A Priority-Based Technique for the Best-Effort Delivery of Stored Video

Wu-chi Feng, Ming Liu, Brijesh Krishnaswami, Arvind Prabhudev
The Ohio State University
Department of Computer and Information Science
Columbus, OH 43210
{wuchi, mliu, krishnas, arvind}@cis.ohio-state.edu

Abstract: In this paper, we introduce a priority-based technique for the delivery of compressed *prerecorded* video streams across *best-effort* networks. This technique uses a multi-level priority queue in conjunction with a delivery window to help *smooth* the video frame rate delivered to the end user while allowing it to easily adapt to changing network conditions. Compared with current approaches, our priority-based approach has several advantages. First, it acts more globally by ensuring that a minimum frame rate for the window interval has been delivered *before* sending enhancement layers. Second, this approach is much simpler to implement than other frame smoothing algorithms that have been presented for the delivery of stored video across best-effort networks. Finally, this approach is directly applicable for the shaping of MPEG-based video encodings with frame dependencies.

Keywords: Video adaptation, video streaming, smoothing, video-on-demand

1. Introduction

The efficient streamed delivery of compressed digital video has received considerable attention from the research community. The delivery of stored video can be particularly challenging for several reasons. First, stored video typically exhibits significantly more long-term burstiness than live video such as a video conference call which contains only a single scene. Second, adapting live video streams is simpler because the data is encoded on the fly, allowing the quality to be changed in the frame capture itself. With stored video applications, however, the video is typically streamed off of a disk and then altered on the fly. This altering typically takes the form of frame dropping for simplicity of implementation. The changing of the quality of the video to fit within available resources is usually accomplished through the storage of multiple versions of the same video stream. As networking technologies continue to advance, it is clear that adaptable video-based application that share network resources will be required.

The adaptable video streaming protocols that have appeared in the literature typically use network feedback to alter a video's frame rate, frame quality, or a mixture of the two in order to reduce the network resource requirements of the stream. Many of the systems that have been developed (as will be discussed in Section 2.1) do not take advantage of the *a priori* information available from stored video streams, resulting in sub-optimal delivery of the video stream. As an example, consider a video stream that has a large number of small size frames followed by a large number of large size frames. While the small frames are being delivered, these adaptive video streaming pro-

tocols do not take into consideration that a large number of large frames will be required in the future playing the small frames at a high frame rate instead of buffering the large frames. As a result, this stream will typically have a larger number of frames dropped when the large frames need to be played back.

More recently, several techniques have been introduced for the adaptable streamed delivery of stored video that take advantage of the *a priori* information. These algorithms extend the notion of *bandwidth smoothing* techniques to work with best effort networks. Bandwidth smoothing techniques use a client-side buffer along with the *a priori* information to create plans for the delivery of video streams that minimize metrics such as the peak bandwidth requirements, the number of bandwidth changes, or the variability of bandwidth requirements. These algorithms, as introduced, are not suitable for best-effort data delivery because they essentially "tell" the network how much bandwidth is need instead of fitting the video into the available network resources. By extending the bandwidth smoothing techniques to work with best-effort networks, these techniques act more globally in the adaptation of the video streams, allowing them to effectively reduce the frame rate variability delivered to the user. The main drawback of these techniques, however, is that they require significantly more computational power than the simple approaches that act more locally in their adaptation. This requirement can have a significant impact on the scalability of the servers that transmit these streams.

In this paper, we introduce a new priority-based video adaptation algorithm for the streamed delivery of stored video across best-effort networks. This technique bridges the gap between algorithms that do short-term video adaptation and the computationally complex algorithms that take advantage of the *a priori* information. Our proposed technique provides for a smoothed frame rate delivery by first ensuring a minimal playback rate can be achieved before adding "enhancement" layer data to increase the frame rate. The advantages this approach provides over previously introduced video adaptation algorithms is that it (1) smooths out the video frame rate delivered to the user, (2) makes implementation of the rate adaptation layer simpler, (3) can easily support frame-dependent video such as MPEG, and (4) is configurable to let users specify the way rate adaptation over time occurs. By acting more globally in the delivery of stored video streams, we will show how the video quality delivered to the user can be held more constant.

In the next section, we describe the relevant background including a discussion of adaptable video delivery techniques for stored video as well as a discussion of bandwidth smoothing techniques. In Section 3, we present our new priority-based technique for the delivery of stored video over best-effort network. In Section 4, we compare and contrast the algorithms with the current techniques using metrics such as average frame rate, minimum frame rate, and frame rate variability that these algorithms achieve. Finally, we conclude with a summary of the experimental results and directions for future work.

2. Background and Related Work

In this section, we review the pertinent background information required for this paper. We break our discussion into three main parts: adaptable video techniques (including video streaming protocols), bandwidth smoothing algorithms, and adaptive bandwidth smoothing algorithms.

2.1. Adaptive Video Transmission Techniques Without A Priori Information

Several approaches have been proposed for the integrated delivery of streamed video across the Internet from both industry and academia. These systems include VIC - a video conferencing tool built for the MBone [11]; The Continuous Media Toolkit (from the University of California - Berkeley) [16]; The Quasar Adaptable Streaming Video Player (from the Oregon Graduate Institute) [20]; Vosaic - University of Illinois' streaming video protocol [2]; and RealAudio/RealVideo Streaming protocol [13].

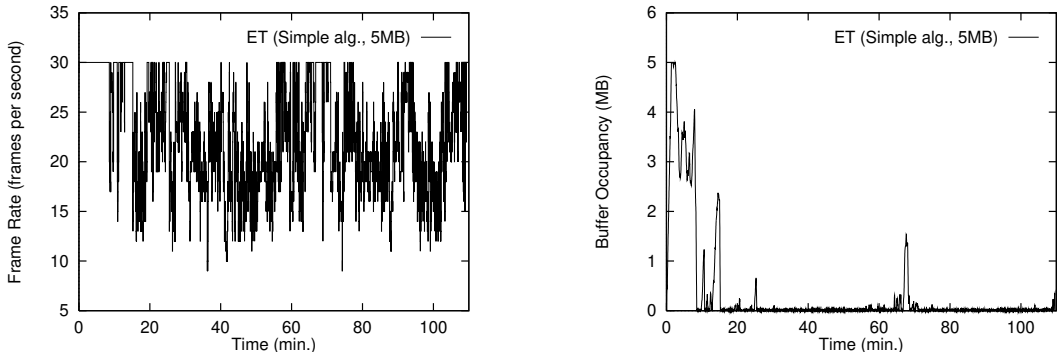


Figure 1: Simple Video Adaptation Algorithm. This figure shows the frame rate and buffer occupancy during the playback. The movie used to generate these graph is *ET* (Quality = 90), and the smoothing buffer used here is 5MB. We used a constant available bandwidth to generate these graphs, which is 70 percent of the average frame size of the whole movie.

