) frames had priority 60%, the predicted (P) frames had priority 80%, and the bidirectional (B) predicted frames had priority 95%. The priority specifies the percentage of packets necessary to recover the message fragment, which reflects the fact that I-Frames are required to decode P-Frames and that P-Frames are required to decode B-Frames, due to temporal dependencies.

Bolot and Turletti [4] described a redundancy scheme for conditional replenishment coders in which DCT blocks updated in packet n would also have more-highly quantized versions transmitted in the following i packets. Thus, assuming a receiver can accommodate a buffer delay of i packets, information about a changed block will be decoded as long as at least one of those i packets is received. In effect, this scheme uses FEC in the form of a simple repetition code and has some degree of unequal protection because redundant DCT blocks are quantized with larger and larger step sizes.

In a more recent study, Boyce [5] introduced the High Priority Partitioning (HiPP) scheme using the data partitioning ideas in MPEG-2 to provide unequal loss protection of MPEG video. In MPEG-2 data partitioning, the data are split into high and low priority partitions. The critical data, such as headers, motion vectors, and low frequency discrete cosine transform (DCT) coefficients, are placed in the high priority partition, while the remaining high frequency coefficients are placed in the low priority partition. A priority breakpoint specifies a cutoff that decides whether the coefficients should go into the high or low priority partition. In Boyce's HiPP scheme, arbitrary splitting is supported and Reed-Solomon FEC is applied only to the high priority partition. It puts all I-Frame data into the high priority partition, puts all B-Frame data into the low priority partition, and splits P-Frame data between the high and low priority partitions.

B. Contribution of this Paper

Our main contribution is in illustrating how the ULP framework can be applied to motion-compensated video (in particular, H.263 video). The use of the ULP framework allows us to improve on previous work that uses FEC to protect motion-compensated video in the following ways:

1) *We apply ULP over fine grained chunks of data.* Leicher protected entire frames and Boyce protected only the high priority partition. We segment the data using ideas from both scalable coding and data partitioning techniques, providing data fragments that are both much smaller and more numerous. By ordering those fragments according to their relative importance, we provide better graceful degradation of video quality.
2) *We optimize the FEC allocation for a given video sequence and estimated network conditions.* In Leicher's scheme, the user must manually assign priorities to each frame type, while in Boyce's scheme, the priority cutoff must be specified.
3) *We specify a target transmission bit rate and trade-off the encoding bit rate and redundancy rate such that the total bit budget is satisfied.* Leicher and Boyce both specify an encoding bit rate and add additional redundancy.

C. Paper Organization

This paper is organized as follows. Section II gives some background on the original ULP framework and the H.263 video coding standard. Section III describes the problems we encountered in adapting the ULP framework to motion-compensated video and how we solved them. Section IV presents our simulations comparing our ULP system to baseline H.263. The results serve to illustrate that the ULP framework effectively provides graceful degradation. Finally, section V contains our conclusions and future work.

II. Background

In this section, we begin with an overview of the ULP framework followed by a brief introduction to the H.263 video coding standard. For an in-depth description of the ULP framework, see [14], [13]. For a good introduction to H.263, refer to [9], [18].

A. ULP Framework

The ULP framework [14] operates by applying unequal amounts of FEC to data that are compressed with an unmodified embedded algorithm and transmitted over lossy packet networks. A diagram of the entire system as it would be used is shown in figure 1. The scheme is modular in that it can use any embedded compression algorithm and obtain graceful degradation of quality with increasing packet loss rate. In the following paragraphs we describe how the ULP framework provides graceful degradation, and the algorithm used for allocation of source and redundancy.

How ULP Provides Graceful Degradation: The ULP framework is based on Priority Encoding Transmission (PET) [1], which assigns unequal amounts of FEC to data sent over lossy packet networks according to user-specified priorities and message fragments. The PET algorithm does not specify how to choose message fragment sizes or how to assign priorities, but

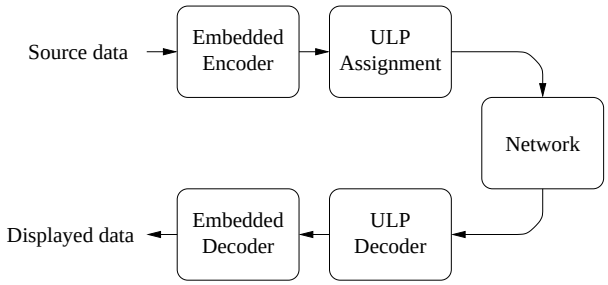

Fig. 1. Block diagram showing the components of a system that uses ULP to protect embedded data.

