MPFD: A Lookahead Based Buffer Management Scheme for MPEG-2 Video traffic

Ashraf Awad, Martin William McKinnon, Raghupathy Sivakumar

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250

***Abstract*— In this work, we propose a priority buffer management scheme for MPEG-2 video traffic called MPFD. The MPFD algorithm is based on constructing a virtual buffer that represents the future buffer occupancy using information about video frames within a future lookahead period. The virtual buffer is then used to determine whether incoming low priority frames need to be dropped in order to protect future high priority frames. MPFD is compared to existing buffer management schemes that has been used for video traffic. MPFD shows a significant frame loss reduction as compared to other buffer management schemes.**

I. Introduction

Multimedia applications have become a significant part of today's networks and many network protocols have been built to support such applications. In order to provide efficient support for multimedia applications such as video, we need to understand its characteristics to achieve optimal treatment when congestion and loss is encountered. For example, video frames are delay constrained and cannot be used if they miss the display time. Also, compressed video formats such as MPEG-2 contain dependencies between frames; therefore, losing a frame may result in having several frames to be incorrectly displayed due to their dependencies on the lost frame. Hence, in order to minimize the loss in video quality, congested network nodes should account for the video traffic characteristics when dropping video packets. Network routers that support service classification can implement service-aware smart packet dropping schemes in order to improve the end-to-end performance.

There are several buffering management approaches proposed in literature to efficiently support prioritized traffic like video streams. Some approaches are based on introducing a threshold in the buffer such as Partial Buffer Sharing (PBS) and Triggered Buffer Sharing (TBS) [1]. In these schemes, a threshold is introduced in the buffer in order to allow one part of the buffer to be shared by high and low priority packets while the other part is only used by high priority packets. The introduced threshold is fixed in the buffer and does not change to accommodate burstiness in high priority packets within a traffic flow or between flows that may be aggregated into the buffer. These schemes are simple to implement and can provide buffer protection for high priority frames, but the overall performance is low because the buffer is not fully utilized due to the fixed partial buffer space that is allocated to low priority packets.

Another priority buffer management scheme is the Push Out Buffer (POB). In this scheme, when a high priority packet arrives at a full buffer, a low priority packet is pushed out of the buffer in order to accommodate the high priority packet. If there are no low priority packets in the buffer, then the incoming packet is dropped. While this scheme is efficient and fully utilizes the buffer space, it is complex to implement because it requires searching the buffer for low priority packets to discard[1].

The simplicity of threshold-based schemes makes them deployable in intermediate network nodes at the expense of their relatively low efficiency. On the contrary, the complexity of POB scheme makes it inappropriate to be deployed in network nodes despite its efficiency and the higher performance it offers.

In this paper, we propose a new dropping scheme that combines the properties of both threshold-based schemes, like PBS, and POB. Multi-Priority Frame Discard (MPFD) has features that exists in both POB and PBS schemes. It has the simplicity of the PBS scheme in dropping incoming packets upon arrival and has the efficiency of POB in minimizing the number of lost frames while keeping the video quality optimal. It also accommodates video traffic characteristics to produce better performance

[1]There are some efficient ways to implement the POB to allow fast search and removal of packets from the buffer, but these ways are complex and not preferable as compared to threshold-based schemes.

than other dropping schemes. MPFD is based on constructing a virtual buffer that represents the future buffer occupancy using information about future video frames. The virtual buffer occupancy is used to decide whether accepting the currently arriving video frame will cause dropping a future video frame of a higher priority or not. MPFD is similar to POB in how both schemes look into a buffer (the physical buffer in POB, and the virtual buffer in MPFD) to determine what frame to drop, as shown in Figure 1. The figure shows an example of dropping a low priority frame (marked with an x) for both MPFD and POB. The same frame is dropped in both cases, but frame dropping happens at different times. In POB, the frame is dropped when the physical buffer is full, while it is dropped earlier as it arrives to the buffer in MPFD. The frame drop occurs in MPFD because accepting it will result in dropping a high priority frame later on as indicated by the virtual buffer.

MPFD utilizes information about the future video frames to construct a virtual buffer in order to predict a future possibility of overflow in the physical buffer. Information about future frames are sent to the network node from the source. If the future overflow occurs while receiving a high priority frame, then incoming low priority frames are dropped in advance in order to avoid that overflow. This way, the physical buffer contents need not to be changed when dropping a frame is necessary, and the POB complexity of searching the buffer is avoided. The information sent by the source is conveyed in a scalable fashion that is inspired by dynamic packet state approaches such as CSFQ.

MPFD is realized by introducing a dynamic threshold in the buffer that determines what the current buffer occupancy should be before accepting the current frame in order to avoid dropping a future higher priority frame (or virtual buffer overflow). The distance between the current frame and the furthest future frame is denoted as the future lookahead step. The dynamic threshold and its relation with the lookahead step and the virtual buffer will be discussed in detail in Section III. Using a threshold in the buffer to trigger a frame drop makes MPFD share the simplicity of PBS.

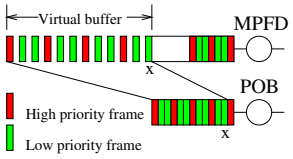

Fig. 1. MPFD virtual Buffer compared to POB physical buffer.

In our performance evaluations, MPFD demonstrates significant performance improvement because the loss in high priority packets is reduced without increasing loss in low priority frames and the whole buffer space can be utilized by low priority frames. In fact, the loss in low priority packets decreased as well in some cases. We also show that MPFD can perform better than POB under certain conditions.

This paper is organized as follows. An overview about video characteristics and existing dropping schemes in related work is given in Section II. In Section III, we describe the MPFD algorithm. In Section IV, the performance of the algorithm is evaluated and discussed. Implementation issues and tradeoffs in the proposed schemes are presented in Section V. Finally, the paper is concluded in Section VI.

II. Background and Motivation

A. Video Traffic Overview

There are three types of frames generated in MPEG-2 video-coded streams: *Intra frames* (I-frames), *Predicted frames* (P-frames), and *Bidirectional frames* B-frames. I-frames are coded using the information in the picture itself only, P-frames are coded with respect to the most recently preceding I-frame or P-frame (anchor frame), and B-frames are coded with respect to the most recently preceding and following I-frame or P-frame. I-frames are independent of other frames and have no temporal compression. P-frames are less compressed than B-frames (because their motion compensation coding is with respect to only one anchor frame), but they are usually smaller in size than the I-frames. B-frames are generally the smallest in size because their coding is dependent on the information contained in the two closest anchor frames through using motion compensation vectors.

