Integrated end-end buffer management and congestion control for scalable video communications

Ivan V. Bajić, Omesh Tickoo, Anand Balan, Shivkumar Kalyanaraman, and John W. Woods

Authors are with the Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA.

This work was supported in part by the ARO grant number 19-00-1-0559

Abstract

In this paper we address the issue of robust and efficient scalable video communication by integrating an end-end buffer management and congestion control at the source with the frame rate playout adjustment mechanism at the receiver. The proposed scheme is based on the observation that when congestion control is implemented at the source, most of the loss occurs at the source and not within the network. Frame rate scalability of encoded video is exploited to match the video transmission rate to the available network rate. The integrated system uses priority information from the encoder and network information from the congestion control scheme to to trade off random loss for the controlled loss of low priority packets. Randomized pacing of packet transmission times reduces the jitter and burst losses suffered by the flow. Based on the overall loss suffered by the flow, frame rate is adjusted at the receiver to maximize the visual quality of the displayed video. We tested our system with both H.26L and a subband/wavelet video coder, and found that it significantly improves the received video quality in both cases.

Index Terms

Buffer management, multimedia networking, robust video communications, scalable video, H.26L.

I. INTRODUCTION

Internet video communications have attracted a lot of research interest in recent years because of the many challenges it poses on the communication system design. Transmission of video typically requires high bandwidth and low delay, while it can tolerate a certain amount of data loss. These requirements are fundamentally mismatched with network protocols such as TCP, that enable lossless data delivery with potentially high delay due to retransmissions. Further, most video coders produce data of varying importance, while networks such as the Internet treat all data equally. To correct this mismatch, several integrated video coding and congestion control approaches have been proposed to simultaneously provide reliable transmission of video and fairness to the competing flows. Some aspects of the interaction between layered video coders and different transport schemes have been studied in [1].

Modern video coders produce layered/scalable bitstreams whose flexibility allows easy adaptation to varying network conditions. Several different forms of scalability are of interest in video communications [2]. In a scalable bitstream, one subset of the bitstream may be used to provide

video at a lower quality (referred to as *signal-to-noise-ratio, or SNR, scalability*), another subset may be used for lower spatial resolution (referred to as *resolution scalability*), and yet another for lower frame rate (referred to as *frame rate scalability*).

When faced with congestion, transmission rate of the video source needs to be reduced. Many proposed schemes for video transmission rate adaptation implicitly or explicitly make use of SNR scalability (e.g. [3], [4], [5], [6]) which favors the reception of lower quality (SNR) video under unfavorable network conditions. While these approaches may posses certain optimality in a rate-distortion sense, they need not produce the best looking video. For example, a recent study of subjective video quality [7] found that in most cases higher quality low frame rate video is preferable to the lower quality full frame rate video. In this work we exploit frame rate scalability for transmission rate adaptation. In our scheme, as the network conditions deteriorate, receiver is more likely to obtain high quality low frame rate video.

We observe that *in a transmission scheme that performs congestion control, most of the packets are dropped at the transmission buffer, while the relative loss inside the network is very low*, as demonstrated in section II-B. Typically, the packets at the transmission buffer are dropped at random by some congestion avoidance mechanism, which makes the loss at the receiver appear to be random. However, by employing intelligent transmission buffer management, random loss can be traded off for controlled loss which may significantly improve the quality of the received video. We propose a simple buffer management scheme implemented at the transmission source which drops low priority[1] packets in response to congestion. The remaining loss in the network may be handled by other means, such as error concealment (which is the approach we take in this paper) or FEC.

The important contributions of this paper are:

- Exploiting frame rate scalability for adaptation to varying network conditions.
- A simple *generic* end-end buffer management scheme that acts as an extension to source coding of video, provides fast adaptation to changing network conditions and converts the random loss a flow suffers to a controlled loss of low priority packets.
- Integration of transmission buffer management and receiver side frame rate adjustment to produce high quality video at the receiver.

[1]In our stystem, priority is related to frame rate.

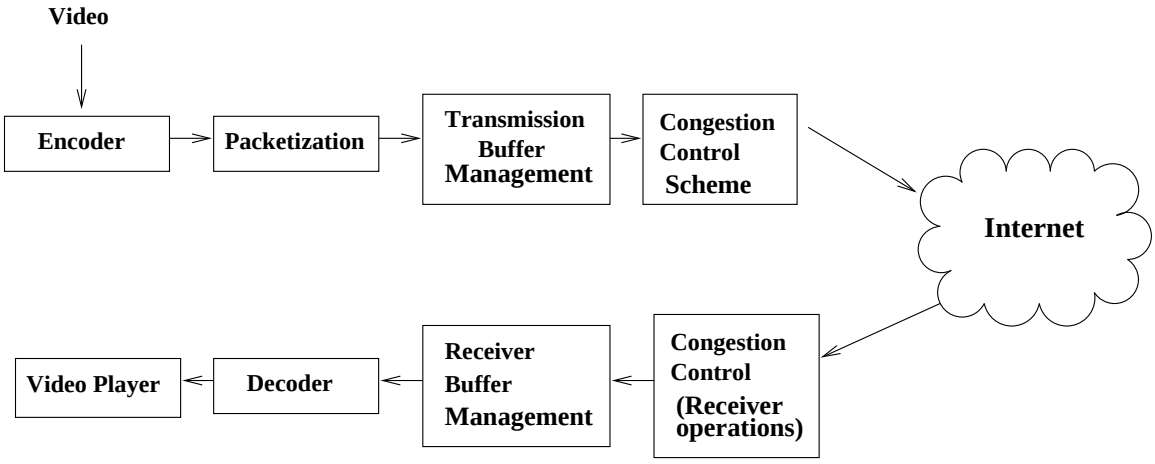


Fig. 1. Video communication system block diagram

- Randomized pacing of packet transmission times which helps reduce jitter effects and correlated losses.
- An integrated video communication system design that produces high quality, low frame rate video in response to congestion.

The rest of the paper is organized as follows. In Section II we present the components of the video communication system, including the two video coders used in our simulations, the proposed buffer management scheme and randomized congestion control. In Section III we describe the video communication system simulation model and present simulation results. The results show significant improvements for both video coders. Conclusions are given in Section IV.

II. VIDEO COMMUNICATION SYSTEM

Figure 1 shows the block diagram of the video communication system. Video is encoded an packetized into individually decodable packets to prevent the propagation of errors caused by the packet loss. On the *large time scale*, at the Group-Of-Pictures (GOP) level, encoder adapts its encoding rate to the current estimate of the available network rate. On the *smaller time scales* (when the GOP is already encoded, but not yet transmitted), the actual transmission rate is regulated by the transmission buffer. The buffer gets feedback from the congestion control scheme about the current network conditions and sends the most important packets within the

available bandwidth. A congestion control scheme serves to minimize burst losses in the network, ensure network stability, and is fair to other flows. At the receiver side, a playout buffer smoothes the flow and reduces jitter. Also, frame rate is adjusted appropriately to improve the quality of the displayed video. Individual components of the system are described in the remainder of this section.

A. Video coding

The generic video communication system presented in this text can utilize any video coding algorithm which produces data of varying importance i.e. different scalability layers. In our experiments we emphasize frame-rate scalability. Results are reported for the recent H.26L video coder [8] and a robust subband/wavelet video coder from [11]. As the results indicate, in both cases buffer management was found to significantly improve the video quality at the receiver, both visually and in terms of the Peak Signal to Noise Ratio (PSNR).

The GOP of the H.26L video coder is similar to that of MPEG and consists of I, P and B frames. I frames are intra-coded without temporal prediction, P frames are coded predictively from the previous I or P frames, while B frames are bi-directionally predicted from the previous and future I or P frames. In this hierarchy, in any given GOP, I frame is the most important, since it is used (directly or indirectly) for predictive coding of all other frames in that GOP. In order of decreasing importance, I frame is followed by P frames, while B frames are the least important. Each frame is divided into blocks of 16×16 pixels, called macroblocks, which are further decomposed by DCT and coded. Consecutive macroblocks (in lexicographic order) are organized into slices. Packetization is performed by the network adaptation layer of H.26L on a slice-by-slice basis. Two packets are produced per each slice: one packet contains header information and motion vector (MV) data, while the other contains coded DCT samples.