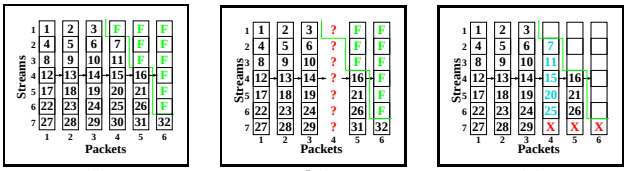

Fig. 2. A message of 32 symbols of data (numbers 1–32) and ten symbols of FEC (F) is divided into seven streams and sent in six packets. (a) The data layout before transmission. (b) The data layout after packet 4 is lost during transmission. (c) The data layout after FEC is used in recovery.

the ULP framework adds these capabilities. An example of the protection provided by the ULP framework is shown in figure 2. It adds FEC to each message fragment such that the fragment and the FEC form a *stream*. The message is divided into L streams, each of which contains one symbol from each of N packets. The property of this type of redundancy allows a given stream containing both data and FEC symbols to be successfully decoded as long as the number of lost packets is less than or equal to the number of FEC symbols. It is important to note that successfully decoding a given stream depends only on the number (not the order) of packets lost.

In Figure 2(a), each of the $L = 7$ rows is a stream and each of the $N = 6$ columns is a packet. This figure shows one possible way to send a message of 32 symbols of data (numbers 1–32) and ten symbols of FEC (F). Notice that more FEC is applied to the earlier parts of the message and less FEC is applied to the later parts. This is done because the ULP framework operates on embedded bit streams. Recall that for embedded bit streams, the earlier parts of the message are more important to the overall quality of the reproduction and that the later parts of the bit stream are meaningless without the beginning parts of the bit stream. By using more FEC for earlier symbols in the message we have the desired property that if we can recover a symbol x, then we are guaranteed to recover all symbols $y < x$. An example where one packet out of six is lost and five are received correctly is shown in figures 2(b) and 2(c). In this case, the first six streams can be recovered because they contain at least one FEC symbol. The last stream cannot be completely recovered as it has no FEC, although a partial decoding is possible. Thus, we can recover a lower fidelity version of the message from the decoded prefix, and each additional stream that is decoded improves the quality of the received message. This illustrates how the ULP framework provides graceful degradation when applied to embedded bit streams.

The ULP Algorithm: The ULP algorithm operates on a description of the source in the form of a utility–cost curve. Utility measures how much benefit the receiver is likely to enjoy by receiving and decoding the data, and cost specifies the number of bits used by the encoder to achieve a given utility. Utility can be measured in many ways, and in this paper we use the total sequence peak signal-to-noise ratio (PSNR). This metric is computed by taking the PSNR value using the average mean squared error over the video sequence. We chose this metric because it reflected our subjective quality fairly well over the video sequences used in our experiments.

Given as inputs the number of packets to be transmitted, the length of those packets, the utility–cost curve representing the source, and an estimate of the packet loss probability in the form of a probability mass function (pmf), the ULP assignment problem is to find an assignment of FEC for each stream such that the expected utility at the receiver is maximized. To determine the amount of data and FEC to assign to each stream, a hill-climbing algorithm was first developed in [14]. Improved ULP assignment algorithms have since been developed [13], [16]. The later algorithms are much faster (two orders of magnitude in some tests) and find allocations with similar or better expected utilities.

B. Baseline H.263

H.263 is a DCT-based, motion-compensated international video coding standard that operates very efficiently at the low bit rates found on the Internet and wireless networks. Version 2 of the standard, commonly known as H.263+, contains 16 optional annexes offering enhancements such as improving compression performance, allowing the use of scalable bit streams, and providing support for custom picture sizes, all at the cost of an increase in complexity. Version 3 of the standard, known as H.263++, contains even more such enhancements. Since we are only interested in demonstrating graceful degradation of video quality, we concentrate on baseline H.263 with no optional annexes for our work and briefly describe the relevant aspects of that standard.

Video Frame Structure: Baseline H.263 supports five standardized picture formats: sub-QCIF, QCIF, CIF, 4CIF, and 16CIF. The luminance component is sampled at the selected resolution while the chrominance components are down-sampled by a factor of two in both the horizontal and vertical dimensions. Each frame in the video sequence is split into 16×16 pixel macroblocks (MB) where each MB is composed of six 8×8 pixel data blocks: four luminance and two chrominance.