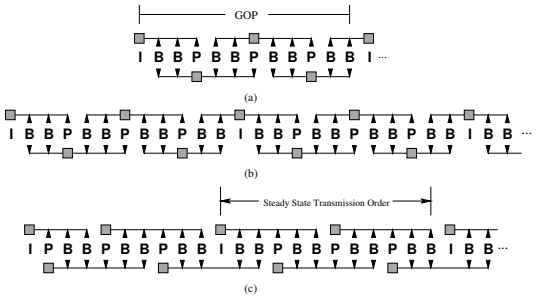

Fig. 2. (a) GOP structure, (b) GOP display order, (c) GOP transmission order.

MPEG-2 frames are generated at a constant rate (e.g. 25 or 30 frames per second). They are generated in a regular, repeating sequence (see Figure 2(a)) called a *Group of Pictures* (GOP). Figure 2(b) and (c) give the order and

dependencies among frames during both frame generation and display, and during transmission, respectively. The frames are reordered prior to transmission so that frame information is always transmitted before all other frames that are dependent upon it. In other words, when a frame arrives at the destination, it will have all information needed from other frames available to decode it.

Due to the structure of MPEG-2 video stream, the impact of lost (or corrupted) frame will not only result in the affected frame no being correctly displayable at the destination, but the impact of the error may "ripple" through other subsequent frames due to their dependence on the corrupted frame. The error propagation will continue till the next synchronization point such as a picture, GOP, or sequence header. For example, losing an anchor frame will result in losing all the following frames in that GOP.

B. Dropping Schemes and related work

There are several data protection mechanisms that has been proposed in order to enhance transport of compressed VBR video over lossy networks. These mechanisms try to minimize the over all video quality degradation due the encountered losses. These algorithms provide more loss protection to anchor frames over B-frames due to their importance. In this subsection we will provide more details about PBS, TBS and POB dropping schemes.

In PBS, a fixed threshold, T is introduced in the buffer to allow more protection to anchor frames. If the buffer occupancy is below a threshold T, both low and high priority packets are accepted into the buffer, otherwise only the high priority packets are accepted until the buffer is full where all packets are discarded [1], [2]. Packets that belong to I- or P-frames (anchor frames) are marked as high priority packets while those belonging to B-frames are low priority packets. Note that when T is equal to the buffer size, PBS becomes like Tail Discard (TD) that does not perform priority dropping.

In TBS, two thresholds are defined, $T_H > T_L$. When the buffer occupancy exceeds T_H it will not accept low priority packets and only high priority packets are allowed into the buffer. The buffer goes back to accept all packets only when the buffer occupancy decreases below T_L [1]. PBS and TBS are simple and easy to implement because they operate on incoming packets and they use a simple decision on the acceptance of a packet. Due to the fixed threshold value in both PBS and TBS, the loss in B-frames are high because the buffer is not fully utilized for B-frames.

In POB, if a high priority packet arrived to the buffer when it is full, then a low priority packet is removed from the buffer. If there is no low priority packets in the buffer, then the incoming packet is discarded. There are two versions fo POB discussed in [1], [3]: FIFO and LIFO. The difference between the two versions lies in the selection of low priority packets to remove. In FIFO, when a low priority packet is to be removed from the buffer, the packet closest to the head of the buffer is removed. In LIFO, however, the packet closest to the tail of the buffer is removed. The POB scheme is usually not preferred due to the overhead and complexity involved in searching and selecting packets from the buffer for removal.

All the above schemes provide protection to high priority packets that belong to anchor frames. However, the properties of compressed video does not only require more protection to anchor frame packets, but it also requires the protected packets to be usable. Our proposed schemes take into account all the video frame dependencies when dropping frames. The frames that are the most dependent in their coding on other frames are dropped then frames that are less dependents are dropped next. By doing this, only useful frames are transmitted and more buffer space will be available for incoming frames. The proposed schemes also minimize the number of frames that are dropped back to back in order to enhance the perceptual quality at the end user because losing individual frames that are scattered through the video stream may not be noticeable by the user.

C. Network model and assumptions

In this Subsection, we start with the assumption about the network environment. We consider network routers that support rate allocation per video flow or class such as core routers that support IntServ or edge routers that support DiffServ. Routers that are application aware are able to implement smart buffer management schemes that can efficiently increase the video transfer quality in congested networks. We also assume that a flow or a class of of flows are allocated a constant buffering space as well. A flow traverses a series of such routers from the source to the destination. The intermediate nodes also have knowledge about the output rate available for each flow or class of flows.

The traffic video source sends frame packets at a constant *frame* rate according to the frame display rate. Also, a frame is composed of a number of packets. Frame packets are transmitted back-to-back at the sender to the edge router then to the network. Due to network jitter, inter-packet time might increase or decrease and therefore, packets might spread out or be compacted. We present the MPFD algorithm for fixed frame boundaries first then we relax this assumption by presenting a method to compensate for the jitter effect.

III. Multiple Priority Frame Discard

A. Overview

The MPFD design is based on the goal of achieving the performance of POB but without having its overhead complexity. In POB, B-frame is pushed out of the buffer when an anchor frame arrives at a full buffer. In MPFD, we would like to drop that B-frame but without the complexity of searching the buffer. In order to avoid searching the buffer for B-frames, we would prefer dropping the B-frame when it arrives to the buffer because it involves no buffer manipulation operations. This is possible only if the network node knows that it needs to drop that B-frame to avoid dropping a future anchor frame. In order for the network node to know this kind of information, it needs to obtain information about the future frames.

Let us assume that the network node knows the arrival pattern of anchor frames within a future lookahead period. This information allows the node to compute the buffer occupancy during the period in which the future frames arrive. As mentioned earlier, we call the predicted buffer state (or occupancy) during that period as the virtual buffer. By constructing a virtual buffer using future frames, frame information can be used to predict if a high priority frame will overflow the buffer in the future. Given the information that the network node has, if a future anchor frame is going to be dropped when the current B-frame is accepted, then that B-frame has to be dropped.

POB and MPFD schemes are similar because both drop low priority frames only if a high priority frame is going to be dropped otherwise. However, MPFD drops low priority frames when they arrive at the buffer while POB drops them when they are already queued in the buffer. PBS and MPFD are also similar because both drop frames when they arrive, however, PBS is not accurate in determining when a B-frame should be discarded because its decision is based on a fixed threshold in the buffer and not on the arrival pattern of frames.

In order to efficiently implement the concept of the virtual buffer, we define a dynamic threshold for each frame type in the GOP structure. The threshold value is computed so that dropping higher priority frames is avoided if possible. The threshold value is dynamically computed at the network node upon frame arrival by using information sent from the video source about the frame sizes of the frames within the future lookahead period. A frame is dropped when it arrives at the buffer if the buffer occupancy exceeds its corresponding dynamic threshold value. Dropping frames upon arrival is fast and simple as compared to other algorithms that require searching the buffer for an appropriate frame to discard.