These systems have several common properties. First, they use reliable transmission streams for control information while using unreliable transport mechanisms for the delivery of the video stream. Some of these provide retransmission of lost data if time permits. Second, with the exception of RealAudio/RealVideo, they all attempt to actively adapt to network congestion (with some more TCP compatible than the others). Finally, particularly for the systems that deliver stored video, *none* of them take advantage of the *a priori* information for the intelligent delivery of streamed stored video. That is, they act immediately on feedback to alter the video stream to fit within available resources.

As an example of the way these algorithms work, we have graphed a *representative* version of these algorithms that continually adapt the video stream to fit within the available resources in . For the rest of this paper, we will refer to this as the *simple video adaptation algorithm*. As shown in the figure, the simple video adaptation algorithm has a frame rate that varies quite a bit over time. The only time any buffer is built up is when the frame rate relative the bandwidth availability is small for an extended period of time (see minutes 0-10, 13-16, and 62-65 in the buffer occupancy figure). In essence, these algorithms are *greedy* in their approach, assuming that additional bandwidth will become available when necessary.

2.2. Bandwidth Smoothing Algorithms

Bandwidth smoothing algorithms *smooth* the bandwidth requirements for retrieval of the video stream from the video server and across the network. By prefetching data ahead of time, a client-side buffer can be used to supplement network bandwidth when the video stream reaches an area of large frame sizes. Given a fixed client-side buffer, these algorithms provide continuous playback of video for the user while

- minimizing the number of bandwidth rate increases required [3, 4],
- minimizing the total number of bandwidth changes required [5],
- minimizing the variability of the bandwidth requirements [17],
- minimizing the buffer utilization of the client-side buffer [6],
- delivering video in constant bandwidth allocation periods [12],
- smoothing across patterns of compressed frame types [18],
- adhering to a buffer residency constraint [8], or
- adhering to on-off bandwidth periods [21].

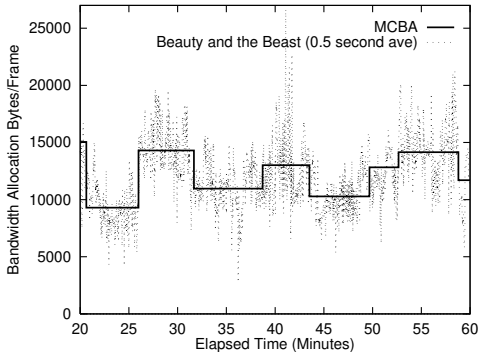


Figure 2: Minimum Changes Bandwidth Allocation Example. This figure shows a sample bandwidth allocation plan generated by the *minimum changes bandwidth allocation* algorithm using a five megabyte buffer for the movie *Beauty and the Beast*. For the given buffer size, the plan generated has, for continuous playback, 1) the minimum peak bandwidth requirement, 2) the maximum-minimum bandwidth, and 3) the smallest number of rate changes possible. The lightly dotted lines represent the half second frame averages.

Many of the above algorithms also minimize the peak bandwidth requirements for the continuous delivery of the video stream given a fixed client side buffer. As an example of how bandwidth smoothing algorithms work, an example bandwidth plan generated by the *minimum changes bandwidth allocation* algorithm for the M-JPEG encoded movie *Beauty and the Beast* is shown in Figure 2. In this figure, we see that the bandwidth smoothing algorithms can result in plans for the delivery of video that have periods of constant bandwidth allocation lasting on the order of 5 to 10 minutes for even a small amount of buffering.

One important theme among these algorithms is that they dictate the bandwidth rates that are required from the underlying network. For dedicated video-on-demand systems, this assumption works fine, allowing the bandwidth smoothing algorithms to minimize the total resources required to deliver the video streams to the users. For best-effort network environments such as the Internet, this assumption *will not allow* the video to be delivered continuously in most circumstances.

2.3. Adaptable Video Techniques Using A Priori Information

There have been several proposed methods for the incorporation of bandwidth smoothing techniques for the delivery of stored video. The *proactive buffer management technique* takes advantage of the *a priori* information to shape the video stream to fit within a single constant bandwidth, smoothing out frame drops over time[9]. Because the algorithm works from the end of the movie to the playing point, this algorithm is more suitable for guaranteed bandwidth allocation channels such as ATM's constant-bit-rate (CBR) service or ATM's available-bit-rate (ABR) service. Under changing network conditions, this technique becomes somewhat impractical because of its large computational requirement.

More recently, we have shown how to extend critical bandwidth allocation techniques (from the bandwidth smoothing world) to work with best-effort networks [10]. Given a fixed client-side buffer, the *modified-CBA algorithm* first creates a bandwidth plan using one of the algorithms presented in the previous subsection (assuming network bandwidth is plentiful). Then in the delivery of the stream, the modified CBA algorithm essentially *clips* the tops off of the bandwidth allocation plan. Because the rate shaping occurs on an already smooth plan, the frame drops tend to be more spread over time. The *adaptable CBA algorithm* combines the calculation of the bandwidth smoothing plan with the rate adaptation algorithm in a more integrated fashion. In particular, it re-calculates a plan

