REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	20 April 2004	Final: 1 October 2000 – 31 December 2003

4. TITLE AND SUBTITLE

Streaming Video Compression for Heterogeneous Networks

5. FUNDING NUMBERS

DAAD 19-00-1-0559

6. AUTHOR(S)

John W. Woods and Shivkumar Kalyanaraman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Center for Image Processing Research
Rensselaer Polytechnic Institute
110 Eighth Street
Troy, NY 12180-3590

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U. S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

40471-EL
• 2

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This is the final report on ARO grant DAAD 19-00-1-0559 titled "Streaming Video Compression for Heterogeneous Networks" and dated 1 October 2000 - 31 December 2003. We present a review of our activities during this three-year project, which treats robust video transmission of compressed video and focuses on highly scalable video compression combined with FEC and intelligent buffer management. In the latter part of the grant, we interacted with the international video standards group MPEG on the establishment of a robust scalable video coder (SVC) standard.

14. SUBJECT TERMS

robust video, video streaming, video compression, buffer management, multiple description, heterogeneous network, FEC, EZBC, MC-EZBC, scalable, subband, wavelet

15. NUMBER OF PAGES

42

16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION ON THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500 **Standard**
Form 298 (Rev.2-89)
Prescribed by ANSI Std. 239-18
298-102

Final Report

on

ARO grant DAAD 19-00-1-0559
(1 October 2000 – 31 December 2003)

Streaming Video Compression for Heterogeneous Networks

by

John W. Woods and Shivkumar Kalyanaraman

Center for Image Processing Research
and ECSE Department
Rensselaer Polytechnic Institute
Troy, NY 12180-3590

20 April 2004

(1) Forward

This is the final report on ARO grant DAAD 19-00-1-0559 titled "Streaming Video Compression for Heterogeneous Networks" and dated 1 October 2000 - 31 December 2003. We present a review of our activities during this three-year project, which treats robust video transmission of compressed video and focuses on highly scalable video compression combined with FEC and intelligent buffer management. In the latter part of the grant, we interacted with the international video standards group MPEG on the establishment of a robust scalable video coder (SVC) standard.

(2) Table of Contents

(3) Statement of Problem Studied

This is the final report on ARO grant DAAD 19-00-1-0559 titled "Streaming Video Compression for Heterogeneous Networks" and dated 1 October 2000 - 31 December 2003. We present a review of our activities during this three-year project, which treats robust video transmission of compressed video and focuses on highly scalable video compression combined with FEC and intelligent buffer management. In the latter part of the grant, we interacted with the international video standards group MPEG on the establishment of a robust scalable video coder (SVC) standard.

This report starts with a review of scalable video coding. Then we concentrate on robust versions of the coded data that are able to withstand packet loss. We first consider dispersive packetization (DP) and provide some intraframe video coding simulation results. This is followed by interframe coding results using the same technique, which is a form of multiple description (MD) coding. Section 4 then introduces FEC into our multiple description (dispersive packetization) coding and we find a significant advantage for DP in the channel mismatch case. The next section features the interaction of video coding with networks and introduces an improved transfer control protocol incorporating randomness in packet sending times.

Then we consider source buffer management and show here that a relatively simple scheme can reduce packet losses in a wired network to small amounts, by following network recommended sending rates. While we use a standard H.26L video coder here, the more scalability that a video coder possesses, the more adaptability it has to network congestion. This is followed by work on using motion compensation to add robustness by using the high quality motion information in our scalable coder MC-EZBC to perform error concealment on GOPs with a lot of missing packets. We thus show that, contrary to common belief, motion compensated interframe coding can be more robust than intraframe coding that does not use motion.

Hybrid video streaming is considered next in a peer-to-peer context, where local neighbors are poled to determine if video information is locally available, thus lowering the load on a network video server. Again here, in a highly scalable video coder all the information is contained in 'atoms.' Hence any information found locally can be used, and only a minimum of information will have to be shipped down the net from the server. There is a section on our work with the MPEG network simulation testbed and our response to their call for evidence in robust scalable coding. Network path aggregation is considered in a final section, wherein improved streaming stability is obtained by efficiently using more than one path for sending the packets. With our out-of-sequence transmission approach, high latency paths can be used to advantage.

(5) Summary of Most Important Results

1. Scalable video coding

Motion compensated video typically consists of several types of data, e.g. motion vectors, frame prediction residuals and possibly some overhead information, such as quantizer data. The structure of the group of pictures (GOP) of the motion compensated 3-D subband/wavelet video coder from [1] is shown in Fig. 1. The top level represents the video at full frame rate. Neighboring frames are decomposed using a motion-compensated filter bank to produce temporal low frequency bands (solid lines) and temporal high frequency bands (dashed lines) at the next lower level. Motion vectors are symbolically shown as arrows. Low frequency bands are effectively the motion compensated average of the two neighboring frames at full frame rate, so they occur at half frame rate. The process is repeated once again to obtain the next lower level (quarter frame rate). In the coder of [1], four neighboring temporal low frequency bands at this lowest level are encoded using motion-compensated temporal DPCM.

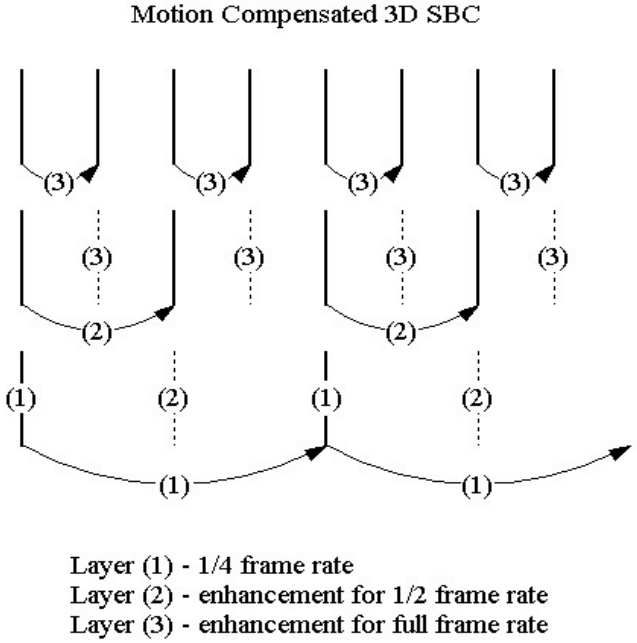


Fig. 1 Early 3-D subband/wavelet coder with GOP=4 frames and hybrid coder base layer.

The structure of the data produced by this coder is naturally divided into three layers. Layer (1) represents the data needed to reconstruct the video at one quarter of the original frame rate. Layer (2) is the enhancement that needs to be added to layer (1) for reconstruction at half the original frame rate. Finally, layer (3) is the enhancement needed for reconstruction at the full frame rate. Looking along the temporal axis, one can notice that within each layer the structure of data is a shuffled sequence of frame prediction residuals and motion vectors.

The coder from [2] extends the GOP structure to 16 frames, enabling five levels of temporal scalability. This is shown in Fig. 2. Here we have five layers of data, from layer (1) at 1/16 of the original frame rate, to layer (5) the final enhancement needed full frame-rate video. Again, along the temporal axis, all layers

(except layer (1)) are represented by a shuffled sequence of frame residuals and motion vectors. In this coder, layer (1) does not contain motion vectors, since it is not encoded using MC temporal DPCM. In early experiments we used the version of this coder with bi-directional motion compensation [3].

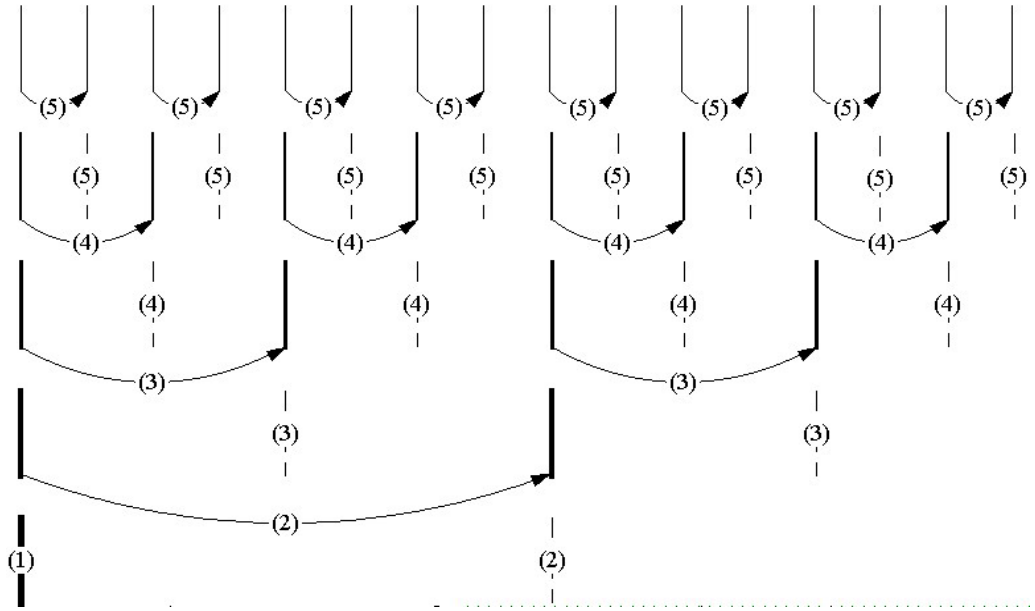


Fig. 2 3D subband/wavelet coder with GOP=16 frames and no hybrid coder base layer.

2. Dispersive packetization

Packetization of the motion compensated scalable video can be accomplished in the following way. Given the desired rate, we first determine the number of bits per GOP. To maximize the packet payload-to-header ratio, we want to create packets that are about 576 Bytes long (this is the largest packet length guaranteed not to be fragmented during Internet transmission). From these parameters we determine the required number of packets per GOP, and hence the number of packets per layer within a GOP. Within each layer, we perform dispersive packetization in 3-dimensions. The set of 3-D packetization masks is obtained starting from one basic 2-D packetization mask, which is repeated, in the temporal direction throughout the GOP. We have recently developed a good method for design of 2-D packetization masks based on lattice partitioning [4].

Within each layer, packets are made to be approximately equally important. This was previously found to be necessary to minimize the variation in received video quality. Equally important packets all contain both motion vectors and portions of frame prediction residuals, and data within the packet is dispersed across space, time and frequency. Additionally, there are several header packets per GOP, containing motion vector maps and quantization parameters. The size of these header packets depends on the encoding rate. In our simulations, the total size was 1-2 K Bytes per layer. During simulation, header packets are assumed to be correctly received. In practice we may need to retransmit these packets should they be lost, or simply send multiple copies of them at-a-time to increase the chance of correct delivery to the receiver.

When a packet is lost, all pieces of information it contained (i.e. subband coefficients and motion vectors) will have a large number of neighbors available (in space, time, and frequency), due to dispersive packetization. This enables us to perform various types of error concealment to approximately recover lost information. In the set of simulations described below, we used median interpolation to recover lost coefficients in the frame residuals, which showed very good results. Lost motion vectors are assumed to be zero, i.e. at this stage no concealment is applied to lost motion vectors. We also found that vector median filtering offered improved good performance for this purpose.

Simulation Results

In this early experiment we examined the effect of error concealment on the video bitstream affected by the packet loss. Here we used robust version of the coder from [1] (MC3D-SBC-FSSQ) to encode and dispersively packetize the *mobile calendar* sequence (color, SIF resolution, 30 fps) at the rate of 350 Kbps. We introduced a 33 % random packet loss in each of the three layers of video in the first GOP, and then reconstructed the full frame-rate video from available packets. Median interpolation from nearest available neighbors was used to conceal the loss in frame prediction residuals. Lost motion vectors were assumed to be zero (i.e. no concealment). Results are summarized in Table 1 and illustrated Fig. 3.

First 16 frames	**Average PSNR (dB)**
No loss	19.75
DP + concealment	16.97
DP, no concealment	10.36

Table 1 PSNR evaluation of dispersive packetization (DP)

As can be seen from Fig. 3, loss occurs in the first GOP and the error propagates through the next three GOPs due to temporal DPCM used at the base level in this coder. Resynchronization occurs at the start of the fourth GOP. Results show that the error concealment gain is about 6.5 dB in the areas affected by packet loss. In a second experiment we investigated the variation of the reconstructed video quality as the link capacity changes. The basis of this experiment is the coder from [3] (BiMC3D-SBC). We use its robust version (P-BiMC3D-SBC) to encode and dispersively packetize the *mobile calendar* sequence (color, SIF resolution, 30 fps) at about 415 Kbps. The error concealment procedure is the same as in the first experiment.

Transmission of the packetized video was simulated using UC Berkeley's *ns-2*. Two types of links were considered: one based on a 10 Mbps bottleneck and the other based on a 1Mbps bottleneck. In both cases our video stream competes against TCP traffic. Results are shown in Table 2 and Figs. 4 and 5. Video streams produced by our P-BiMC3D-SBC coder consist of five layers (Fig. 2), but for simplicity we show the aggregate packet loss, not the loss divided among the layers. Most of the loss, however, occurs in layer (1), since this layer accounts for almost half the packets.