We choose B-frames to be the first to be dropped when the buffer becomes congested. The B-frame is the lowest priority frame in the frame dependency hierarchy and losing it will not affect decoding another frame at the destination. The spread of B-frames in the GOP structure causes B-frame loss to be distributed through the sequence and less likely to be noticed by end user. Also, the frequency of occurrence of B-frames makes B-frames available for dropping most of the time. Furthermore, due to the relatively small size of the B-frame as compared to other frames types, the possibility of buffer underflow after the frame drop is reduced.

P-frames are usually dropped mainly to protect the following I-frame and other future P-frames that have lower sequence number (or higher priority). In general, all P-frames are subject to be dropped for the protection of the following I-frame. Dropping the P-frame with the highest sequence number (last P-frame in a GOP) will not cause another P-frame drop. In the next subsection, we will describe the MPFD algorithm using the dynamic threshold with no jitter or losses in the incoming video traffic. These assumptions will be relaxed later in this section.

Before we present the details of the MPFD algorithm, we stop to outline the three basic stages in the MPFD approach:

- The source stamps each frame with the frame sizes of the following L anchor frames and the frame inter-arrival time[2].
- When the frame arrives at a network node, the stamped information is extracted from the frame header and then used along with the current buffer occupancy and the output rate to construct the virtual buffer. As will be shown in the next section, the virtual buffer construction is reduced to the evaluation of some simple equations to compute the dynamic threshold value.
- The computed dynamic threshold will indicate if the current frame can be admitted to the buffer or if it has to be discarded. If admitted to the buffer, the frame is enqueued and transmitted to the next network node.

In the rest of the section, we elaborate on how the virtual buffer construction is emulated through an appropriate set of equations. We also discuss deployment considerations for MPFD and the modifications required to handle jitter and loss.

[2]Including full information in each packet may be redundant, however, we will assume first that each frame is stamped with all the information needed for its admission at the network node. Then, we will discuss in a later subsection several ways of reducing the amount information sent with each packet.

B. MPFD algorithm

We define a dynamic threshold, T_{XY}, for each frame type in the GOP structure, where X is the frame type whose admission to the buffer is restricted by that threshold and Y is the furthest future higher priority frame to be protected. This means that T_{XY} is defined such that X and Y and all high priority frames between X and Y can be admitted to the buffer if the current buffer occupancy is less than T_{XY}. In other words, T_{XY} is highest current buffer occupancy that guarantees no dropping of higher priority frames in the virtual buffer. The threshold value depends on the future frame sizes that are needed to be protected. For example, when a B-frame arrives at the buffer, the threshold value, T_{BA_L}, is calculated in order to determine whether the current available buffer space is enough to accept the future L anchor frames and the arriving B-frame. We will refer to L as the number of lookahead steps for the dynamic threshold for the rest of the paper. If the buffer occupancy is below T_{BA_L}, the B-frame is accepted, otherwise it is discarded. T_{BA_L} should be carefully selected so that all L anchor frames are accepted. Other future B-frames might need to be discarded as well in order for the virtual buffer to accommodate the anchor frames.

Fig. 3. GOP frames assignments

Let the GOP size be N frames, and the distance between anchor frames be M frames. These video frames are labeled as in Figure 3. B-frames are labeled as B_1 and B_2 because their corresponding threshold value computation differs slightly as will be shown later in this subsection.

Before computing T_{XY}, we define another threshold value, T_X^Y, as the threshold imposed on frame X in order to guarantee the acceptance of frame Y only. It is different than T_{XY} because it does not necessarily guarantee acceptance of high priority frames between X and Y.

Let r_o be the output service rate and μ_X be the size of frame X in cells[3]. Now we can compute $T_{B_1}^{A_i}$, $T_{B_2}^{A_i}$, and $T_p^{A_i}$. A_i represents the i-th anchor frame to arrive after the current frame we are computing the threshold for. In order for the virtual buffer to accept the i-th anchor frame, there should exist enough initial available buffer space to start with to buffer any burst while receiving the ith anchor frame given that earlier frames do not overflow the virtual buffer. In other words, the buffer occupancy should be small enough to accommodate any bursts during that period. The maximum starting buffer occupancy (just before accepting frame X) is given by $T_X^{A_i}$ as shown in Equation (1), where t_i is the time period during which the buffer receives both frames X and A_i, and μ_i is their frame sizes. This equation means that, given an output rate r_o, the buffer can accept μ_i bytes during t_i without overflowing if the the buffer occupancy is below $T_X^{A_i}$.

$$T_X^{A_i} = B - \mu_i + t_i \times r_o \tag{1}$$

By using Equation (1), we can compute the thresholds for each frame type by computing the appropriate corresponding μ_i and t_i. Equations (2)-(4) show the values for $T_{B_1}^{A_i}$, $T_{B_2}^{A_i}$, and $T_P^{A_i}$ with their correspond μ_i and t_i values, where t_f is the frame inter-arrival time, t_X is the burst time for frame X, $t_X \leq t_f$, and $S(x)$ is the sequence number of frame x.

$$\begin{aligned} T_{B_1}^{A_i} &= B - \mu_i + t_i \times r_o \quad \forall i \in \{1, 2, \cdots, L\} \\ t_i &= (2 + (i-1)M) \times t_f + t_{A_i} \\ \mu_i &= \mu_{B_1} + \sum_{l=1}^{i} \mu_{A_l} \end{aligned} \tag{2}$$

$$\begin{aligned} T_{B_2}^{A_i} &= B - \mu_i + t_i \times r_o \quad \forall i \in \{1, 2, \cdots, L\} \\ t_i &= (1 + (i-1)M) \times t_f + t_{A_i} \\ \mu_i &= \mu_{B_2} + \sum_{l=1}^{i} \mu_{A_l} \end{aligned} \tag{3}$$

$$\begin{aligned} T_P^{A_i} &= B - \mu_i + t_i \times r_o \quad \forall i \in \{1, 2, \cdots, \frac{N}{M} - 1\} \\ t_i &= (iM) \times t_f + t_{A_i} \\ \mu_i &= \mu_P + \sum_{\forall l: S(A_l) \leq S(A_i)} A_l \end{aligned} \tag{4}$$

Notice that when $T_P^{A_i}$ is computed in equation (4), future B-frames are not accounted for because T_{PA_i} only determines if there is a need to drop the current P-frame to protect a future frame of a higher priority given that all possible B-frames are dropped.