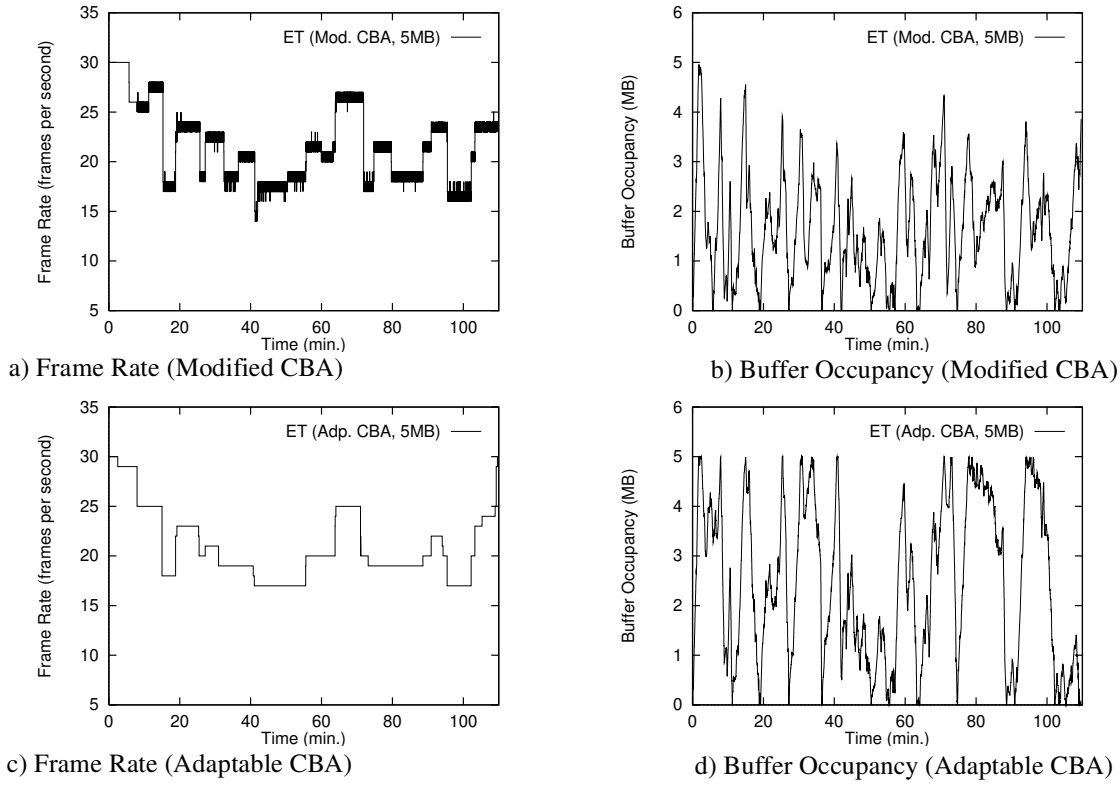


Figure 3: Modified CBA and Adaptable CBA Algorithm Examples. This figure shows the frame rate and buffer occupancy graphs of the movie ET (Quality = 90) using the modified CBA and adaptable CBA with 5 megabyte smoothing buffer, and available bandwidth is constant at 70 percent of the average frame size.

for the delivery of stored video every second and then alters the video stream to fit within the available resources. While this algorithm can result in the best performance in terms of frame rate variability delivered to the user, it is computationally more expensive than the other algorithms presented thus far. In particular, the adaptable CBA algorithm requires the recalculation of the frame rate every time the bandwidth changes. Note, that we assume sufficient additional buffering is in place to have bandwidth targets that are constant over periods of seconds or that the streaming protocol provides a more constant bandwidth stream to the application [1].

To illustrate these algorithms and to compare it with the simple video adaptation algorithm, we have graphed two representative examples in Figure 3 using the same network conditions as in Figure 1. Here we see that the modified and adaptable CBA algorithm reduces the frame rate variability delivered to the user through the use of bandwidth smoothing techniques. In particular, we see in these examples that the buffer occupancy is greater than in the simple video adaptation algorithm. We will show an example delivery plan for the modified and adaptable CBA algorithms in the experimentation section.

3. A Priority-Based Video Adaptation Algorithm

In this section, we introduce our proposed priority-based video adaptation algorithm. We first describe the adaptation technique at a high-level and then move into a more in-depth description of the various parameters that can be specified within the algorithm.

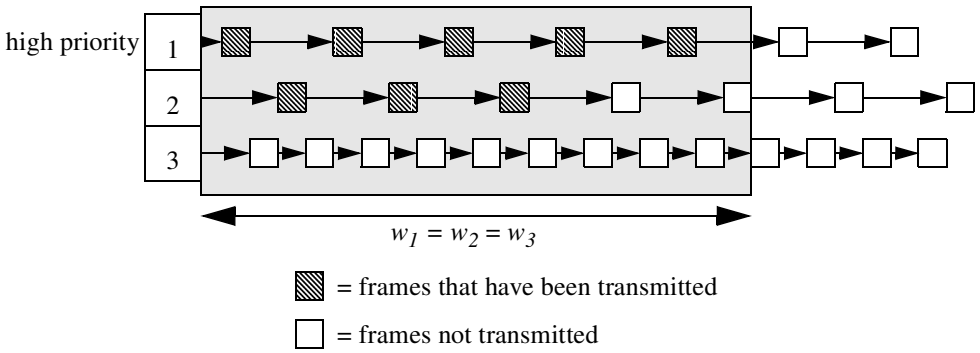


Figure 4: Priority-Based Delivery Model. This figure shows an example of the priority-based shaping algorithm. As shown, sufficient network resources exist to support the highest priority layer of the video stream and part of the second layer of the stream. The window-size of 19 in this example is smaller then would be expected for use in this algorithm. As shown in this example, the user will be guaranteed at least 15 frames per second over the duration of the window

3.1. The Model

In our model, each frame within the video is assigned to one of the *m* layers in the priority queue. These layers can be assigned when storing the video or can be assigned dynamically during the retrieval of the video stream. Each of the *m* layers is then mapped directly into a *multi-level priority queue* with the highest priority layer starting with 1. For clarity of discussion, we will refer to the highest priority layer as the *base layer.* Each priority layer *i* is also assigned a window-size, w_i, which represents the ideal prefetch interval that the user would like to have sitting in the local buffer. The frames within the video stream are consecutively numbered from 1 to *n*. We will discuss the choice of layers and window size in Section 3.2 and Section 3.3, respectively.

Given the priority queue and the window sizes, let the current frame that is being play at the client be *curr_playback*. The algorithm always tries to guarantee that the frames in the highest priority queue from *curr_playback* to *curr_playback*+w_1 are delivered to the client buffer *first.* If enough network resources exist to allow these frames to be delivered it then attempts to meet the same requirement for all the "enhancement" layers, 2 to *m*, in that order. Thus, the window-based mechanism ensures a certain minimum frame rate (assuming the base layer fits within the available resources) *over* the window defined by w_0 and the multiple levels in the priority structure allow the algorithm to send enhancement layers when sufficient network resources exist. The rate shaping (or dropping of frames) is accomplished by monitoring the head of the queues. If the transmission instant has passed, then the frame is discarded because insufficient network resources did not permit transmission of the frame in time to be viewed. As a result, the dropping of frames occurs quite naturally with the layers allowing the data to be dropped more evenly over a larger period of time. An example of this model is shown in Figure 4.

So, how does this algorithm differ from previous approaches? Compared with the algorithms that actively use the *a priori* knowledge from the video stream (as discussed in Section 2.3), this algorithm is computationally much more efficient and much simpler to implement, particularly when the bandwidth availability over time varies. By using a window-based priority mechanism, this algorithm accomplishes similar prefetching of larger frames, smoothing the frame rate delivered to the end user over time. Compared with the algorithm that adaptively shape the video stream on the fly without the *a priori* knowledge, this algorithm provides a more global shaping of the video stream by first ensuring that a large number of frames are buffered to provide a minimal frame rate. The priority-based algorithm also allows for the efficient shaping of videos with multiple frame dependencies such as MPEG's I, P, and B frames. Using the priority-based algorithm, the I-frames can be assigned the highest priority

```
1     for(curr_play=0;curr_play < num_frames;curr_play++)
2       {
3       send_amt = bandwidth available during this slot;
4       while((!done)&&(send_amt>0))
5         {
6           Find a target priority level to send from;
7             This target layer t does not have its window completely full;
8
9           If all layers have their window full, do not send any data
10
11          Otherwise, transmit the next frame at the target layer that is
12            greater than the current playback instant and can be transmitted
13            and displayed on time;
14        }
15       }
```

Figure 5: Pseudo-code for the Priority-Based Adaptation Algorithm. This figure shows the pseudo-code for the proposed algorithm.