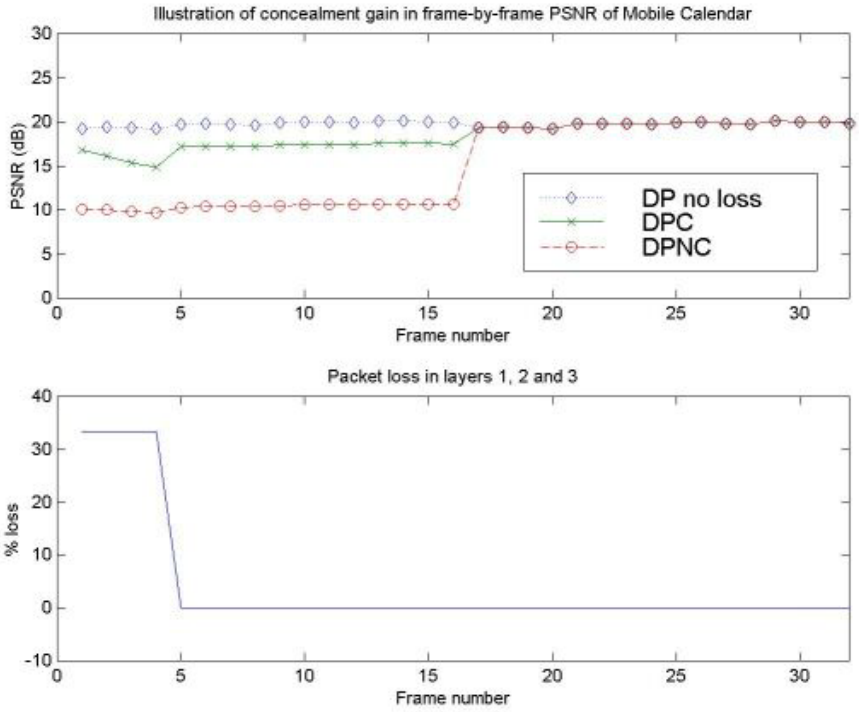


Fig. 3 Gain in PSNR and packet loss versus frame number.

Coding method	Average PSNR (dB)	Average packet loss (%)
BiMC3D-SBC	22.32	
P-BiMC3D-SBC (no loss)	20.21	
P-BiMC3D-SBC, 10Mbps	19.34	3.4 %
P-BiMC3D-SBC, 1Mbps	19.07	5.9 %

Table 2 Achieved PSNRs versus percent packet loss.

One can see that at very low bit rates (415 Kbps), modifications introduced into the coder to enable dispersive packetization resulted in a loss of about 2 dB. This loss mostly depends on the coding method used for coding the frame residuals, which is FSSQ [5] in this case. It could be reduced by using different coding methods (e.g. EZBC [9]). Also, the loss is expected to be smaller at higher bit rates. Simulated video transmission through the 10 Mbps bottleneck resulted in 3.4 % aggregate packet loss, which induced about 0.9 dB loss in PSNR. When the link capacity was reduced to 1 Mbps, the loss rose to 5.9 %, which reduced reconstructed video quality by additional 0.3 dB. Video quality degraded gracefully with packet loss increase.

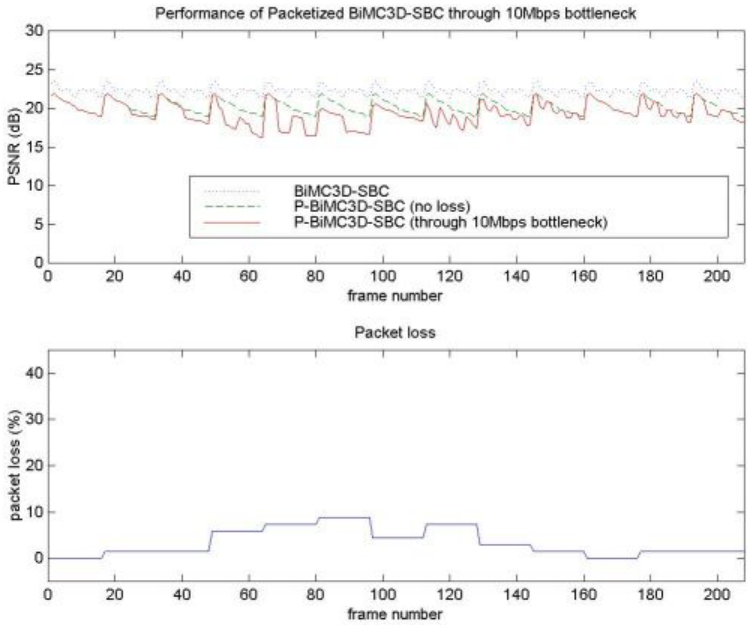


Fig. 4 P-BiMC3D-SBC performance with 10 Mbps bottleneck versus frame number.

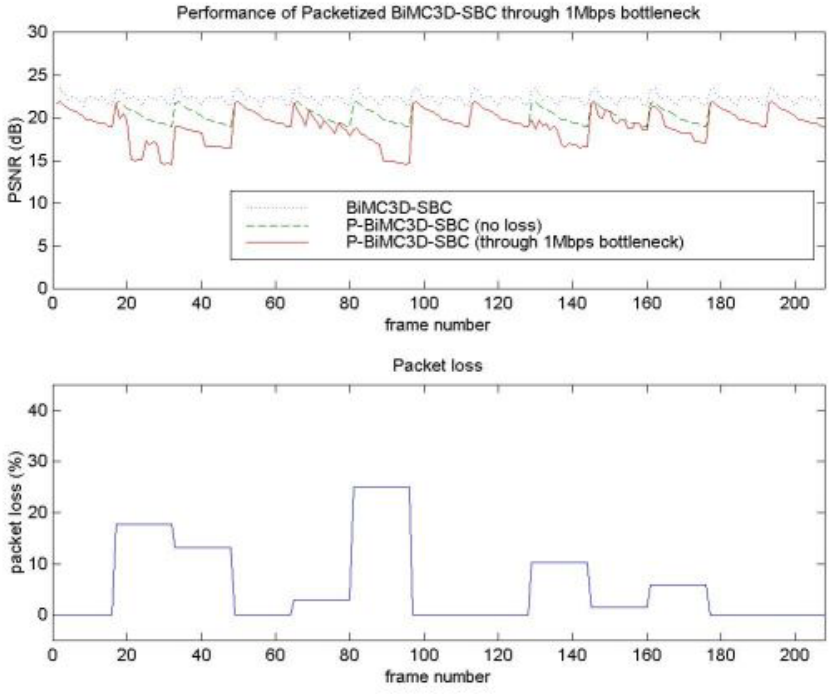


Fig. 5 P-BiMC3D-SBC performance with 1 Mbps bottleneck versus frame number.

Portions of the video clips from this experiment can be seen at the web site [6].

3. Domain-based Multiple Description Coding of Video

Following upon our work on dispersive packetization (DP) of images and intraframe-coded video [7], we developed a robust motion compensated 3-D subband video coder [8,9]. Robustness is achieved by partitioning the domain of the video signal in such a way that missing pieces of the data (subband samples or motion vectors) are likely to have many available neighbors that are used for error concealment at the decoder. The details of domain partitioning are discussed in [21]. Due to the fact that packets (descriptions) are created by partitioning the domain of the signal, we also refer to this process as domain-based multiple description (MD) coding.

As mentioned earlier, in our coder video data are divided into frame rate scalability layers – the base layer provides the video at the lowest frame rate (currently 1/16 of the original frame rate), while the other layers provide enhancement for higher frame rates (1/8, 1/4 and 1/2 of the original frame rate). To enable scalability, each layer is packetized independently so that the video at a lower frame rate can be reconstructed from subsets of packets corresponding to a higher frame rate. Within each layer, data is coded and packetized in a dispersive manner, so that subband samples from any given space-time-frequency neighborhood appear in different packets, which enables good error concealment of lost samples from the available neighboring samples. Also, all the packets from a given layer carry approximately the same amount of information about every frame in that layer, which minimizes the variation of video quality at a fixed packet-loss rate. It was found that this dispersive packetization approach to robust coding effectively combats the packet loss and eliminates the need for forward error correction (FEC) at packet-loss rates up to several percent. Performance at higher loss rates can be further improved through the use of FEC, or by intelligent buffer management strategies described later in this report.

To evaluate the performance of our dispersive packetization scheme, we compared it to the interleaved slice-based packetization (SP) scheme that is part of the H.323 recommendation for packet-lossy environments. In both cases we used the same algorithm for error concealment of missing data – median filtering for missing subband samples and vector median filtering for missing motion vectors. Results of three groups of experiments are reported here. The first group (experiments 1 and 2) is based on the *Mobile calendar* sequence encoded at 0.45 Mbps with a GOP of 16 frames and the average PSNR of 22.3 dB. The second group (experiments 3 and 4) is based on the *Football* sequence encoded at 0.97 Mbps with a GOP of 4 frames and average PSNR of 26.3 dB. The third group (experiments 5 and 6) is based on the *Table tennis* sequence encoded at 0.94 Mbps with a GOP of 16 frames and average PSNR of 31.4 dB. All sequences were grayscale, SIF resolution, at 30 fps. Average PSNR results are summarized in Table 3.

Experiment	Avg. packet loss	PSNR for SP	PSNR for DP	Gain of DP over SP
1	8.2%	18.9 dB	19.7 dB	+0.8 dB
2	12.4%	17.8 dB	19.1 dB	+1.3 dB
3	5.4%	24.0 dB	24.6 dB	+0.6 dB
4	10.4%	21.8 dB	23.2 dB	+1.4 dB
5	9.2%	23.5 dB	25.7 dB	+2.2 dB
6	16.0%	20.0 dB	22.8 dB	+2.8 dB

Table 3: Average PSNR results comparison of DP versus SP.

We notice that DP has the PSNR advantage over SP for all the experiments, with the greatest advantage +2.8 dB at the highest average packet loss rate. It is also important to realize that the DP decoded frames tend not to have the washed out areas present in SP decoded ones. Instead, there is a much more uniform loss across the entire frame and sequence. This visibility advantage is in addition to the just mentioned 0.6-2.8 dB objective advantage. Very simple error concealment schemes were used to provide estimates of missing image and motion data. We are continuing this work by looking at more powerful methods to restore the missing data. These methods should result in improved subjective performance for the DP approach. Figure 6 shows a frame-by-frame PSNR plot from experiment 4.

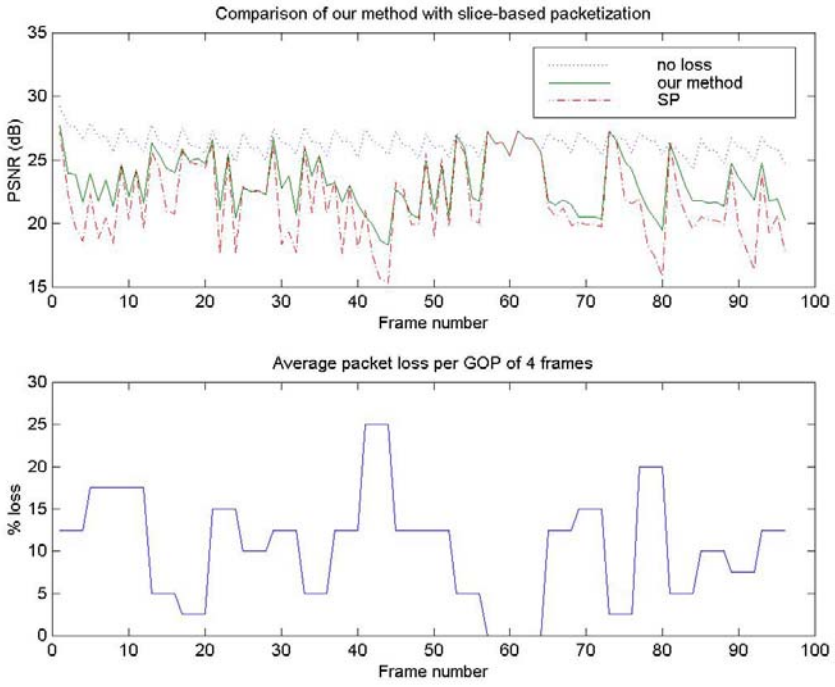
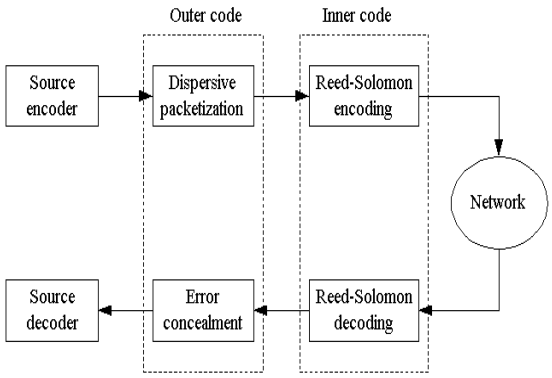


Fig. 6: PNSR plot from experiment 4 comparing PSNR performance DP and SP.

4. Concatenated Multiple Description Coding of Video

In concatenated channel coding, the idea is to combine two codes with different error correcting capabilities to produce a more powerful code. Following this intuitive approach, we have combined our dispersive packetization (DP) with FEC-based multiple descriptions [22]. As shown in Figure 7a (left), DP represents the outer code, while the FEC is introduced by applying a Reed-Solomon code as the inner code. In this way, portions of the video stream, which cannot be recovered by FEC, are estimated by error concealment. Details are given in [25,31].