Since each anchor frame within L-lookahead steps is guaranteed to be accepted, the threshold value for L-lookahead steps has to be equal to the smallest i-lookahead threshold for that frame. In other words, T_{XA_L} is the minimum value of all $T_X^{A_i}$, $\forall i \in \{1, 2, \cdots, L\}$ to guarantee acceptance of all L anchor frames. $T_{B_1A_L}$, $T_{B_2A_L}$ and T_{PA_L} are shown in equations (5), (6), and (7), respectively.

[3]Frames are segmented into fixed size packets, which we will call cells.

$$T_{B_1A_L} = min(T_{B_1A_i}) \quad \forall i \in \{1, 2, \cdots, L\} \qquad (5)$$

$$T_{B_2A_L} = min(T_{B_2A_i}) \quad \forall i \in \{1, 2, \cdots, L\} \qquad (6)$$

$$T_{PA_L} = min(T_{PA_i}) \quad \forall i \in \{1, 2, \cdots, \frac{N}{M} - 1\} \qquad (7)$$

In addition to the above threshold values, another threshold is defined on each frame type in order to prevent an overflow from happening in the middle of the frame reception. By introducing this threshold, the MPFD scheme will guarantee acceptance of complete frames. The threshold, T_X, is given in equation (8). The network node algorithm to compute the threshold for each frame is given in Figure 4, where μ_c and t_c are the current frame size and the its burst time, respectively.

As shown in the previous equations, the threshold is a function of the output rate, buffer size and the frame sizes. The actual sizes for corresponding future frames are sent from the sender to the network node either in packet headers or in separate packets. Since the threshold computation is a function of the service rate and buffer size, this algorithm adapts to the current varying network condition. If the network becomes congested and the buffer's service rate decreases, this algorithm would still be effective in protecting the anchor frames over B-frames. If the network is not congested and the output service rate increases, the threshold value will increase allowing more B-frames to be accepted, which will not happen in other fixed threshold dropping schemes such as TBS and PBS [1].

$$T_X = B - \mu_X + t_X \times r_o \qquad (8)$$

When a frame is received:

$T = B - \mu_c + t_c \times r_o$ // Threshold to accept current frame
if *(frame type = B_1-Frame)*
  for *all* $i < L$
    $t_i = (2 + (i-1)M) \times t_f + t_{A_i}$
    $\mu_i = \mu_c + \sum_{l=1}^{i} \mu_{A_l}$
    $T = min(T, B - \mu_i + t_i \times r_o)$
else if *(frame type = B_2-Frame)*
  for *all* $i < L$
    $t_i = (1 + (i-1)M) \times t_f + t_{A_i}$
    $\mu_i = \mu_c + \sum_{l=1}^{i} \mu_{A_l}$
    $T = min(T, B - \mu_i + t_i \times r_o)$
else if *(frame type = P-Frame)*
  for *all* $i < \frac{N}{M} - 1$
    $t_i = (iM) \times t_f + t_{A_i}$
    $\mu_i = \mu_P + \sum_{\forall l: S(A_l) \leq S(A_i)} A_l$
    $T = min(T, B - \mu_i + t_i \times r_o)$
if $(b > T)$
  Drop(current frame and all future dependent frames)
else
  Accept(current frame)

Fig. 4. MPFD algorithm

C. Deployment considerations

Thus, so far we have assumed that each frame header contains frame sizes of all L future anchor frames. This will create redundant information with a high overhead that is carried on the network[4]. In this section we will address different ways in which the source can provide the network nodes with the necessary information but with lower overhead.

One way of sending information from source to the network node is to provide the network node with incremental information in each frame header. In this method, each frame header will contain the frame size of the L-th anchor frame that follows that frame. Each network node will maintain an explicit state that contains L data items, and each item contains the frame size of an anchor frame within the lookahead period. When a frame arrives, the L-th anchor frame size is extracted and updated in the state. Since the distance between anchor frames (M) is typically three (see Figure 3), anchor frame size information can be sufficiently sent every three frames. However, this information can be duplicated in each of the three frames. If a frame is dropped in a network node, then subsequent nodes will not be updated with frame information that was carried in that frame, which will affect the functionality of MPFD. To overcome this problem, the information that was carried in the dropped frame can be appended to the information carried in the next frame header. Maintaining a state of L values does not add a significant overhead to the network node and it is much simpler than manipulating the buffer contents as it happens in POB scheme.

The choice of the amount of information sent in the frame header depends on two main parameters that governs the method of sending information to the network: available bandwidth and router capabilities. If MPFD is implemented on routers with limited capabilities, then stamping full information in each frame will eliminate maintaining state at the router but at the expense of redundancy in sent information. At the other extreme, routers with sufficient capabilities can maintain state that contains future frame sizes in order to minimize the bandwidth

[4]Although, the lookahead step required to match the performance of a complex algorithm such as POB is relatively small (4 to 6 in our evaluations).

overhead.

D. Network jitter

In Subsection III-B, we derived the value for the dynamic threshold assuming fixed frame boundaries. In this section, we study the effect of jitter on the computation of the threshold values. Due to the jitter and queueing delay that the network introduces, packets might arrive spaced out or back-to-back. This behavior will certainly affect the performance of MPFD.

Consider a back-logged buffer at network node j and let the buffer occupancy at that node be b. Let us assume that the buffer contains n video frames. Since those n frames are already in the buffer, they will be transmitted back-to-back with a rate equal to the buffer output rate. Since video frames vary is size, their frame boundaries vary accordingly and will not be constant any more. When considering jitter, we cannot assume fixed frame boundaries of future frames because this will produce inaccurate presentation of the virtual buffer, which will result in wrong dynamic threshold values. In order to accurately compute the dynamic threshold in node $j+1$, the node needs to know the arrival time of all future anchor frames that are included in the threshold computation so that the actual frame boundaries are determined. In the rest of this subsection, we define MPFD with jitter compensation algorithm, MPFD-JC, to solve the jitter problem.

In MPFD-JC, a state of timing information about future frames is maintained at the network node and updated each time a new frame arrives. The updated timing information is passed at the node buffer to the frame that is about to leave the buffer. For example, let us consider network node j with current buffer occupancy b. Assume that the first packet of an anchor frame, A_i, arrives to the buffer. Let us also assume that the first packet of a video frame, F_j, is about to leave the buffer. By knowing the buffer occupancy and the output rate, r_o, the inter-arrival time between F_j and A_i, t_{a_i}, is calculated as shown in equation (9). The inter-arrival time for A_i can be sent with F_j to network node $j+1$ and be used to calculate MPFD dynamic threshold at node $j+1$.

