Proactive Buffer Management for the Streamed Delivery of Stored Video

Wu-chi Feng, Brijesh Krishnaswami, Arvind Prabhudev
The Ohio State University
Department of Computer and Information Science
Columbus, OH 43210-1277

Abstract

Abstract: Many techniques have been introduced for the adaptive delivery of video across networks such as the Internet. These algorithms can react to network load by dynamically altering the quality of the video stream in real-time, either through a reduction in quality of image or frame rate. For stored video streams, however, these techniques do not take advantage of the *a priori* information available from the stored data. Bandwidth smoothing techniques, which are effective in reducing the resource requirements for the streamed playback of stored video, use this *a priori* information but cannot adapt to changes in network conditions. In this paper, we propose the *proactive* management of streamed stored video delivery over best effort networks like the Internet. Our proposed scheme attempts to maximize the sustained rate of the video by proactively monitoring the available bandwidth and the buffer. Our results show that the quality of video delivered to the desktop can be improved with our technique.

1. Introduction

The ability to efficiently support the streamed delivery of video has received much attention from the research community, ranging from video server design and retrieval techniques, to adaptive coding of video, to low-level network protocols that support streamed real-time video applications. One of the main research threads in delivering streamed video across networks has been focused on the adaptability of applications to changing network conditions. By using feedback from the network, the sender can dynamically alter the video quality through a change in quantization scales in encoding, a change in the frame rate, or a prioritization of the more important parts of the video. These techniques, however, typically continue to dynamically adapt the quality of the video to the available network resources rather than using the *a priori* information about the video to make decisions of what quality of video should be transmitted across the network. In particular, they typically use small amounts of buffer to remove only network jitter instead of proactively monitoring the frame data to be sent, the amount of data in the buffer, and the bandwidth currently available from the network.

Bandwidth smoothing techniques have been offered as a potential alternative to delivering stored video streams. Many of these algorithms are provably optimal (under certain conditions) by *completely* analyzing all the *a priori* information about a movie, namely its frame sizes. For example, the *minimum changes* *bandwidth allocation algorithm* results in a plan for the delivery of video that has the minimum peak bandwidth requirement and the minimum number of rate changes for continuous uninterrupted playback for a fixed client side buffer. While these techniques are extremely useful in dedicated video-on-demand systems (where bandwidth is typically more plentiful), these algorithms are woefully inadequate for delivery across best-effort, Internet-like networks. The main reason for this inadequacy is the fact that these algorithms "tell" what bandwidth is required rather than altering the bandwidth requirement of the video dynamically into the available bandwidth.

In this paper, we propose a *proactive buffer management technique* for the delivery of compressed prerecorded video streams across best-effort networks. This algorithm is different than previous streamed video approaches because it attempts to maximize the minimum frame rate that the user sees by using the *a priori* information that is available with the stored source. By proactively managing the buffer, the video server can use the buffer to keep the quality more constant. That is, in times of low network bandwidth relative to the frame sizes, the algorithm drains the buffer. In times of high network bandwidth relative to the frame sizes the algorithm starts to build the buffer more *rather than playing at a higher frame rate*. By proactively managing the buffer, we show that the frame rate viewed by the client can be smoothed, giving the appearance of a more stable video stream to the user.

In the following section, we review some of the related work in streaming protocols, adaptive video techniques, bandwidth smoothing techniques, and list the assumptions that we have made for this paper. In Section 3, we describe our *proactive buffer management scheme* and describe a possible alternative to the proactive buffer management scheme. In Section 4, we describe the experiments that we have conducted and show how the proactive buffer management scheme improves the stability of the received frame rate. Finally, we conclude with a summary and possible directions for future work.

Contributions of this paper: In this paper, our main contribution is the idea of proactive buffer management for the delivery of stored video streams across best-effort networks. To date, we are unaware of any techniques that use the buffer ostensibly to "smooth" out the frame rate of the video received. The techniques shown in this paper are the first step in making video transmission protocols that take advantage of the *a priori* information that may help stabilize the quality of the streamed video. Finally, this work complements the current work on streaming protocols and adaptable video algorithms (as we will describe in the Background and Related Work section).

2. Background and Related Work

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

2.1. Adaptable Video Techniques

The research in video streaming protocols has occurred at several layers in the network protocol stack. Several approaches provide an integrated meddler layer between the application and the network, while other approaches provide only the streaming protocol or the techniques to adaptively alter video streams.

