REAL-TIME INTERNET VIDEO USING ERROR RESILIENT SCALABLE COMPRESSION AND TCP-FRIENDLY TRANSPORT PROTOCOL

Wai-tian Tan and Avideh Zakhor

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720 USA
E-mail: {dtan,avz}@eecs.Berkeley.EDU

ABSTRACT

We introduce a point to point real-time video transmission scheme over the Internet combining a low-delay TCP-friendly transport protocol in conjunction with a novel compression method that is error resilient and bandwidth-scalable. Compressed video is packetized into individually decodable packets of equal expected visual importance. Consequently, relatively constant video quality can be achieved at the receiver under lossy conditions. Furthermore, the packets can be truncated to instantaneously meet the time varying bandwidth imposed by a TCP-friendly transport protocol. As a result, adaptive flows that are friendly to other Internet traffic are produced. Actual Internet experiments together with simulations are used to evaluate the performance of the compression, transport, and the combined schemes.

1 Introduction

Supporting low latency video communication over the Internet is an important yet challenging task. A few possible applications include video conferencing, tele-medicine, and interactive access to prerecorded videos stored in remote databases. There are two main requirements for Internet video communications: (1) bandwidth adaptability; (2) error-resilience. By bandwidth adaptability, we mean the ability to change the video bit rate according to network conditions. Lack of bandwidth adaptability results in non-adaptive streams with two sets of disadvantages. First, it leads to congestion collapse when the aggregate bandwidth of the video traffic exceeds network capacity.

Second, it competes unfairly with other adaptive traffic, such as TCP, which reduces transmission rate in face of network congestion. Existing approaches to flow control for video traffic over the Internet often require online encoding. By dynamically changing the parameters of the encoder, it is then possible to limit the data rate [1, 2]. For pre-encoded video, Jacobs and Eleftheriadis [3] have proposed a scheme in which selective DCT coefficients from MPEG streams are dropped to yield lower rates. However, both schemes are typically unable to match the desired rate instantaneously so that output buffering becomes necessary, adding to total latency. Yet another possible approach to flow control is transcoding, but it results in increased computational complexity and delay.

Lack of error resilience results in error propagation, and hence widely varying video quality as a function of time. Current approaches to mitigate the effects of error propagation include error control mechanisms at the transport level. This typically takes the form of retransmissions or forward error correction (FEC). Retransmission based error control methods often fail to be real-time, particularly when round-trip propagation delay is large. FEC schemes on the other hand, are often ineffective when losses are bursty [4]. In terms of error control, existing real-time Internet video applications either employ only intra block coding [5] which in general yields low compression efficiency, or prohibit the use of lost regions as reference for motion prediction [1], which does not extend to pre-encoded videos.

To address the above problems of flow and error control, one possible solution is to re-engineer the network to provide the necessary quality of service (QOS) guarantees via reservations and admission control [6]. Besides requiring changes in the current infra-structure, this approach also introduces call blocking when resources become temporarily over-subscribed. Furthermore, even when such QOS guarantees are widely available, it is likely to be more costly than the plain old best-effort service.

An attractive alternative is to use bandwidth scalable video with feedback rate control whereby transmission sources adjust their rates in response to changing network conditions. By bandwidth-scalability, we mean the ability to produce an embedded bit-stream which allows decoding at multiple rates. Bandwidth adaptation is then easily achievable for pre-encoded sources by simple filtering operations. This would solve the flow control problem while the error control problem can be solved by further requiring the compression method to be resilient to packet losses. In this paper, we will propose one such compression method based on 3D subband coding. 3D-subband decomposition is preferred to motion compensated residual coding (MCRC) because MCRC uses previously coded frames as reference for coding future frames, introducing data dependency and thus error propagation when the reference frames are lost. In addition, schemes employing 3D subband decomposition have the potential of generating embedded representations with fine granularity of available bit rates [7]. In contrast, scalable compression schemes based on MCRC either employ multiple prediction loops whereby precluding fine scalability due to the prohibitively high complexity, or suffer from drift or compression efficiency when motion vectors from base layer is used for coding at higher rates. Furthermore, the computational requirements of 3D subband decomposition is considerably lower than motion estimation, thereby enabling real-time encoding.

Taubman and Zakhor [7] have recently proposed a 3D subband coding scheme that allows decoding at many finely grained rates without sacrificing compression efficiency. In this paper, we modify the basic approach in [7] to enable real-time, software-only encoding and decoding. This is achieved by replacing the arithmetic coding algorithm of [7] by hierarchical block coding. We also propose new data partitioning and packetization schemes in order to achieve error resilience. This is accomplished by imposing the requirement to produce individually decodable and equally important packets. Because the proposed compression method is finely scalable, it is compatible with virtually any flow control algorithms. However, in order to be fair to TCP traffic, we have chosen to use a TCP-friendly transport protocol to show the effectiveness of our scheme for real-time video transmission over best effort packet networks such as the Internet.

The remainder of this paper is organized as follows. We will describe and evaluate the proposed compression method in Sections 2 and 3. A TCP-friendly transport protocol will be described in Section 4. Experimental results over the Internet for the combined scheme are presented in Section 5. Our results show that the proposed scheme competes fairly with TCP for bandwidth and has significantly smaller distortion under loss compared to schemes that produces inter-dependent packets.

2 Error-Resilient, Bandwidth-Scalable Video Compression

Traditionally, compression algorithms are designed for relatively error-free environments. In particular, Shannon's joint source-channel coding theorem predicts that there is no loss in optimality by considering source and channel coding separately assuming a stationary and memoryless channel with infinite delay and complexity. However, given the real-time constraint, the non-stationarity of the Internet, and the limited capabilities of real systems, compression schemes that tailor to the characteristics of the channel may provide better performance. For example, bandwidth scalable video compression schemes are often designed with transport prioritization in mind and produce packets that are inter-dependent. In particular, SNR-scalable video compression algorithms [7, 8, 9] produce packets that are *linearly* dependent, i.e., for every K frames N packets are produced so that if packet i is lost, error propagation would prevent decoding of packets $i+1$,...,N. While this would work well in ATM-type networks where different end-to-end quality of service guarantees can be specified for different flows, it is unsuitable for Internet transmissions because the lack of prioritized transport causes packet losses to appear random, resulting in large variability in received video quality. To see why linearly dependent packets yield large variability under random loss, consider independent packet reception rate of p. The probability that we can decode exactly i packets out of a total of N transmitted packets is then $(1-p)p^i$ for $i \neq N$, and p^N for $i = N$, a bimodal distribution that is geometric but with a tall spike at $i = N$. The distribution for $N = 20$ and $p = 0.9$ is shown in Fig. 1. Over 70% of the time we can either decode all 20 or at most 6 packets, resulting in large variability in the quality of the received video.

In this section, we will describe a bandwidth scalable compression scheme that produces individually decodable packets of equal importance. The scheme is based on 3D subband decomposition and data partitioning in the subband coefficient domain, which provides error resilience, and progressive quantization which provides bandwidth-scalability. Hierarchical block coding technique is used for speed and compression efficiency.

2.1 Error Resilience

To eliminate error propagation, we need every packet to be individually decodable. One way to achieve this is to employ a forward decomposition of the source material into M *components* and then compress each component independently to form a packet. Each packet can then be decoded to a co-image where the sum of all co-images form the original images.

There are many such decompositions for still images. One example is the polyphase decomposition which takes every M consecutive pixels and distributes one pixel to every component. Each component then would clearly be individually decodable and approximately of equal importance. This scheme suffers from low compression efficiency. Another approach is to use block based coding in the pixel domain. However, when one packet contains all information about a spatial location, its loss will cause all information in that location to be lost. Yet another approach is to use subband decomposition to divide the source into subbands that can be compressed independently. However, the DC-subband contains most of the energy for natural images. If each subband goes into a packet, this skewness would cause large variability in decoded picture quality under lossy conditions.

To overcome the problems of the above approaches, we use an alternative data partitioning scheme for subband based compression: instead of making each subband a component, we partition each subband into an equal number of coefficient blocks. Each coefficient block in a subband carries information about some localized region in the original frames. The components are then formed by grouping from each subband, equal number of coefficient blocks that correspond to different spatial regions of the source. As an example, Fig. 2 shows the formation of one component out of a total of nine. Since there are seven subbands, it would take at least seven packet losses to completely eradicate a particular spatial region. Conversely, a single packet loss contributes to at most one frequency band in any spatial location.

To extend the framework from still image to video, one possible way is to use 2D subband decomposition with motion compensation. However, since motion compensation does not perform well when required to produce finely scalable video, and also introduces error propagation, a scheme based on 3D subband coding is used [7, 8]. A component then consists of coefficient blocks of different spatial locations from the set of spatio-temporal subbands.

2.2 Real-time Implementation

In this section, we will describe the application of hierarchical block coding (HBC) [8] to real-time scalable video based on 3D subband coding. HBC is a fast and efficient way to exploit correlations between bit planes of subband coefficients and is motivated by the fact that arithmetic coding based schemes previously used for scalable video coding [7] are often too compute-intensive for real-time applications. For example, in decoding video of size 320×240 coded at 1.5 $Mbps$ using the scheme of [7], it is found that over 70% of the overall complexity can be attributed to arithmetic coding.

In coding every bit-plane, we maintain a *significance map* which is a binary map indicating whether each coefficient is significant (i.e. non-zero) if we stop at the current bit-plane. When a coefficient is significant in a certain bit-plane, we will transmit its *value* uncoded. The significance map with the values are sufficient to reconstruct the bit-planes.