The network node maintains the timing information for each anchor frame in the buffer. Timing information for an anchor frame, A_i is mainly the buffer delay experienced before starting the frame transmission, denoted as t_{a_i}. Inter-arrival time for each received frame is computed according to equation (9) and then used to update the timing information. Each time a packet leaves the buffer, t_{a_i} will be decremented by t_{packet}, which is the packet transmission time. For each departing frame, the timing information is included in the frame header. *Note that the buffer contents are not accessed in the process of updating the inter-arrival times*. Information is gathered at frame arrival and is written at frame departure, i.e., when a packet arrives or leaves the buffer.

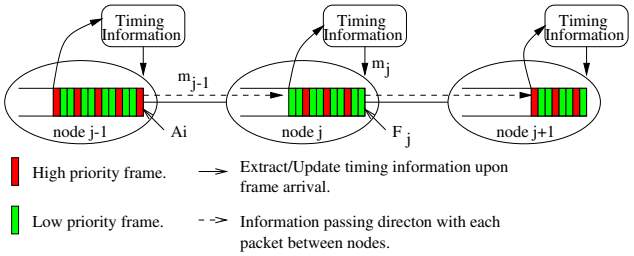

Fig. 5. Passing inter-arrival information to past frames.

$$t_{a_i} = b \times r_o \tag{9}$$

For L-lookahead step MPFD-JC, the buffer size may not be large enough to accommodate L anchor frames and all B-frames among them. Let us assume that the buffer has only $m_j < L$ anchor frames, we can obtain the inter-arrival times for the remaining $L - m_j$ frames from the last arriving frame (A_i in the above example) to node j from node $j-1$. Since the arriving frame to node j carries inter-arrival times for the following m_{j-1} at the least, this information can be passed to F_j as shown in Figure 5. After a frame traverses a number of network nodes, arrival information about the following L anchor frames will potentially be obtained.

Now, for the L-lookahead step MPFD-JC, each network node will maintain timing information state that contains L values that correspond to the L anchor frames. Each value represents the time a frame needs before it starts its transmission out of the buffer. The timing information state will be updated upon packet departure (by subtracting t_{packet} from each value) or frame arrival (by updating the inter-arrival times that are carried in the incoming frame header). It is noted that when links are congested, inter-arrival time information are updated faster due to larger buffer occupancies in the congested nodes. MPFD can be simply modified to account for network jitter. Equations (2)-(4) stay the same expect for t_i value. The values for t_i in equations (2)-(4) are redefined in equation (10), where t_{a_i} is the inter-arrival time between the current frame and A_i. The pseudo code for MPFD-JC algorithm is given in Figure 6, where $FH_t[i]$ is the inter-arrival time between the current frame and A_i that is carried in the current frame header.

$$t_i = t_{a_i} + t_{A_i} \tag{10}$$

When frame j is received:
$t_{a_j} = b \times r_o$
for *all* $j \leq i < L$ // Update timing information.
$t_{a_i} = FH_t[i] + t_{a_j}$
$T = B - \mu_c + t_c \times r_o$
if *(frame type = B_1-Frame or B_2-Frame)*
for *all* $i < L$
$t_i = t_{a_i} + t_{A_i}$
$\mu_i = \mu_c + \sum_{l=1}^{i} \mu_{A_l}$
$T = min(T, B - \mu_i + t_i \times r_o)$
else if *(frame type = P-Frame)*
for *all* $i < \frac{N}{M} - 1$
$t_i = t_{a_i} + t_{A_i}$
$\mu_i = \mu_P + \sum_{\forall l: S(A_l) \leq S(A_i)} A_l$
$T = min(T, B - \mu_i + t_i \times r_o)$
if $(b > T)$
Drop(current frame and all future dependencies)
else
Accept(current frame)
When a packet is transmitted:
for *all* $i < L$
$t_{a_i} = t_{a_i} - t_{packet}$
For each departing frame header:
for *all* $i < L$
$FH_t[i] = t_{a_i}$

Fig. 6. MPFD-JC algorithm

The complexity involved in maintaining a state of timing information is not significant as compared to buffer maintenance as in POB scheme. The state information may only correspond to an array L variables, where L is a small number. In our simulation evaluations, $L = 10$ was sufficient to outperform all existing dropping schemes.

E. Frame loss

The MPFD algorithm depends on the source to provide network nodes with information about future frames in order to create the virtual buffer. The virtual buffer is created correctly at each node if we assume that there is no anchor frame loss in the upstream nodes because the information carried in a frame will correctly describe future frames within the lookahead period. However, anchor frames may be discarded resulting in incorrect or not updated frame information carried by previous frames. To illustrate this problem, let us consider a scenario of a B-frame that carries information about a later anchor frame, A_i. The B-frame is accepted and buffered at network node j. Assume that A_i is dropped when it arrives at node $j-1$. Now the B-frame will have incorrect information about a dropped frame. The B-frame proceeds to node $j + 1$ where it gets dropped to avoid dropping A_i. The B-frame is dropped because of incorrect or not updated information about A_i causing more frame drop than needed.

Therefore, anchor frame loss needs to be updated in previous frames that are still in the physical buffer to avoid incorrect dropping of frames. To solve this problem we follow a method similar to MPFD-JC and define an extended MPFD algorithm with loss compensation, MPFD-LC. In MPFD-LC, the network node maintains information state about anchor frames sizes in the lookahead period and updates this information of any changes carried in incoming frames. The state includes L frame sizes that correspond to the L anchor frames in the lookahead period. If a node drops an anchor frame, it updates its corresponding maintained frame size with a zero. The node updates the information carried in each departing frame with any frame drop. A pseudo code for the MPFD-LC algorithm is shown in Figure 7.

When frame j is received:
for *all* $i < L$ // Update size information.
$\mu_{A_i} = FH_\mu[i]$
if *anchor frame*
if *frame dropped*
$\mu_{A_0} = 0$
For each departing frame header:
for *all* $i < L$
$FH_\mu[i] = \mu_{A_i}$

Fig. 7. MPFD-LC algorithm

Note that dropping a B-frame does not require any update because B-frame information is not carried in previous frames and is not used for lookahead. Anchor frame loss is rare because a loss will first occur in B-frames, therefore, updating information about lost anchor frame is rare and may not introduce significant overhead.

IV. Performance Evaluation

A. Simulation Environment

In order to study the performance of MPFD, we consider a network node with a simple architecture. Since each flow in the node has a designated buffer space and output rate, we can look at one video flow in the simulation as shown in Figure 8. When a packet arrives to the buffer, a decision is made on whether to accepted it or not depending on the buffer management scheme applied.

We segment video frames into fixed size packets that we will call cells. This segmentation is used because Variable Bit Rate (VBR) video frames vary in size and so do the slices that compose a picture. Therefore, measuring

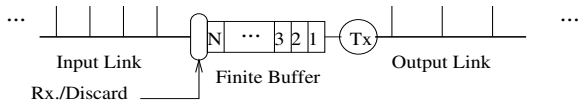

Fig. 8. Queueing System Description.