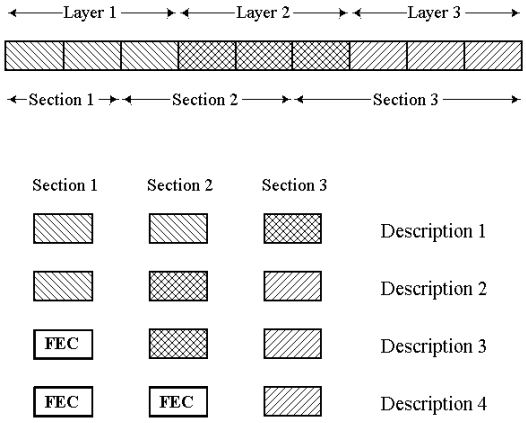


Fig. 7a: Concatenated multiple description coding

Fig. 7b: Application of FEC to packets

The process of applying FEC is illustrated in Figure 7b (right). The dispersively packetized video is divided into segments and different sections receive different amounts of FEC, according to their importance. More important sections contain the data corresponding to the lower frame rates. In Figure 7b, section 1 is protected by the (4,2) RS code, section 2 by the (4,3) RS code, and section 3 is unprotected. Each description is mapped to a reasonable sized packet to avoid possible fragmentation.

We illustrate the operation of a concatenated MD system by example. With reference to Figure 7b, suppose that four out of the twelve segments are lost, as shown in Figure 7c. In this case segment 4 is in section 1 which is protected by the (4,2) RS code, so it can be recovered by FEC. Similarly, segment 5 in section 2 is protected by the (4,3) RS code, and can be recovered by FEC. Segments 3 and 6 in section 3 cannot be recovered. Hence, segment 3 is approximately recovered by error concealment using the data from other segments in layer 2, while segment 6 is approximately recovered from the other segments in layer 3.

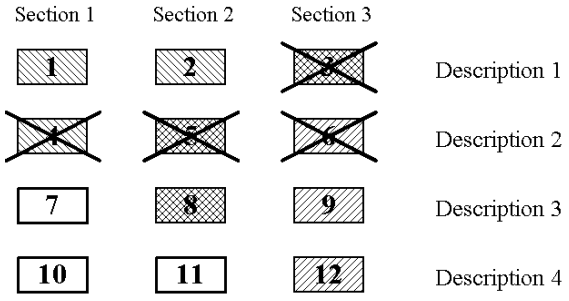


Fig. 7c: Illustration of segment loss scenario

Figure 8 shows comparisons of the performance of the concatenated MD system with its component systems. In one case (labeled EC), we use the total rate budget for source coding and apply error concealment to recover the missing data. In the other case (labeled FEC), we assign an appropriate

amount of FEC (for the given network packet loss rate p), so that the source coding rate plus the channel coding rate are within the total rate budget, and use only the FEC at the receiver to recover the missing data. The third case (labeled FEC+EC) is a combination of FEC and error concealment. Figure 8a (left) shows the performance in the case when perfect channel-state information is available (i.e. the correct value of p is known in advance). Figure 8b (right) shows the performance in the case of a mismatch between the assumed value of p and the actual value of p (the figure shows what happens when FEC is designed for $p = 0.09$ while the actual p varies). As the results show, concatenated system significantly outperforms its component systems at higher loss rates, and also shows increased robustness in the case of a channel-state mismatch.

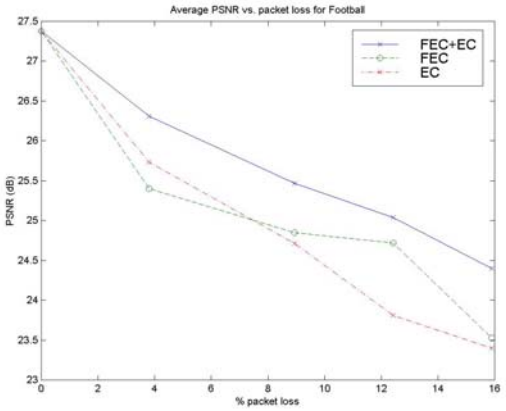


Fig. 8a: PSNR versus percent packet loss

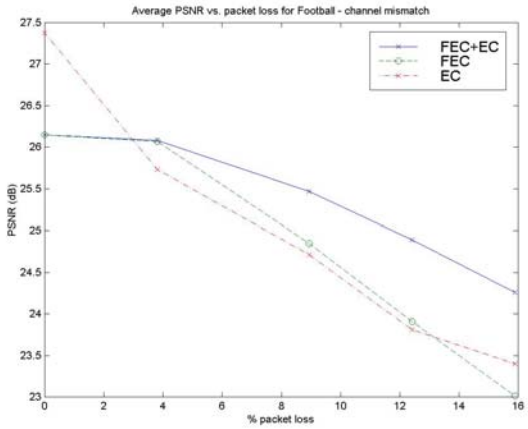


Fig. 8b: PSNR in channel mismatch case

5. Robust video over the Internet

The networking aspects of the ARO-funded research program focused on three topics:

1. Application-level congestion management issues for video-transmission over the Internet. This work is divided into two parts: randomized TCP congestion control, and source-buffer-management protocols.
2. A new hybrid download-streaming paradigm that blends aspects of "download" and "streaming" to provide enhanced video delivery capabilities in emerging overlay and peer-to-peer networking contexts.
3. Exploratory research activity in the area of leveraging massive path diversity through a combination of end-to-end delay compensation and FEC techniques.

Traditionally Internet video applications run over transport layer protocols such as RTP and UDP. These off-the-shelf network protocols *do not specify congestion control*, i.e., mechanisms that adapt the sending rate in response to loss or other congestion indications from the network. The issue of congestion control for video poses an important dilemma. Congestion control results in rate-adaptation and indeed lower rates during periods of high packet loss. This is a challenge for video coding schemes, which then have to handle both

a) variable and lower rates and
b) be robust to higher packet loss rates!

At the same time, without congestion control, it is possible to have congestion collapse, network instabilities and beat-down of competing TCP flows - an undesirable scenario. The requirement to avoid network instability and provide TCP-friendliness has to be balanced against the end-to-end performance demands of video applications. The investigation of this dynamic tradeoff of video over the best-effort Internet was a core focus of the project.

The strategy we chose is to equip the end-system with middleware to provide TCP-friendly congestion control, and services like adaptive packetization, adaptive packet marking, relatively smooth transmission rate, randomized transmission (to tackle network synchronization issues) and source buffer management. Further, the strategy increases the amount of "channel" specific information available to source coding and error-concealment schemes. Such mechanisms maximize application-perceived performance and provide graceful degradation of performance under congested conditions. It complements joint source-channel coding approaches like UEP (unequal FEC error protection), error concealment, and efficient source coding. The approach is illustrated in Fig. 9.

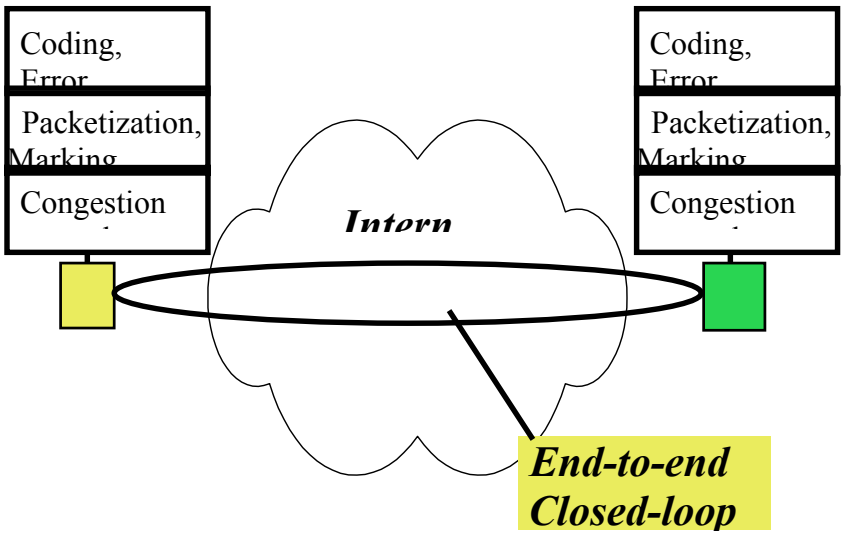


Fig. 9: End-to-End Protocol Support for Video over the Internet

In the first year of the program, we built and evaluated several alternatives for the congestion management piece of the end-to-end architecture. We evaluated the additive increase/multiplicative decrease (AIMD) scheme against the recently developed TCP Friendly Rate Control scheme (TFRC) [10] and the binomial increase/decrease algorithms [11,16]. These are considered to be important algorithms that could potentially be used to control video transmission over the Internet. In particular, they display lower volatility compared to the TCP AIMD algorithm, while aiming to be friendly to the installed base of TCP congestion control.

We studied these algorithms carefully and developed new randomized versions of these algorithms, that have special benefits when operating over drop-tail FIFO queues like those widely deployed in the Internet today. This performance analysis has led us to a better understanding of the right type of algorithms to choose for controlling video transmission over the Internet. These results have been reported in a paper accepted for publication [15]. We discuss the randomization solution below.

Current implementations of TCP and similar congestion control schemes suffer from serious performance problems like unfairness to flows with higher round trip times (RTTs), synchronization and phase effects in flows and correlated losses, leading to throughput degradations under a wide range of scenarios. We proposed a solution to these issues by randomizing the packet sending times at TCP sources (called Randomized TCP). Specifically, we space successive packet transmissions with a time interval *Delta = RTT(1+x)/cwnd*, where *x* is a zero mean random number drawn from a Uniform distribution. We found that a Uniform distribution on *U*[-1,1] is near optimal with respect to metrics like throughput, fairness, timeouts, losses, and de-synchronization. We showed that the scheme is better than Paced TCP (another variant of TCP) as well as TCP Reno (the traditional TCP implementation) in a large number of test scenarios. The proposed scheme is also fair with TCP Reno. We showed through simple simulations that Randomized TCP reduces phase effects and synchronization even when multiplexed with TCP Reno. Also, we extended randomization to other window-based schemes like the family of binomial schemes and showed that their fairness to TCP Reno increases dramatically even over traditional drop-tail queues inside the network.

5.1 Performance analysis of randomized TCP congestion control

Figure 10 shows the topology that was used for the simulations to study the behavior of various congestion control schemes. It consists of *N* sources and *N* receivers. Schemes are compared pair-wise. We deploy one scheme on half the senders (i.e. on *N/2* flows) and compare it against a competing scheme that is deployed on the remaining half of the senders (*N/2* flows). The link between the two routers is the bottleneck, typically running at 1 Mbps or 10 Mbps depending upon the simulation.

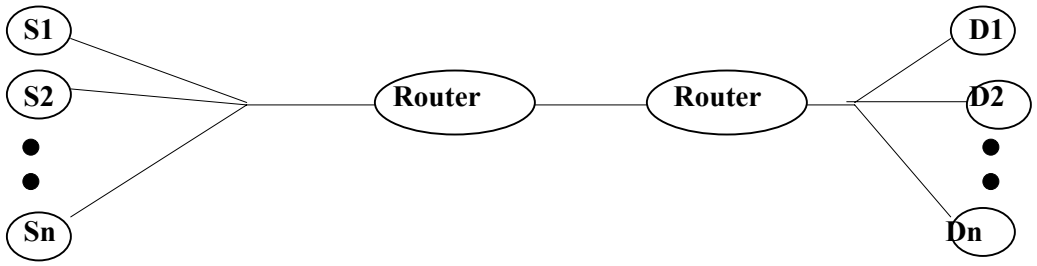


Fig. 10: Topology used for simulation-based comparison of congestion schemes

In our simulation results, we observe that *for medium-low speed bottlenecks with drop-tail queuing* the TFRC scheme behaves very unpredictably and in an unfair manner when competing with TCP [15,16]. Similar results are seen (though to a lesser degree) when alternative increase/decrease (binomial algorithms) [11] compete against TCP. The latter behavior is illustrated in Figure 11. The figure plots congestion window sizes of one selected flow implementing the AIMD scheme (highly oscillatory curve)

vs. the congestion window sizes of a selected flow implementing the binomial scheme (IIAD) [11]. The congestion window measures the number of packets in transit in the network. Larger congestion window curves indicate higher throughput for that flow. If the curves are far apart, it indicates that the distribution of throughputs is unfair.

Without our randomization enhancement, the IIAD scheme (despite its nice property of lower volatility) is unfair to TCP AIMD scheme. This is because the IIAD window curve is consistently larger than the corresponding AIMD congestion window curve. Figure 12 shows that once the randomization is added, the scheme competes fairly with TCP AIMD, i.e., the curves are coincident. We have tested the performance of this randomization in several other scenarios and it almost consistently performs better than the non-randomized case. This suggests that we could use a TCP friendly/compatible congestion control scheme (which has low volatility) with randomization introduced as the basis of video-transmission over the Internet. Future work would closely integrate this congestion management strategy with coding/error concealment schemes at the application layer.