The second coder we use in our work is a robust motion compensated (MC) 3-D subband/wavelet video coder from [11]. The typical GOP structure of this coder is shown in Figure 2. The top level of frames represents the video at full frame rate. Neighboring frames are decomposed using a MC filter bank to produce temporal low frequency bands (solid lines) and temporal high frequency bands (dashed lines) at the next level. Motion vectors (MVs) are symbolically shown as arrows. High temporal frequency frames and associated MVs are coded and dispersively packetized as described below. Low temporal frequency bands at the second

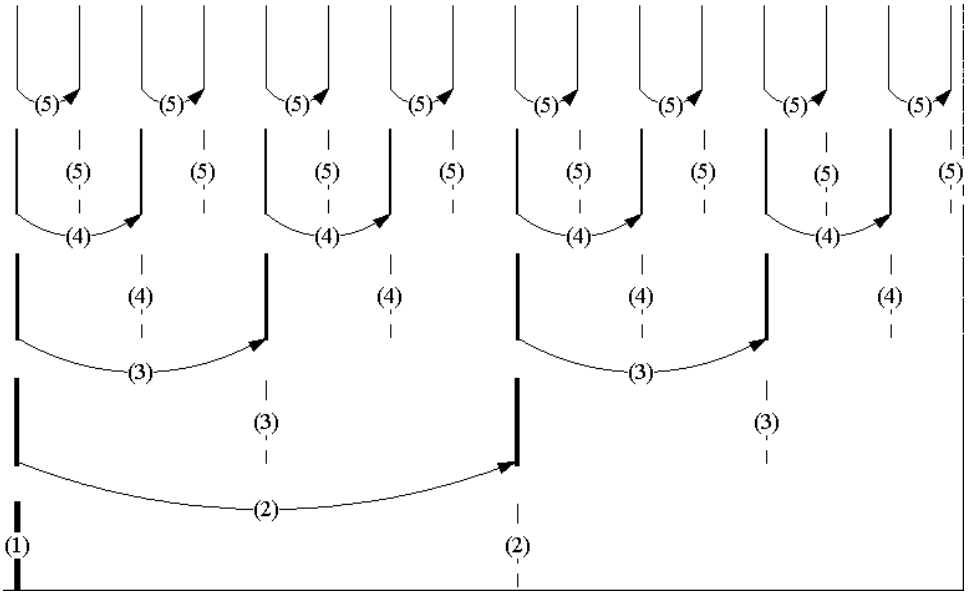


Fig. 2. GOP structure of the MC 3-D subband/wavelet video coder

level occur at $1/2$ of the full frame rate. They are further decomposed to get the video at $1/4$ frame rate, etc. In Figure 2, the last level corresponds to $1/16$ of the full frame rate. Transmitted data in this case is naturally divided into five layers of temporal scalability, labeled (1) through (5) in the figure. Decoders can reconstruct the video at $1/16$ frame rate from layer (1), $1/8$ frame rate from layers (1) and (2), etc.

Each layer is packetized independently so that the video at lower frame rates can be reconstructed from a subset of the packets corresponding to higher frame rates. Within each layer, data is coded and packetized in a dispersive manner, so that subband samples from the common space-time-frequency neighborhood appear in different packets, which enables easy error concealment of lost samples from the available neighboring samples. Also, all the packets from the same 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.

B. Source buffer management

From extensive simulations we infer that for multimedia transmission into a TCP based network, most loss occurs at the point of transmission i.e. the source, and not at the nodes

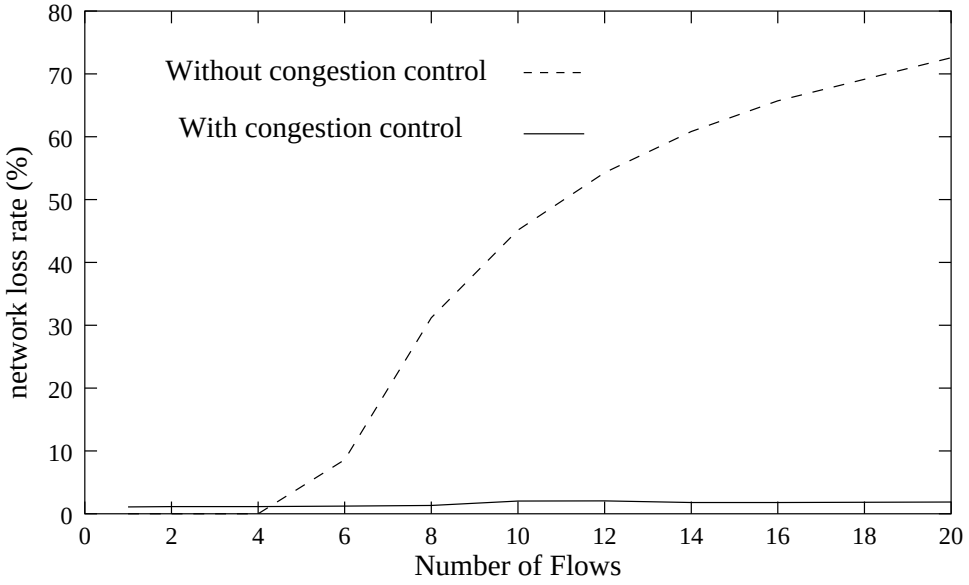


Fig. 3. Comparison of average network loss (%) with and without congestion control.

Algorithm 1 : source buffer management

for Layer k Packet Arrival **do**
  calculate the queue size $s(t)$
  if $s(t) > q_{max}$ **then**
    Drop the packet
  else
    if $s(t) > T_s$ **then**
      Drop the packet with probability $1 - p_k$
    else
      Enqueue the packet
    end if
  end if
end for

inside the network. This is *contrary to the belief* that the packet loss in the network due to congestion is the major contributor to the total loss a TCP flow suffers. Our simulations show that in response to congestion the transmission queues at the sources increase which finally leads to packet drops at the source and it is this dropping at the source that is *the major contributor* to the aggregate loss of the flow. As an example, Figure 3 shows the average loss within the network for $N = 1, 2, ..., 20$ flows through a single bottleneck of 5 Mbps bandwidth. With congestion control (TCP) the loss within the network remains fairly low (below 3%) as the number of flows increases, since most of the packets which violate the available rate constraint are dropped at the source. On the other hand, without congestion control (UDP) the loss within the network increases significantly with the number of flows. This provides a *strong incentive to use a suitable end-end congestion control for video transmission* because the end-end can control the packets that are dropped at the source buffer by employing a suitable packet filter.

The packet filter can be designed in conjunction with the video codec employed. In this section we propose a simple buffer management strategy for video encoded into multiple priority layers. Each GOP is encoded into a number of packets belonging to different priority levels. Packets produced by the encoder are stored in the buffer prior to transmission. The buffer adapts the actual transmission rate to the available network rate which, at the time of transmission, may be different from the encoding rate. When the current network rate $R_n(t)$ is less than the encoding rate R_a, the flow loses packets at the rate of $R_a - R_n(t)$ packets per second. The buffer management algorithm performs a greedy strategy on the packets being transmitted, trying to maximize the quality of the video which can be reconstructed from the transmitted packets. The contribution of each packet to the video quality is indicated by its priority layer.