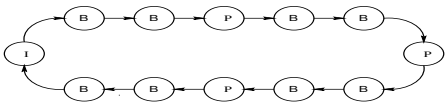

Fig. 9. Markov Chain video traffic model.

losses in terms of frames or slices would not be accurate as opposed to fixed size cells as a measuring unit.

B. Video traffic Model

Segmented video frames arrive to the buffer where the order of frames (in repeating GOPs) is governed by the twelve-state machine shown in Figure 9. As a result, the deterministic ordering between the frame types is preserved within each GOP to resemble better modeling accuracy than 3-state MMBP statistical ordering mentioned in [4].

In this work, frames are sized according to lognormal distributions [5]. They are bounded by a maximum frame size. The mean size of each frame type is selected according to the average of statistics for real MPEG-2 coded movies shown in [6], which were prepared by Oliver Rose of the Computer Science Institute at University of Wuerzburg. The I-, P-, and B-frames are set to have normalized mean frame sizes, $\mu_I : \mu_P : \mu_B$, of $1 : 0.3 : 0.13$ and relative standard deviations, $\sigma_I : \sigma_P : \sigma_B$, of $1 : 0.76 : 0.32$, respectively.

Maintaining the previously mentioned average frame size ratios limits the system's offered load[5] to 23% because, even when μ_I is maximal, μ_B is limited $0.13\mu_I$, which results in low system load. Furthermore, a 100% load can only be achieved when a continuous stream of cells is injected in the system; that type of stream is unrealistic. Even though the maximum load is low, video streams are time sensitive, which cause the input arrival rate to reach the peak rate during a packet burst (frame). This arrival pattern can cause buffer overflow when the output link is congested. We are also evaluating the algorithm's performance at an output rate that falls in the range of the average input rate.

[5]Offered load = (full slots/Total slots) at the input link.

C. Metrics

Part of the network node intelligence is the ability to distinguish between video frame types. This allows the network node to implement frame dependency drop, i.e., when a frame is dropped then all of its dependent incoming frames are dropped as well. By using frame dependency in dropping frames, only usable frames will be passed to the next network node, which will reduce the network load and cause some congestion situations to be eliminated. Dependency dropping is used in this paper in order to evaluate the performance in terms of goodput.

Goodput is defined as the ratio of the cells belonging to correctly displayable frames that are transmitted by the node's output link to the total number of cells. Furthermore, we define $P_{loss} = 1 - goodput$; we choose these definitions of goodput and loss probability because they are more representative measures of the integrity of the end user's perceived video quality than other measures [7], [8]. Similar work that used goodput as performance measure is presented in [9], [3], [10]

It was shown in Section III, MPFD is designed to consider the video traffic dependency between frames. However, PBS and TBS were not designed for video traffic that has dependency between frames. Since we are measuring the goodput (or the usability of transmitted video frames), it will not be be fair to compare MPFD with the PBS and POB as they are because their goodput performance will be low. Therefore, we implemented modified PBS, TBS, and TD schemes that are able to perform dependency dropping. We call the modified PBS, TBS, and TD as PBS-D, TBS-D, and TD-D, respectively.

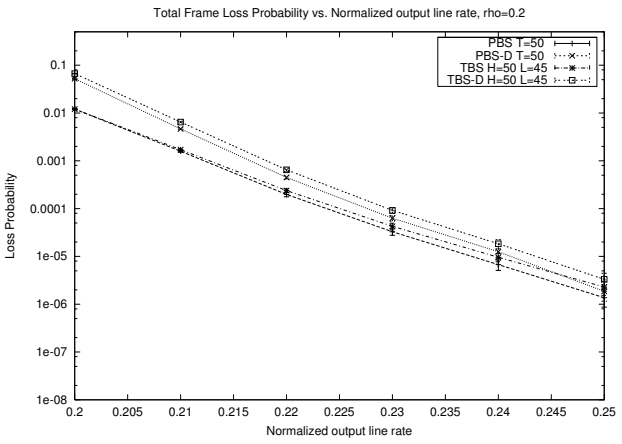

Fig. 10. Total loss.

The goodput performance of PBS-D and TBS-D is better than the original PBS and TBS schemes as shown in Figure 10. Since the corrupted or incorrectly decodable

frames are dropped upon arrival, the buffer will have more space to accept usable frames, which increases the goodput.

D. Selected Results

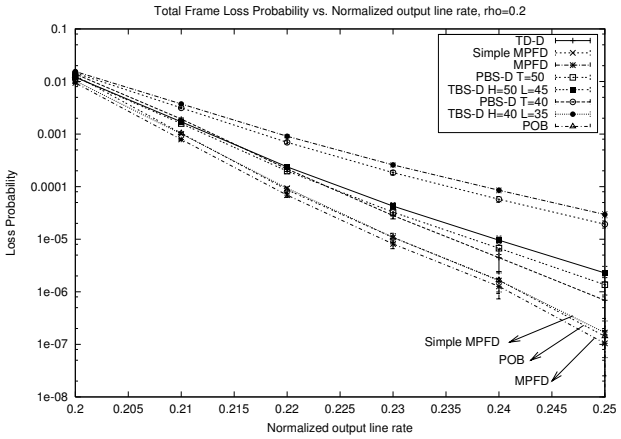

Fig. 11. Total loss.

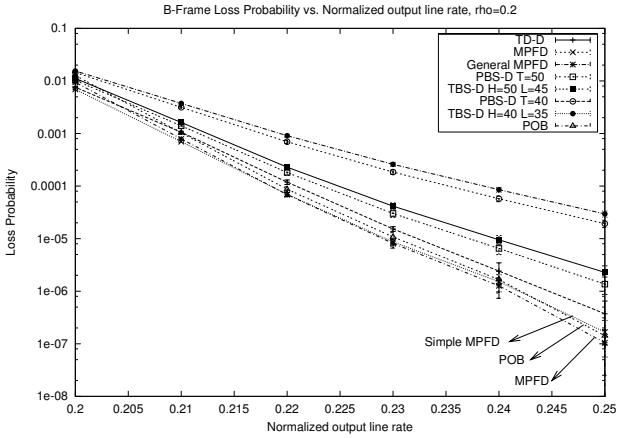

Fig. 12. Loss in B-frames.

We compared the results of MPFD with four dropping schemes: TD-D, POB, PBS-D and TBS-D algorithms. In Figures 11–14, we plotted cell loss probability for all dropping schemes versus normalized output line rate[6] for a load of 20% and with a confidence of 90%. Figures 15–18 show similar results for a load of 23%. In addition to the results for MPFD with $L = 10$, we plotted results for simple MPFD with $L = 1$ to show the lower performance bound of MPFD. It will be shown below in this subsection that MPFD with 10 lookahead steps is sufficient to achieve better performance than all threshold-based dropping schemes.