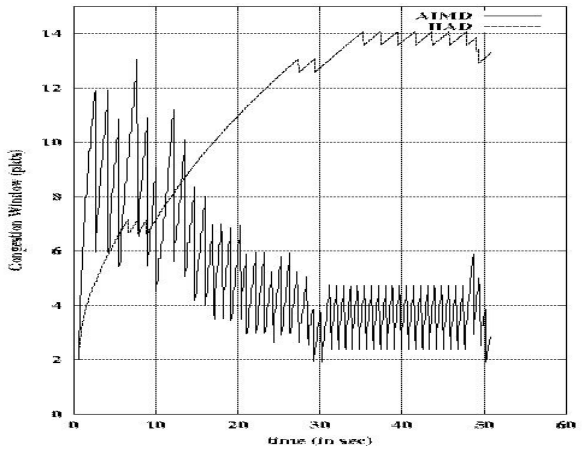


Fig. 11: AIMD v/s binomial scheme (IIAD), no randomization

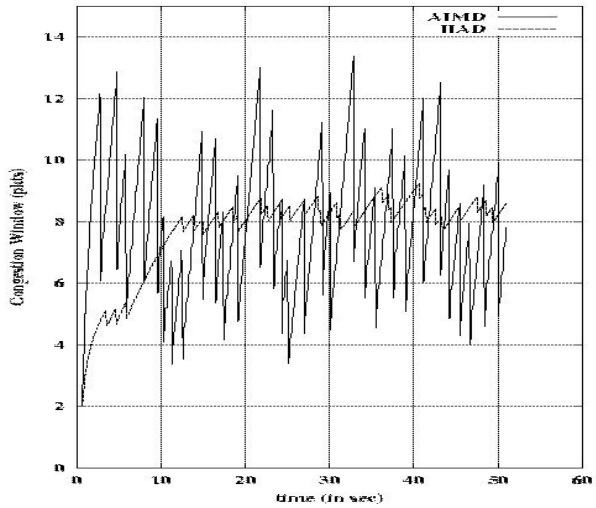


Fig. 12: AIMD v/s binomial scheme (IIAD), with randomization

6. Packet Loss Behavior and Integrated Source-Buffer Management

Besides the *fairness* characteristics, we examined the *packet loss behavior* of congestion control schemes. A key to providing reliable video transmission over lossy packet networks is the understanding of the mechanisms causing the packet loss, and points in the network where the loss occurs. Typically, the finite size of internal router buffers is quoted as the main cause of network packet loss. However, in a real-time transmission of high-bandwidth content such as video this is no longer the case, as illustrated below.

A typical model for video transmission over wired networks is shown in Figure 13. A server application sends the video data (solid lines) over the Internet to the client application, and receives feedback control information (dashed lines) from the client. This feedback information typically includes the estimates of the round-trip time (RTT), packet loss, etc. In this model, the "network" is considered to be everything between the server application and the client application.

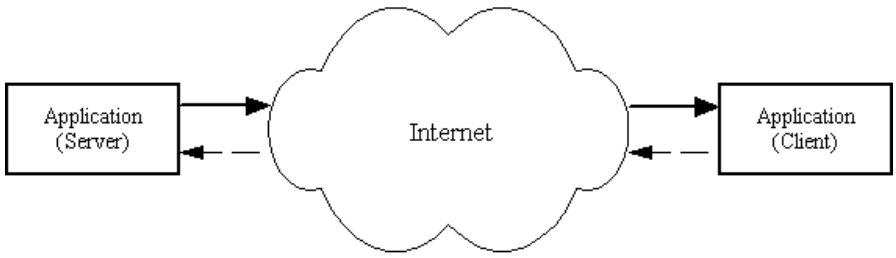


Fig. 13: A simple model for network video transmission

A more realistic model is shown in Figure 14, which also accounts for congestion control modules at the server and the client. These modules monitor the conditions on the network along the transmission path and regulate the transmission rate so that potential congestion situations can be quickly resolved. In this case, the "network" is everything between the two congestion control modules.

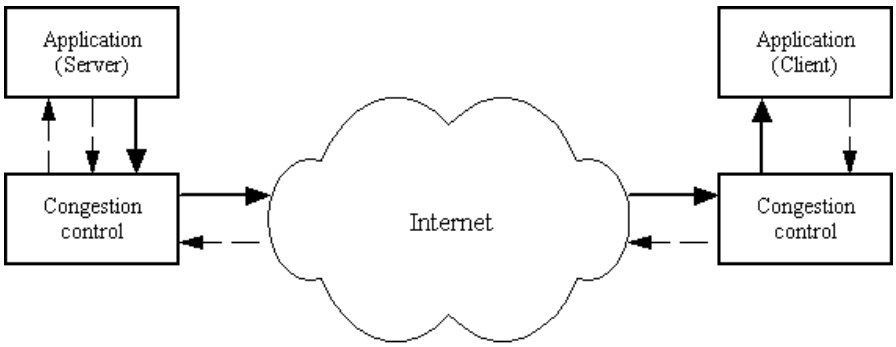


Fig. 14: A model for network video transmission involving congestion control

Figure 15 shows the comparison between the "network" loss in the two models illustrated above. The plot was obtained by measuring the "network" loss versus the number of flows transmitted through a single bottleneck. The figure shows that, as the number of flows increases, the overall loss between the applications increases significantly (as expected), but the loss between the congestion control modules remains fairly low – a few percent. Hence, *most of the loss is caused by congestion control itself, i.e. at the source of transmission.*

In other words, with congestion control, the surprising observation is that the packet loss rate within the network is *fairly small* even for large degrees of multiplexing. This is because the congestion control algorithm uses loss merely as a mechanism to detect congestion and reduce the source rates. As a result *the key overload point is at the source itself*! If the source coder sends at a rate faster than that allowed by the congestion control scheme, a backlog builds at the source, and once buffers are full, packets are lost more often at the source than in the network! Packets are also dropped at the receiver if they do not arrive in time for the playback.

In a typical scenario, the congestion control mechanism drops the packets at random, which makes the loss at the receiver appear to be random. However, if the congestion control is done in an intelligent way so that the packets dropped are less important for the application than the transmitted ones, performance can be improved. Scalable video coders can be very useful in conjunction with intelligent congestion control, since their bit streams can be easily manipulated to achieve different trade-offs between transmission rate, quality, frame rate and resolution.

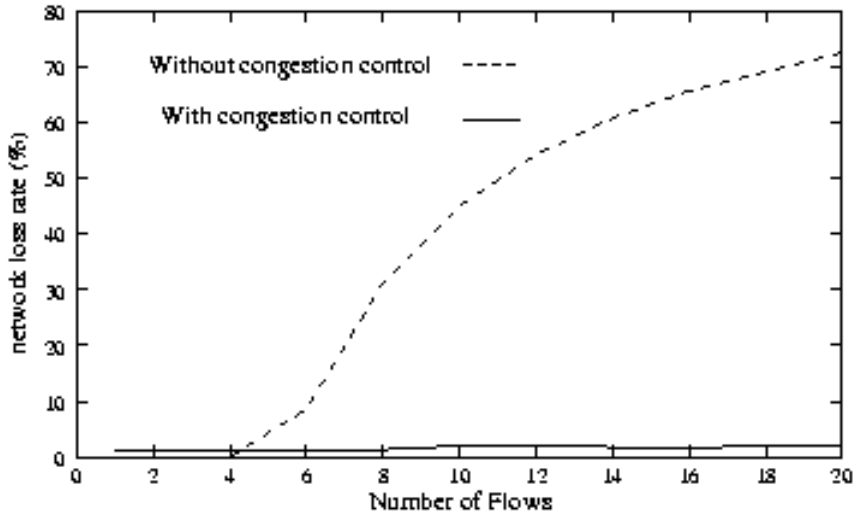


Fig. 15: Comparison of network loss for the models from Figures 13 and 14

This observation presents a clear opportunity to manage the *allocation* of packet losses among various layers of the encoded video stream. In particular, since higher layers are less important than lower layers, they can be dropped first, if dropping is necessary. We developed a simple source-buffer management strategy based upon this observation as illustrated in Figure 16.

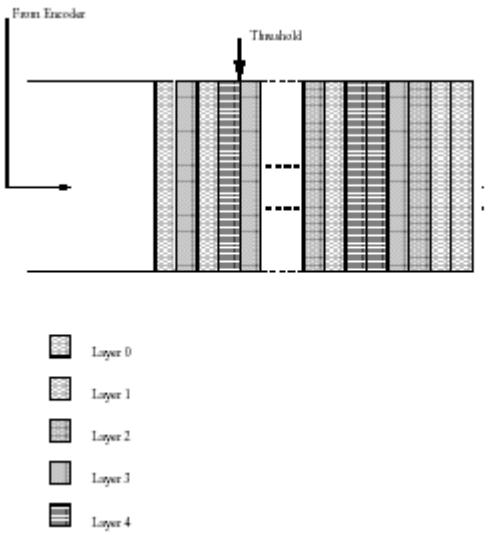


Fig. 16: an illustration of source-buffer management

A simple first-in-first-out buffer is used at the source. Packets enter the buffer based upon the output of the video encoder. The buffer management algorithm makes a decision whether to accept the packet or not. Packets are clocked out of the buffer based upon the rate set by the end-to-end congestion control scheme. In [12] we proposed one such integrated scheme for end-to-end congestion control. We tested the scheme with two video coders: our robust frame-rate scalable subband/wavelet video coder from [19] and the recently developed international standard video coder H.26L. Video is encoded into packets that are then passed on to the transmission buffer, along with their priority information. The priority of each packet is related to its corresponding frame rate. In the case of our subband/wavelet coder, the priority of a packet is determined by its temporal scalability layer, so that packets corresponding to enhancement layers for the highest frame rate have the lowest priority, while those corresponding lower frame rates have the higher priorities. In the case of the H.26L coder, *B*-packets have the lowest priority, followed by *P*-packets and *I*-packets. The transmission buffer regulates the transmission rate.

Since both the output and input rates may vary, we use a threshold at half the buffer size to trigger the operation of the buffer management scheme. Once the queue grows larger than the threshold, the difference between average input and output rates is viewed as the target overall packet-loss rate that has to be allocated. Given the average arrival rates of various layers, the buffer management scheme allocates a loss probability for each layer starting from layer 5, such that the weighted sum of loss probabilities (weighted by their fraction of arrival rates) equals the target overall packet-loss rate. When a packet arrives, and the queue is above this threshold, the appropriate loss probability is applied depending upon the layer to which the packet belongs and the packet may be dropped. Essentially the scheme tries to allocate the losses as far as possible to the higher layers before affecting the lower layers.

The buffer management mechanism effectively performs congestion control and drops packets at the onset of congestion. Lowest priority packets are dropped first, followed by packets of the next higher priority, and so on. In this way the receiver is more likely to receive a high-quality low frame-rate version of the video than a low-quality high frame-rate version. Hence, at the receiver side, the displayed frame rate is adjusted based on the overall loss, so that visual quality is maximized. The remaining loss in

the network (i.e. the loss caused by the overflow at internal router buffers) is handled by error concealment at the receiver.

In our experiments we used the *Football* test sequence (SIF resolution, 30 fps) encoded with our subband/wavelet coder, and the *Flower Garden* sequence (SIF resolution, 30 fps) encoded with the H.26L coder. Both sequences were encoded at a total bit rate of 1.7 Mbps. We tested the scheme for two scenarios:

(1) five sources transmitting through a single-bottleneck network, and
(2) eight sources transmitting through a multiple bottleneck network.

The results are shown in Tables 4 and 5, respectively. We compare the average PSNR of the received video with the proposed intelligent Buffer Management and Frame-rate Adjustment (BM+FA) scheme, and without it. As can be seen from the results, significant PSNR gains are achieved with the proposed scheme. Subjective quality is also significantly improved. Sample video clips may be found at our site [14].

Sequence	Bottleneck bandwidth (Mbps)	Avg. PSNR (dB) without BM+FA	Avg. PSNR (dB) with BM+FA	Gain (dB)
Football	7.5	22.1	28.4	+6.3
Football	6	18.4	24.8	+6.4
Flower Garden	7.5	12.7	29.3	+16.6
Flower Garden	6	10.2	28.9	+18.7

Table 4: Average PSNR results for a single-bottleneck network

Sequence	Avg. PSNR (dB) without BM+FA	Avg. PSNR (dB) with BM+FA	Gain (dB)
Football	18.5	27.9	+9.4
Flower Garden	17.1	32.0	+14.9

Table 5: Average PSNR results for a multiple-bottleneck network

The proposed buffer management scheme may be thought of as a mechanism for concentrating the loss in the less important layers of the video, i.e. for providing unequal loss protection (ULP). A similar effect can be achieved by using forward error correction (FEC) codes for ULP. This raises the question of which method is better. To this end, we also compared our proposed buffer management scheme with the concatenated multiple description (MD) coding scheme from [18], which uses Reed-Solomon codes for ULP. Comparison was made on the *Football* test sequence encoded with our subband/wavelet coder, as above. The FEC was designed for a binary erasure channel with loss probability and bandwidth matched to the two cases in Table 4. The frame rate of the received video is adjusted as in the BM+FA case explained above. As can be seen from the results shown in Table 6, both methods for providing ULP can significantly improve the performance over the case when ULP is not used. Also, in addition to being computationally simpler, it seems that our BM+FA method works slightly better than FEC+FA. The

reason for this is suspected to be the fact that in the FEC+FA case the packet loss is random, while in the BM+FA case most of the loss is deterministic rather than random.