Motion Compensation: There are two types of frames in baseline H.263: intra (I-Frame) and inter (P-Frame). The I-Frame contains only intra macroblocks while the P-Frame can contain both intra and inter macroblocks. An intra MB is coded independently, whereas an inter MB is first motion-compensated and then the differences are encoded. Motion-compensated prediction models the pixels within the current MB as being a translation of pixels in the previously encoded frame and represents that translational motion by two-dimensional displacement vectors called motion vectors (MV). In baseline H.263,

there is one motion vector for each inter MB. Inter-frame coding gives very good compression efficiency; however, it also leads to temporal error propagation since the P-Frames depend on earlier frames in the video sequence. In baseline H.263, the first frame in the video sequence is an I-Frame and all other frames are P-Frames.

Transform and Entropy Coding: Each 8×8 block of original or motion-compensated pixels is transformed using the DCT [17]. The transform decorrelates the block of pixels and compacts its energy into a small number of coefficients, which are then quantized using the same quantization step size. The quantized coefficients are entropy-coded using variable-length codes (VLCs). Prior to entropy coding, the 8×8 DCT transformed block is scanned into a one-dimensional array, ordered from low frequency to high frequency coefficients, and coded using a three-dimensional run-length VLC table representing the triple (LAST, RUN, LEVEL). The symbol LAST is one bit representing whether or not this codeword is the last in the current block. The symbol RUN is the distance between two nonzero coefficients in the array of coefficients. The symbol LEVEL is the quantized value of the nonzero value immediately following a sequence of zeros. The motion-vectors are also entropy-coded using VLC codewords after being predicted from nearby MVs.

III. Applying ULP to H.263

This section details our application of the ULP framework to compressed H.263 video and the mechanisms that we use to overcome the challenges inherent in that process. Several changes to the original ULP system that was illustrated in figure 1 were necessary. These modifications are described below and addressed in the following subsections.

1) *We must create independent subsequences.* Because streaming video exists as a continual sequence of frames, we must divide that sequence into discrete independent subsequences. Each of these subsequences is protected in the fashion that was illustrated in figure 2.
2) *We must reorder the compressed bit stream.* The H.263 coder does not produce an embedded bit stream (the data are not ordered in any way). Because the ULP algorithm assumes that the bit stream is embedded, we must reorder the compressed bit stream to make it appear embedded.
3) *We must select the encoding bit rate.* The ULP framework relies on having a utility-cost curve that describes the utility of the bitstream when it is truncated at various points. It is difficult to generate a single utility-cost curve for an input source since it depends on the chosen encoding bit rate of the H.263 encoder.

Figure 3 shows the entire system which is modified to protect H.263 motion-compensated video. We first decide on the number of frames to use in our independent subsequences which we call a group of pictures (GOP). A decision for the encoding bit rate along with the video source is used as input to the H.263 encoder and a compressed bitstream is produced. We then pass the data to an ordering module that performs the reorganization described in Section III-B. The ULP assignment module determines an assignment for the ordered bitstream and

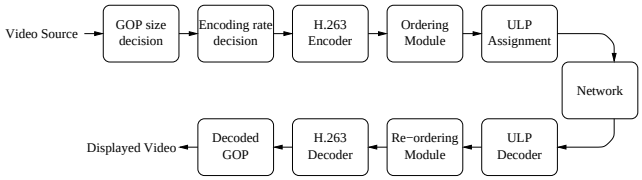

Fig. 3. Block diagram showing the components of a system that uses ULP to protect H.263 motion-compensated video. Protection is provided separately over each independent subsequence.

creates packets to be transmitted over the network. The network may drop packets, the surviving packets are given to the ULP decoder, and a prefix of the ordered bitstream is created. The re-ordering module converts that prefix into a standards-compliant syntactically-legal compressed bitstream, which the H.263 decoder uses to display the video sequence.

A. Creating Independent Subsequences

Video exists as a continual sequence of frames, so an important design decision is the number of frames that each subsequence contains. This group of pictures (GOP) size is influenced by our application of interest, streaming media applications over the Internet which has the following properties:

1) For video-on-demand, small start-up delays are often encountered.
2) The Internet uses large packet sizes of 500 to 1500 bytes so that the overhead of packet headers is minimal.
3) Longer FEC codes that spread information among more packets make the data more resistant to bursty losses.

These points imply a trade-off between long subsequences that impose long delays, yet are effective against bursty losses, and short subsequences that impose short delays, yet are less effective against bursty losses. We study GOP sizes that induce delays of one to three seconds, less than the five seconds common in current implementations by industry. Also, to ensure that subsequences are independent, we encode the first frame in a GOP as an I-Frame so that our ordering schemes need not worry about error propagation between GOPs.