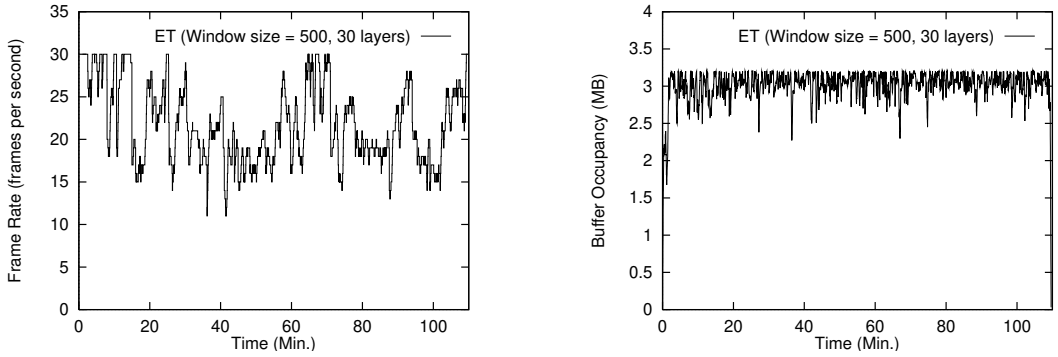


Figure 6: Priority-Based Frame Delivery. This figure shows an example of the priority based delivery algorithm for the movie *E.T.* with a constant bandwidth availability of 0.7 times its average frame size.

while the P and B frames can be assigned to lower priorities. The example illustration in Figure 4 shows a possible prioritization for an MPEG movie with the encoding:

IBPBIBPBI....

As a result of this assignment we are guaranteed that a P or B frame will *not* be delivered unless the associated frame that it depends on has also been delivered. Finally, this priority-based algorithm (because it does not depend on having access to all *a priori* knowledge can be used for on-line smoothing over best-effort networks [14].

The pseudo-code for this algorithm can be found in Figure 5 and an example delivery of a stored video stream using a single constant bandwidth availability can be found in Figure 6. As shown in the figure, we see that the algorithm has much higher buffer utilization than the algorithms discussed in the background section. We also see that the frame rate delivered to the user is smoother than the simple adaptation algorithm but not quite as smooth as the algorithms that require constant recalculation of frame rates. We will show an example of a delivery plan under changing network conditions in the experimental section. In the following several subsections, we will describe some of the design trade-offs in choosing the number of levels and the window size for the adaptive algorithm.

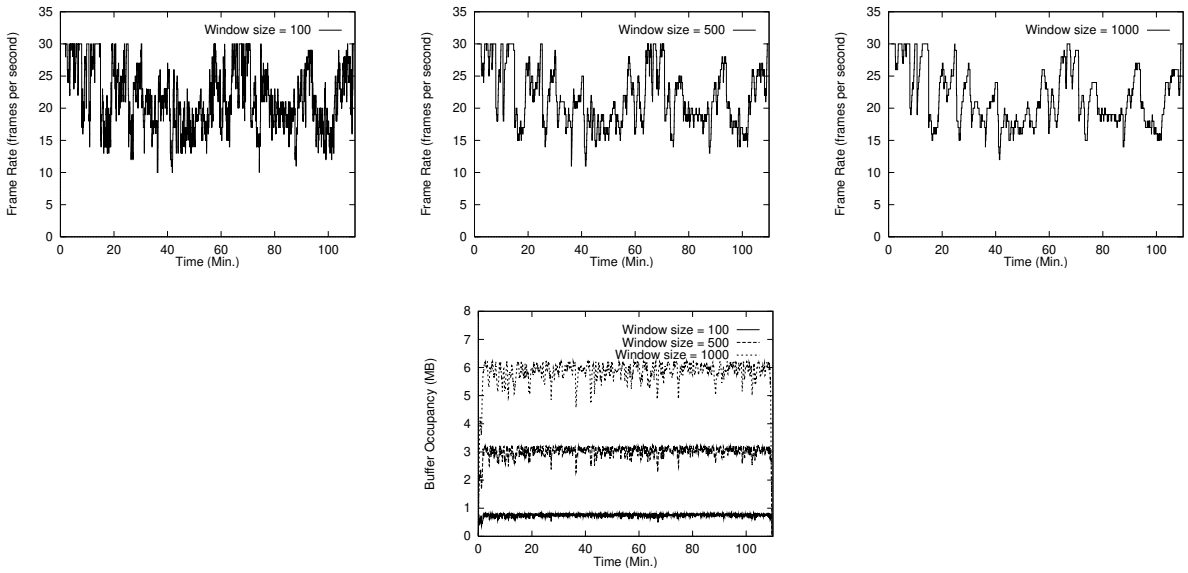


Figure 7: Window Sensitivity Example. This figure shows the performance of three different window sizes for the delivery of *E.T.* using a constant bandwidth allocation of 0.7 times the average frame rate for the movie.

3.2. Window Sensitivity

One of the main parameters of this algorithm is the size of the window. As the window size increases, three important things happen. First, the algorithm will start biasing more toward the base or highest priority layer. Second, the minimum frame rate delivered to the user will increase because the amount of time the server spends transmitting the base layer has increased. Finally, the buffer requirement at the client will increase because it needs to store more prefetch data. Using a really small window will result in the delivery of data similar to the simple adaptation algorithm. That is, it is much more sensitive to fluctuations in the bandwidth availability across the network. As more bandwidth becomes available, the frame rate immediately increases; as less bandwidth becomes available, the frame rate immediately decreases.

To illustrate the importance of using a reasonably size window, we have graphed three examples in Figure 7. In these examples, we have used 30 priority levels (each containing one frame per second) and a constant bandwidth allocation of 70% of the average bit-rate of the video stream. Here, we see that using a window of 50 frames, results in a very unstable delivery of video data to the user, highlighting the fact that the algorithms is not "looking" ahead very far. Going to a larger window of 500 frames results in a buffer utilization that is higher but improves the frame rate delivered to the user. Finally, we see that as the window size increases to 1000, it results in diminishing returns in the smoothing of the video rate.