Bottleneck bandwidth (Mbps)	Avg. PSNR (dB) without ULP	Avg. PSNR (dB) with FEC+FA	Avg. PSNR (dB) with BM+FA
7.5	22.1	27.4	28.4
6	18.4	22.7	24.8

Table 6: Comparison of two different ULP schemes

6.1 Computational complexity of BM-FA and FEC-FA

The BM-FA strategy is computationally very simple. All that is required is to assign a drop probability for each layer of encoded video. For the greedy assignment [12], drop probabilities can be computed in closed form, so the complexity of the algorithm is $O(N_l)$, where N_l is the number of layers. Let N be the number of packets in a coding unit (e.g. a GOP). Since there is at least one packet per layer in the frame-rate scalability case, we have $N_l \leq N$, so finally the complexity of the drop probability assignment in BM-FA is at most $O(N)$. On the other hand, the fastest known algorithm for FEC assignment [17] in FEC-FA has the complexity $O(NL)$, where L is the size of the packet in symbols (typically Bytes). For Internet transmission where large packets are preferred (e.g. L = 500 Bytes), one usually has $N \leq L$ in a typical coding unit. Both algorithms need to execute their assignments whenever channel conditions change sufficiently, so the frequency of their execution is expected to be the same.

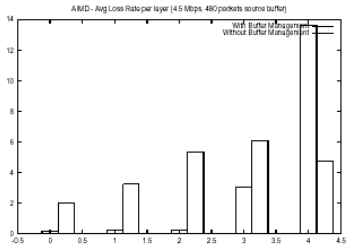
Consider an example of a video encoded at 1.5 Mbps with a GOP size of 16 frames. In this case we have $N_l = 5$ temporal scalability layers. With a packet size L = 500 Bytes, we have about $N = 200$ packets per GOP. Now, the FEC assignment in [17] consists of two parts: the first part is an $O(N)$ algorithm to find an initial guess to the solution (the "rate-optimal solution" in [17]); the second part is a sequence of iterative improvements to this solution, with the overall complexity $O(NL)$. In the worst case, the second part requires $2L(N - 1) + 2 = 199002$ multiplications, the same number of additions, and $L(N - 1) = 99500$ comparisons. On the other hand, the probability assignment in [12] requires in the worst case $2N_l = 10$ multiplications, the same number of additions, and also the same number of comparisons.

Figure 17 shows additional simulation results illustrating the effect of the source-buffer management scheme, when different congestion-control schemes are used. Consider Figure 17a, which shows the loss probabilities for various layers with and without buffer management, where Randomized AIMD is the congestion control scheme used. In this particular scenario, the per-flow average rate (after congestion control) is about 870 kbps and about 30-40 kbps of information has to be dropped, i.e., a target overall loss rate of 4%. Observe that the distribution of loss probabilities is substantially skewed to the right (i.e. the higher layers) with source-buffer management. This is exactly what we want, which would translate into higher PSNR and perceptual video quality upon playback, albeit at a reduced frame rate.

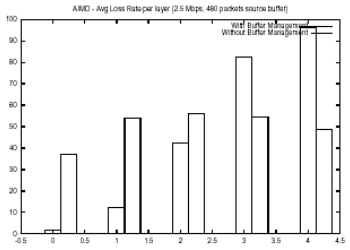
Figure 17b shows a similar scenario, but the per-flow rates are much lower, i.e. the target overall loss rate is between 40-50%! Now, under such high loss conditions the impact of the source-buffer management is even greater. Observe that without buffer management, there is not much skew in the distribution of loss rates across layers, but with the buffer management the losses are largely allocated only to the higher layers.

Figure 17c and d repeat this experiment using Randomized IIAD (a binomial scheme) that displays lower volatility of per-flow rates. Observe that the benefits of source buffer management are much higher in this case (compared to Figures 17a and b) because the lower volatility of the per-flow rates allows better short-term estimation of target loss rates and allocation of loss rates to different layers. As a result, with IIAD and source buffer management, the lower layers are allocated close to a 0% loss rate, and all the loss is concentrated in the higher layers.

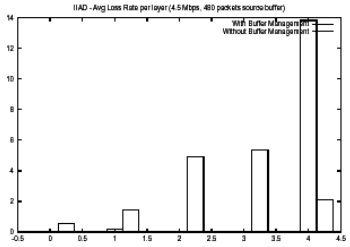
This feature is displayed again in Figures 17e and f where the source-buffer management histograms for AIMD and IIAD are directly compared. Figures 17g and h show the queuing behavior (at the source queue, which is managed) for the two schemes: IIAD and AIMD. The AIMD curves show substantial oscillation.



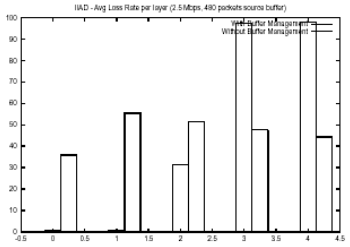
(a) AIMD - Average Loss Rate per layer



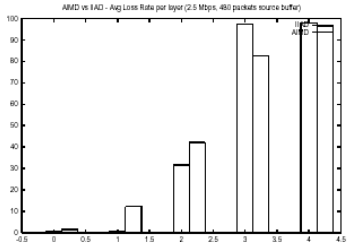
(b)AIMD - Average Loss Rate per layer



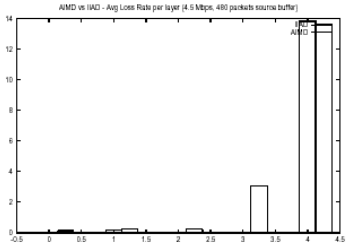
(c) IIAD - Average Loss Rate per layer



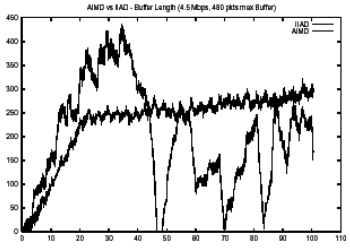
(d) IIAD - Average Loss Rate per layer



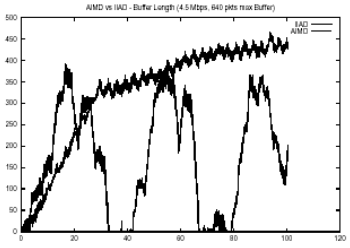
(e) AIMD vs IIAD - Average Loss Rate per layer



(f) AIMD vs IIAD - Average Loss Rate per layer



(g) AIMD vs IIAD - Source Buffer Length



(h) AIMD vs IIAD - Source Buffer Length

Fig. 17: Simulation Results: Source Buffer Management + Congestion Control Schemes

Therefore we conclude that large per-flow rate volatility is bad for streaming video, and that source-buffer management can dramatically improve performance. Again there is the need for scalable coding to be able to rapidly respond to the network conditions.

7. Motion compensated error concealment for MC-EZBC video

Although different forms of ULP may reduce the probability of loss of important video data, they cannot eliminate such possibility. In the case of intelligent buffer control, the flow of video packets still encounters some loss within the network. The FEC-based ULP is not immune to the loss of important video data either, especially in the case of mismatch between actual channel parameters and those used in the FEC code design. In such cases it is beneficial that the decoder perform some form of error concealment on the corrupted part of the video data. We demonstrated in [25] that combining FEC and error concealment increases robustness of the system to channel parameter mismatch. We have recently developed [27] a motion-compensated (MC) error concealment strategy for the Embedded Zero Block Coded (EZBC) [28] video. The basic concept is shown in Figure 18 on the next page.

This figure shows three groups of pictures (GOPs) of MC-EZBC, a motion-compensated subband/wavelet video coder, here with two levels of temporal decomposition. The original or input frames are in the top level. Low temporal frequency frames are shown as solid lines, and high temporal frequency frames are shown as dashed lines. Motion vectors are shown as solid arrows. Suppose that GOP n is corrupted through the loss of spatial data (lowest temporal frequency frame and high temporal frequency frames), which constitute most of the data for a typical GOP. In this example, forward motion compensation is used so that lowest temporal frequency frame is temporally aligned with (and looks similar to) the first frame in the GOP. We therefore use the motion field and the last frame of GOP $n-1$ to estimate the lowest temporal frequency frame of GOP n, and then use the motion vectors of GOP n to recover the frames through motion compensated temporal synthesis. Had backward motion compensation been used, the lowest temporal frequency frame would have been temporally aligned with the last frame of the GOP, so we would use the data from GOP $n+1$ for estimation.

While it is commonly assumed that motion compensation makes a compressed video more fragile and hence less robust. This example shows that a good motion field can add robustness too. Motion fields that are used in scalable MCTF type coders must be good in order to perform well in frame-rate scalability. This is because the MCTF actually filters along these temporal motion paths. If they are not good, then the lower frame-rate video will suffer from artifacts. So a well operating MCTF type coder will have this more reliable type of motion field.

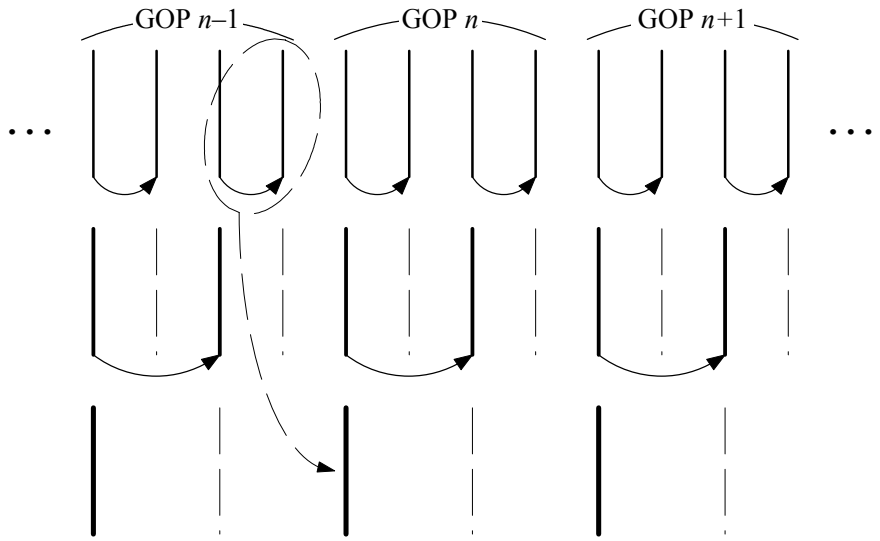


Fig. 18: Three consecutive GOPs of an MC subband/wavelet video coder

The method relies on the availability of motion vectors from the corrupted GOP. However, this is not a very hard requirement to satisfy, since motion vectors typically constitute a relatively small portion of the GOP data (10% – 20%). Hence, they may be heavily protected for transmission, or subject to re-transmission as needed. The scalable coders based on MCTF, such as MC-EZBC, have a much higher quality motion field than do typical MPEG hybrid coders, and this feature is used to advantage here.

We found that this type of error concealment significantly improves the quality of corrupted GOPs. The method works reasonably well for sequences with moderate motion content. Some PSNR results are shown in Table 7 for *Mobile calendar* sequence (SIF resolution, 30 fps) and *Foreman* sequence (CIF resolution, 30 fps). Both were encoded at 1 Mbps and in both cases they were "corrupted" by decoding the second GOP at a very low rate of about 50 kbps. Visual performance is illustrated in Figure 19 for the *Mobile calendar* sequence. Corresponding video clips may be found at the site [29]. A journal article on this work has been subsequently completed and submitted [40].

Sequence	No concealment	MC error concealment	Gain
Mobile Calendar	13.4	19.5	+6.1
Foreman	14.7	27.4	+12.7

Table 7: PSNR results (in dB) for MC error concealment of corrupted GOPs.