Algorithm 1 describes the source buffer management operation. When the instantaneous source buffer size $s(t)$ exceeds maximal allowed buffer size q_{max}, any new arriving packet is dropped. On the other hand, when $T_s < s(t) \leq q_{max}$, where $T_s < q_{max}$ is a certain threshold, the selective packet drop policy is enforced, whereby a packet from layer k is dropped with probability $1 - p_k$ (i.e. transmitted with probability p_k). The packet transmission probabilities are calculated as follows. Let there be K_{max} layers ordered in decreasing order of importance, so that layer 1 is the most important layer and K_{max} is the least important. Let r_k be the rate of layer k, so that $\sum_{k=1}^{K_{max}} r_k = R_a$. Initially, $p_k = 1$ for $k = 1, 2, ..., K_{max}$. When $R_n(t) < R_a$, causing the transmission buffer to fill up and exceed the threshold T_s, we find new values of p_k which

Algorithm 2 : transmission probability assignment

for $(k = K_{max}$; $k > 0$ and $\sum_{j=1}^{K_{max}} p_j r_j > R_n(t)$; $k--)$ **do**
  if $\sum_{j=1}^{k-1} p_j r_j > R_n(t)$ **then**
    $p_k = 0$
  else
    $p_k = (R_n(t) - \sum_{j=1}^{k-1} p_j r_j)/r_k$
  end if
end for

satisfy the network rate constraint

$$\sum_{k=1}^{K_{max}} p_k r_k \leq R_n(t). \tag{1}$$

The greedy assignment of transmission probabilities is shown in Algorithm 2. Starting from the least important layer, transmission probabilities are reduced until the network rate constraint is satisfied. It is easy to see that Algorithm 2 assigns probabilities which satisfy the rate constraint (1) with equality. In our case, priority layers match the frame rate scalability layers, so this probability assignment favors the reception of higher quality video at the lower frame rates, rather than low quality video at the highest frame rate.

The choice of the source buffer threshold T_s is important for the overall system performance. Having a small threshold will lead to unnecessary packet drops at the source buffer, while having a large threshold will increase the overall delay and eventually cause the receiver buffer underflow. In the remainder of this subsection we derive the near optimal value for the source buffer threshold T_s.

Assume the packet size is approximately constant over the duration of video transmission. Let D be the GOP size in packets (i.e. without loss, the decoder decodes D packets in one GOP); B be the receiver side pre-buffer size (i.e. the receiver waits for the buffer to have B packets before it starts decoding and playing out the video initially); $s(t)$, as before, be the instantaneous source buffer size (in packets); $b(t)$ be the instantaneous receiver buffer size (in packets); R_a be the application rate (i.e. encoder encodes R_a packets/second and decoder decodes R_a packets/second); $R_n(t)$, as before, be the network rate (in packets/second); and RTT be the Round Trip Time (in seconds).

The overall loss in the video communication system consists of three components: (1) loss at the source (packets dropped due to congestion control), (2) network loss (random packet drops inside the network), and (3) loss at the receiver (due to receiver buffer underflow caused by excessive delay - this loss includes the packets not delivered in time for playback). In our analysis, we consider only the loss at the source and the loss at the receiver, for two reasons: (a) we have no control over the random loss inside the network and (b) as shown before, the network loss is fairly small when proper congestion control is done.

First, consider the case when $T_s = 0$. In this case, any packet arriving from the encoder is subject to packet drop policy (Algorithm 1). If the transmission of the packet would violate the current network rate constraint, the packet is dropped according to the probabilities assigned in Algorithm 2. Hence, whenever the instantaneous network rate falls below the application rate, we encounter some loss at the source buffer. Therefore, in this regime, loss at the source is highest, because buffering the packets which violate the instantaneous network rate would possibly allow for their transmission in the future, and hence would not incrase the loss at the source. On the other hand, the loss at the receiver is minimized due to absence of the delay in the source buffer.

Next, consider what happens when $T_s \rightarrow \infty$ (hence also $q_{max} \rightarrow \infty$). In the limit, there is no loss at the source since the packet drop policy is not enforced for any finite source buffer size $s(t)$ (see Algorithm 1). All the packets produced by the encoder are stored in the source buffer until they can be transmitted. However, in this regime, the delay is unbounded and all the loss is due to receiver buffer underflow.

Based on the analysis above, as T_s increases from 0 to ∞, the loss at the source decreases, while the loss at the receiver increases. The optimal setting for T_s would balance these two types of loss to achieve the overall minimum. We now derive the upper bound on $s(t)$ for which the receiver buffer will not face an underflow in the steady state. A packet that comes to the source buffer from the video application will face the following delays before it is decoded.

$$delay = \frac{s(t)}{R_n(t)} + \frac{RTT}{2} + \frac{b(t)}{R_a}$$

This delay must be less than B/R_a (the initial play-out buffer delay) for the packet to arrive at the decoder in time. Assuming that $\frac{RTT}{2}$ is small compared to other delays (necessary in practice, for avoiding jitter),

$$\frac{s(t)}{R_n(t)} + \frac{b(t)}{R_a} \leq \frac{B}{R_a} \Rightarrow \frac{s(t)}{R_n(t)} \leq \frac{B-b(t)}{R_a}.$$

Source buffering can cause excessive delays only when $R_n(t) \leq R_a$. In this (worst) case, from the previous equation we have $s(t) \leq B - b(t)$. Since the decoder decodes D packets (1 GOP) at at time, receiver buffer will not underflow only if $b(t) \geq D$. Therefore,

$$s(t) \leq B - D. \tag{2}$$

Hence, if the source buffer size is maintained below $B - D$, the receiver buffer will not underflow in the steady state and loss at the receiver will be minimized. On the other hand, loss at the source occurs only when $s(t) > T_s$. Hence, we choose $T_s = B - D$. We point out that this is only a near-optimal value for T_s due to several assumptions used in the derivation. However, it performs very well in practice, as demonstrated in Figure 8 in Subsection III-B.

C. Congestion control

Our congestion control mechanism is based on binomial algorithms [12] coupled with randomized pacing of packet transmission times [13]. As such, it provides smoothly varying transmission rate suitable for video flows, and helps reduce jitter effects. Binomial algorithms and randomized pacing are discussed in the remainder of this subsection.

Binomial congestion control algorithms are very suitable for multimedia transfer. Their advantage is that the reduction in transmission rate upon encountering congestion is not as drastic as the conventional TCP. These algorithms use a generalized form of TCP's additive increase policy by increasing the congestion window in steps inversely proportional to a power k of the current window ($k = 0$ for TCP). They also generalize the TCP's multiplicative decrease policy by decreasing proportional to a power l of the current window ($l = 1$ for TCP). The authors [12] show that if $k + l = 1$, the schemes compete fairly with TCP and thus the class of algorithms is name binomial algorithms. It is further shown that if $k + l > 0$, $k > 0$, and $l > 0$, the binomial schemes converge to fairness under a synchronized feedback assumption. The general increase/decrease equations for binomial algorithms are given as:

- Increase: $w_{t+RTT} \longleftarrow w_t + \alpha/w_t^k; \alpha > 0$
- Decrease: $w_{t+\delta t} \longleftarrow w_t - \beta w_t^l; 0 < \beta < 1$

where w_t is the congestion window size at time t, and α and β are constants.

We show by simulation results that the choice of transport scheme matters in multimedia transmission. The performance of the integrated scheme we present is highly dependent on the congestion control algorithm used by the transport scheme. We report our results with the randomized versions of IIAD (Inverse Increase and Additive Decrease) scheme with $k = 1$ and $l = 0$ in the equations above and AIMD (Additive Increase and Multiplicative Decrease) scheme with $k = 0$ and $l = 1$ in the equations above. The simulation results show a clear improvement in performance with IIAD as compared to AIMD.

The randomization of packet transmission times was first introduced in [13]. The randomization is shown to reduce bias against flows with higher RTTs, window synchronization, phase effects in flows and correlated losses. The randomization does not send back-to-back packets but spaces successive transmissions with a time interval $\Delta = RTT(1+x)/w_t$, where x is a zero mean random number drawn from an uniform distribution.

D. Video decoding and playout

As discussed earlier in the text, the overall loss in the video comunication system consists of the loss at the source, the loss inside the network, and the loss at the receiver (due to receiver buffer underflow). The source buffer management has been designed to minimize the effects of the loss at the source (by dropping least important data first) and to prevent receiver buffer underflow. The remaining loss, i.e. the loss inside the network, is handled by error concealment, whose task is to improve the reconstructed video quality using the available data.