B. Reordering the Compressed Bitstream

The assumption that the data are embedded gives the ULP framework guidelines for the relative importance of the data and allows the bit stream to be truncated at arbitrary points. Because H.263 is not an embedded coder, our approach is to order the compressed bit stream to make it appear embedded.

Figure 4 illustrates the main idea for our reordering scheme which we call frequency-based ordering (because we reorder the VLC codewords from low to high frequency content, reflecting their order of importance to visual quality). In the normal H.263 bit stream, data for each frame are interleaved in raster-order (left to right, top to bottom). This creates the effect that the important data (such as header information and motion vectors) are spread throughout the bit stream as shown in the figure. The figure shows that our reordering scheme attempts to order the data for a group of pictures in order of importance. We do this by extracting data from each frame and rearranging

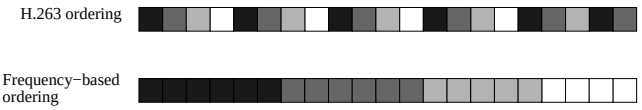

Fig. 4. H.263 bit stream layout and frequency-based bit stream layout. The darker colors represent data that are more important to the quality of reconstruction.

the bit stream. The particular ordering of importance that we use in our frequency-based ordering scheme is:

1) The first I-Frame in a GOP.
2) Picture headers from all frames in a GOP.
3) MB headers from all frames in a GOP.
4) MV data from all MBs in a GOP.
5) First VLC codewords from each block in a GOP.
6) Second VLC codewords from each block in a GOP.
7) *Etc.*

While our ordering scheme does not create a truly embedded bit stream, it is ordered according to relative importance. For example, the first data that we receive are the I-Frame, allowing us to reconstruct the first frame in a group of pictures. Next comes various control information like headers and motion vectors. Finally, we group the kth VLC codewords from each block (where the groups are ordered from low to high frequency content). The effect of this grouping is that we receive some data from all blocks in all frames allowing us to reconstruct a coarse version of our video sequence. The more data we receive, the more VLC codewords from each block we can decode allowing us to achieve progressively better quality.

Validation of our Frequency-Based Ordering Scheme:

To test how well our frequency-based ordering scheme works to create an embedded bit stream, we compare it against a scheme similar to the one Leicher used [11] (which we call a PET-style ordering) and an HiPP [5]-style ordering. In the PET-style ordering, each frame is assigned its own priority segment. The I-Frame has the highest priority, followed by the P-Frames in order from the earliest to latest in a GOP. This order reflects the relative order of frame importance due to temporal dependencies. In the HiPP-style ordering, we have two priority segments (high and low). All I-Frame data go into the high priority segment and P-Frame data are split into the high and low segments, with the priority cutoff set at the one that provides the most graceful degradation for each experiment.

To compare the various ordering schemes we took H.263 encoded bit streams, ordered them according to the various ordering schemes, and simulated the effects of applying ULP and sending the data over a lossy network. Because we are using the ULP framework, we can simulate various amounts of loss by simply truncating the ordered bit stream, since this is exactly the kind of priority protection that ULP gives you. By comparing the reconstructed sequences after truncation, we can get an idea of which ordering scheme approximates an embedded bit stream the best.

When watching the video sequences, we noticed that the PET-based scheme degrades the most and that our frequency-based scheme degrades the least. The PET-based scheme loses data in chunks, such that many frames at the end of a GOP

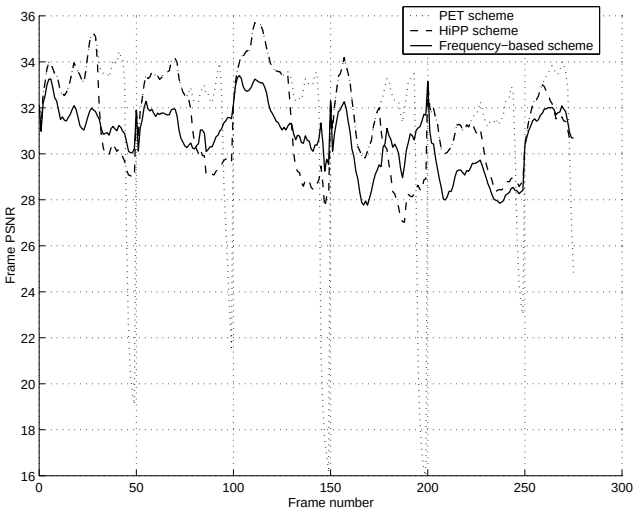

Fig. 5. Frame by frame PSNR for the foreman video sequence encoded at 128 kbps when each set of 50 frames experiences 10% data loss. The x-axis plots the frame number of all encoded frames in the video sequence and the y-axis plots the frame's PNSR.