We code the significance map of the first bit-plane with significant elements using a Quad-tree based approach [10]. Because significance maps for subsequent bit-planes can only grow, we exploit the fact to our advantage and skip information that can be inferred from previous significance maps. An example for a 4×4 block is illustrated in Fig. 3. The codeword for the first significance map starts with an “1” indicating that the 4×4 block has significant values. The next 4 bits “1100” shows whether each of the 2×2 sub-blocks contain significant values, starting with the top-left sub-block and counting clockwise. For each 2×2 sub-block containing significant values, 4 bits are added to indicate which 1×1 sub-blocks are significant. To code the significance map for the next bit-plane, we observe from the previous significance map that the 4×4 block as well as the first two 2×2 sub-blocks contains significant values. Those information can therefore be skipped, improving compression performance. The effectiveness of HBC is based on the fact that coefficients in the high frequency subbands are often quantized to zero so that quad-tree based approaches which assign short codewords to blocks with many zeroes are efficient.

A frequently used operation for quad-tree coding is testing whether a block contains significant values. Since the significance map is binary, it takes only 16 bits to represent the significance map of a 4×4 block. Thus, it takes only one 16-bit compare to check if a 4×4 block is significant and only two 32-bit compares for an 8×8 block. As a result, HBC admits a low complexity implementation.

2.3 Bandwidth Scalability and Packetization

In this section, we will describe packetization of compressed information into a rate-scalable structure. Rate-scalability is desirable in real applications not only because of its usefulness in flow control, but also because it provides complexity scalability whereby receivers with limited decoding power can reduce data that is being decoded.

Instead of achieving rate-scalability by skipping components, we compress each component into a multi-rate representation. Rate-scalability is then obtained by modulating the size of the compressed representation. The scheme is depicted in Fig. 4 where $R_3 > R_2 > R_1$. This is in contrast to traditional packetization of layered video where the inter-dependency and varying importance of the packets often require transport prioritization [11, 9].

Each component contains coefficient blocks which are independently compressed using progressive quantization followed by hierarchical block coding [8] described in more details in Section 2.2. When subband coefficients are represented using signed-magnitude format, progressive quantization reduces to bit-planing with an appropriate choice of quantizer. Successive bit-planes of a coefficient block are then inter-coded to produce small codewords which provides fine granularity for layered packetization.

In Fig. 4, in order to produce good quality for all the rates R_1, R_2 and R_3, we wish packetization to be carried out so that data is roughly packed in the order of their importance. Because hierarchical block coding exploits correlations between bit-planes of the same coefficient block, the more significant bit-planes must be packed before the less significant ones. Given that constraint, our packetization strategy then tries to form an ordering of all the bit-planes in all coefficient blocks in a component. However, there is in general no bit-plane packing scheme that simultaneously minimizes distortion at all rates. One practical heuristic is the greedy approach: at any one time, we examine every coefficient block in a component to find the most significant bit-plane that has not already been packed. We then choose from the set of most significant bit-planes the one that offers the most expected energy reduction per bit, and pack the entire bit-plane. However, coding the decision of the packing order as described above incurs large overhead since one decision is needed per bit-plane per coefficient block. Instead, we take an approximate approach as follows. Every time the encoder finishes subband analysis, it computes and parametrizes the energy distribution of the subbands into 8 parameters totaling 32 bits which can then be compactly transmitted per packet. The parameters and their explanations are given in Table. 1 and its caption. Both the decoder and the encoder then reconstruct from the parameters an approximate energy distribution which is used for estimating the next bit-plane that would yield the best rate-distortion performance. After a bit-plane is compressed, its actual energy contribution is used to update the approximate energy distribution both at the decoder and the encoder. The parameters described above indicates the relative importance of different subbands averaged over all spatial locations. Since each component contains only one coefficient block corresponding to only one spatial location in each subband, the relative importance of the coefficient blocks within a component may be different from that of the subbands. As a result, the update is necessary to exploit the spatial variation of energy within a subband.

A schematic diagram for the procedures at the encoders and decoders are shown in Figs. 5 and 6 respectively.

3 Performance of Resilient, Scalable Compression

In this section, we evaluate the performance of the proposed compression algorithm in terms of its compression and speed performances. We will also compare its performance against several other compression methods when subjected to random packet loss.

3.1 Compression Performance

Since the proposed compression method is required to produce independently decodable packets and be bandwidth scalable, it is necessarily more restrictive than compression schemes that do not. In particular, it lacks the motion models of MPEG and H.263 and does not exploit correlation between subbands, as is done in SPIHT [12] or Shapiro's embedded zero-tree scheme [13]. Furthermore, correlation between coefficient blocks within the same subband is also deliberately ignored. As a result, there is in general a decrease in compression efficiency when there is no packet loss.

We compare PSNR for two sequences: "Raider of the Lost Ark", and "Mother and Daughter", with 600 and 300 frames respectively. The *Raider* sequence is a high motion fighting scene at 24 fps whereas *Mother* is a low motion head and shoulder sequence at 30 fps. The image size for both sequences is 320×224. We perform PSNR comparisons of our scheme with H.263 and MPEG-1 using group of picture (GOP) sizes of 4 and 8. We do not consider longer GOP patterns because our intended application is transmission over lossy networks where long GOP patterns are more susceptible to error propagation due to packet losses.

An important issue to deal with when comparing codecs is rate control. Our approach is to ensure that all frames are coded by each codec and that the overall bit rates are identical. Specifically, we generate the H.263[1] results using fixed quantization parameters (QP) across all frames. The resulting bit-rate is then controlled by changing QP in the range [1, 31]. When the

[1] The software from the University of British Columbia is used.

desired bit-rate cannot be reached by changing QP alone, the input sequences are subsampled to result in lower frame rates. This occurs when we try to compress *Raider* using GOP sizes of 4 and 8 and *Mother* using GOP size of 4 at a rate below 128 *kbps*. The MPEG results are generated using an MPEG-1 software [14] with 10 slices per frame and exhaustive search. Target bit-rates are set so as to match those produced by H.263. The MPEG-1 software dynamically adjusts the compression quality of each frame using its own rate control mechanism, which is essentially based on a leaky bucket algorithm.

For our proposed scheme, we use the same number of bits per GOP instead of more elaborate rate control algorithms. This is because for our intended networking applications, the number of bits used per GOP is dictated by the flow control algorithm to be described later in Section 4. Given a fixed bit budget for a GOP, we propose two ways to distribute bits among the components. The simpler approach is the constant bit rate (CBR) allocation in which each component receives the same number of bits. With variable bit rate (VBR) allocation, the compressed bits of each component are divided into fixed size cells and distortion tags representing the energy reduction of each cell are stored. A greedy algorithm for energy reduction is then used to allocate bits among the components. Since the distortions tags are not needed to decode the components but are only used to perform bit allocation, they do not need to be transmitted for applications such as video playback over networks where the video server is performing the rate allocation and filtering. We use 40 bytes for a cell and 2 bytes for a distortion tag, which are found experimentally to work well. This yields a 5% overhead in using VBR allocation.

The results of comparing three codecs using two sequences are shown in Table. 2. When using our method, one embedded bit-stream is produced for every frame rate. Subsets of the bit-stream are then extracted to decode a video at the same frame rate but at a different bit rate. For H.263 and MPEG, a separate bit-stream is generated for every frame rate and every bit rate. We see that the performance of the proposed scheme is more or less identical in the CBR and VBR cases with the largest differences being at the lower bit rates. H.263 outperforms MPEG-1 and our proposed scheme for almost all the rates that it can be applied to[2]. We observe that the gap between H.263 and our proposed scheme increases as the bit rate and GOP size increases. Specifically, for GOP size of 4, the gap is in the range of (-0.7, 0.3), (-0.2, 0.7), (1.2, 1.4) and (0.9, 1.1) dB for bit rates about 100, 230, 500 and 1500 $kbps$ respectively. Similarly, for GOP size of 8, the gap is 0.2 dB at about 100 $kbps$, and is in the range of (1.9, 2.5), (2, 2.4) and (1.8, 2) dB for bit rates about 200, 500, 800 and 1300 $kbps$ respectively. The mild performance penalty is expected due to the requirements to produce bandwidth scalability, independently decodable packets for error resilience, and real-time implementation issues.

Our proposed compression scheme currently does not address frame rate scalability in the sense that the reconstructed video from a single embedded bit stream has the same frame rate regardless of the bit rate. This limits the inherent use of the scheme at very low bit rates such as 10 - 20 $kbps$. We are currently investigating ways of introducing frame rate scalability in addition to bit rate scalability [15].

3.2 Performance under Loss

We next compare the performance of our proposed scheme (P) under 5% simulated packet loss. Besides scheme (M), two other schemes are considered: (S) 3D subband coding where every packet contains one subband and, (T) the scalable compression scheme of [8] which produces linearly dependent packet. Packetization of MPEG bit-stream is performed so that no slice is split across packets unless the slice size exceeds packet size [16]. The results are shown in Fig. 7. We see that only scheme (P) enjoys a uniform high quality of received video. Even though packets under scheme (S) are independent, the skewness in their energy causes large variability in received video quality. Schemes (T) and (M) suffer from error propagation and show even greater variability. In particular, as a consequence of motion compensation in MPEG, we can see from Fig. 7 that

[2]For $Mother$, no quantization step size yields bit rates between 2-4 $Mbps$

errors have longer "tail" with longer GOP pattern. For scheme (P), simple error concealment is performed on the DC-subband where every missing coefficient is estimated by the average of its surviving neighbors. Similar concealment techniques are not applicable to scheme (T) under which the all spatial locations in the DC subband are compressed and transmitted together.