In the case of subband/wavelet video coder, median filtering is employed to recover missing pieces of data. Missing subband samples are estimated as the median of the available neighboring samples, while missing MVs are estimated as a vector median of the available neighboring MVs.

Due to the source buffer management policy, the loss at the receiver is concentrated in the higher enhancement layers, i.e. those corresponding to higher frame rates. If this loss is high, it may be advantageous to reduce the frame rate of the displayed video, since the lower frame rate version is received with lower loss and hence, lower objective distortion (mean squared error - MSE). On the other hand, reducing the frame rate may degrade the subjective quality of high-motion video. We would like to obtain a simple rule which tells us under which loss conditions should the frame rate be reduced for playout at the receiver. In order to model subjective video quality and to come up with such a rule, we indroduce the following simple

and intuitive distortion metric:

$$Distortion = MSE + \lambda \sum_{k=1}^{K_{max}} MSEMV_k, \tag{3}$$

henceforth referred to as the *visual distortion*. In equation (3), MSE is the mean squared error in the decoded frames and $MSEMV_k$ is the mean squared error (per pixel) in motion vectors in layer k (see Figure 2). The error in motion vectors is the difference between the "original" motion vectors (those used for motion compensated temporal filtering) and those used in decoding/playout, which may be degraded due to loss, or set to zero when the frame rate is reduced. For example, if the frame rate is reduced from full to a half, all motion vectors in layer $k = K_{max}$ are set to zero, so in this case $MSEMV_{K_{max}}$ is the mean squared value (average energy) of the "original" motion vectors in layer $k = K_{max}$. Factor λ in (3) serves to balance the two contributing terms and match the quantitative visual distortion in (3) to the subjective distortion. We found experimentally that $\lambda = 10$ was a reasonable choice. To motivate the intuition behind (3), observe the following. If the video contains a static scene (no motion), there is no penalty in reducing the frame rate, since all the motion vectors are zero anyway. The higher the motion in the video, the greater the motion vector energy, and more penalty is introduced when frame rate is reduced.

The following example illustrates the properties of the visual distortion metric (3), and shows how it can be used to derive simple rules for reducing the frame rate at the receiver in case of packet loss. The Football sequance with SIF resolution (352×240) at 30 frames per second (fps) was encoded into seven streams with bit rates between 500 kbps and 2500 kbps. Each stream is then decoded at full, half and quarter frame rate, and visual distortion (3) was computed for all three cases. The plot of distortion versus bit rate is shown in Figure 4. Observe that at higher bit rates, reducing the frame rate degrades the visual quality, while at very low bit rates, reducing the frame rate seems to improve the visual quality. This in accordance with the results of [7], which indicate that at low bit rates, lower frame rates are preferrable by human observers.

The seven streams were then subject to random packet loss in the few highest layers, and visual distortion (3) was computed for full, half and quarter frame rates. Figure 5 shows the plot of distortion versus stream number under five different conditions: (1) full frame rate, no loss; (2) full frame rate, 25% loss in layer 5; (3) half frame rate, no loss; (4) half frame rate, 25% loss in layer 4; (5) quarter frame rate, no loss; As seen in the graph, for all seven streams (i.e.

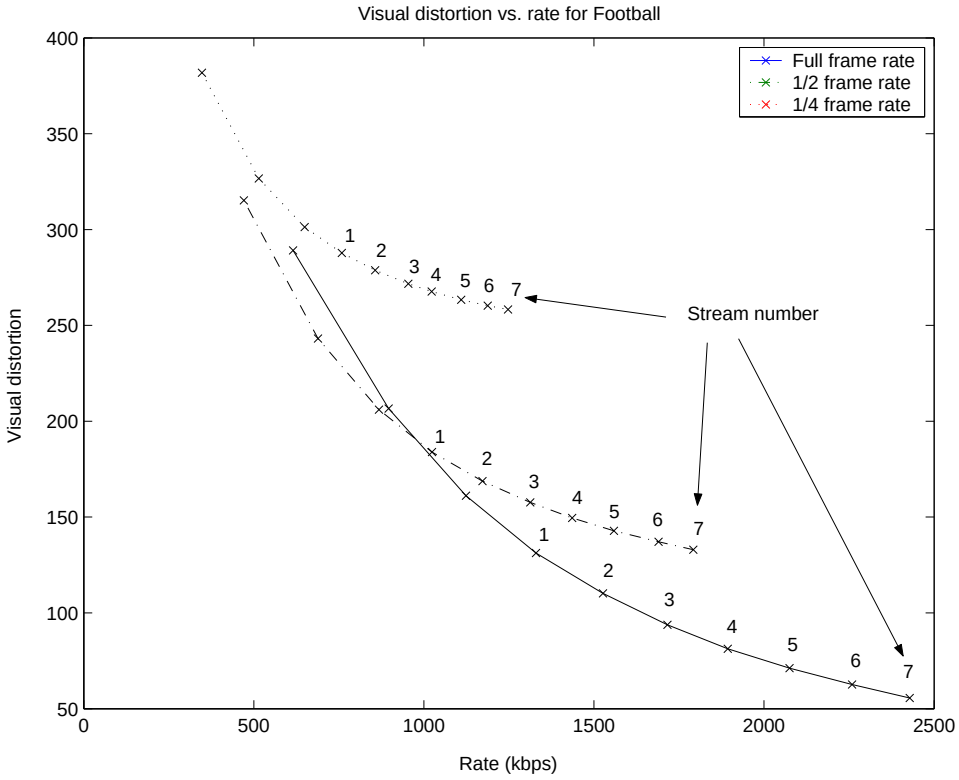


Fig. 4. Visual distortion versus bit rate for Football

across the rage of bit rates), a 25% loss in layer 5 produces video at a full frame rate which is visually inferior to the corresponding video at a half frame rate. Similarly, a 25% loss in layer 4 produces video at a half frame rate which is visually inferior to the corresponding video at a quarter frame rate.

Based on these observations, the simulations with subband/wavelet coder were carried out with the following rule: the frame rate of the decoded/displayed video is reduced until the loss in all the remaining layers is less than 20%. Although this operation does not necessarily reduce the mean squared error (MSE) of the decoded frames, it does tend to improve visual quality of the displayed video.

Error concealment operations performed by H.26L video decoder are specified in [8]. Samples from the missing macroblocks in I frames are estimated as the weighted sum of the samples from the available neighboring macroblocks, as proposed in [9]. For other frames, MV associated with the macroblock is used to copy the corresponding macroblock from the reference frame.