The appropriate choice of the window size will ultimately depend on the user's preference on the frame rate delivered versus the stability of the frame rate. A larger window will result in a slightly lower average frame rate delivered to the user and fewer frames from the enhancement layers (assuming bandwidth is limited). A smaller window size will tend to maximize the number of frames delivered but can result in larger swings in the frame rate delivered to the user. We can, however, specify an approximate limit to the window size that can be used if the client has a fixed size buffer. This maximum window size should adhere to the following:

$$\text{Approx. Maximum Window Size} \leq \frac{\text{Buffer Size}}{\text{Ave. Bandwidth Available (in bytes/frame)}}$$

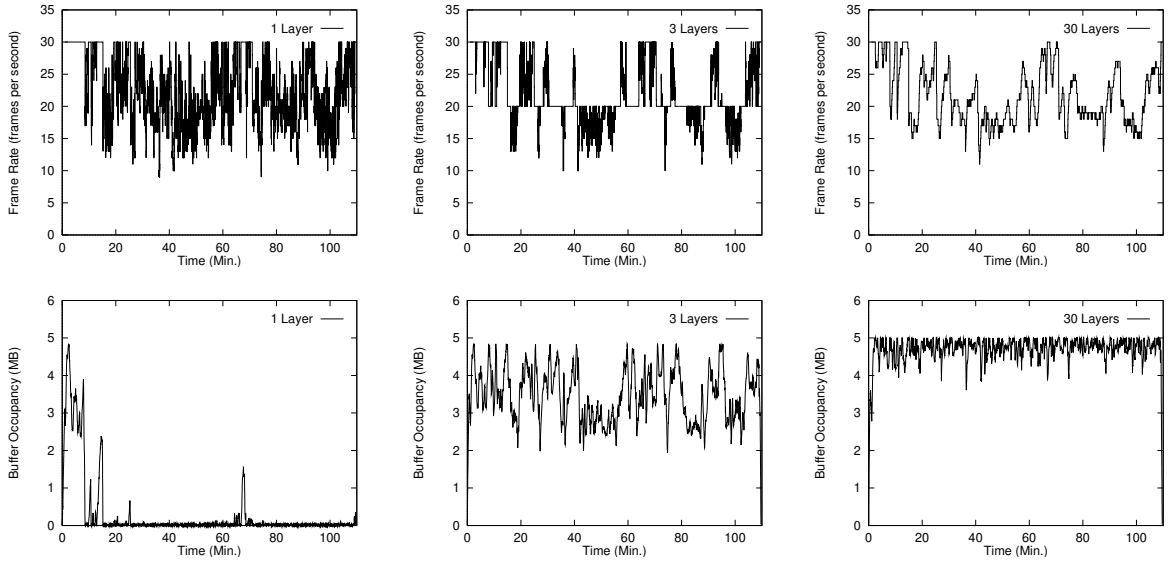


Figure 8: Effect of Number of Layers. These graphs show the performance of delivering *E.T.* with constant bandwidth of 0.7 times the average frame size for 1, 3, and 30 layers.

That is, this equation tells us how many frames the buffer can hold at a given time. A window size much larger than this approximation will increase the chances that the buffer will overflow.

3.3. Number of Priority Levels

Another important parameter for this algorithm is the number of levels to use for the frame rate shaping. Using a single level for the priority queue results in the same delivery as in the simple adaptation algorithm presented in the background section *with one exception*. The window size will limit the buffer that is built up on the client because the window size limits the number of frames transmitted. Recall, the simple adaptation algorithm does not have this constraint. Using a large number of levels allows the frame rate to be more gradually adapted and allows the dropped frames to be more evenly spaced out. The main disadvantage, however, is that there is a certain amount of overhead involved in "context-switching" between the various layers. As we will discuss in the next section, this problem can be somewhat alleviated with a technique we call *thresholding* for the layers.

To illustrate the effect of using different layers, we have graphed several examples with different numbers of layers in Figure 8. As shown in the figure, using a single level results in a plan that is nearly the same as the simple video adaptation algorithm shown in Figure 1. Using 3 layers with each layer containing 10 frames per second results in frame rates that are more "blocky" centered around 10, 20, and 30 frames per second. This highlights the fact that using too many frames per second in any layer will result in a clumping of frame drops as the network bandwidth changes. Finally, we see using many layers tends to spread the frame rate delivered more evenly. Through our simulations, we have found that using a base layer (highest priority) with more frames approximately equivalent to the available bandwidth does not effect the frame rate drastically, as one would expect. As a result, using a layering method such as 15 frames per second in the highest priority followed by 15 layers of 1 frame per second works well.

3.4. Thresholding

The window-based mechanism can result in excess overhead in switching between the multiple layers. For example, a frame from the highest priority layer may be delivered, followed by a frame at the next highest priority. If sufficient time has elapsed, another frame from the highest layer will need to be sent, resulting in frames from the multiple layer being finely interleaved. To make implementations simpler, *thresholding* can be used to force the algorithm to send frames from a single layer for a slightly longer period of time before switching to another layer. Using thresholding, we assign a threshold t_i to each layer in the priority queue with window size w_i ($t_i \prec w_i$). Once a particular layer has it's w_i frame requirement met, we allow the window to shrink to w_i - t_i *before* we begin sending data from that layer again. Our results have shown that setting a threshold of approximately 10% of the window size does not significantly impact the performance of the algorithms but will provide for simpler implementations.

4. Experimentation

In this section, we compare and contrast our proposed algorithm with the simple video adaptation algorithm as well as the adaptable video streaming algorithms presented in Section 2.3. We first show the performance of the algorithms under variable bandwidth availability. We will then describe the performance of the priority-based algorithm compared with the previous approaches using the metrics of minimum frame rate, average frame rate, and variability of frame rate delivered to the user. The variability in frame rate measures how large the fluctuations in frame rate tend to be (with smaller variation in frame rate being better [15]). While the frame rate variability does not tell the entire story of frame rate fluctuations, it gives us the approximate range of frame rates that the user will see. For example, we may have one plan that alternates between 15 and 30 frames per second while another plan that has 30 frames per second for the first half of the video and 15 frames per second in the second half; each of these end up with the same frame rate variability. Of course, the latter does *not* typically occur so the frame rate variability is fairly indicative of the frame rate fluctuations seen by the user.[1]

For the experiments in this section, we have chosen to use the M-JPEG (similar to all I-frame MPEG) encoded movie *E.T. - The Extra Terrestrial* with a quality index of 90. This quality index results in a compression value of 1.3 bits/pixel (corresponding to “good/excellent” picture quality [19]). For the priority-based smoothing algorithm we have chosen to use a 30-layer model with one frame per second at each layer in order to cover the broadest range of bandwidth availability. As previously described, this will allow us to have the smoothest frame delivery.