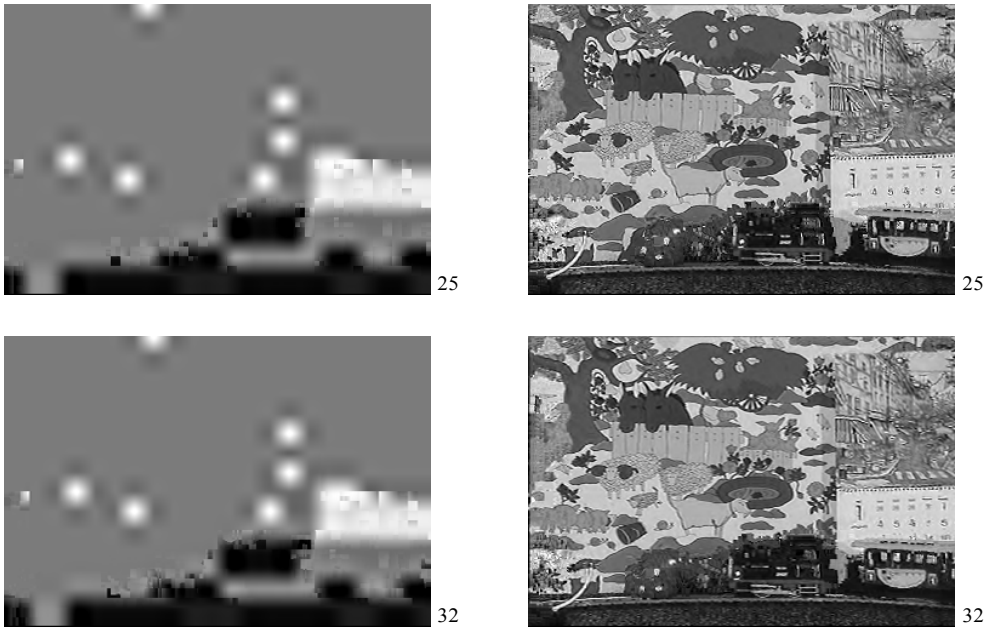


Fig. 19: Frames without error concealment (left) and with MC error concealment (right)

8. Hybrid video streaming/downloading

Video streaming over the current best-effort Internet is challenging due to factors such as the required high bit rate, delay, and loss sensitivity of video data. The traditional client/server based video streaming structure may suffer from video quality degradation when the network is congested. With the pervasive deployment of computers and networks, more and more resources (computation, bandwidth, and storage) are available over the Internet. How to efficiently use these resources to assist video streaming is a challenging problem, also due to the dynamic and heterogeneous characteristics of the supplying peers. Our proposed hybrid video downloading/streaming (HDS) architecture is shown as Figure 20. The HDS scheme efficiently integrates a traditional client/server based video streaming system (streaming mode) and a peer-to-peer based media data distribution system (downloading mode), in order to reduce the server load and increase the availability of video data on the receiver side. In our hybrid architecture, the two modes complement each other. To simplify the description, we outline our scheme with one video server, one "supplying" peer (i.e. with cached content), one "requesting" peer, and constant bit-rate (CBR) video.

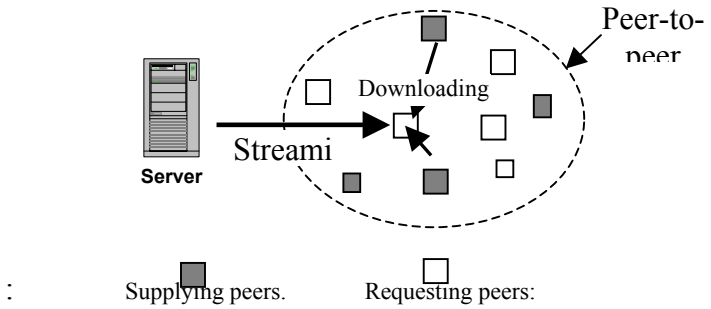


Fig. 20: Hybrid video downloading/streaming architecture

The building blocks of our HDS system include:

(1) a receiver-driven coordination control scheme (RDCC) and a memory disk cooperative buffering (MDB) scheme at requesting peer;
(2) a scheduler at the video server; and
(3) a scheduler at the supplying peer.

These building blocks efficiently cooperate to reduce the server load and maximize the availability of video content on the receiver side.

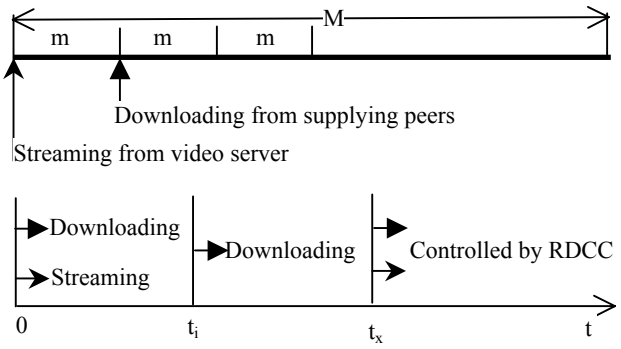


Fig. 21: HDS Timing behavior diagram

Whenever a receiver (requesting peer) decides to watch a video clip, it first sends out a request to the video server. The video server can be a traditional video server or a content distribution network (CDN) server. In the meantime, the requesting peer performs a content lookup service on a peer-to-peer network, such as *Chord*, to find the supplying peers who currently have the requested video content. After the receiver gets the video profile from the server and the addresses of the supplying peers from the network, it analyzes the video content according to the video profile and segments the video content into *N* slices as seen in Figure 21 (upper subfigure). We define a *slice* as a contiguous segment of video data in a video bit stream; one slice may include several video frames. The receiver runs the proposed RDCC algorithm to coordinate downloading and streaming modes described in Figure 21 (bottom subfigure). The RDCC algorithm is the key part of the overall HDS scheme. It computes the availability of the video content in the receiver buffer, estimates the available link bandwidths, and coordinates downloading and streaming modes. Figure 22 shows the server load using our HDS scheme, which is significantly reduced. A conference paper has been published on this work [30].

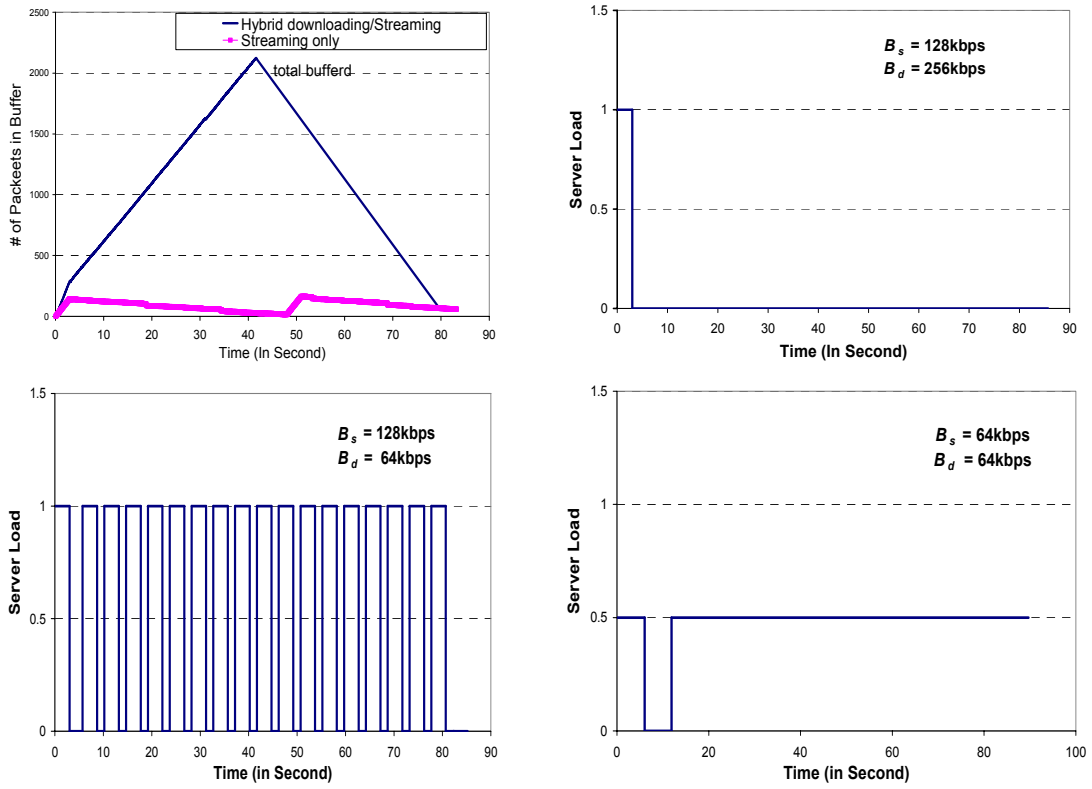


Fig. 22: (a) top left, buffer occupation; (b) top right, (c) bottom left, (d) bottom right: server load in different available bandwidth (The test video is *Foreman*, QCIF format, H.263+ CBR encoded, average bit rate is at B = 128 kbps. The length of the sequence is 81 seconds. We define the server load as "1" if the server sends video at 128 kbps and as "0.5" at 64 kbps)

9. Interaction with MPEG on Multimedia Testbed

In this section we describe our MC-EZBC coding results [32] and the details of experiments as requested in Annex D of the MPEG call for evidence [34] on scalable video coding. In particular we look at Test 2 of this call, which involved simulation on their multimedia test bed [33]. The purpose of the test was to determine how various coding algorithms can respond to variable network bandwidth in a repeatable test environment.

Our submitted MC-EZBC bit stream consisted of two portions:

1. Initial non-scalable portion containing headers and motion vectors – for the sequences in Test 2 this portion takes 40 – 130 kbps.
2. A fully embedded portion containing progressively coded subband/wavelet coefficients.

Since the second portion of the bit stream is fully embedded, we were able to maintain the bit rates at virtually exact values as required. For example, for the target rate of 1024 kbps, the actual rate turns out

to be 1024.0575 kbps, within 0.006% of the target; for the target rate of 256 kbps, the actual rate turns out to be 256.0575 kbps, within 0.03% of the target.

We used a GOP size of 32 frames, which, at 30 fps translates to 1.067 seconds. In a sequence of 900 frames, there are 28 GOPs of size 32, and the last GOP has 4 frames.

Encoding and decoding delay are both equal to 1 GOP, which in our case is 32 frames. For each sequence, encoding takes about 3 hours on a 2.4 GHz Pentium 4 machine with 512 MB of RAM under Windows XP. Decoding takes about 10 minutes.

Algorithm Complexity

Motion estimation is done through hierarchical variable-size block matching (HVSBM), with block sizes varying between 64×64 and 4×4. The search range for block matching is –10 to +10 at the top level of the temporal pyramid, and it approximately doubles for each temporal level down. Motion estimation is bi-directional, using one previous reference frame and one future reference frame. For a given block, if a good match is not found in the future frame, we search the past frame, and the better of the two matches is chosen as the reference block.

Scene changes are detected using the fraction of unconnected pixels – if there are too many unconnected pixels, MCTF is not performed on that pair of frames and a scene change is declared. The fraction of unconnected pixels needed to declare a scene change was set to 0.3, except for the sixteenth and twentieth GOP of *MadCyclist* (frames 481 – 512 and 609 – 640, respectively), where it was set to 0.2. We found that compared to the original threshold value of 0.3, the new setting of 0.2 in these segments improved the visual performance by removing the artifacts of block-based motion compensation, but on the other hand it lowered the PSNR in these segments by about 0.5 dB.

Coding of one GOP requires that all frames of the GOP be in the memory. The amount of memory (in bytes) needed for storage of YUV 4:2:0 data is

$$1.5 \times |\text{GOP}| \times \text{sizeof(float)} \times \text{frame\_width} \times \text{frame\_height},$$

where |GOP| is the number of frames in a GOP, and sizeof(float) is the number of bytes in a floating-point number, in our case 4.

The only non-scalable portion of the bit stream is the initial portion containing the header information (frame size, frame rate, etc.) and motion vectors. For the sequences in Test 2, this non-scalable part is worth 40 - 130 kbps, depending on the complexity of the motion. The remaining part of the bit stream is fully embedded, and decoding can stop at any point.

To make as much use of the test bed functionalities developed for MPEG-4 FGS as possible, we decided to split the MC-EZBC bit stream into two layers:

- Base Layer (BL), which contains the non-scalable portion of the bit stream (headers and motion vectors), and
- Enhancement Layer (EL), which contains the scalable portion of the bit stream (progressively encoded subband/wavelet coefficients).

In accordance with the strategy used in MPEG-4 FGS streaming, re-transmissions are used for BL, but not for EL. For the sequences in Test 2, BL takes 40 - 130 kbps, depending on the motion complexity.

We use the original rate adaptation provided in the test bed by the MPEG testbed **Streamer** object. To perform rate adaptation, it is essential to assign a time stamp to each application packet. Rate adaptation is then performed by the **Streamer** based on the application packet size and its time stamp. However, in a temporal pyramid used in MC-EZBC, data may correspond to different time scales and need not be localized in time. This section explains how we assign time stamps to MC-EZBC data.

At the start of a streaming session, time stamps for both BL and EL are initialized to zero. Recall that BL, apart from a small header, contains mostly motion vector (MV) fields. If there are |GOP| frames in a GOP, there will be |GOP|−1 MV fields in that GOP: |GOP|/2 at the top level of the temporal pyramid, |GOP|/4 at the next temporal level down, and so on, until we get to a single MV field at the bottom. Hence, in every GOP the number of MV fields is one less than the number of frames. We have a running MV field counter whose function is similar to the existing **m_uiFrameCount** counter in the **MediaDatabase** object - it counts the MV fields that have been processed (packetized and sent). This counter is incremented by one at the start of each GOP, to account for the difference between the number of frames and the number of MV fields in a GOP, and it is also incremented by one for each processed MV field. Analogously to the FGS case, when the counter value becomes equal to the frame rate specified by **m_uiFrameRate,** meaning that one second worth of data has been processed, the time stamp is incremented and the counter is reset to zero. The process repeats until the end of the streaming session.

In the EL, time stamps are assigned based on the number of bytes processed. If the EL for a given GOP has *N* bytes, then on average each frame is worth *N*/|GOP| bytes. The counter for EL increments by one for each chunk of *N*/|GOP| bytes processed. As above, when the counter becomes equal to the frame rate, the time stamp is incremented and the counter is reset to zero. The process repeats until the end of the streaming session.