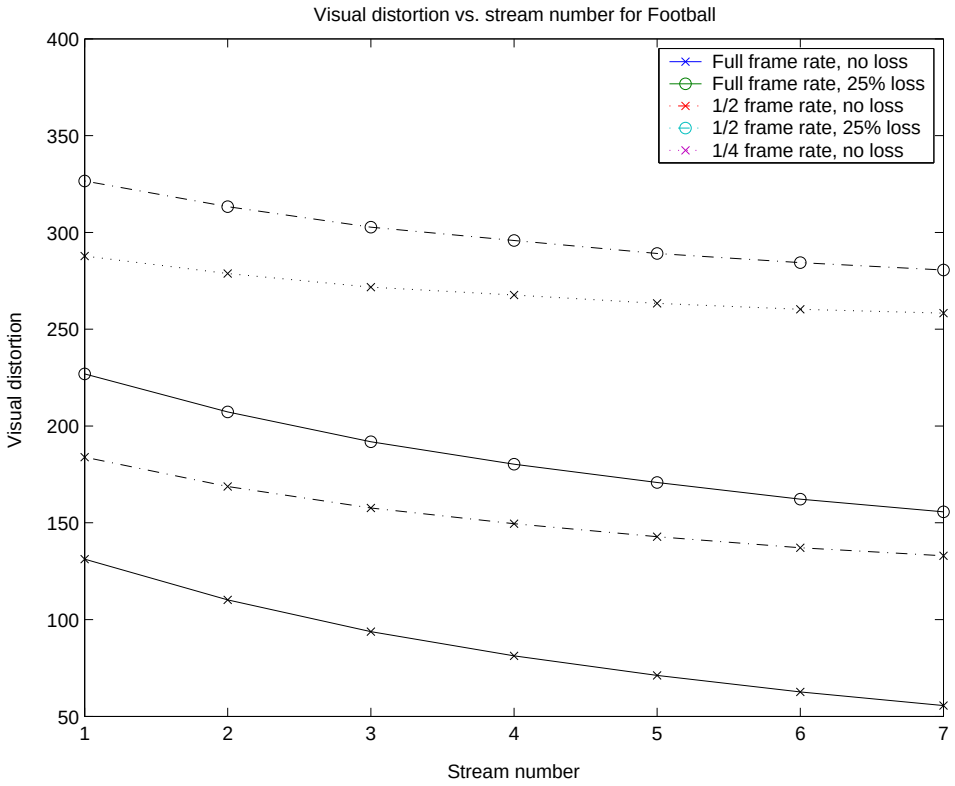


Fig. 5. Visual distortion for Football in case of packet loss

If the MV for the missing macroblock is also missing, it is first estimated using the approach suggested in [10], whereby one of the neighboring MVs, which optimizes a certain smoothness criterion, is chosen as the estimate of the missing MV. Analogously to the subband/wavelet case, we reduce the frame rate of the displeyed video if the loss becomes large. In particular, if the loss in a certain frame exceeds 20%, that frame is not displayed.

III. Simulation model and results

A. Simulation setup

The simulations are done using the *ns*-2 simulator. We have integrated each component of Figure 1 within *ns*-2. The setup for the single bottleneck simulations is the simple dumb-bell configuration shown in Figure 6. The nodes S1 to Sn are the source stations that transmit the media files using different transport/buffering schemes as dictated by the case configuration. The media flows pass through the bottleneck link B1, B2 finally terminating at nodes D1 to Dn. The

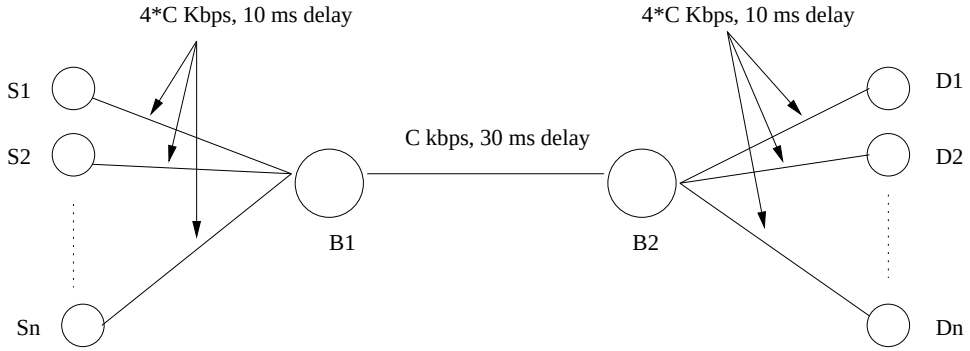


Fig. 6. Single bottleneck topology used in the simulations.

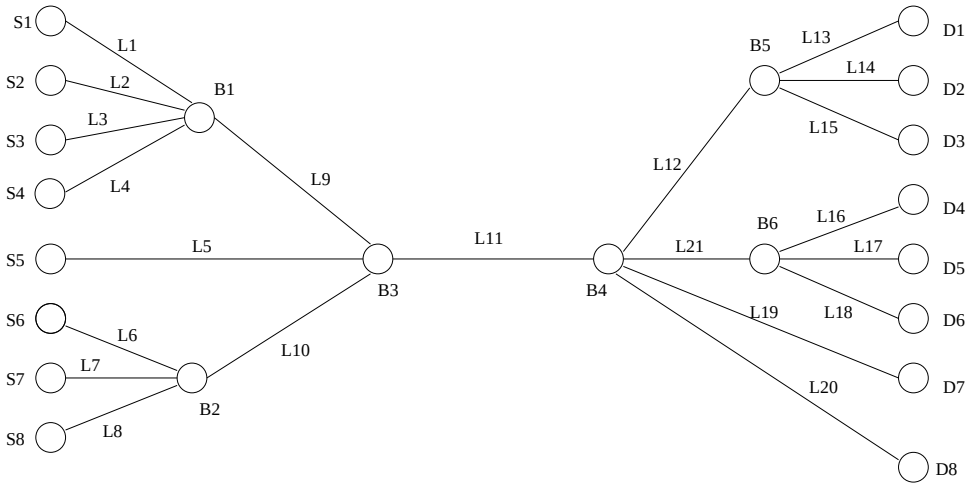
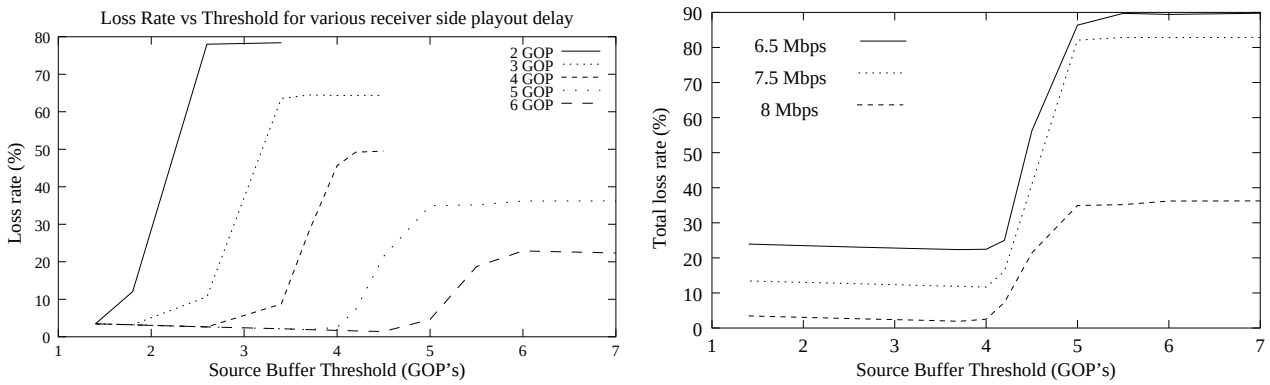


Fig. 7. Multiple bottleneck topology used in the simulations.

bottleneck has a capacity C Kbps and delay of 30 ms. All the access links have capacity $4 * C$ Kbps and a delay of 10 ms. The bottleneck buffer is set at 40 packets for each simulation.

For multiple bottleneck simulations we used the topology shown in Figure 7, which is the same topology as used in [4]. Eight sources S1 to S8 are the source stations that transmit the media files using different buffering schemes as dictated by the case configuration. The media passes through bottleneck nodes B1 to B6 and finally terminates at the destination nodes D1 to D8. In our simulations we used the following link statistics. Links L1 to L8 and L13 to L20 had a bandwidth of 40 Mbps and a delay of 10 ms. Links L9, L10, L12 and L21 had a bandwidth of 16 Mbps and a delay of 10 ms. Link L11 had a bandwidth of 20 Mbps and a delay of 30ms.

Two standard test video sequences, Football and Flower garden, were used in the experiments.



(a) 7.5 Mbps (b) 5 GOP's playout delay

Fig. 8. Loss rate vs. source buffer threshold: near-optimal source buffer threshold is $D = 1$ GOP less than receiver side initial playout delay B. The result holds true under all network rate conditions examined.

Both were SIF resolution (352×240 pixels) at 30 fps. The Football sequence is encoded using the subband/wavelet coder at a bit rate of 1.7 Mbps with a GOP size of 16. The average packet size was 725 bytes. The Flower garden sequence was encoded using 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. Each GOP had an average of 160 packets. Our goal was not to compare the coders, but to illustrate that source buffer management brings generic improvement to the received video quality, i.e. it improves the quality of the received video in both cases. Simulations also show the advantages of using a smoothly varying congestion scheme like IIAD over AIMD.