are lost. This results in a jerky video sequence caused by frame skipping. On the other hand, the frequency-based scheme spreads the effects of loss among the DCT-coefficients in all frames in a GOP. The data are slightly degraded, but entire frames do not get dropped. The HiPP-based scheme is similar to our frequency-based scheme but the loss is not spread in as fine a granularity because there is only one priority cutoff. Another drawback to this approach is that the priority cutoff must be user specified. An example of the degradation effects can be seen by looking at Figure 5, which shows the utility for each encoded frame in the foreman video sequence. The sequence is encoded at 128 kbps, a GOP has 50 frames, and each encoded GOP experiences 10% truncation. Within a GOP, the frequency-based ordering scheme has the lowest PSNR for the first frame but the least variation in PSNR among the frames in a GOP. The loss is spread out among the various frames providing a slightly degraded but good quality video sequence. We also observed the same trend for other video sequences, encoding bit rates, GOP sizes, and truncation amounts. Because our frequency-based scheme provides the best graceful degradation in the face of increasing truncation, we conclude that it behaves the closest to an embedded coder (the data are ordered by importance).

C. Selecting the Encoding Bit Rate

For a true embedded encoder, the bit stream would be the same if the encoder were asked to produce a bits, or if it were asked to produce b bits and those $b > a$ bits were truncated to a bits. With the ordering scheme above, however, an encoding of a bits will almost certainly be different from an encoding of $b > a$ bits that is truncated to a bits (due to the monolithic nature of the encoder). To differentiate our ordered monolithic encodings from the usual embedded or scalable encodings that continually refine data, we refer to the ordered encodings as *truncatable*.

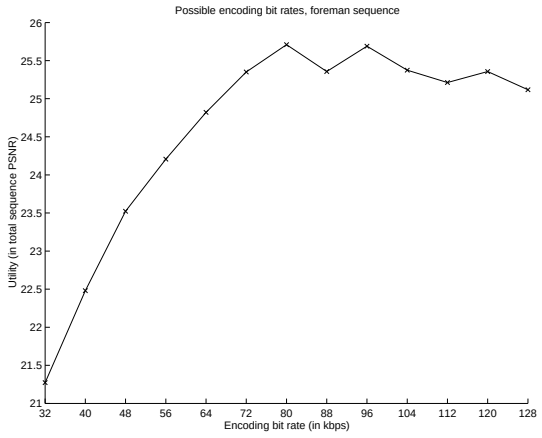

Fig. 6. Illustration of the utility (in total sequence PSNR) for different attempted encoding bit rate. We choose the encoding bit rate that yields the best utility.

Given a mismatch between a sequence targeted for a bits and one truncated to a bits, generating a single utility–cost curve to describe the encoded bit stream is problematic. Instead, we generate a family of utility–cost curves, where each curve represents a single ordered encoding truncated to points that correspond to the order listed in Section III-B. We then run the ULP assignment algorithm independently over each member of the family and use the encoding bit rate and FEC assignment that yields the best expected utility. An example of the various expected utilities after running the ULP algorithm on the utility–cost curves generated for different encoding bit rates is shown in figure 6. The results are shown for the foreman video sequence, a target bit rate of 128 kbps, and an exponential loss model with a mean loss rate of 10%. We tried encoding bit rates of 32 kbps to 128 kbps in increments of 8 kbps. By looking at the figure we see that the best encoding bit rate is 80 kbps, so we would encode the source at 80 kbps and use the amount of redundancy that was used in generating the given utility.

IV. Results

In this section, we present some of our experimental results obtained using simulations. We present comparisons of our ULP system with baseline H.263 to show that our system can provide good graceful degradation in the presence of packet loss. Given an estimate of the network loss model, we compare our ULP system to baseline H.263 by looking at expected frame PSNR values and frame PSNR values corresponding to different packet loss rates. The results give us insight into the quality of the video sequences and how it varies as the video is played. Before going into our results we describe the methodology used in our simulations.

A. Methodology

For our experiments, we implemented the system illustrated in figure 3. For the H.263 encoder and decoder we used the implementation of the TMN version 3.0 H.263 codec [8]. The only change made to the decoder causes it to insert duplicate frames into the reconstructed video sequence when a frame is skipped, which is needed to compare the original and reconstructed video sequences on a frame by frame basis. We could have achieved the same effect without changing the decoder, but this implementation was considered easiest. We use the approximately optimal ULP algorithm described in [13]. We simulate rather than use an actual network to allow greater control over the packet loss conditions. Finally, we compare the reconstructed video sequence both subjectively (by viewing the sequences) and objectively (by examining frame by frame utilities). It is important to note that the graphs for the frame utilities show sharp drops in utility at the start of each GOP. This is due to the fact that the H.263+ encoder skips frames at the low bit rates used in our experiments. We decided to include frame utilities even for skipped frames for fairness since our ULP system and baseline H.263 will skip different frames (since they encode at different bit rates).