3.3 Speed Performance

For the 600 frames *Raider* sequence of size 320 $\times$ 224 sampled at 24 fps, the encoding speeds on a 400 MHz Pentium are given by 34 to 21 frames per second in the range of 200 *kbps* to 2 *Mbps* respectively. The decoding speeds in the same range varies from 60 to 25 frames per second. The reported times exclude disk access and display time [8]. We see that real-time software-only encoding and decoding are possible on current computers.

Fig. 8 shows the speed comparison of our proposed scheme and that of an MPEG-1 software [14] on an 170 MHz Ultra-Sparc. For the *Raider* sequence at bit rates ranging from 500 *kbps* to 3 *Mbps*, our encoding proceeds at 18.2 to 11.2 fps and decoding proceeds at 17.5 to 12.5 fps respectively. On the same machine MPEG encoding proceeds at 0.4 to 1.6 fps using exhaustive and logarithmic search respectively. The reported speed excludes disk access and display times. We see that real-time software-only video encoding and decoding of our proposed scheme is possible on today's computers.

3.4 Visual Quality

Fig. 9(a) shows original "Lena" image at 512×512. Five levels of spatial decomposition, using a 5/3-tap biorthogonal filter, are performed on the image to get 16 subbands. Each subband is then divided into 256 coefficient blocks. The largest coefficient block is 16×16 while the smallest is 1×1. We form 256 components and compress each component using layered block coding method described in [8] to get 256 packets which are then subjected to a 22% random packet loss. The image reconstructed from the survived packets is shown in Fig. 9(b). No error concealment has been applied to the image. We see that errors are dispersed over a wide support and while the image is uniformly blurred and the total energy is diminished, all features of the original image are still visible. Furthermore, even though block based coding is employed, there are no sharp discontinuities because data partitioning is performed in the frequency domain instead of the pixel domain.

4 TCP Friendly Transport Protocol

We next consider the problem of designing a flow control algorithm for unicast video communication. To achieve fair sharing of networking bandwidth, it is desirable that a transport protocol share resources fairly with multiple instances of itself and with TCP, the dominant source of Internet traffic. TCP cannot be used directly for video transport because its reliability is achieved at the expense of time varying delay and throughput. Fig. 10 shows the end-to-end delay in seconds when 300 seconds of video material generated at 600 and 630 *kbps* respectively are transmitted back to back from Toronto to Berkeley using TCP at noon time. In both cases, since the delay experienced at the end of the experiment is close to zero, the long term average throughput of the TCP connections must exceeds the data rates. However, we observe that the instantaneous end-to-end delay can still be significant, up to 6 seconds in our experiments.

There are two sources of latency in TCP: (1) backlog of data when throughput temporarily drops below the data generation rate; (2) retransmission latency. To achieve low latency objective, we eliminate backlog of data by (a) filtering a scalable video bit-stream to meet the instantaneous throughput and (b) not performing retransmissions. One way to ensure fair competition with TCP for bandwidth is to match exactly the traffic pattern of the adaptive window flow control algorithm of TCP [17]. However the sending window size of TCP is usually halved on determination of a single packet loss, resulting in traffic pattern with abrupt changes. Instead of matching the TCP traffic pattern exactly and instantaneously, a more relaxed form of fairness can be obtained by matching the TCP throughput on a macroscopic scale. On a lightly loaded network, the TCP sending window has the same size as the amount of buffer space (B) available in the receiver, yielding an average throughput (T) of:

$$T = \frac{B}{RTT} \tag{1}$$

where RTT is the dynamic estimate of the round-trip time. When the window size is changing due to the congestion avoidance algorithm [17, 18], Mahdavi and Floyd [19], Mathis *et.al.* [20] have derived expressions relating TCP throughput to the packet loss rate (p):

$$T = k \; \frac{MSS}{RTT \times \sqrt{p}} \tag{2}$$

where MSS, the maximum segment size, is the amount that the TCP sending window increases per RTT when there is no packet loss. k is a constant between 0.7 [20] and 1.3 [19] depending on the particular derivation of Equation 2.

A non-TCP flow sharing the same path as a TCP flow will experience and measure similar network conditions. By choosing B and MSS, and measuring p and RTT appropriately, it is possible for the non-TCP flow to estimate the throughput of the TCP flow and subsequently control its transmission rate accordingly. Even though B and MSS are implementation dependent, they are constants and do not play an important role in the dynamic evolution of throughput. The dynamic estimates RTT and p however, must be calculated in accordance with actual TCP implementation.

In TCP, a new estimate of RTT is obtained per RTT using a coarse clock that typically has a resolution as low as 500 ms [18]. These coarse estimates are then used to update the average round trip time ($srtt$) as well as the mean deviation using first order autoregressive estimators [17]. These quantities are in turn used to determine the timeout in which an unacknowledged packet is declared lost [18]. To achieve close correspondence with TCP, the same estimator is kept to determine timeout. Packet losses within one RTT are considered as a single congestion event and p is determined by [20]:

$$p = c/D \tag{3}$$

where c is the number of congestion events in an observation window while D is the amount of data in units of MSS transmitted in the same observation window. Since the coarse granularity of the TCP clock inadvertently causes much fluctuation in the measurement of RTT, the throughput estimated by Equations 1, 2 will show large variability. Instead, the throughput is calculated using an additional RTT estimate that is measured using an accurate clock.

Given estimates of TCP throughput, we construct TCP-friendly rate based transport protocols (TFRP) by first choosing a desired transport packet size. A source attempts to make a fixed number of transmissions per second and the size of each transmission is modulated by the flow controller so as not to exceed throughputs given by Equations 1 and 2. We reduce transmission overhead by combining consecutive small transmission attempts if the combined size does not exceed the desired transport packet size. Larger packets are transmitted as is and without fragmentation.

5 Internet Experiments

In this section, we will describe Internet experiments using the proposed compression scheme in conjunction with TFRP. We will first show that uncontrolled flows may lead to severe congestion so that rate control algorithms should always be used. We then demonstrate the fairness of TFRP when competing with TCP for bandwidth. The results of simultaneous transmissions of the proposed video scheme (P) and its non-resilient counterpart (T), both using TFRP will then be presented. Finally, we will compare the proposed compression scheme against scheme (T) with adaptive FEC employed. We also experimented with using FEC with the proposed scheme.

Because video traffic consumes considerable network resources, non-adaptive flows can cause severe congestion. Fig. 11 shows the results of a non-adaptive flow from Hong Kong to Berkeley carrying video compressed using Scheme (T) at 1 Mbps. The video source is a 600 frames *Raiders* sequence encoded at 12 fps and is being transmitted in a loop. Every 4 frames is packetized into linearly dependent packets to provide low latency and limit the effects of error propagation. We see that the high transmission rate induces much congestion with packet loss rate averaged about 33%. A very high average Mean Square Error (MSE) of 777 is sustained despite the high data rate.

Since scheme (T) is bandwidth-scalable, we can apply TFRP to avoid transmission at excessive rates. The result is shown in Fig. 12. Comparing the results in Fig. 12 with those of Fig. 11 corresponding to uncontrolled flow, we see that much of the congestion is eliminated even though occasional peaks are still observed due to the linear dependency of packets under scheme (T). The average MSE for the experiment is 67.

To achieve flow control in a TCP friendly manner, we propose the use of TFRP. Figs. 13 and 14 show simultaneous transmission of TCP and TFRP from Hong Kong to Berkeley and Toronto to Berkeley respectively. A desired transport packet size of 500 bytes is chosen, which roughly parallels the transport packet size of most TCP implementations, and 70 packets are generated per second with an observation window of $32 \times RTT$ to measure p. An initial estimate of 5% is used for p which is eventually replaced by the estimate of Equation 3 after $32 \times RTT$. Both channels are congested, showing an average loss rates of 1.4% and 0.5% respectively, with occasional bursty losses as high as 30%. The average throughputs for TCP are 241 and 414 $kbps$ respectively while that of TFRP are 239 and 370 $kbps$ respectively. This demonstrates that TFRP can coexist with TCP. We see that while the protocols show similar throughput in both cases, TFRP shows significantly less variability than TCP due to the use of an averaging window to measure p. This is particularly useful for real-time multimedia transport where high and frequent quality fluctuations are not desirable. The use of a larger window will provide further smoothing, but at the expense of longer reaction time to changes at network conditions.

Fig. 15 shows the bit-rate trace of TFRP from Toronto to Berkeley. Initially the throughput stabilizes at around 700 $kbps$ until two more TFRP instances are started 80 seconds into the experiment. This causes temporary congestion before each instance of the protocol adjusts its transmission rate to around 480 $kbps$. There is little actual packet loss except when the protocol is reacting to the change in network conditions. This indicates that TFRP is successful in sharing the available bandwidth to avoid excessive losses.

We next compare the simultaneous Internet transmission of scalable video compressed under schemes (P) and (T). TFRP is used to carry both streams. The packet loss rate is measured in 1/3 second intervals and is shown in the top graph of Fig. 16. There are moderate and bursty packet losses, with an average around 3.8% and variation between 0 to 23%. We see that scheme (P) out-performs (T) significantly both in having a lower average MSE of 105 versus 244, and in having smaller variability. Visually, packet losses under (P) appear as blurred patches in different locations in different frames whereas in (T) we observe uniformly blurred frames. However, the data dependency between packets in (T) causes high temporal variability in quality, resulting in oscillations between high quality and very poor quality pictures. Such wild oscillations do not exist for scheme (P) since the packets do not depend on each other. As a result, scheme (P) produces much more visually pleasing decoded video than scheme (T).