B. Buffer threshold

In this section we provide the experimantal verification of near-optimality of the source buffer threshold setting derived in section II-B. Experiments are based on the Football sequece encoded using the subband/wavelet coder, as described in the previous subsection. Figure 8 (a) plots the total loss rate vs. source buffer threshold T_s (in GOP's) for various values of receiver side initial playout delay B (in GOP's). In these units, the GOP size is $D = 1$ GOP. The figure shows that the total loss rate is minimized when the threshold is near $B - D = B - 1$ GOP's, which agrees with the results of section II-B. For example, when initial playout delay is $B = 5$ GOP's, the optimal value of the source buffer threshold is near $B - 1 = 4$ GOP's. Figure 8 (b) shows the

loss rate vs. source buffer threshold for various values of bottleneck rate, with receiver playout delay fixed at $B = 5$ GOP's. We see that the optimal value for the source buffer threshold is approximately 4 GOP's across the range of bottleneck rates. Again, the optimal value of the source buffer threshold is approximately $D = 1$ GOP less than the receiver playout delay, confirming the validity of results in section II-B.

C. Congestion control

In this section we present some congestion control results and show how buffer management helps trade off random loss for controlled loss of low priority packets. Experiments reported in this section are based on the single bottleneck topology with 5 sources, each of them sending the same video sequence. We ran two sets of simulations - one where the bottleneck bandwidth was set at 7.5 Mbps and the other where bandwidth was 6 Mbps. The decoder playout buffer was of length 6 GOP's and the pre-buffer was of length $B = 5$ GOP's. The source buffer threshold was set at $T_s = B - 1 = 4$ GOP's. The simulation was run with both AIMD and IIAD, with and without buffer management.

Figures 9 (a) and (b) compare the average loss rate of the 5 flows per layer in steady state, with and without buffer management for IIAD. Figure 9 (a) corresponds to a bottleneck bandwidth of 7.5 Mbps. The overall loss rate in this case was 11% with buffer management and 13.3% without buffer management. Figure 9(b) corresponds to 6 Mbps. The loss rates are 25.2% and 30.6% with and without buffer management respectively. The results show that with buffer management we can intelligently drop more of the low priority packets whereas without buffer management, the loss is distributed randomly and across all layers. Concentrating the loss in low priority packets helps improve the received video quality through appropriate frame rate adjustment.

Figure 10 plots the instantaneous source buffer length versus time for IIAD and AIMD. Both simulations were run over a time interval of about 100 seconds at 7.5 Mbps with source buffer threshold $T_s = 4$ GOP's. The high variation of the buffer length in AIMD is caused by the high rate variation in AIMD. Due to the high variation in AIMD, the delay or jitter associated with AIMD is higher. Therefore, even though both AIMD and IIAD lose more of the low priority packets, AIMD has far more losses in high priority layers than IIAD and the performance suffers.

Figure 11 shows the results for the Flower garden sequence encoded using the H.26L encoder. Figure 11(a) shows the average loss rate per layer with and without buffer management for 7.5

Mbps bottleneck bandwidth. The total loss rate for each case are 9.8% and 10.6%, respectively. Figure 11(b) shows the results for 6 Mbps bottleneck capacity. Here the steady state total loss rate was 29.8% and 29.3% with and without buffer management, respectively.

Overall, the results in this section illustrate how the proposed buffer management scheme concentrates the loss in the low priority packets. In the next section we show how this effect coupled with frame rate adjustment improves the quality of the received video.

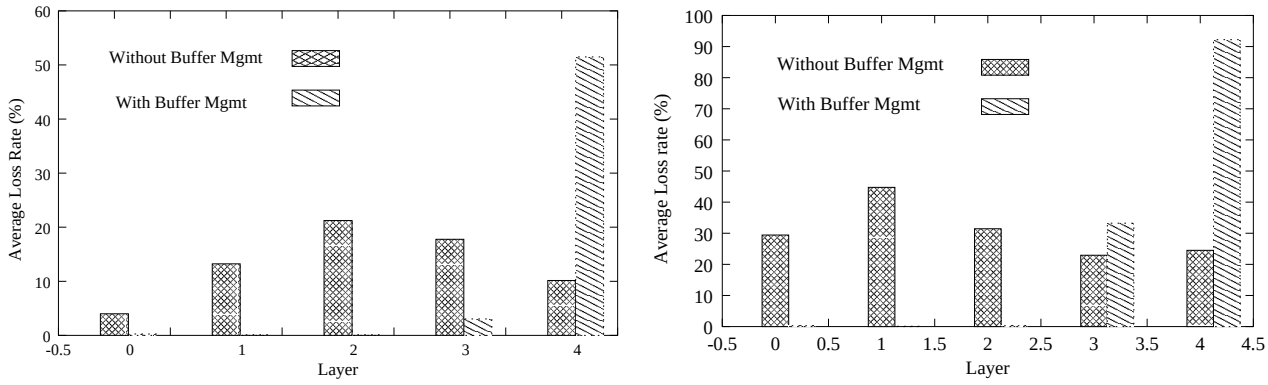
D. Video transmission results

In this section we present the experimental results of video transmission over single and multiple bottleneck network topologies. The results illustrate improvement in video quality (both objective and subjective) brought by the proposed buffer management scheme coupled with frame rate adjustment. Objective video quality is usually measured by the peak signal-to-noise ratio (PSNR) in dB, defined as

$$PSNR = 10 \log_{10} \frac{255^2}{MSE},$$

where MSE is the mean squared error between the original (uncompressed) video and the decoded video.

Since the video is played out at a reduced frame rate if the loss becomes too large, there is a question of what is the ”original” low frame rate video sequence to which the decoded sequence



(a) IIAD - 7.5 Mbps (b) IIAD - 6 Mbps

Fig. 9. Average Loss Rate per Layer using Subband/wavelet encoder: Buffer Management helps to move the losses to low priority packets (layer 4)

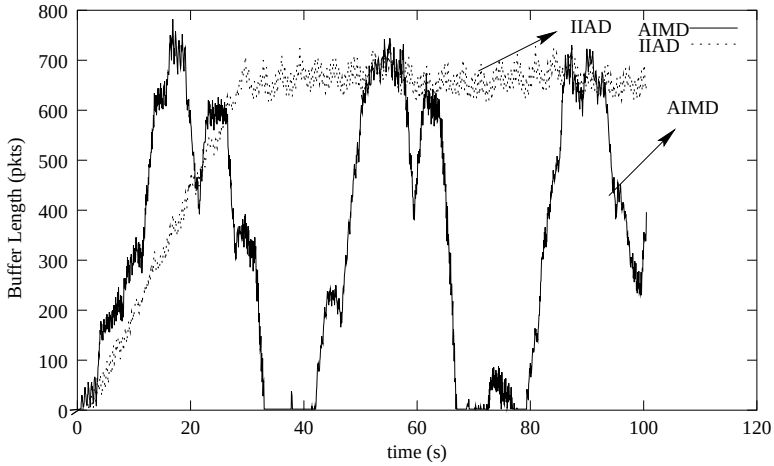
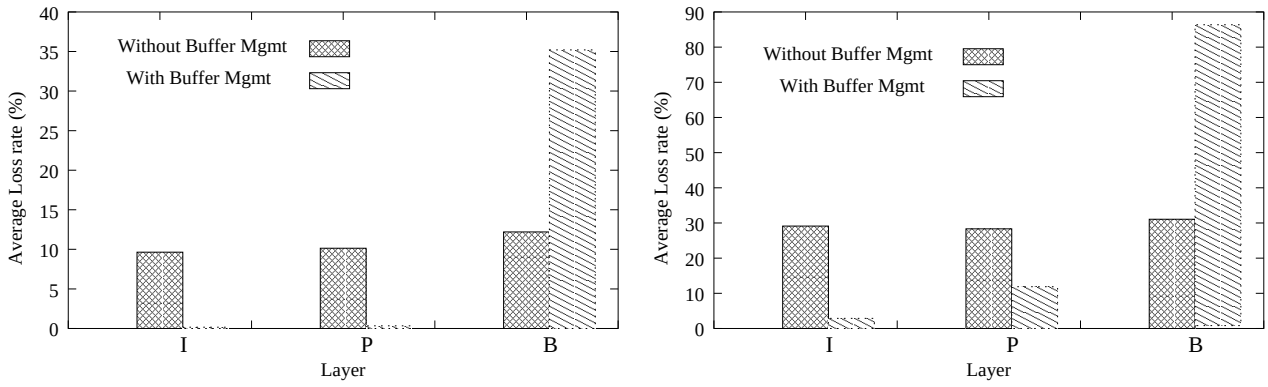