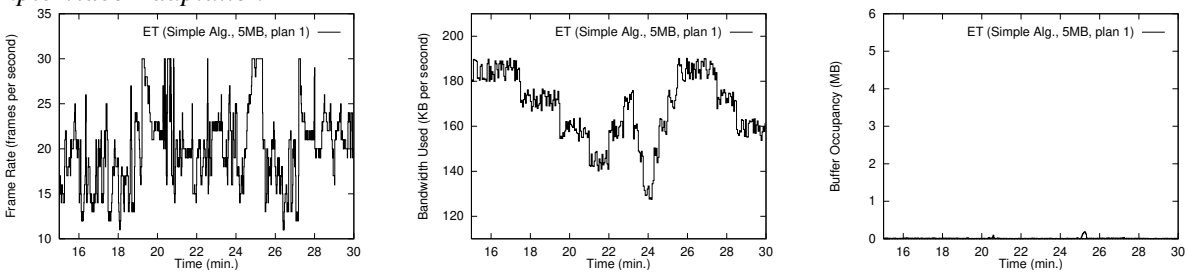
4.1. Performance Under Variable Bandwidth Availability

In this section, we highlight the performance of the priority-based adaptation algorithm compared with the other adaptation algorithms under bandwidth availability that is changing over time. The bandwidth availability graph is shown in Figure 9(b) of the simple adaptation algorithm and ranges from 50% to 70% of the average frame rate for the video stream (132 Kbytes/sec to 185 Kbytes/sec). For the simple adaptation algorithm, we see that it suffers from significant frame rate variability delivered to the user, ranging from 11 frames/second to 30 frames/second. In (c), we see that very little buffering of data occurs because the bandwidth availability is never greater than the frame rate for extended periods of time.

In Figure 9 (d) to (i), we have graphed the similar graphs for the modified critical bandwidth allocation algorithm as well as the adaptable critical bandwidth allocation algorithm. In these figures, we see that these algorithms result in frame rates that are much smoother over time compared with the simple adaptation algorithm. In particular,

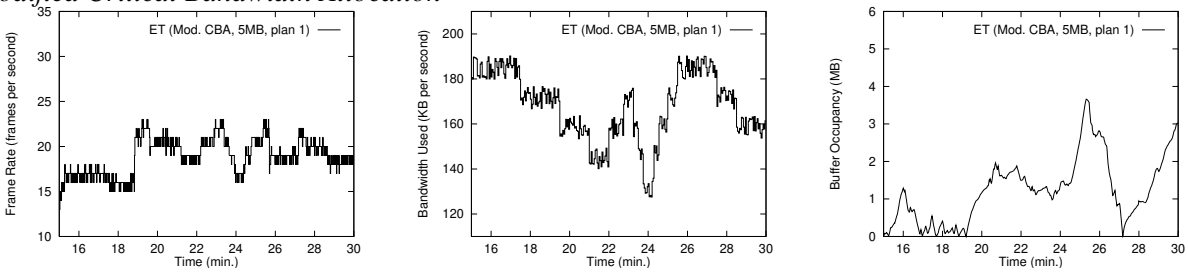
1. If the reviewers know of a metric that more accurately captures the essence of frame rate fluctuations, we would appreciate your suggestions.

Simple Video Adaptation



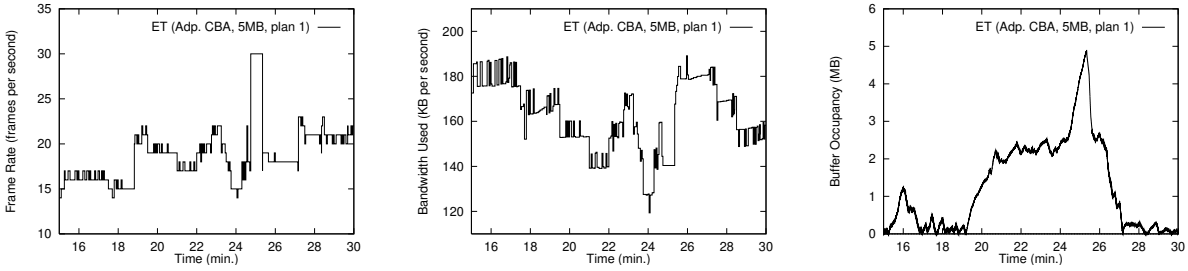
(a) (b) (c)

Modified Critical Bandwidth Allocation



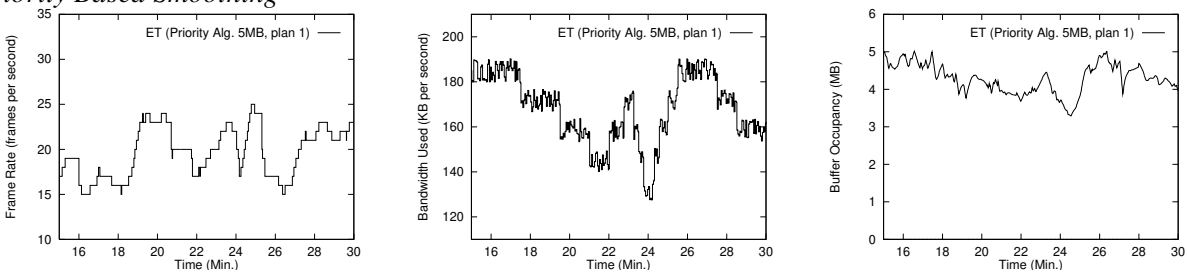
(d) (e) (f)

Adaptive Critical Bandwidth Allocation



(g) (h) (i)

Priority Based Smoothing



(j) (k) (l)

Figure 9: Performance Under Varying Bandwidth Availability. These figures show the comparison of the priority-based delivery algorithm compared to the other video adaptation algorithms for bandwidth availability that varies over time. The actual bandwidth availability is reflected in the bandwidth allocation figures for the simple adaptation algorithm as well as the priority based smoothing algorithm. The left figures show the frame rates delivered to the user (measured by counting frame delivered in every 30 frame interval). The right graphs show the associated buffer occupancy graphs for the respective algorithms.

we see that the buffer builds up over time in order to ensure a higher frame rate in the future. We also note the small differences between the modified and adaptable CBA-based algorithms. First, the modified CBA approach is a more conservative approach making sure that the buffer will never overflow [10]. As a result, we see that the buffer does not approach the full 5 megabytes in this sample. Second, in figure (g) we see that the frame rate goes to 30 frames per second for a short period of time (minutes 24.5 to 25.5). During this time, two things have happened: (i) the buffer is nearly full and (ii) the bandwidth availability has increased. As a result, the algorithm can no longer prefetch ahead (because of the full buffer) and therefore uses the bandwidth to fill in as many frames as possible.

In Figure 9 (j - l), we have graphed the same graphs for the priority-based smoothing algorithm. In (l), we immediately see the difference in approach to the previous algorithms: the buffer utilization remains fairly high throughout the delivery of the video stream. As shown by (j), the priority-based algorithm effectively smooths the video frame rate over the interval. Because the algorithm does not look at *all* the frames in the future when doing the rate shaping, it results in periods where the frame rate is higher than the adaptable algorithms (e.g. minutes 19-21) and times when the frame rate is less (e.g. minutes 25 to 27).