5.1 Using Forward Error Correction with Non-resilient Video

We will next consider the incorporation of FEC to non-resilient compression methods. We will restrict our discussion to the comparison between schemes (P) and (T) because both are finely scalable and can be easily used in conjunction with TFRP and are similar in that both employ hierarchical block coding in addition to 3D subband analysis.

Fig. 17 shows the distortion-rate characteristics when the *Raider* sequence under schemes (P) and (T) are subjected to different simulated losses and with no FEC added. We see that the two schemes have comparable performance when there is no loss. However, the performance of scheme (T) deteriorates much faster than that of scheme (P) as loss rate increases. If we were to know the packet loss rates *a priori*, one possible way to improve the quality of scheme (T) transmission is to apply unequal error protection (UEP) whereby more important packets are protected more. Furthermore, since scheme (T) is bandwidth scalable, the UEP scheme can be performed without overall data expansion by reducing the number of informational packets to make room for redundancy or parity packet.

One common approach to achieving unequal error protection for wireless environments is to use Rate-Compatible Punctured Convolutional (RCPC) codes [21] under which it is possible to assign different levels of protection to different blocks of bits. Such schemes however, are relatively computation intensive. Instead, we will achieve unequal error protection for scheme (T) using packet replication [22].

Given a bandwidth constraint C, and a known packet loss rate p_{loss}, our proposed UEP scheme minimizes the expected distortion of scheme (T) video given by:

$$\mathbf{E}D = \sum_{i=1}^{M} p_i D_i \tag{4}$$

where p_i is the probability that all packets up to packet i are received while packet $i+1$ is lost. D_i is the expected distortion when the first i packets are decoded and is approximated by the 0 loss curve in Fig. 17. M is the least important packet that is being transmitted and is given by the bandwidth constraint:

$$\sum_{i=1}^{M} n_i R_i \leq C \tag{5}$$

where n_i is the number of copies that packet i is transmitted and R_i is the size of packet i. To reduce the number of variables, and hence computational complexity, we restrict all packets to have equal size; i.e. $R_i = R \quad \forall i$. This is a reasonable assumption since unequal packet size introduces considerable difficulties in actual implementation. A further assumption is that packet losses are independent. Under that assumption, p_i can be calculated from probability of packet loss (p_{loss}) according to the following formula:

$$ps_0 = 1.0 \tag{6}$$

$$p_i = ps_{i-1} \times p_{loss}^{n_i} \tag{7}$$

$$ps_{i+1} = ps_i - p_i \tag{8}$$

where ps_i denotes the probability of receiving the first i layers correctly.

Given the above assumptions, the only remaining variables for the constrained optimization problem are M and n_i. The optimization is performed using exhaustive search over all possible

M and n_i that satisfy Equations 4 and 5. The size of the search space however, is prohibitive; specifically, it is equal to the number of ways to distribute $N = \lfloor \frac{C}{R} \rfloor$ balls into M bins where M can range from 1 to N. We can reduce the size of the search space by enforcing that more important packets have at least the same level of protection as less important packets. This is an intuitively pleasing assumption for scheme (T) since loss of packet i would render packets $i+1$, $i+2$, ... useless. Mathematically, this translates into $n_i \geq n_j$ for $i < j$.

For our experiments, the packet size R is chosen to be 300 bytes to yield fine grained bandwidth-scalability. Decreasing R further will provide finer granularity at the expense of higher transmission overhead. The packet loss rate p_{loss} is approximated using a running estimate that is updated at least 3 times per second. The n_i are pre-computed using exhaustive search for 16 fixed loss rates ranging from 0.25% to 20%. Table 3 shows the size of the search space for various values of N, the total number of packets to send that is dictated by the flow control algorithm of TFRP, as well as the optimal values for n_i when $p_{loss} = 0.05$. When p_{loss} is increased to 0.1, the optimal values of n_i changes according to Table 4. As expected, at higher loss probability, the optimization algorithm opts to send fewer packets, but protects those few packets with more duplicates than at lower loss probability.

5.2 Using Forward Error Correction with Resilient Video

Similarly for our proposed scheme (P), given a fix rate to code a component, it is possible to reduce that rate to make room for piggy-backing low rate versions of other components. Following a formulation similar to that of Equations. 4 and 5, an optimal FEC scheme can be obtained for every assumed packet loss rate.

5.3 Results Using FEC

We carry out an actual Internet experiment again between Hongkong and Berkeley. The results are shown in Fig. 18. The average MSE for schemes (P) and (T) with FEC are 100.53 and 201 respectively. Both show an improvement over the case when FEC is not used. A comparison of

Figs. 16 and 18 indicates that FEC reduces the variability in the decoded video for scheme (T). However, scheme (T) with FEC still shows higher average distortion and larger variability compared to scheme (P) with or without FEC. The better performance of the error resilience scheme compared to that of the non-resilient scheme with FEC is due to the mismatch between the assumed and actual packet loss rates. Since only delayed channel estimates are available for the Internet, such mismatch is unavoidable. However, FEC schemes are typically optimized only for a fixed assumed loss rate and are sensitive to such mismatch. The proposed error-resilient compression scheme however, is designed without assuming a packet loss rate and therefore tends to perform better when the channel states are not precisely known.

6 Conclusion

In this paper, we described a low latency video transmission scheme consisting of a TCP-friendly flow controller with a bandwidth-scalable compression scheme that produces individually decodable packets. The compression scheme is found to admit real-time encoding and decoding with little loss in compression efficiency compared to MPEG-1 at no loss. Low latency is established by the elimination of buffering in flow control so that the overall latency is propagation delay plus decoding time. We perform simulated packet loss studies and find that the proposed compression scheme produces relatively constant video quality in face of packet losses as compared to MPEG and several subband-based packetization methods. We also perform actual Internet experiments comparing another bandwidth-scalable compression scheme (T) which is similar to the proposed compression scheme except with dependencies among packets. We find that the ability to independently decode packets reduces the resulting distortion significantly even after an adaptive FEC scheme is applied to protect the non-resilient bit-stream.

7 References

[1] T.Turletti and C.Huitema. Video-conferencing on the Internet. *IEEE/ACM Trans. Networking*, Vol. 4, No. 3, pp 340-351, June 1996.

[2] H.Kanakia, P.Misha and A.Reibman. An Adaptive Congestion Control Scheme for Real Time Packet Video Transport. *IEEE/ACM Trans. Networking*, Vol. 3, No. 6, pp 671-682, December 1995.

[3] S.Jacobs and A.Eleftheriadis. Streaming Video using Dynamic Rate Shaping and TCP Congestion Control. submitted to *Journal of Visual Communication and Image Representation*, January 1998.

[4] I.Cidon, A.Khamisy and M.Sidi. Analysis of Packet Loss Processes in High-speed Networks. *IEEE Trans. Info. Theory*, Vol. 39, No. 1, pp 98-108, January 1993.

[5] S.McCanne, M.Vetterli and V.Jacobson. Low-complexity Video Coding for Receiver-driven Layered Multicast. *IEEE J. Selected Areas in Comm.*, Vol. 15, No. 6, August 1997.

[6] L.Zhang, S.Deering, D.Estrin, S.Shenker and D.Zappala. RSVP: A New Resource Reservation Protocol. *IEEE Network Magazine*, Vol. 7, No. 5, pp 8-18, September 1993.

[7] D.Taubman and A.Zakhor. Multirate 3-D Subband Coding of Video. *IEEE Trans. Image Proc.*, Vol. 3, No. 5, pp 572-88, September 1994.

[8] W. Tan, E. Chang, and A. Zakhor. Real Time Software Implementation of Scalable Video Codec, *Proc. ICIP*, Vol. 1, pp 17-20, September 1996.

[9] D. Taubman and A. Zakhor. A Common Framework for Rate and Distortion Based Scaling of Highly Scalable Compressed Video. *IEEE Trans. CSVT*, Vol. 6, No. 4, pp 329-354, August, 1996.

[10] M. Kunt. Block Coding of Graphics: A Tutorial Review. *Proc. of IEEE*, Vol. 68, No. 7, July 1980

[11] M.Garrett and M.Vetterli. Joint Source/Channel Coding of Statistically Multiplexed Real-Time Services on Packet Networks. *IEEE/ACM Trans. on Networking*, Vol. 1, No. 1, pp 71-80, February 1993.

[12] A.Said and W.Pearlman. A New, Fast, and Efficient Image Codec based on Set Partitioning in Hierarchical Trees. *IEEE Trans. CSVT*, Vol. 6, No. 3, pp 243-250, June 1996.

[13] J.Shapiro. Embedded Image Coding using Zerotrees of Wavelet Coefficients. *IEEE Trans. Sig. Proc.*, Vol. 41, No. 12, pp 3445-3462, December 1993.

[14] K.Patel, B.Smith, L.Rowe, Performance of a Software MPEG Video Decoder, *Proc. of the ACM Multimedia '93* pp 75-82.

[15] W.Tan and A.Zakhor. Coding and Transport of Low Bit-rate Scalable Video, Submitted to *ICIP 1999*.

[16] D.Hoffman, G.Fernando and V.Goyal. RTP Payload Format for MPEG1/MPEG2 Video. RFC 2250, January 1998.

[17] V.Jacobson. Congestion Avoidance and Control. *Proc. SIGCOMM '88*, pp 314-219, Stanford, CA, August 1988.