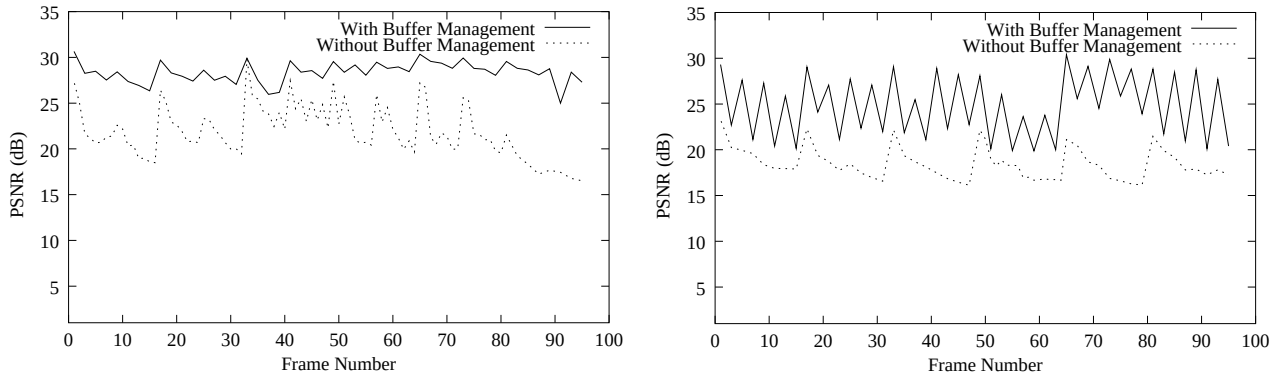
Fig. 10. Comparison of source buffer length using AIMD and IIAD schemes: IIAD scheme is smoother than AIMD and reduces jitter in the received video.



(a) 7.5 Mbps (b) 6 Mbps

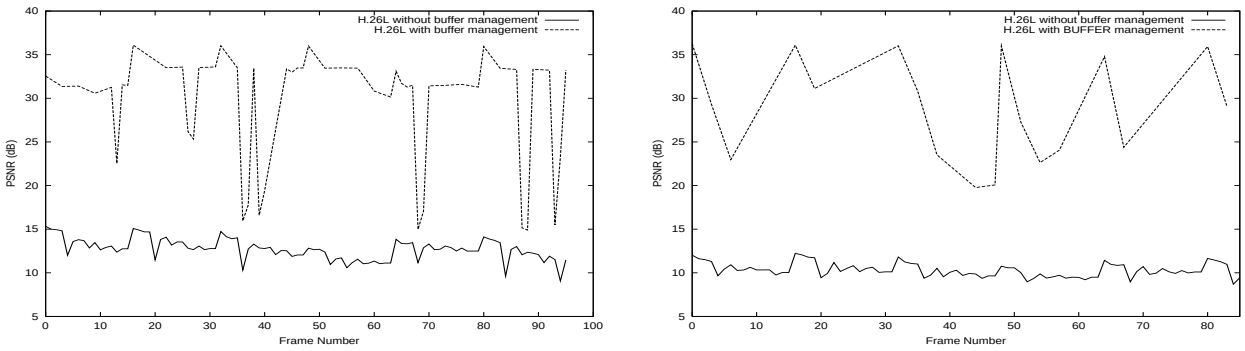
Fig. 11. Average loss rate per layer using H.26L coder: buffer management helps to move the losses to low priority B packets.

should be compared. This question is still unresolved in the video processing community. Our view is that each coder implicitly produces its own ideal low frame rate ”original” depending on its mechanism of providing frame rate scalability. For the motion-compensated subband/wavelet video coder, the low frame rate ”original” is the sequence obtained by passing the full frame rate original video through the motion-compensated filter bank (without quantization) and sub-sampling in the temporal direction. For the H.26L coder, the low frame rate ”original” is simply the subsampled unquantized version of the original video. In our PSNR calculations, for each coder we used its own low frame rate ”original” sequence. We stress again that our goal here is



(a) 7.5 Mbps (b) 6 Mbps

Fig. 12. PSNR comparison for the single bottleneck case: Football sequence encoded using the subband/wavelet coder.



(a) 7.5 Mbps (b) 6 Mbps

Fig. 13. PSNR comparison for the single bottleneck case: Football sequence encoded using the H.26L coder.

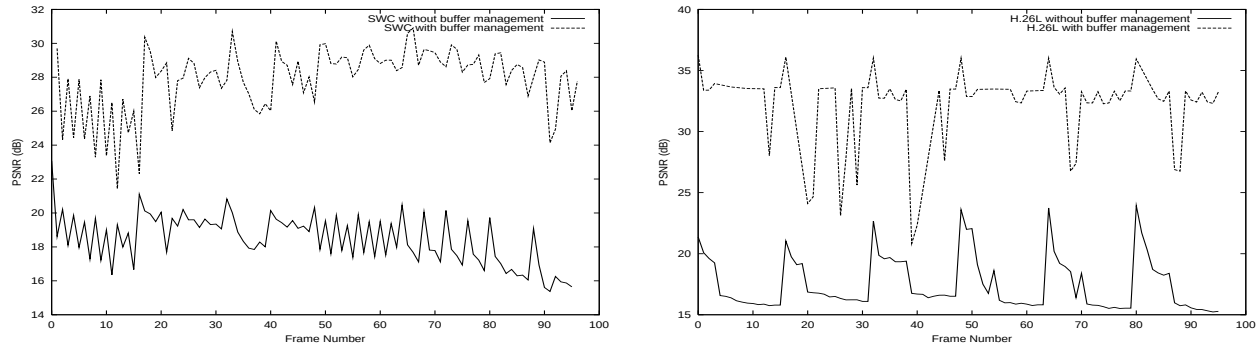
not to compare the two coders, but to demonstrate that the proposed buffer management scheme can improve the performance in both cases.

In Figures 12 and 13, and Table I, we present PSNR results for video transmission over a single bottleneck network. PSNR graphs represent ensemble averages of the frames from the 5 flows transmitted through the bottleneck, while the table shows time time averages of these ensemble averages. Hence, the results illustrate the aggregate performance of all 5 flows. Figure 12 shows the frame-by-frame PSNR of the Football sequence encoded using the subband/wavelet coder.