We present results using the 400 frame foreman sequence and the first 200 frames of the news sequence. These video sequences were downloaded from the web site: *http://kbs.cs.tu-berlin.de/~stewe/vceg/sequences.htm*. The results presented in this section use a target transmission bit rate of 128kbps and a GOP size of 50 frames (corresponding to a 2 second delay). We use an exponential loss model with a mean loss rate of 10% for each GOP. We simulate the packet loss by randomly choosing which packets are dropped. This produces some counter-intuitive results when looking at baseline H.263 where sometimes the performance looks better at higher loss rates. This happens because the order of packets lost affects the quality of the reconstructed video. We do not see this effect for our ULP system because only the number (not the order) of packets lost affects successful decoding of the data as described in section II. We also experimented with other video sequences and various settings of parameters (target transmission bit rate, GOP size, and network loss model) and found those results to be consistent with those presented in this section.

B. Expected Frame PSNR

In this section, we compare our ULP system to baseline H.263 by looking at the performance measured in expected frame PSNR. This metric is computed by averaging the performance over all possible loss rates with the performance weighted according to the probabilities given by the loss model (exponential with a 10% mean loss rate). The results for the foreman sequence are shown in figure 7 and the results for the news sequence are shown in figure 8. The results show that the expected performance of our ULP system is much better than baseline H.263 due to the packet loss protection used by ULP. Notice again that both ULP-protected sequences skip frames at the ends of GOPs due to packet loss.

C. Frame PSNR at Various Packet Loss Rates

In this section, we compare our ULP system to baseline H.263 by looking at frame PSNR values for different loss rates. Figure 9 shows the results for the foreman sequence and figure 10 shows the results for the news video sequence. Both figures show the same trends. For no packet loss baseline H.263 performs better since it encodes at the target transmission bit

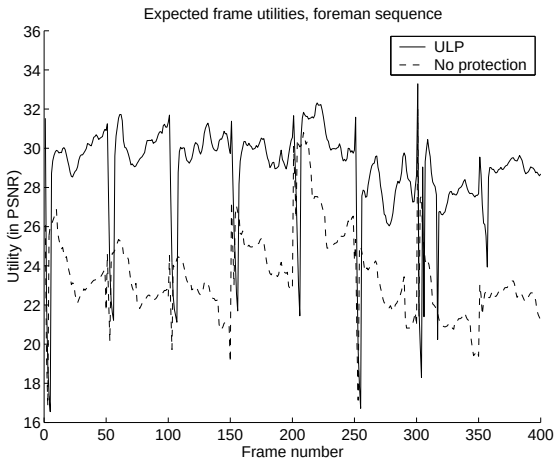

Fig. 7. Expected frame PSNR values for the foreman video sequence using an exponential loss model with a 10% mean loss rate for each GOP (50 frames).

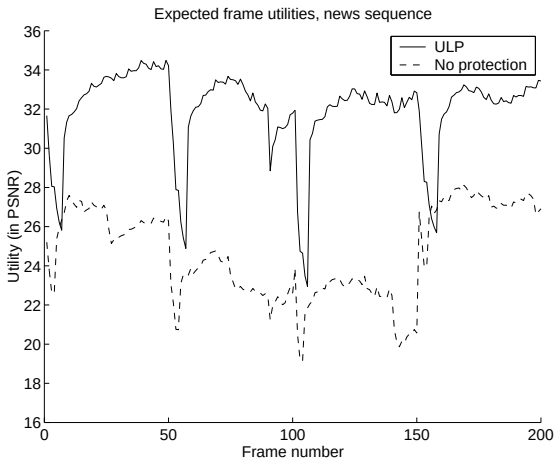

Fig. 8. Expected frame PSNR values for the news video sequence using an exponential loss model with a 10% mean loss rate for each GOP (50 frames).