[18] W.Stevens. *TCP/IP Illustrated*, Vol. 1, Addison-Wesley, 1996.

[19] J.Mahdavi and S.Floyd. TCP-Friendly Unicast Rate-Based Flow Control. Technical note sent to the end2end-interest mailing list, http://www.psc.edu/networking/papers/tcp_friendly.html, January 1997.

[20] M.Mathis, J. Semke, J.Mahdavi, T.Ott. The Macroscopic Behavior of the TCP Congestive Avoidance Algorithm. *CCR*, Vol. 27, No. 3, July 1997.

[21] J.Hagenauer. Rate-Compatible Punctured Convolutional Codes (RCPC Codes) and their Applications. *IEEE Trans. Comm.*, Vol. 36, No. 4, April 1988.

[22] Y.Wang and Q.Zhu. Error Control and Concealment for Video Communication: A Review. *Proc. IEEE*, Vol. 86, No. 5, May 1998.

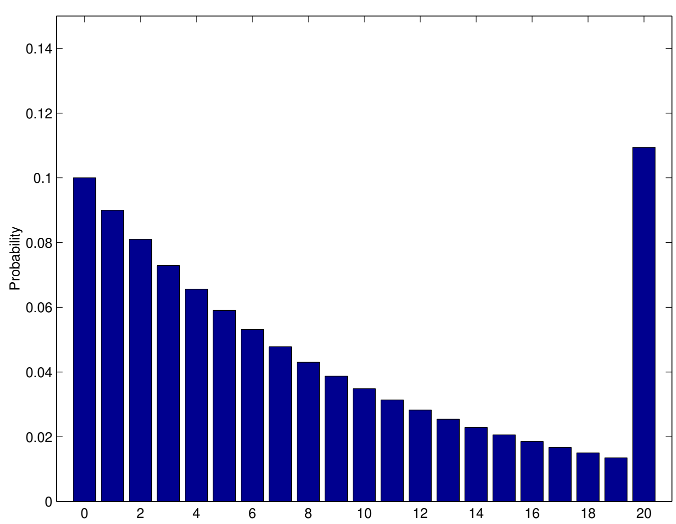

Figure 1: Distribution of the number of decoded packets for 20 linearly dependent packets at 10% loss

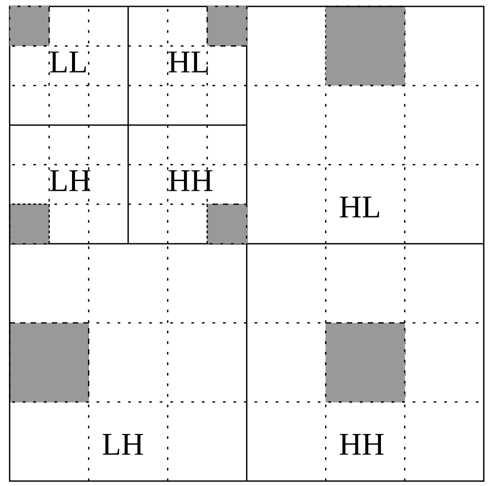

Figure 2: Grouping coefficient blocks from different subbands to form a component.

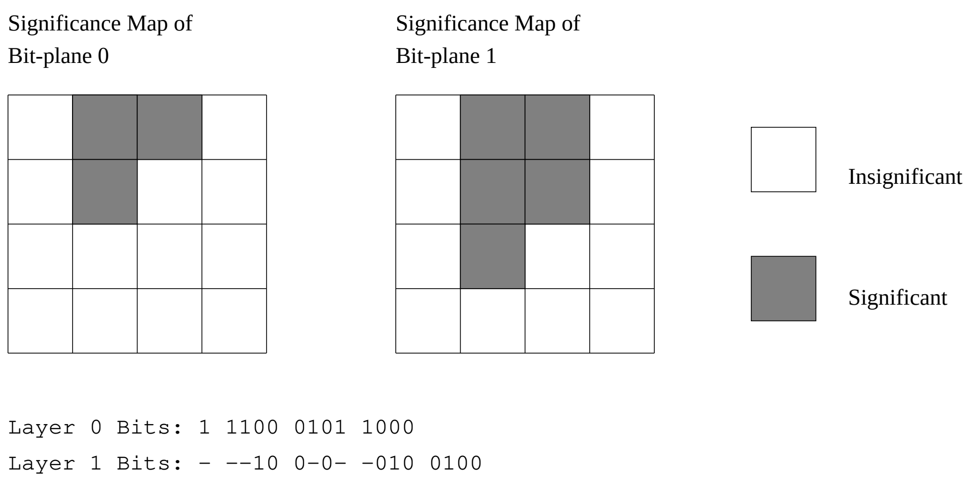

Figure 3: Coding significance maps using hierarchical block coding. “-” indicates values that can be deduced.