TABLE I

COMPARISON OF AVERAGE PSNR FOR THE SINGLE BOTTLENECK CASE: SIGNIFICANT GAINS OBTAINED WITH BUFFER MANAGEMENT

Video Sequence	Bandwidth (Mbps)	PSNR (dB) with buffer management	PSNR (dB) w/o buffer management
Football	7.5	28.4	22.1
Football	6	24.8	18.4
Flower garden	7.5	29.3	12.7
Flower garden	6	28.9	10.2



(a) (b)

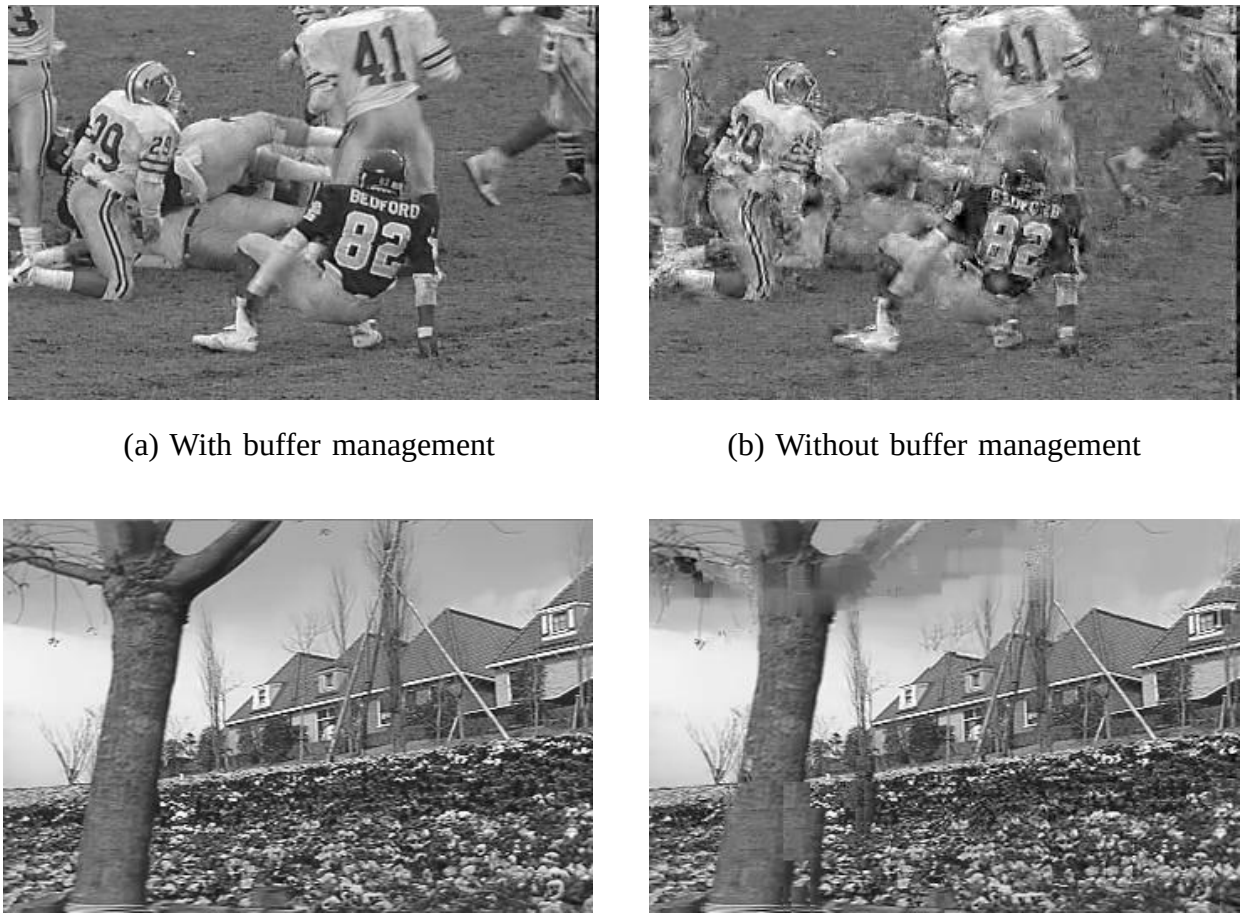
Fig. 14. PSNR comparison for the multiple bottleneck case: (a) subband/wavelet coder, Football sequence; (b) H.26L coder, Flower garden sequence.

Comparison is made between IIAD video transmission with and without buffer management. Figure 12 (a) corresponds to the 7.5 Mbps bottleneck bandwidth, while Figure 12 (b) corresponds to the 6 Mbps case. The graphs illustrate the advantage of using the proposed buffer management scheme. The average PSNR gain is is substantial - over 6 dB. Figure 13 shows similar comparison for the Flower garden sequence encoded using the H.26L coder. Similar experiments were performed on the multiple bottleneck network and the results are summarized in Figure 14 and Table II. The results are qualitatively the same as in the single bottleneck case and and demonstrate a clear asvantage of using the proposed buffer management scheme in terms of the objective video quality (i.e. PSNR).

TABLE II

COMPARISON OF AVERAGE PSNR FOR THE MULTIPLE BOTTLENECK CASE: AGAIN, SIGNIFICANT GAINS OBTAINED WITH BUFFER MANAGEMENT

Video Sequence	PSNR (dB) with buffer management	PSNR (dB) w/o buffer management
Football	27.9	18.5
Flower garden	32.0	17.1



(a) With buffer management (b) Without buffer management

(c) With buffer managemant (d) Without buffer management

Fig. 15. Snapshots of video sequences: buffer management in conjunction with frame rate adaptation improves the visual quality of displayed video frames.

Improvements in visual quality are illustrated in Figure 15, where we show snapshots of the frames from the two video sequences. These frames were obtained from the single bottleneck

simulations with a bandwidth of 7.5 Mbps. Sample video clips produced in the simulations are available at [14].

IV. Conclusions

An integrated video communication system that controls the packet drops at the source, along with an intelligent choice of congestion control was proposed to solve the problems faced by a video flow in a congested network. It was shown that most of the packet drops occur at the source buffer when congestion control is employed. This allows us to design a suitable end-end buffer management scheme for the video flow. The transmission buffer management works together with receiver side frame rate adjustment mechanism to provide high quality low frame rate video in response to congestion. Different congestion control schemes were examined and it was shown that binomial congestion control schemes that do not vary their rates very much, along with randomized pacing, help reduce jitter effects. The proposed buffer management scheme was tested with two different video coders and different network topologies, and has shown significant improvements in objective and subjective video quality in all cases.

References

[1] N. Feamster, D. Bansal, and H. Balakrishnan, "On the interactions between layered quality adaptation and congestion control for streaming video," *11th International Packet Video Workshop (PV2001)*, April 2001.

[2] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video processing and communications*, Prentice-Hall, 2002.

[3] W.-T. Tan and A. Zakhor, "Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol," *IEEE Trans. Multimedia*, vol. 1, no. 2, pp. 172-186, June 1999.

[4] R. Puri, K.-W. Lee, K. Ramchandran, and V. Bharghavan, "An integrated source transcoding and congestion control paradigm for video streaming in the Internet," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 18-32, March 2001.

[5] P. A. Chou and Z. Miao, "Rate-distortion optimized sender-driven streaming over best-effort networks," *Proc. IEEE Workshop on Multimedia Signal Processing*, Cannes, France, October 2001.

[6] Z. Miao and A. Ortega, "Optimal scheduling for streaming of scalable media," *Proc. Asilomar Conf. on Signals, Systems, and Computers*, Pacific Grove, CA, November 2000.

[7] M. Masry and S.S. Hemami, "An analysis of subjective quality in low bit rate video," *Proc. ICIP'01* Thesaloniki, Greece, October 2001.

[8] Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, "Joint model number 1, revision 1 (JM-1R1)," JVT-A003r1, January 2002. available at ftp://standard.pictel.com/video-site/h26L/

[9] P. Salama, N. B. Shroff, and E. J. Delp, "Error concealment in encoded video streams," in *Signal recovery techniques for image and video compression and transmission*, A. K. Katsaggelos and N. P. Galatsanos (Eds.), Kluwer, 1998.

[10] W.-M. Lam, A. R. Reibman, and B. Liu, "Recovery of lost or erroneously received motion vectors," *Proc. ICASSP'93*, vol. 5, pp. 417-420, April 1993.

[11] I. V. Bajić and J. W. Woods, ”Domain-based multiple description coding of images and video,” *Proc. SPIE 4671 (VCIP’02)*, pp. 124-135, January 2002.

[12] D. Bansal and H. Balakrishnan, ”Binomial congestion control algorithms,” *Proc. IEEE INFOCOM’01*, April 2001.

[13] K. Chandrayana, S. Ramakrishanan, B. Sikdar, S. Kalyanaraman, A. Balan, and O. Tickoo, ”On randomizing the sending times in TCP and other window based algorithms,” *RPI ECSE Networks Lab Technical Report, ECSE-NET-2001-1*, July 2001.

[14] http://networks.ecse.rpi.edu/˜balana/sequences