9.1 Results

This section contains the results of our experiments with the MC-EZBC coder. We report three sets of results in this section: constant bit rate results, "ideal" streaming results, and test bed streaming results. All the results are reported in terms of the Y-component PSNR (denoted PSNR_Y) and the Mean PSNR, defined as

$$\text{Mean PSNR} = (4\times\text{PSNR\_Y} + \text{PSNR\_U} + \text{PSNR\_V})/6.$$

Constant Bit-Rate Results

For each of the sequences of Test 2, the constant bit rate (CBR) results are compiled from four streams with the following rates: 256 kbps, 512 kbps, 768 kbps and 1024 kbps. These are summarized in Tables 8 and 9.

Sequence	Coder	256 kbps	512 kbps	768 kbps	1024 kbps
BigShips	MPEG-4 FGS anchor	33.90	34.79	35.66	36.40
	MC-EZBC	35.28	38.66	40.66	42.12
Crew	MPEG-4 FGS anchor	32.95	34.04	35.14	36.11
	MC-EZBC	33.78	36.82	38.71	40.13
MadCyclist	MPEG-4 FGS anchor	26.26	27.40	28.55	29.61
	MC-EZBC	26.75	30.31	32.30	33.86

Table 8: PSNR_Y in dB.

Sequence	Coder	256 kbps	512 kbps	768 kbps	1024 kbps
BigShips	MPEG-4 FGS anchor	36.02	36.73	37.43	38.05
	MC-EZBC	37.56	40.97	42.86	44.22
Crew	MPEG-4 FGS anchor	34.89	35.83	36.79	37.65
	MC-EZBC	35.43	38.34	40.08	41.38
MadCyclist	MPEG-4 FGS anchor	29.02	29.92	30.89	31.79
	MC-EZBC	29.27	32.50	34.28	35.67

Table 9: Mean PSNR in dB.

"Ideal" Streaming Results

"Ideal" streaming refers to the idealized situation where the encoder/server has the knowledge of the future channel conditions (available bit rate) ahead of time. In this case it is assumed that the sending rate is maintained exactly at the level provided by the network, and there are no losses in the network. To obtain the results for "ideal" streaming, we pull the appropriate segments from the CBR streams in the previous section: for the initial 5 seconds we use a 1024 kbps stream, for the next 5 seconds we use the 768 kbps stream, and so on. The PSNR results for these segments are then concatenated to produce the result for "ideal" streaming. Average values of PSNR_Y and Mean PSNR are shown in Tables 10 and 11, respectively.

Sequence	MPEG4-FGS anchor	MC-EZBC
BigShips	35.16	39.30
Crew	34.48	37.63
MadCyclist	27.79	30.99

Table 10: PSNR_Y in dB.

Sequence	MPEG4-FGS anchor	MC-EZBC
BigShips	37.02	41.49
Crew	36.17	39.02
MadCyclist	30.26	33.09

Table 11: Mean PSNR in dB.

Actual Test Bed Streaming Results

For each sequence we used one MC-EZBC stream at 1024 kbps at the server. This rate corresponds to the maximum transmission rate provided by the network profile. The network profile is as recommended by the call for evidence: "Bit-rates are changing each 5s from 1024 Kbps to 768 Kbps, 512 Kbps, 256 Kbps, 512 Kbps, and 768 Kbps."

Table 12 contains the average PSNR_Y of decoded videos. Frame-by-frame PSNR_Y is plotted in Figs. 23a-c. The results in terms of the Mean PSNR are shown in Table 13 and Figs. 23d-f.

Sequence	MPEG4-FGS anchor	MC-EZBC
BigShips	35.18	37.84
Crew	34.44	36.29
MadCyclist	27.87	29.44

Table 12: PSNR_Y in dB.

Sequence	MPEG4-FGS anchor	MC-EZBC
BigShips	37.04	40.05
Crew	36.13	37.77
MadCyclist	30.32	31.68

Table 13: Mean PSNR in dB.

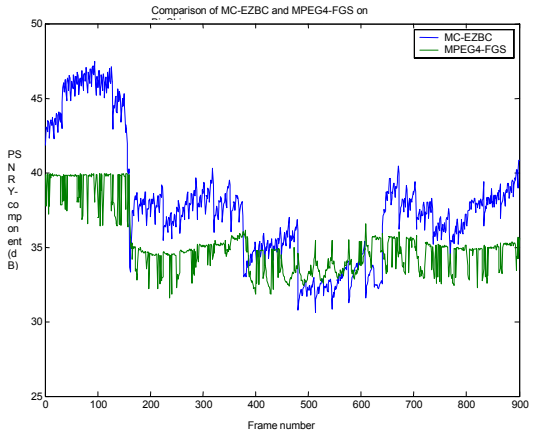


Figure 23a: PSNR_Y in dB for *BigShips*.

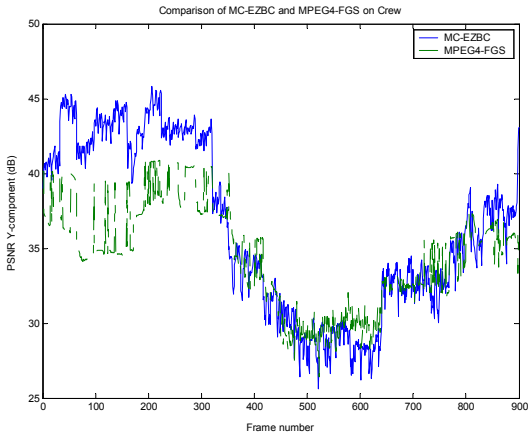


Figure 13b: PSNR_Y in dB for *Crew*.

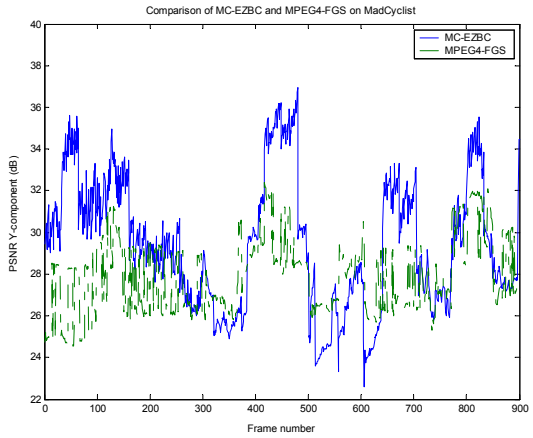


Figure 23c: PSNR_Y in dB for *MadCyclist*.

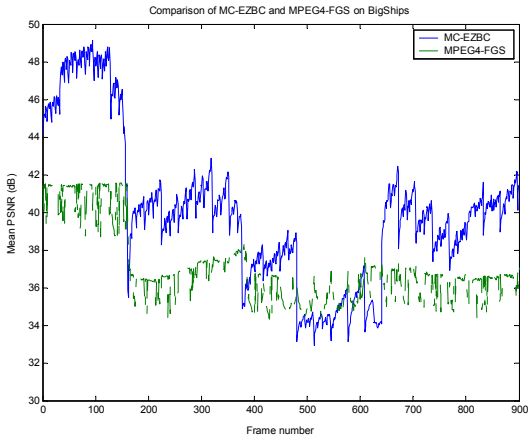


Figure 23d: Mean PSNR in dB for *BigShips*.

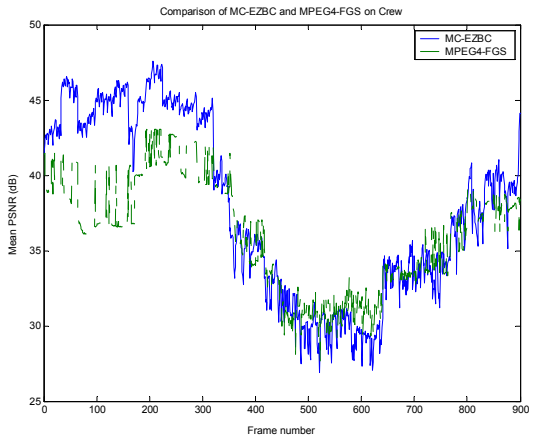


Figure 23e: Mean PSNR in dB for *Crew*.

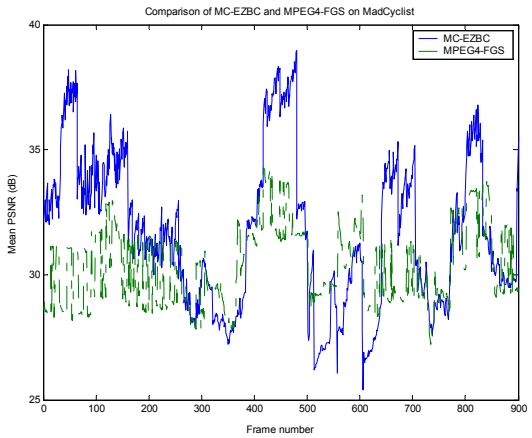


Figure 23f: Mean PSNR in dB for *MadCyclist*.

10. Network Path Aggregation

In this section we present an efficient multiplexing and error control system to improve the streaming video performance over path aggregates. While providing the application with increased aggregate bandwidth, the scheme reduces the performance degradation due to high path latencies and loss rates. The reduction in effective loss and delay is achieved by smart multiplexing and by exploiting the high latency paths to the user's advantage. A novel out-of-sequence transmission algorithm utilizes the higher latency paths to transfer suitable frames from within the transmit buffer. We also present an FEC strategy for our scheme that decouples the transmission of error correction frames from the associated data. This provides protection against correlated losses. Our scheme, while not completely optimized, provides high gains at a considerably lower complexity compared to other approaches.

Multimedia transmission over the Internet has gathered considerable interest in recent years. Apart from the unreliability of the best-effort Internet, the limited bandwidth and path latencies also pose major hurdles to efficient multimedia transport. Previous studies have shown that the use of path aggregates can overcome the bandwidth deficiency [35-38]. It is important to consider the effect of a path diversity scheme on performance with respect to a change in degree of diversity (path loss, delay, bandwidth, number of paths etc.). Multimedia transmission is highly susceptible to path loss characteristics and latencies. Thus, it is important that a path aggregation scheme not be limited to efficient bandwidth aggregation but must also optimize the end user experience. We achieve this by jointly reducing the effective loss rate and latency/jitter in addition to maximizing transmission rate. Our scheme, although not optimal, increases the video performance by an order of magnitude while keeping the implementation complexity much lower, with respect to an optimized approach [38].

We present a novel partitioning scheme that takes advantage of unequal packetization of the video and also uses an out of sequence transmission scheme to utilize high latency paths. Conventionally, the transport level optimizations have been decoupled from the application level information. Our scheme demonstrates that better streaming performance is achieved by using a content aware transport scheme. To deal with the short-term temporal dynamism of the best-effort path characteristics, we use a TCP friendly congestion response scheme on every path that is suited to multimedia delivery [39]. We refer to our scheme as *Smart Multi-path Capacity Aggregation* (SMCA**)**. An extended version of SMCA utilizes the path loss information to generate FEC and to decouple the FEC packets from the associated video packets over the path aggregates.

10.1 Details

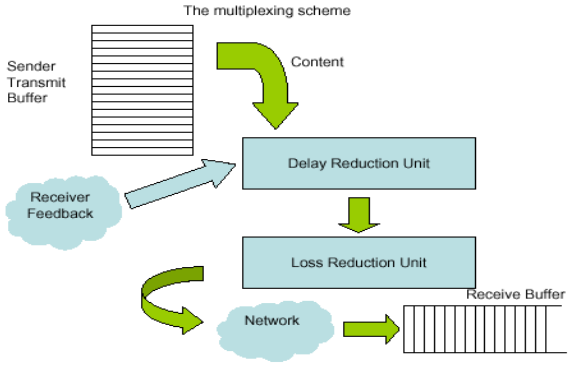


Fig. 24 illustration of SMCA system architecture

The SMCA architecture is represented in Fig. 24. The sender transmit buffer is filled by the application with video packets in a serial fashion. The packets corresponding to the frame to be transmitted/decoded earliest occupy the head of the buffer. The SMCA scheme is used to *choose* frames for transmission from the transmitter buffer depending on both the content information and the network feedback. There are two main stages in mapping the video frames to the appropriate paths. The first stage assigns individual GOPs to a set of paths under delay constraints. T he delay reduction unit in Fig. 24 represents this stage. The second stage protects each frame with FEC and then maps each video and redundancy frame to a path using the content information. The loss reduction unit in the figure represents this second stage. The delay reduction unit is so named because it helps minimize effective delay by using out-of-sequence transmissions on high latency paths. Similarly, content based multiplexing of frames and error correction reduces the impact due to the network losses (hence the name loss reduction unit).

Delay Reduction Unit:
The delay reduction unit minimizes the effective delay experienced by the video data by taking advantage of available high latency paths. These high latency paths, which otherwise are not suitable for video transmission, are used to transmit frames that are to be decoded later in time. The delay reduction unit ranks the paths according to latency and selects appropriate data suitable to be transmitted on each path. The process results in a direct increase in the available bandwidth for the application. Some frames higher up in the transmit buffer may be transmitted before the frames preceding them due to the suitability of available high latency links. Thus a non-sequential transmission of video frames is realized that utilizes the normally unsuitable links to provide reduction in jitter and increase in the transmission rate.