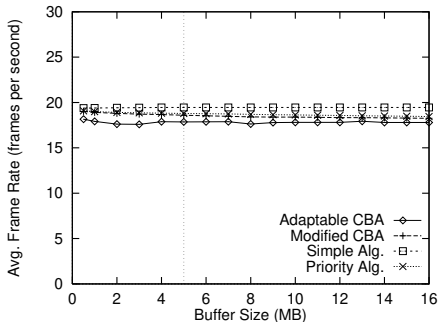
4.2. Effect of Buffer Availability

To compare the various algorithms, we have graphed the average and minimum frame rate as well as the frame rate variability delivered to the user for various buffer sizes. In these experiments, we held the bandwidth availability constant at 60% of the average frame rate (158 Kbytes/second) and allowed the window size to increase until it full utilized the buffer. In Figure 11 (a), we see that all three algorithms deliver approximately the same average frame rate with the simple adaptive algorithm always delivering the best average frame rate followed by the priority-based algorithm, the modified CBA algorithm, and the adaptable CBA algorithm (in that order). Because the adaptable CBA algorithm attempts to smooth the video stream over longer periods of time than the other algorithms, it will always try to prefetch the very large frames in the future to ensure the smoothest frame rate. These large frames essentially replaces one or more smaller frames that could have been delivered resulting in the slightly smaller average frame rate. The simple adaptive algorithm results in a constant average frame rate independent of the buffer size. This is mainly due to the fact that at 60% of the average frame rate, the buffer does not ever have a chance to build up any buffer. Finally, the priority-based algorithm results in similar performance to the modified CBA algorithm.

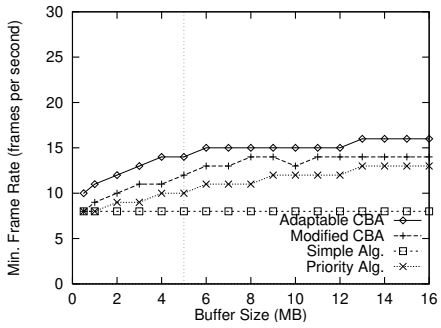
In figures (b) and (c), we see that the slight reduction in the average frame rate allows the adaptive delivery algorithms to deliver a stream with higher minimum frame rates as well as smaller frame rate variation. In (c), the priority-based and adaptive algorithms improve the variation in frame rate because they actively prefetch data. For buffers larger than 8 megabytes, the priority-based algorithm shows a reduction of 2 frames per second while the adaptive algorithms show a reduction of 3 frames per second in the frame rate variability. Finally, we note that in order to more fairly compare these algorithms, we have used a *single constant bandwidth allocation per run*. As a result, it is expected that the modified and adaptable CBA-based algorithms will have a more difficult time achieving higher frame rates than the priority-based algorithm, which provides a simpler adaptable mechanism.

4.3. Effect of Bandwidth Availability on Frame Rate

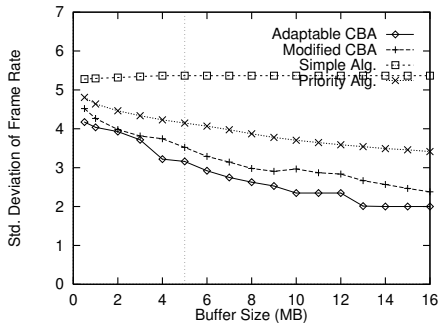
In Figure 11, we have graphed the performance of the algorithms using a 5 megabyte smoothing buffer while adjusting the window size of the priority-based algorithm to take advantage of the buffer. As expected (and as shown in (a)), the more bandwidth that is available, the higher the frame rate that can be delivered to the user. In particular, the ordering of the various algorithm is the same as in the previous section. In (b), for bandwidth availability less than the average movie bit-rate, the priority-based algorithm falls in between the simple adaptation algorithm and the critical bandwidth-based algorithms, as expected. For bandwidth availability greater than 130% of the



(a) Average Frame Rate



(b) Minimum Frame Rate



(c) Standard Deviation in Frame Rate

Figure 10: Effect of Buffer Availability. These figures show the effect of smoothing buffer size on the frame rate average, minimum, and variability. In these experiments, the bandwidth availability was constant at 60% of the average frame rate. In addition, for the priority-based algorithm, the window size was increased to completely fill the buffer at one point during the video. The thin vertical line shows the "intersection" point between the graphs in this figure and those shown in Figure 11.

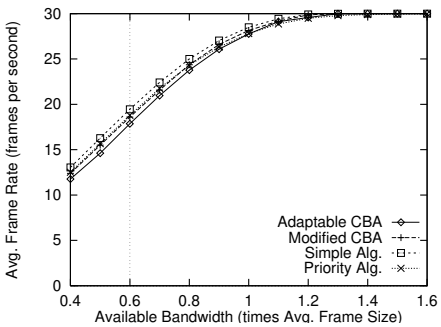
average movie bit-rate, the priority-based algorithm tends to does *worse* than the other algorithms for both the minimum frame rate and the frame rate variability. There is a relatively simple explanation for this phenomena. Suppose, we have a window size for all layers of 300 and a frame i at the lowest layer in the priority queue. Under the priority-based algorithm, all the frames in the window between i and $i+300$ must be sitting in the buffer before frame i is delivered to the client buffer. Under the other algorithms, the video is more "streamed" in that frame i is always delivered before frames greater than i. This result suggests that under extremely plentiful bandwidth conditions, consolidating all the layers into a single priority will be beneficial for the priority-based smoothing algorithm. We will discuss this particular point in more detail in the Conclusion and Future Work Section.

5. Acknowledgments

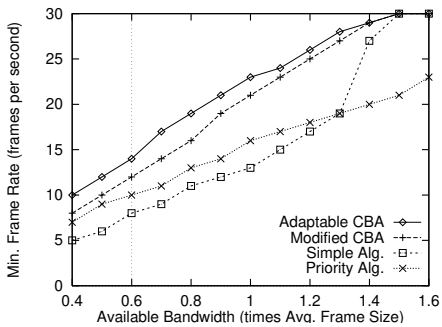
We would like to thank (in advance) any useful comments the reviewers can give us concerning this document. This work was supported in part by grants from the Ohio State University Research Foundation and Ameritech. The work in this paper does not necessarily reflect the views of the funding agencies.