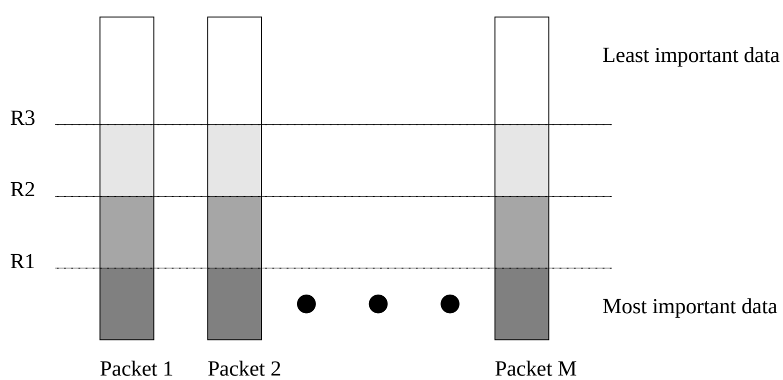

Figure 4: Achieving rate-scalability

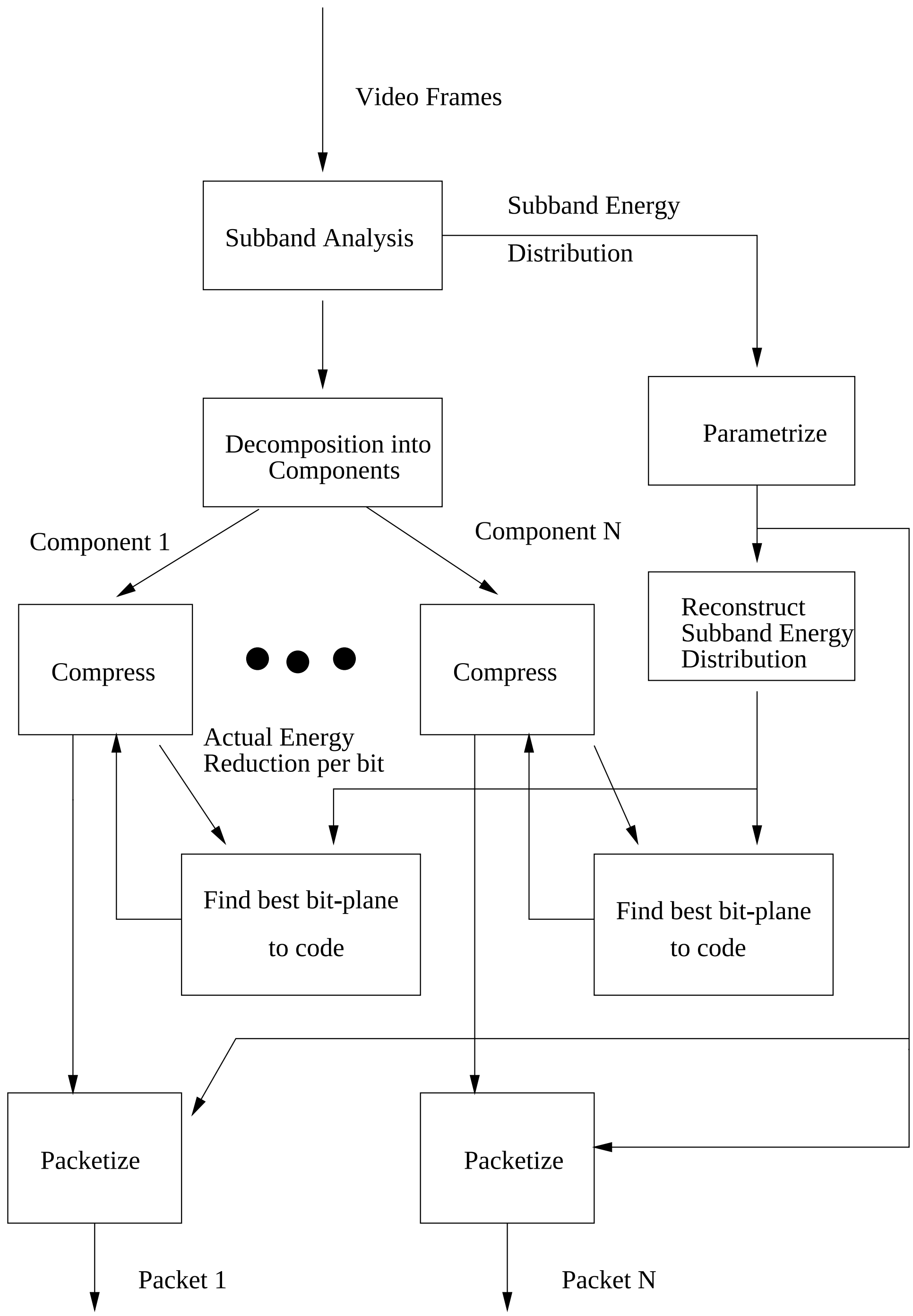

Figure 5: Encoding Procedures

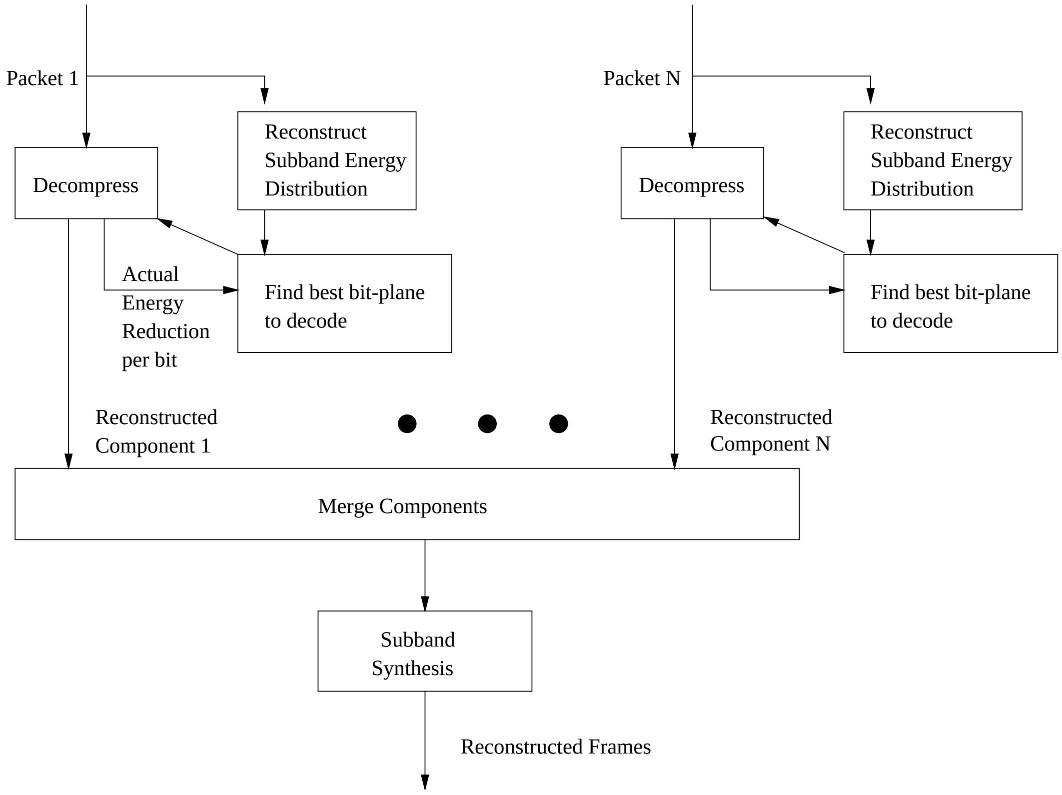

Figure 6: Decoding Procedures

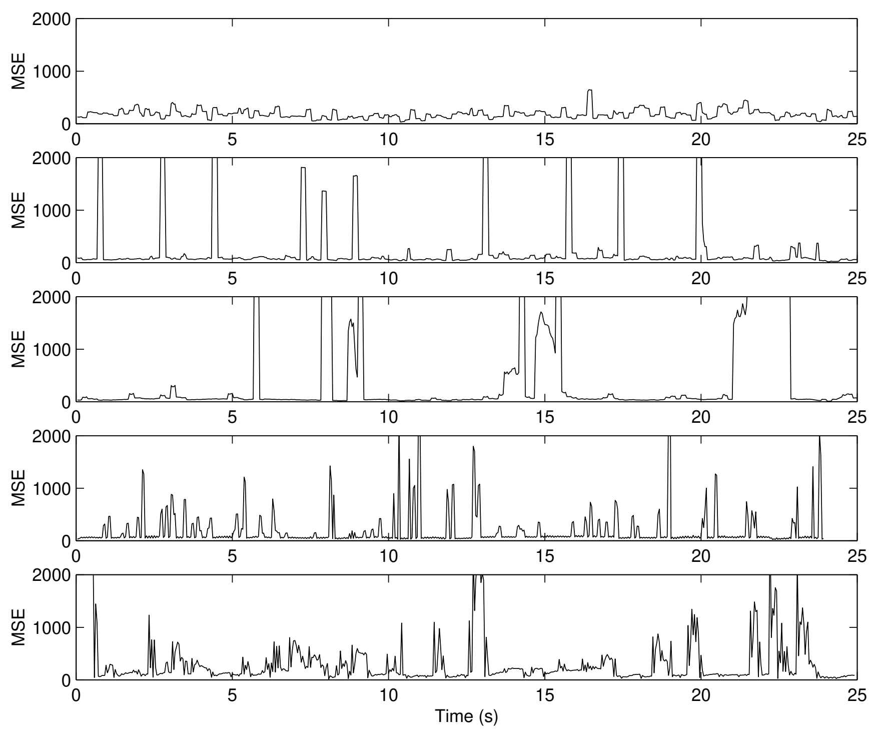

Figure 7: Variability of quality at 5% simulated random packet loss. From top to bottom: (P), (S), (T), (M) with GOP 2, (M) with GOP 15.

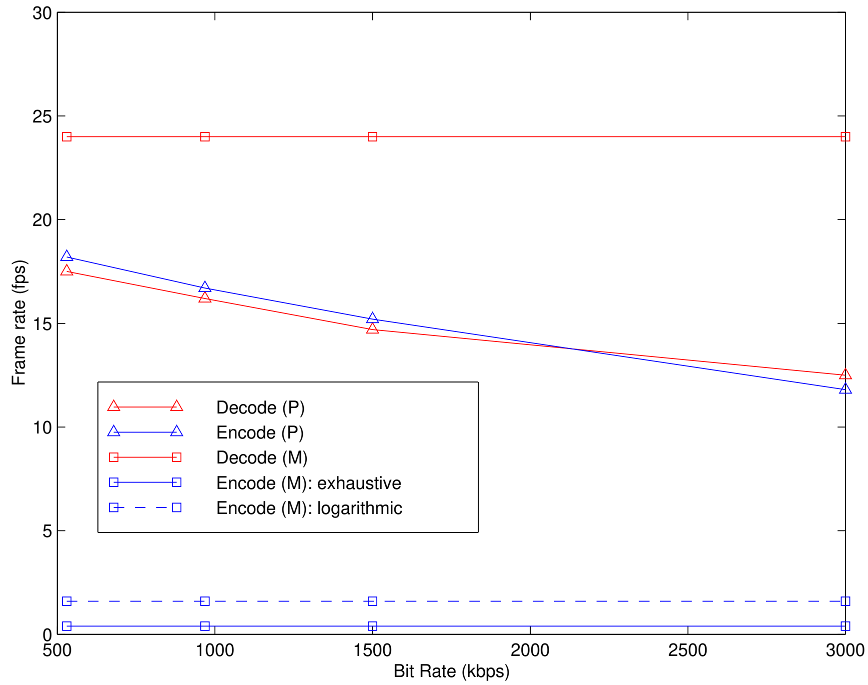

Figure 8: Speed comparisons of the proposed scheme (P) and MPEG-1 (M) on an 170 MHz Ultrasparc. “Exhaustive” and “logarithmic” refer to the different strategies in motion estimation of MPEG

(a) (b)

Figure 9: Original Lena (a) and Lena at 0.3 bits/pixel and 22% loss (b).

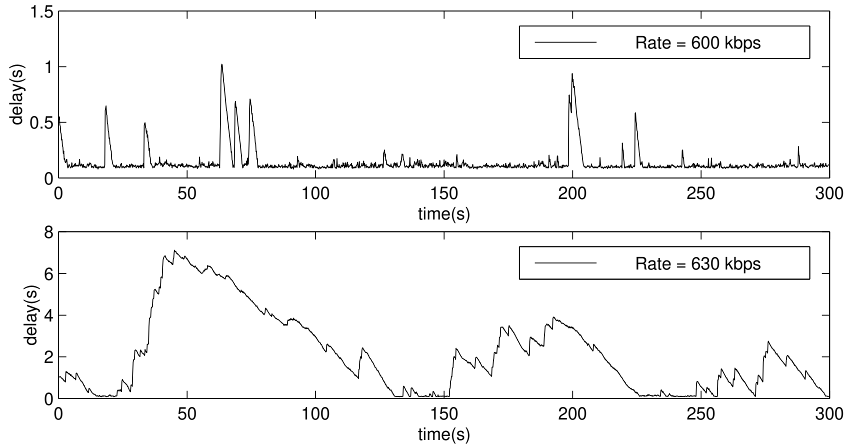

Figure 10: Delay for video transport using TCP from Toronto to Berkeley (noon, May 11, 1998).