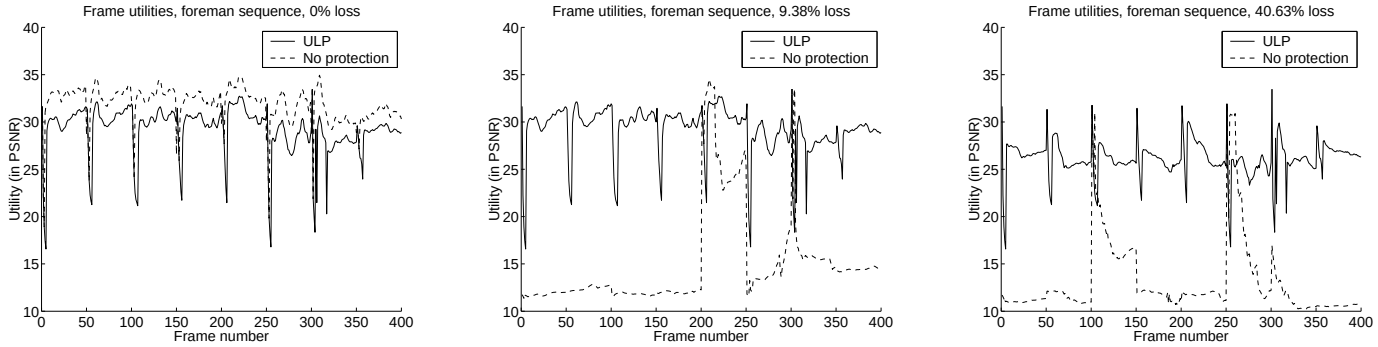

Fig. 9. Frame PSNR values for the foreman video sequence when experiencing no packet loss (left), 9.38% packet loss (center), and 40.63% packet loss (right) for each GOP (50 frames).

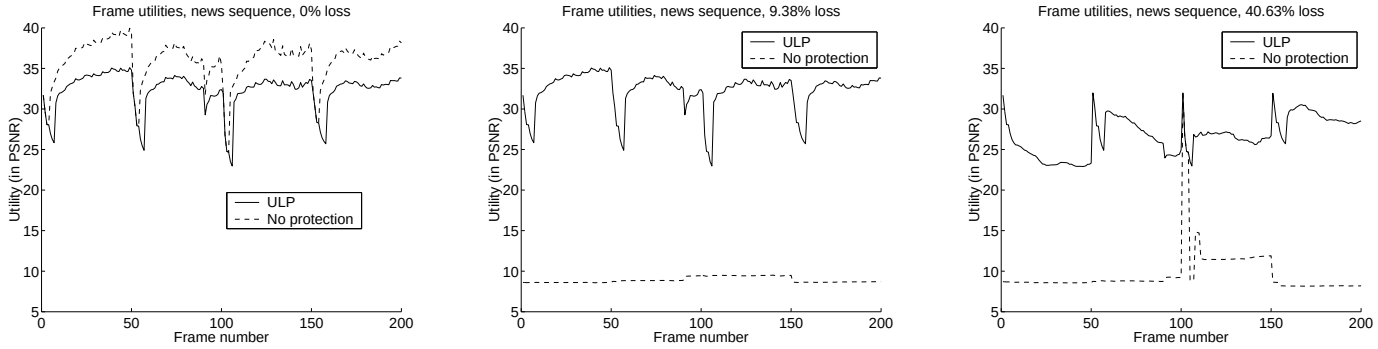

Fig. 10. Frame PSNR values for the news video sequence when experiencing no packet loss (left), 9.38% packet loss (center), and 40.63% packet loss (right) for each GOP (50 frames).

rate while ULP encodes at a lower bit rate and uses redundancy. When experiencing packet loss rates near to the mean loss rate (9.38% packet loss), we see that the ULP system is able to provide good quality while baseline H.263 has an unusable video sequence. Finally, at a very high packet loss rate (40.63% packet loss), ULP still manages to provide a good quality video sequence which is only slightly degraded. The results show that ULP successfully provides graceful degradation in the presence of increasing packet loss.

V. Conclusions and Future Work

The goal of this work was to show that we can provide graceful degradation of motion-compensated video quality in the presence of packet loss. We demonstrated that possibility, despite the various challenges involved with non-embedded, monolithic, motion-compensated video coders like H.263. We introduced a frequency-based ordering scheme to make the encoded bit stream appear embedded, then adapted the ULP algorithm to find the encoding bit rate and FEC allocation that provides the best expected utility for a given packet loss estimate. We compared our ULP system to baseline H.263 in the

presence of packet loss to illustrate that our system provides graceful degradation. We would like to mention that it is also possible to consider a system that uses FEC to provide equal loss protection (ELP), where all data are considered of equal importance and have the same protection. Such a system provides two levels of quality for its clients. A client receives no data until a sufficient number of packets are received (allowing the decoding of the original data). Once the client can decode the original data, the receipt of additional packets does not help the video quality. The advantage of an ELP system over our ULP system is its simplicity since it only provides one level of protection. However, the amount of protection still needs to be chosen. Preliminary results show that ELP performs pretty well.