[6]The normalized output rate is the ratio between the output line rate and the input line rate. It can be looked at as a measure of link congestion.

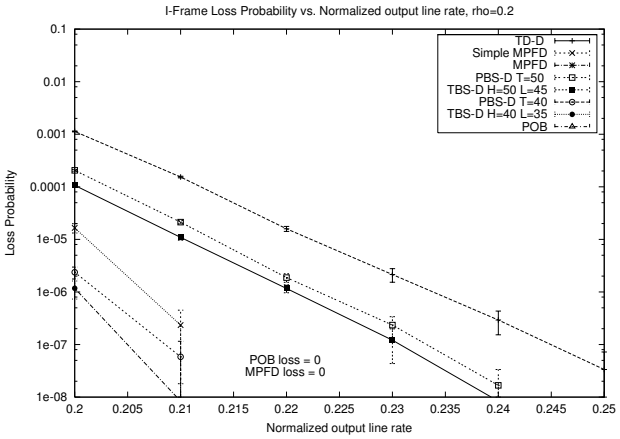

Fig. 13. Loss in I-frames.

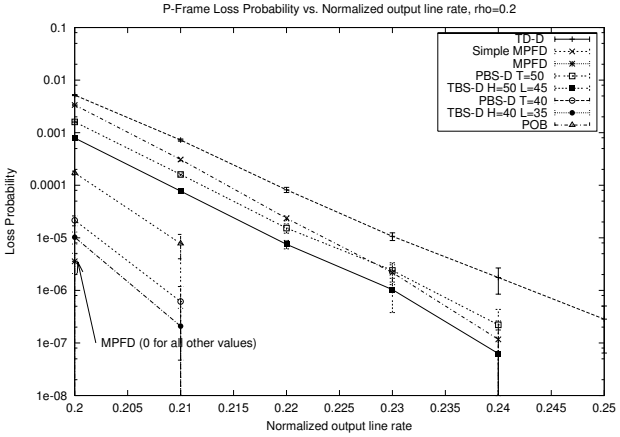

Fig. 14. Loss in P-frames.

Figure 11 shows the total loss probability for all dropping schemes. The total loss of complete frames in both MPFD and simple MPFD appears to be the smallest of all threshold based schemes while it achieves a comparable performance of POB scheme. This indicates that the MPFD scheme results in better bandwidth utilization than TD-D, PBS-D and TBS-D. Similar results are also shown in Figure 15. Notice that, with simple MPFD, total loss is significantly reduced, which results in a better video quality. The reason behind this significant improvement can be seen in Figures 12 and 16 where we can see that the reduction in the total loss is mainly caused by a significant reduction in the B-frame loss. This loss drop is one advantage of dynamic threshold in MPFD over fixed threshold(s) as in PBS-D and TBS-D.

Notice that the performance of PBS-D and TBS-D is governed by the appropriate selection of the threshold value. The B-frame loss in PBS-D is increased by an order of magnitude when the threshold value is changed from 50 to 40. Therefore, the threshold value has to carefully chosen in order to meet the burstiness of the video stream. In

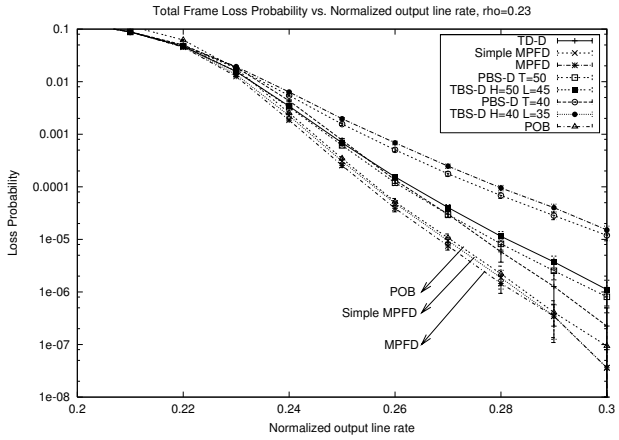

Fig. 15. Total loss.

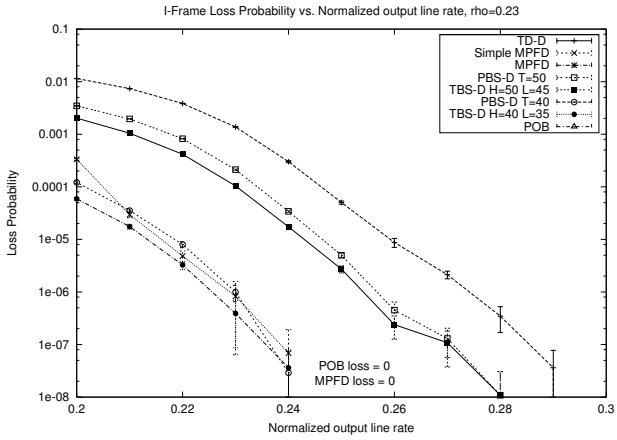

Fig. 17. Loss in I-frames.

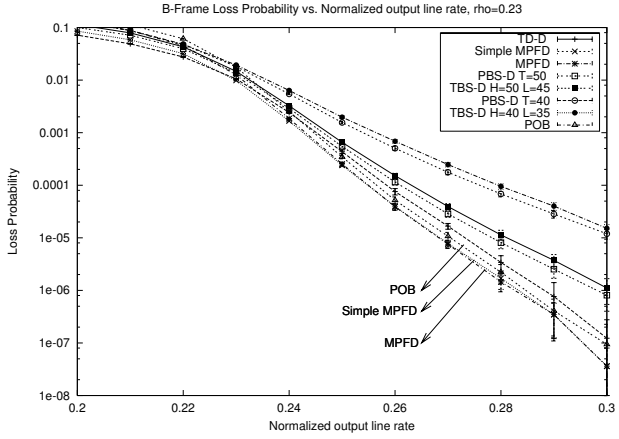

Fig. 16. Loss in B-frames.

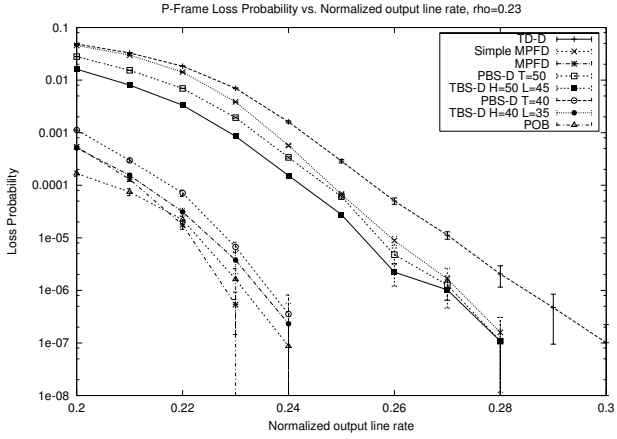

Fig. 18. Loss in P-frames.