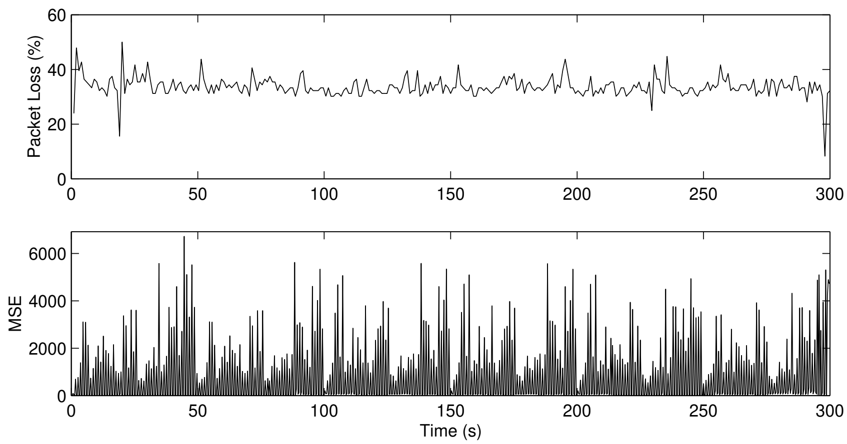

Figure 11: Non-adaptive flow at 1 Mbps from Hong Kong to Berkeley using scheme (T) (2 pm, 9/8/98)

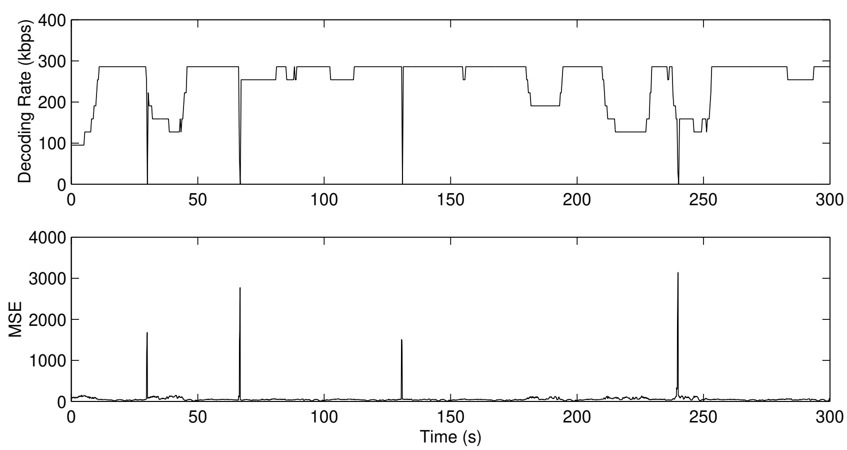

Figure 12: Adaptive flow from Hong Kong to Berkeley using scheme (T) (3 pm, 9/8/98)

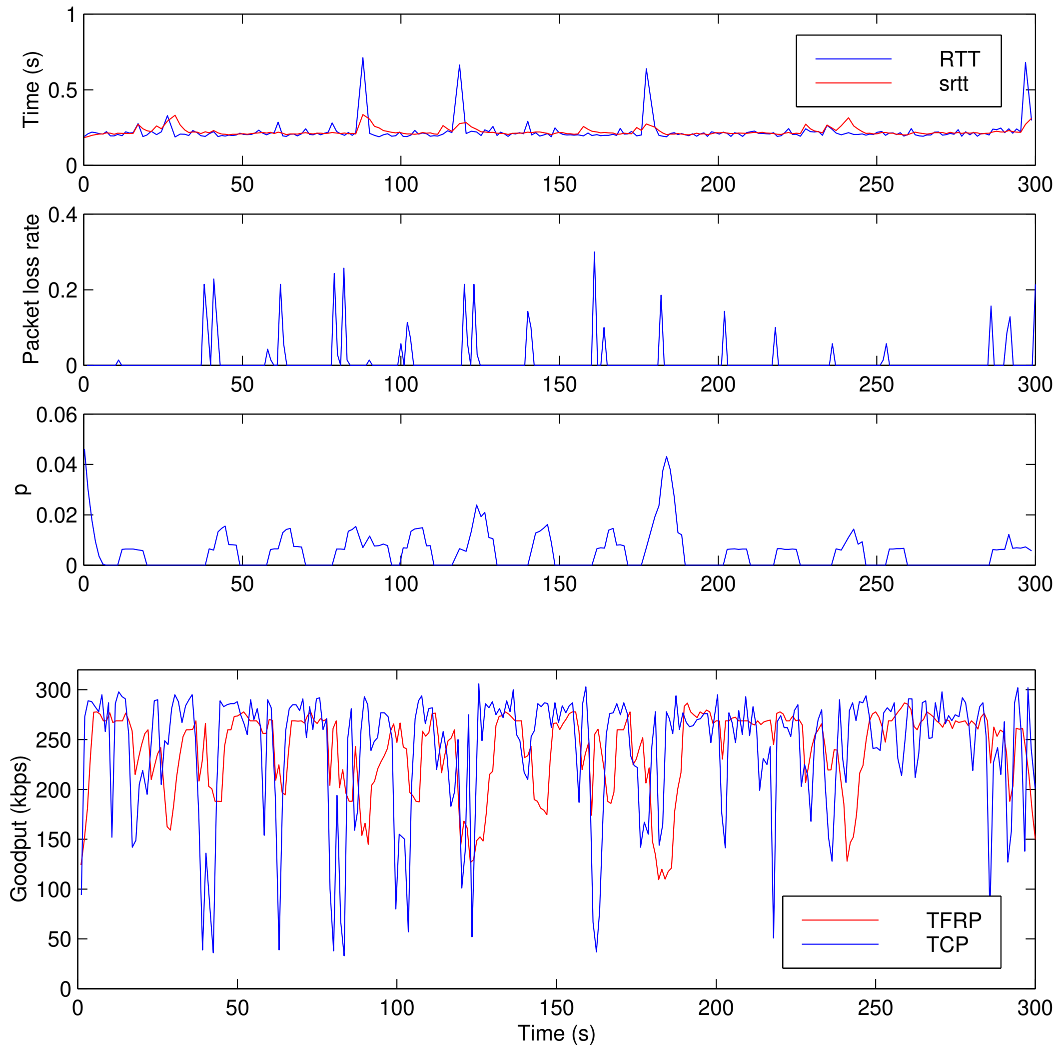

Figure 13: Simultaneous transmission using TCP and TFRP between Hong-Kong and Berkeley. First graph shows instantaneous RTT and the average RTT ($srtt$). Second graph shows packet loss rate observed at the receiver. Third graph shows p as estimated at the sender. Fourth graphs shows the bandwidth consumes by TCP and TFRP

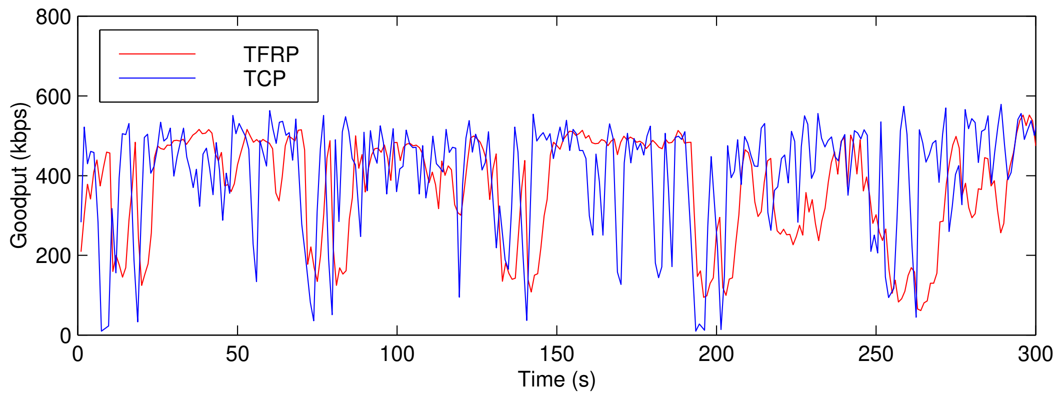

Figure 14: Throughput of simultaneous transmission of TCP and TFRP from Toronto to Berkeley

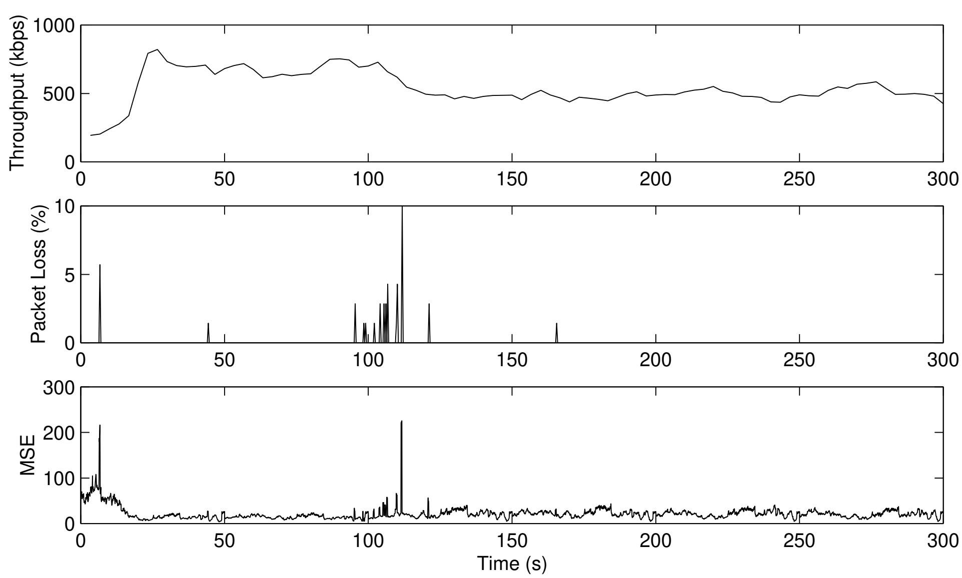

Figure 15: Throughput (top), packet loss rate at receiver (middle) and video quality (bottom) for the proposed compression scheme (P) with TFRP from Toronto to Berkeley (6 pm, June 23, 1998).