Loss Reduction Unit:
The delay reduction unit associates each frame with a set of paths suitable for its transmission. The loss reduction unit uses the frame content information and the path loss characteristics to determine the exact path a frame will be transmitted on. The mapping of the frames to paths is based on heuristics. Results show that the heuristic based loss reduction unit along with the delay reduction unit can achieve close to optimal performance at a much lower complexity. Higher importance content is transmitted on the paths with low loss rates while the lesser important content is transmitted on paths with higher loss rates. The loss reduction unit is integrated with an optional FEC module that unequally protects the video according to the current loss characteristics. The FEC frames are decoupled from the associated data frames for transmission providing an additional degree of robustness to correlated network losses.

10.2 Results

Tables 14-16 show the gains in PSNR achieved by SMCA when compared with an opportunistic packet-mapping scheme (OPMS) and a completely optimized pruned tree (PT) packet multiplexing scheme [38]. The simulations were performed using the UC Berkeley *ns-2* simulator. The topology used consisted of 5 independent network paths between the source and the receiver. The paths were populated by the video traffic under consideration plus background data traffic. We used the *Flower Garden* video test sequence in the simulations. The sequence was SIF resolution (352 x 240 pixels) at 30 fps. The Flower Garden sequence was encoded using a standard H.26L encoder. The GOP had 16 frames in the following order: IBBPBBPBBPBBPBBP. The bit-rate was 1.7 Mbps and the average packet size was 700 Bytes.

Table 14 presents the gains in PSNR achieved with a changing number of paths. The substantial gain of more than 7 dB when the network resources are diversified among 5 paths shows that SMCA uses path diversity to the user's advantage. The total number of paths was varied from one path (n = 1) to five paths (n = 5) while keeping the aggregate bandwidth fixed at 1.3 Mbps, average loss probability fixed at 0.1, and average path delay fixed at 30 ms.

The gains in performance with varying loss characteristics are presented in Table 15. The average path delay was set at 30 ms. All the PSNR values are averaged over 30 simulation runs. We note that the SMCA scheme performs much better than the OPMS under conditions of high average path loss. The gains in performance with varying delay characteristics are shown in Table 16. The average path loss probability was set at 0.1. Again, all the PSNR values are averaged over 30 simulation runs. We note that the SMCA scheme performs much better than the OPMS under conditions of high average delay. Under both conditions the performance of SMCA is comparable to the optimized PT approach, but SMCA achieves this performance at a much lower complexity.

Num. Of Paths	1	2	3	4	5
PSNR (dB)	20.98	22.48	25.42	26.02	28.04

Table 14. Average PSNR variation with number of paths

Avg. Loss Prob.	**SMCA PSNR(dB)**	**PT PSNR(dB)**	**OPMS PSNR(dB)**
0.05	29.32	31.82	26.06
0.1	29.03	29.02	24.43
0.35	26.32	26.86	18.21
0.4	22.78	20.31	11.64

Table 15. Gains with loss variation

Avg. Delay (ms)	**SMCA PSNR(dB)**	**PT PSNR(dB)**	**OPMS PSNR(dB)**
30	30.12	31.83	27.96
50	28.32	29.46	24.33
100	25.12	24.21	19.19
300	21.78	18.73	11.03

Table 16. Gains with delay variation

(6) Listing of Publications and Reports

(a) journal articles

1. I. V. Bajic and J. W. Woods, "Maximum minimal distance partitioning of the Z^2 lattice," *IEEE Trans. Inform Theory*, vol. IT-49, April 2003, p. 981-992.
2. I. V. Bajic and J. W. Woods, "Domain-based multiple description coding of images and video," *IEEE Trans. Image Processing*, vol. IP-12, October 2003, p. 1211-1225.

(b) conference articles

1. I. V. Bajic and J. W. Woods, "EZBC video streaming with channel coding and error concealment," to appear in *Proc. SPIE VCIP'03*, Lugano, Switzerland, July 2003.
2. I. V. Bajic and J. W. Woods, "Concatenated multiple description coding of frame rate scalable video," *Proc. IEEE ICIP'02*, vol. II, pp. 193-196, Rochester, NY, September 2002.
3. I. V. Bajic and J. W. Woods, "Domain-based multiple description coding of images and video," *Proc. SPIE 4671 (VCIP'02)*, pp. 124-135, San Jose, CA, January 2002.
4. Y. Shan and S. Kalyanaraman, "Hybrid video streaming/downloading over peer-to-peer networks," *Proc.IEEE ICME 2003*.
5. I.V. Bajic, O. Tickoo, A. Balan, S. Kalyanaraman, and J. W. Woods, "Integrated end-end buffer management and congestion control for scalable video communications," *Proc. ICIP 2003*, Barcelona, Sept. 2003.

(c) meetings and reports

1. I. V. Bajic, *Robust Subband/Wavelet Coding and Transmission of Images and Video*, PhD thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, August 2003.
2. A. Golwelkar, I. Bajic, and J. W. Woods, "Response to Call for Evidence on Scalable Video Coding," ISO/IEC JTC1/SC29/WG11, MPEG2003/m9723, July 2003, Trondheim, NO.

(d) submitted

1. I. V. Bajic, O. Tickoo, A. Balan, S. Kalyanaraman, and J. W. Woods, "Integrated end-end buffer management and congestion control for scalable video communications," submitted to *IEEE Trans. Multimedia*, October 2003.
2. I. V. Bajic and J. W. Woods, "MC-EZBC Video Streaming with Unequal Loss Protection and Motion-Compensated Error Concealment," submitted to *IEEE Trans. Image Processing*, April 2004.

(7) List of participating scientific personnel

Co-Principal Investigators: John W. Woods and Shivkumar Kalyanaraman

Graduate Research Assistants: Ivan Bajic, Anand Balan, Omesh Tickoo, and Yufeng Shan

Also supported graduate student A. M. Chaudry obtained MS degree in Dec. 2000

Ivan Bajic obtained his PhD degree in August 2003.

(8) Report of Inventions

There are some invention disclosures and having to do with scalable video coding, as part of our work with the MPEG standardization effort. However none of these are directly the result of the robustness research we have performed on this grant.

(9) Bibliography

1. S-J. Choi and J. W. Woods, “Motion-compensated 3-D subband coding of video,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 8, pp. 155-167, Feb. 1999.
2. S-T. Hsiang and J. W. Woods, “Invertible three-dimensional analysis/synthesis system for video coding with half-pixel-accurate motion compensation,” in *Proc. SPIE 3653, Visual Communications and Image Processing '99*, Jan. 1999.
3. P. Chen, *Contributions on Image and Video Coding*, PhD Thesis proposal, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, Nov. 2000.
4. I. V. Bajic and J. W. Woods, “Maximum minimal distance partitioning of the Z^2 lattice,” *IEEE Trans. Inform. Theory*, vol. IT-49, April 2003, p. 981-992.
5. T. Naveen and J. W. Woods, “Subband Finite State Scalar Quantization,” *IEEE Trans. Image Processing* , January 1996.
6. Domain Based Multiple Description: http://www.cipr.rpi.edu/~ivanb/mcvideo.htm
7. I. V. Bajic, *Robust Coding and Packetization of Images and Intraframe-coded Video*, Masters Thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, August 2000.
8. I. V. Bajic, J. W. Woods, and A. M. Chaudry, “Robust transmission of packet video through dispersive packetization and error concealment,” in *Proc. Packet Video Workshop (PV 2000)*, Sardinia, Italy, May 2000.
9. S-T. Hsiang and J. W. Woods, "Embedded video coding using motion compensated 3-D subband/wavelet filter bank," in *Proc. Packet Video Workshop (PV 2000)*, Sardinia, Italy, May 2000.
10. S. Floyd, et al, "Equation-based Congestion Control for Unicast Applications," *Proc. of SIGCOMM'2000*, Stockholm, August 2000.
11. D. Bansal, H. Balakrishnan, "Binomial Congestion Control Algorithms," *Proc. of INFOCOM 2001*, Anchorage, AK, April 2001.
12. I.V. Bajic, O. Tickoo, A. Balan, S. Kalyanaraman, and J. W. Woods, “Integrated end-end buffer management and congestion control for scalable video communications,” *Proc. ICIP 2003*, Barcelona, Sept. 2003.
13. Y. Shan and S. Kalyanaraman, “Hybrid video streaming/downloading over peer-to-peer networks,” *Proceedings of IEEE ICME 2003*.
14. Video Test Sequences: http://networks.ecse.rpi.edu/~balana/sequences
15. K. Chandrayana, S. Ramakrishnan, B. Sikdar, S. Kalyanaraman, A. Balan, O. Tickoo, ``On Randomizing the Sending Times in TCP and other Window Based Algorithms," to appear in *Journal of Computer Networks*, 2004.
16. K. Chandrayana, B. Sikdar and S. Kalyanaraman, ``Comparative Study of TCP Compatible Binomial Congestion Control Schemes," *Proceedings of IEEE HPSR*, Kobe, Japan, May 2002.
17. V. Stankovic, R. Hamzaoui, and Z. Xiong, “Packet loss protection of embedded data with fast local search,” *Proc. IEEE ICIP 2002*, vol. 2, pp. 165-168, Rochester, NY, September 2002.
18. I.V. Bajic and J. W. Woods, “Concatenated multiple description coding of frame-rate scalable video,” *Proc. IEEE ICIP 2002*, vol. II, pp. 193-196, Rochester, NY, September 2002.
19. I. V. Bajic and J. W. Woods, “Domain-based multiple description coding of images and video,” *Proc. SPIE 4671 (VCIP’02)*, pp. 124-135, San Jose, CA, January 2002.
20. I. V. Bajic and J. W. Woods, “Domain-based multiple description coding of images and video,” *IEEE Trans. Image Processing*, vol. IP-12, October 2003, p. 1211-1225.
21. I. V. Bajic and J. W. Woods, “Maximum minimal distance partitioning of the Z^2 lattice,” *IEEE Trans. Inform. Theory*, vol. IT-49, April 2003, p. 981-992.

22. R. Puri and K. Ramchandran. "Multiple Description Source Coding Through Forward Error Correction Codes," *Proc. Asilomar'99*, Pacific Grove, CA, October 1999.
23. I. V. Bajic, O. Tickoo, A. Balan, S. Kalyanaraman, and J. W. Woods, "Integrated end-end buffer management and congestion control for scalable video communications," submitted to *IEEE Trans. Multimedia*, October 2003.
24. I. V. Bajic and J. W. Woods, "Domain-based multiple description coding of images and video," *Proc. SPIE vol. 4671 (VCIP'02)*, pp. 124-135, San Jose, CA, January 2002.
25. I. V. Bajic and J. W. Woods, "Concatenated multiple description coding of frame-rate scalable video," *Proc. IEEE ICIP'02*, vol. II, pp. 193-196, Rochester, NY, September 2002.
26. V. Stankovic, R. Hamzaoui, and Z. Xiong, "Packet loss protection of embedded data with fast local search," *Proc. IEEE ICIP'02*, vol. 2, pp. 165-168, Rochester, NY, September 2002.
27. I. V. Bajic and J. W. Woods, "EZBC video streaming with channel coding and error concealment," *Proc. SPIE VCIP'03*, Lugano, Switzerland, July 2003.
28. S.-T. Hsiang and J. W. Woods, "Embedded video coding using invertible motion compensated subband/wavelet filter bank," *Signal Processing: Image Commun.*, vol. 16, pp. 705-724, May 2001.
29. MD-FEC example clips: http://www.cipr.rpi.edu/~ivanb/mcec_videos.htm
30. Y. Shan and S. Kalyanaraman, "Hybrid video streaming/downloading over peer-to-peer networks," accepted for publication, *IEEE ICME 2003*.
31. I. V. Bajic, *Robust Subband/Wavelet Coding and Transmission of Images and Video*, PhD thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, August 2003.
32. A. Golwelkar, I. Bajic, and J. W. Woods, "Response to Call for Evidence on Scalable Video Coding," ISO/IEC JTC1/SC29/WG11, MPEG2003/m9723, July 2003, Trondheim, NO.
33. C.-J. Tsai, M. van der Schaar, and Y.-K. Lim, "Working draft of ISO/IEC TR21000-12 resource delivery test bed," ISO/IEC JTC1/SC29/WG11/N5494, December 2002.
34. "Call for evidence on scalable video coding advances," ISO/IEC JTC1/SC29/WG11/N5559, Pattaya, TI, March 2003.
35. J. G. Apostolopoulos, "Reliable Video Communication over Lossy Packet Networks using Multiple State Encoding and Path Diversity," *Visual Communications and Image Processing,* January 2001.
36. W. Xu, S. S. Hemami, "Efficient Partitioning of Unequal Error Protected MPEG Video Streams for Multiple Channel Transmission," *Proc. IEEE Int. Conf. on Image Processing*, Rochester, NY, Sept. 2002.
37. T. Nguyen and A. Zakhor, "Path Diversity with Forward Error Correction System for Packet Switched Networks," *INFOCOM,* April 2003.
38. W. Xu, S. S. Hemami, "Delay-Optimized Robust Transmission of Image over Multiple Channels," *Proc. IEEE Int. Conf. on Multimedia and Expo,* Baltimore, MD, Jul. 2003.
39. D. Bansal and H. Balakrishnan, "Binomial Congestion Control Scheme," *Proc. of IEEE INFOCOM*, Tel-Aviv, Israel, March 2000.
40. I. V. Bajic and J. W. Woods, "MC-EZBC Video Streaming with Unequal Loss Protection and Motion-Compensated Error Concealment," submitted to *IEEE Trans. Image Processing*, April 2004.