One direction for future work is to study the execution time of various stages of the system operation to verify that our approach is suited to video-on-demand applications. Another direction for future work is to study the use of the H.263+ or H.263++ optional annexes. These annexes would make the ordering process more complex, but may lead to improved results. Finally, since an assumption of the ULP algorithm is that the source data are embedded, another direction is to compare the scheme introduced in this paper against applying ULP to naturally embedded video coders such as 3D-SPIHT [10].

REFERENCES

[1] Andres Albanese, Johannes Blömer, Jeff Edmonds, Michael Luby, and Madhu Sudan. Priority encoding transmission. *IEEE Transactions on Information Theory*, 42:1737–1744, November 1996.

[2] R. Aravind, M. Reha Civanlar, and Amy R. Reibman. Packet loss resilience of MPEG-2 scalable video coding algorithms. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(5):426–435, October 1996.

[3] J.C. Bolot. End-to-end packet delay and loss behavior in the Internet. In *SIGCOMM*, pages 289–298, September 1993.

[4] J.C. Bolot and T. Turletti. Adaptive error control for packet video in the Internet. In *Proceedings of ICIP*, volume 1, pages 25–28, 1996.

[5] Jill M. Boyce. Packet loss resilient transmission of MPEG video over the Internet. *Signal Processing: Image Communication*, 15(1-2):7–24, September 1999.

[6] Jill M. Boyce and Robert D. Gaglianello. Packet loss effects on MPEG video sent over the public Internet. In *ACM Multimedia*, pages 181–190, 1998.

[7] C. W. Chen, P. Cosman, N. Kingsbury, J. Liang, and J. W. Modestino (Guest Editors). Error-resilient image and video transmission. *IEEE Journal on Selected Areas in Communications*, 18(6):809–1144, June 2000.

[8] TMN 3.0 (H.263) codec. Released by the signal processing and multimedia group, University of British Columbia, http://spmg.ece.ubc.ca.

[9] Guy Côté, Berna Erol, Michael Gallant, and Faouzi Kossentini. H.263+: Video coding at low bit rates. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(7):849–866, November 1998.

[10] B. Kim, Z. Xiong, and W. Pearlman. Low bit-rate scalable video coding with 3D set partitioning in hierarchical trees (3D SPIHT). *IEEE Transactions on Circuit and Systems for Video Technology*, 10:1374–1387, December 2000.

[11] Christian Leicher. Hierarchical encoding of MPEG sequences using priority encoding transmission (PET). Technical Report TR-94-058, The International Computer Science Institute, November 1994.

[12] Adam H. Li, Surin Kittitornkun, Yu Hen Hu, Dong-Seek Park, and John D. Villasenor. Data partitioning and reversible variable length codes for robust video communications. In *Data Compression Conference*, pages 460–469, March 2000.

[13] Alexander E. Mohr, Eve A. Riskin, and Richard E. Ladner. Approximately optimal assignment for unequal loss protection. In *Proceedings of ICIP*, volume 1, pages 367–370, sep 2000.

[14] Alexander E. Mohr, Eve A. Riskin, and Richard E. Ladner. Unequal loss protection: Graceful degradation of image quality over packet erasure channels through forward error correction. *IEEE Journal on Selected Areas in Communication*, 18(6):819–829, June 2000.

[15] Vern Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*, 18(6):277–292, June 2000.

[16] Rohit Puri and Kannan Ramchandran. Multiple description source coding through forward error correction codes. In *Proc. 33rd Asilomar Conference on Signals, Systems, and Computers*, volume 1, pages 342–346, October 1999.

[17] Ramamohan Rao and Patrick Yip. *Discrete Cosine Transforms: Algorithms, Advantages, Applications*. Academic Press, 1990.

[18] CCITT recomendation H.263. Video coding for low bit rate communication, 1998.

[19] Yao Wang, Stephan Wenger, Jiangtao Wen, and Aggelos K. Katsaggelos. Error resilient video coding techniques. *IEEE Signal Processing Magazine*, pages 61–82, July 2000.

[20] Yao Wang and Qin-Fan Zhu. Error control and concealment for video communication: A review. *Proceedings of the IEEE*, 86(5):974–997, May 1998.

[21] Stephan Wenger, Gerd Knorr, Jörg Ott, and Faouzi Kossentini. Error resilience support in H.263. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(7):867–877, November 1998.