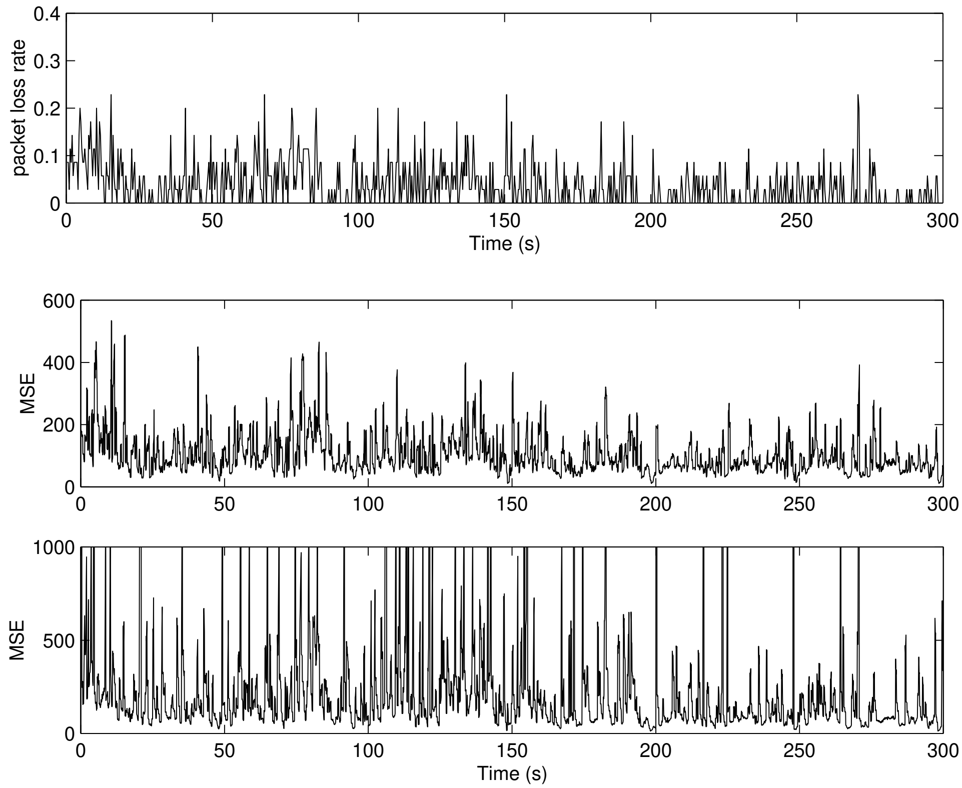

Figure 16: Simultaneous video transmission from Hong Kong to Berkeley with rate-control. (3 pm on 10/20/98). Top shows loss rate, middle and bottom correspond to MSE for schemes (P) and (T) respectively.

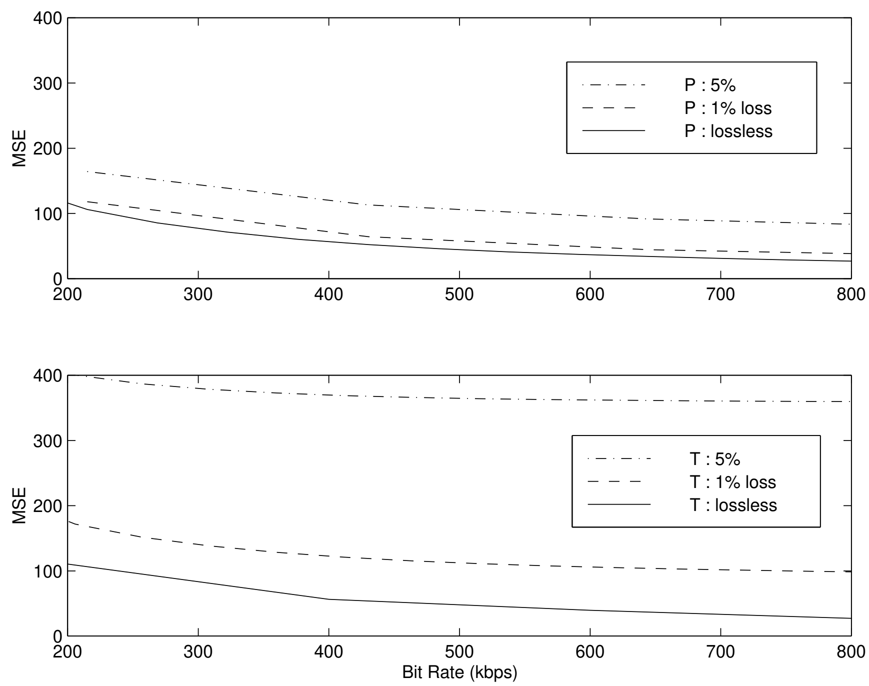

Figure 17: Distortion-rate characteristics of schemes (P) at 0, 1 and 5% packet loss.

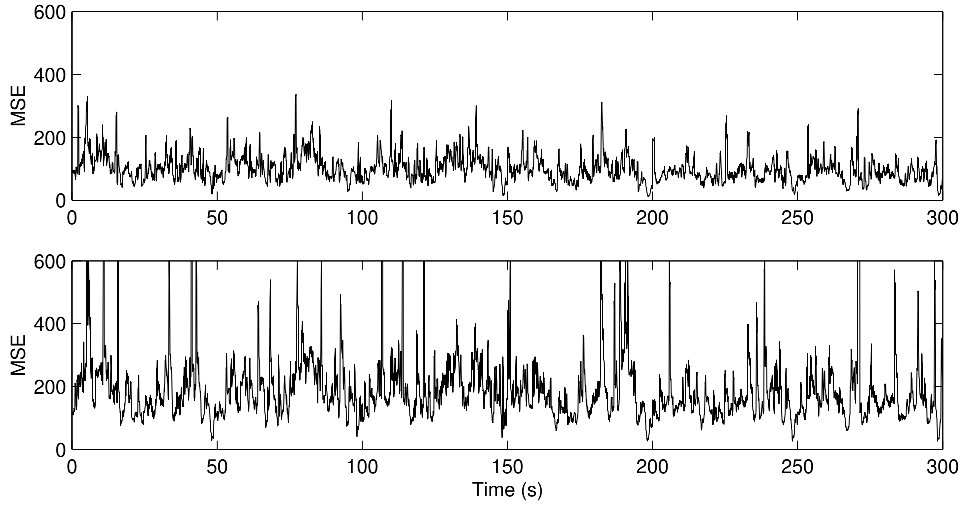

Figure 18: Simultaneous video transmission from Hongkong to Berkeley using TFRP and adaptive FEC. Top is scheme (P) and bottom is (T).

Description	No. bits	min value	max value
T_{LL}/T_{LH}	5	1	100
T_{LH}/T_H	3	0.5	7
S_{LH}/S_{HH}	5	1	32
S_{HL}/S_{HH}	5	1	32
S_{LL}/S_{HH}	6	0.5	2048
$S_{LL}/(S_{LH}+S_{HL}+S_{HH})$	2	1	8
δ	3	1/8	24
$\log_2(\alpha)$	3	9	16

Table 1: Parameters for the energy distribution of subbands. T_{LL} and S_{HL} stand for the energies of the temporally low frequency bands and the horizontally high but vertically low frequency bands respectively. δ is the decay factor for the spatially high-frequency subbands (e.g., the ratio of energies in the two HL subbands in Fig. 2) and α is the energy density of the DC subband.

Bit Rates (kbps)	103 (8 fps)	236	507	937	1463	3183
Raider (P_c)	29.5	29.3	32.6	35.5	38.0	43.0
Raider (P_v)	29.5	29.1	32.6	35.5	38.0	43.0
Raider (H)	28.8	29.1	33.8	36.9	39.1	43.0
Raider (M)	-	-	31.8	34.8	36.9	40.6
Bit Rates (kbps)	100 (15 fps)	232	494	1097	1664	3024
Mother (P_c)	31.3	31.5	36.2	40.1	42.5	46.0
Mother (P_v)	31.8	33.0	36.4	40.3	42.7	46.1
Mother (H)	32.1	33.7	37.8	41.5	43.6	-
Mother (M)	-	-	36.4	40.4	42.1	43.0
Bit Rates (kbps)	126 (12 fps)	200	613	843	1344	3010
Raider (P_c)	28.7	28.3	33.2	34.6	37.2	42.1
Raider (P_v)	28.5	28.2	33.2	34.6	37.1	42.1
Raider (H)	28.7	29.0	35.1	36.6	38.9	42.7
Raider (M)	-	-	32.6	34.0	36.1	39.4
Bit Rates (kbps)	105	241	549	837	1299	3024
Mother (P_c)	31.0	34.2	37.2	39.2	41.4	46.1
Mother (P_v)	31.5	34.2	37.2	39.2	41.4	46.1
Mother (H)	31.7	36.2	39.7	41.6	43.4	-
Mother (M)	-	-	38.0	40.3	42.0	42.8

Table 2: Compression Performance for proposed scheme using CBR (P_c) and VBR (P_v) allocations, MPEG-1 (M) and H.263 (H).Top half and bottom half show results for GOP sizes of 4 and 8 respectively.

	search space	n_1	n_2	n_3	n_4	n_5	n_6	n_7
N = 5	7	2	1	1	1			
N = 10	42	3	2	1	1	1	1	1

Table 3: Distribution of duplicate packets for 5% packet loss and total transmitted packet of N.

	n_1	n_2	n_3	n_4	n_5	n_6	n_7
N = 5	3	1	1				
N = 10	4	2	2	1	1		

Table 4: Distribution of duplicate packets for 10% packet loss and total transmitted packet of N.