MPFD, however, it is not required to define a fixed optimum value for the threshold.

Figures 13 and 14 show the cell loss probabilities in I-frames and P-frames, respectively. The simple MPFD results show a loss reduction in I-frames and P-frames and better performance than TD-D, but not not as good as other threshold-based dropping schemes. In MPFD, however, the I-frames encountered no loss and the P-frames had only 3.5×10^{-6}, which is lowest loss among all dropping schemes including POB. Similar results are shown in Figures 17 and 18 for a load of 23%. The general MPFD performs the best among the other dropping schemes. It is important to notice that, for comparable results for anchor frame loss, the corresponding B-frame loss in MPFD is one to two orders of magnitude less than that for TBS-D and PBS-D.

The simple MPFD has shown a significant loss reduction in I-frames and P-frames without increasing loss in B-frames. With knowledge about only one future anchor frame size, it provided I-frame loss reduction better than PBS-D with $T = 50$, and TBS-D with $H = 50, L = 45$. The loss in P-frames is comparable to PBS-D with threshold value of 50 for a normalize output rate greater than 0.23 as shown in Figure 18, which is equal to the average input traffic rate. Since simple MPFD represents the lower performance bound for MPFD, the results showed that MPFD is efficient in protecting high priority frames while increasing the usability of the video frames.

V. Implementation Issues

There are several issues about MPFD scheme limitations and deployment. We address some of them in this section.

- The MPFD requires the source to send information about future frames. Sending information about frame sizes adds bandwidth overhead to the established connection. However, the overhead is small. Two bytes could be sufficient to include size and timing information for each frame size.
- With an appropriate congestion control at the source, MPFD can be implemented at the source. In this

case, threshold computation will be more accurate since network jitter, loss and delay effects do not exist at the source. At the network side, no modification will be required at the intermediate nodes, which makes this algorithm easy to deploy. MPFD at the source may not be as powerful as when it is deployed in the network, but it can perform better than other buffer management schemes such as POB because of it is simpler to implement and because a bigger lookahead enables the source to construct the virtual buffer further in the future, which results in a better performance.

- We mentioned that MPFD can drop complete frames. This means that when a frame is dropped then all packets that belong to that frame has to be dropped. Since the network node knows the size of the arriving frame when it receives frame header, it can drop all packets that belong to that frame. We also assume that all network nodes in the video flow path are MPFD enabled. This implies that network nodes are aware of the packets that they drop because frame dropping occurs after extracting the information from the frame header.
- The MPFD algorithm can be applied for Video On Demand video streaming to enhance the video quality because information about future frames is available. In real-time video, however, this type of information is not available. Therefore, MPFD can be limited to short look-ahead distance and utilize information about small number of future frames.
- MPFD can also be applied on a per class level instead of per flow level. One lookahead sequence can be constructed for the whole class using information carried by the flows in the class. The lookahead sequence is then used to construct the virtual buffer to determine what frames have to be dropped. In per class environment, thresholds defined on B-frames will still be applied in the same a way it is applied in a per flow environment. However, dependency dropping cannot be fully implemented, but dropping the highest sequence (lowest priority) of anchor frames first reduces the need to implement dependency drop.

VI. Conclusion

In this paper, we proposed a new buffer management scheme to enhance video transport over the network. The proposed MPFD scheme is based on constructing a virtual buffer that represents the future buffer occupancy using information about future video frames. The virtual buffer is used to determine whether low priority frames need to be dropped in order to avoid dropping a high priority frame. This scheme is compared with TD, PBS, TBS, and POB dropping schemes. It was shown in this paper that MPFD scheme performs better than other dropping schemes in protecting high priority frames as well as in increasing output link utilization.

These schemes can be applied in network nodes in order to handle network congestion in multimedia connections while maximizing the goodput of video sequences. The MPFD is easier to implement than POB because it operates on incoming frames and does not search the buffer for frames to drop. It can be applied in layer-four switching domain to maximize goodput.

We are interested in extending the current work to study MPFD incremental deployment of MPFD at the source. Also, we are looking into ways to minimize the algorithm overhead at the network node and to try to apply MPFD in a core stateless network environment.

References

[1] Pedro Cuenca, Antonio Garrido, Francisco Quiles, and Luis Orozco-Barbosa. Performance Evaluation of Cell Discarding Mechanisms for the Distribution of VBR MPEG-2 Video Over ATM Networks. *IEEE Transactions on Broadcasting*, 44(2), June 1998.

[2] C.G. Kang and H.H. Tan. Queueing Analysis of Explicit Priority Assignment Partial Buffer Sharing Schemes for ATM Networks. *INFOCOM*, 2:810 –819, April 1993.

[3] Tao Tian, A.H. Li, Jiangtao Wen, and J.D. Villasenor. Priority dropping in network transmission of scalable video. *International Conference on Image Processing*, 3:400–3, Sept. 2000.

[4] C. Ohta, K. Shinagawa, and Y. Onozato. Cell Loss Properties for Multiplexing of MPEG Video Sources Considering Picture Coding Types in ATM Networks. *ICC*, 3:23–7, June 1996.

[5] M. Krunz, R. Sass, and H. Hughes. Statistical Characteristics and Multiplexing of MPEG Streams. *INFOCOM'95*, 2:455–62, April 1995.

[6] M. Frey and S. Nguyen-Quang. A Gamma-Based Framework for Modeling Variable MPEG Video Sources: The GOP GBAR Model. *IEEE/ACM ToN*, 8(6):710–9, December 2000.

[7] Olivier Verscheure, Pascal Frossard, and Maher Hamdi. Joint Impact of MPEG-2 Encoding Rate and ATM Cell Losses on Video Quality. *GLOBECOM '98*, 1:71–6, Nov. 1998.

[8] P. Frossard and O. Verscheure. AMISP: A Complete Content-Based MPEG-2 Error-Resilient Scheme. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(9):989–98, Sept. 2001.

[9] J. R. Li, X. Gao, L. Qian, and V. Bharghavan. Goodput Control for Heterogeneous Data Streams. *NOSSDAV*, June 2000.

[10] Bing Zheng and M. Atiquzzaman. TSFD: Two Stage Frame Dropping for Scalable Video Transmission Over Data Networks. *IEEE Workshop on High Performance Switching and Routing*, pages 43 – 47, May 2001.