6. Conclusion and Future Work

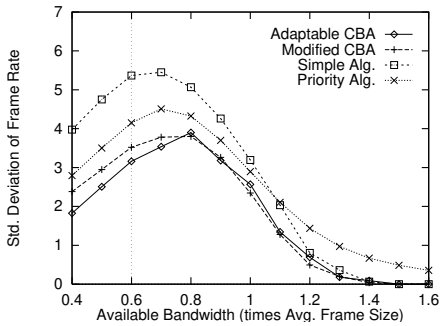
In this paper, we have introduced a new priority-based technique for the delivery of compressed prerecorded video streams across best-effort networks. This technique bridges the gap between algorithms that do short-term



(a) Average Frame Rate



(b) Minimum Frame Rate



(c) Standard Deviation in Frame Rate

Figure 11: Effect of Bandwidth Availability. These figures show the effect of bandwidth availability on the frame rate average, minimum, and variability. In these experiments, the buffer was set to 5 megabytes. The window size for the priority-based algorithm was then adjusted to take advantage of the buffer. The thin vertical line shows the "intersection" point between the graphs in this figure and those shown in Figure 10.

video adaptation and the computationally complex algorithms that take advantage of the *a priori* information. The proposed approach has several advantages over previous approaches. First, it provides a simple mechanism for which to specify frame dependencies such as the I, P, and B frames found in MPEG, allowing the stream to be shaped *without* having to use complex calculations to figure out which frames to drop. Second, this algorithm is much simpler to implement than the previous adaptive shaping algorithms while providing better smoothing than the simple adaptation algorithm. Finally, this approach allows the user to easily specify their preferences in terms of frame rate stability versus the total number of frames delivered through the specification of the window size.

Our results have shown that using the priority-based adaptation algorithm allows us to reduce the frame rate variability and to increase the minimum frame rate delivered to the user at the cost of a slightly lower average frame rate (compared with the simple adaptation algorithm). We have also shown that the priority-based algorithm gives us the advantages of the adaptable critical bandwidth based approaches without the overhead of complex calculations in the delivery of the stream.

We are currently implementing our priority-based adaptation algorithm into the Berkeley Continuous Media Toolkit and hope to have a demonstration of its capabilities by the conference. Future work entails the dynamic management of the window size and the layering to changing network conditions. In particular, we are investigating techniques for altering the layering as the bandwidth availability becomes large relative to the frame rate required.

7. References

[1] Shanwei Chen, Johnathan Walpole, Calton Pu, “Flow and Congestion Control for Internet Media Streaming Applications”, In *Proceedings of SPIE Multimedia Computing and Networking 1998*, Kevin Jeffay, Dilip Kandlur, Timothy Roscoe, Editors, Proceedings of SPIE Vol. 3310, pp 250-264.

[2] Zhigang Chen, S.M. Tan, R. Campbell, Y. Li, “Real Time Video and Audio in the World Wide Web”, in *Fourth International World Wide Web Conference*, Boston, Massachusetts, December 1995.

[3] Wu-chi Feng, S. Sechrest, “Smoothing and Buffering for Delivery of Prerecorded Compressed Video”, In Proceedings of *IS&T/SPIE Multimedia Computing and Networking* , Feb. 1995, San Jose, CA, pp. 234-242.

[4] Wu-chi Feng, S. Sechrest, “Critical Bandwidth Allocation for the Delivery of Compressed Prerecorded Video”, *Computer Communications*, Vol. 18, No. 10, Oct. 1995, pp. 709-717.

[5] Wu-chi Feng, F. Jahanian, S. Sechrest, “An Optimal Bandwidth Allocation Strategy for the Delivery of Compressed Prerecorded Video”, *ACM/Springer-Verlag Multimedia Systems Journal,* September 1997.

[6] Wu-chi Feng, “Rate-Constrained Bandwidth Smoothing for the Delivery of Stored Video”, in *Proceedings of the 1997 SPIE Multimedia Computing and Networking Conference*, San Jose, CA, Feb. 1997.

[7] Wu-chi Feng, Jennifer Rexford, “A Comparison of Bandwidth Smoothing Techniques for the Delivery of Compressed Prerecorded Video”, *Proceedings of IEEE INFOCOM ‘97*, Kobe, Japan, pp. 58-66, April 1997.

[8] Wu-chi Feng, “Time-Constrained Bandwidth Smoothing for Interactive Video-on-Demand Systems”, *Proceedings of the 13th International Conference on Computer Communications 1997*, pp. 291-303.

[9] Wu-chi Feng, B. Krishnaswami, A. Prabhudev, “Proactive Buffer Management for the Delivery of Stored Video Across Best Effort Networks”, *to appear ACM International Conference on Multimedia*, Bristol, UK, September 1998.

[10] Wu-chi Feng, Ming Liu, “Adaptable Critical Bandwidth Allocation for Stored Video Delivery Across Best-Effort Networks”, *submitted for publication.*

[11] Steve McCanne, V. Jacobson, “VIC: A Flexible Framework for Packet Video”, Proceedings of *ACM Multimedia 1995*, November 1995.

[12] Jean McManus, K. W. Ross,“Video on demand over ATM: Constant-rate transmission and transport,” In *Proceedings of IEEE INFOCOM*, pp. 1357-1362, March 1996.

[13] RealNetworks - http://www.real.com

[14] J. Rexford, S. Sen, J. Dey, W. Feng, J. Kurose, J. Stankovic, D. Towsley, " OnlineSmoothing of Live, Variable-Bit-Rate Video ", in the *International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '97)*, May 1997.

[15] L.A. Rowe, B.C. Smith, “A Continuous Media Player”, *Proceedings of the 3rd International Workshop on Network and Operating Systems Support for Digital Audio and Video,* San Diego, CA, Nov. 1992.

[16] L.A. Rowe, K. Patel, B.C. Smith, K. Liu, " MPEG Video in Software: Representation, Transmission and Playback," *Proc. of IS&T/SPIE 1994 Int'l Symp. on Elec. Imaging: Science and Technology*, San Jose, CA, February 1994.

[17] J.D. Salehi, Z.L. Zhang, J.F. Kurose, D. Towsley, “Optimal Buffering for the Delivery of Compressed Prerecorded Video”, in *Proceedings of ACM SIGMETRICS*, pp. 222-231, May 1996.

[18] H.M. Smith, M.W. Mutka, “Pattern Smoothing for Compressed Video Transmission”, *In Proceedings of IEEE International Conference on Communications*, June 1997.

[19] Gregory Wallace, “The JPEG Still Picture Compression Standard”, *Communications of the ACM*, Vol. 34, No. 4, April 1991, pp. 30-44.\

[20] Jonathan Walpole, Rainer Koster, Shanwei Cen, et. al, "A Player for Adaptive MPEG Video Streaming Over The Internet", *Proceedings 26th Applied Imagery Pattern Recognition Workshop*, SPIE, Washington DC, October 15-17, 1997.

[21] J. Zhang, J. Hui, “Traffic Characteristics and Smoothness Criteria in VBR Video Traffic Smoothing”, In *Proceedings of IEEE Multimedia Systems 1997*, June 1997.