Integrated Adaptable Video Delivery Techniques

Many 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 [10],
- The Continuous Media Toolkit (from Berkeley) [15],
- Quasar's Adaptable Streaming Video Player (for Oregon Graduate Institute) [17],
- Vosaic - University of Illinois' streaming video protocol [2], and
- RealAudio/RealVideo Streaming protocol [14].

to name a few. These integrated streaming approaches have several common features. First, they all provide end-to-end delivery of video streams. That is, they encompass both the presentation and transport layers of the protocol stack. Several of these, such as VIC, also provide the end application as well. Second, they adapt to changes in the network load and scale the media to fit within the available bandwidth. Finally, with the exception of RealAudio/RealVideo, these protocols use small amounts of buffering to remove network jitter that is introduced but *do not* use the buffer as a tool to smooth the quality of the video that is viewed. That is, they reactively adjust the quality of the video to match network resources. RealAudio/RealVideo is the notable exception to this. In their approach, they buffer some data in the client and try to stream the video at the rate available. If the buffer runs out, "jerkiness" is perceived in the playback and either the quality of the video stream is reduced or a pause is inserted to again buffer some more data. Our approach for the *proactive buffer management* of streamed stored video is to use the protocols that these researchers have proposed but augment it with buffer management techniques. In particular, we attempt to *smooth* out the video playback rate by not adjusting the quality of video immediately to what is available.

Video Streaming Protocols

In addition to the work of their integrated MPEG streamed video player, researchers at OGI have started to examine flow and congestion control for Internet media streaming [1]. Their protocol, called the Streaming Control Protocol (SCP), is a UDP based implementation that attempts to provide TCP-like flow and congestion control mechanism for streamed media applications. SCP attempts to smooth out the packet rate that is sent, allowing upper layers (such as our proposed proactive buffer management scheme) to have a more stable bandwidth target to aim for.

Adaptive Video Techniques

In addition to the basic rate adaptation techniques used by the integrated end-to-end delivery mechanisms, which include frame rate and picture quality, several other works have proposed the prioritization of digital video streams. The main idea of these works is to prioritize the data through mechanisms such as the cell-loss priority bit in the ATM header to allow the network to alter the video on-the-fly by dropping the less important data. This work has taken two flavors: prioritizing the video stream structure and DC components of a video stream with higher priority (losing AC values during network congestion)[12], or using the cell-loss priority bits to induce video frame-losses instead of random packet losses [13].

2.2. Bandwidth Smoothing Techniques

Bandwidth smoothing algorithms have been proposed for the delivery of compressed prerecorded video streams. These algorithms *smooth* the bandwidth requirements for retrieval of the video stream from the video server and transmission across the network. By prefetching data ahead of time, the buffer can be used to supplement network bandwidth when the video stream reaches an area of large frame sizes. By completely analyzing the *a priori* information available from a stored video source and a fixed size client-side buffer, several optimal bandwidth smoothing algorithms are possible. In particular, given a fixed client-side buffer, it is possible to minimize the peak bandwidth requirement for the continuous delivery of the video stream while

- minimizing the number of bandwidth rate increases required [5],
- minimizing the total number of bandwidth changes required [6],
- minimizing the variability of the bandwidth requirements [16],
- minimizing the buffer utilization of the client-side buffer [7],
- delivering video in constant bandwidth allocation periods [11], or
- adhering to a buffer residency constraint [9].

The 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 shared network environments such as the Internet, this assumption will not allow the video to be delivered continuously in most circumstances.

The notion of bandwidth smoothing, however, can be useful in the delivery of stored video streams in shared network environments. In particular, using a client-side buffer while analyzing the *a priori* information should allow us to have fine-grained control over the frame rate (as well as quality) of video delivered to the client. By taking advantage of the *a priori* information available, we can use both the buffer and the bandwidth availability to help determine the frame rate that we should send data.

One algorithm that our proposed scheme requires an understanding of is the *rate constrained bandwidth smoothing algorithm* (RCBS) which minimizes the buffer utilization for a given rate constraint. This algorithm, for a given rate r, starts at the end of the movie and works towards the beginning of the movie. Any frames that are larger than the rate r are prefetched in advance, filling in valleys that occur with prefetch data. If the rate is small relative to the frame sizes, the RCBS algorithm will build a large prefetch buffer requirement with the bandwidth requirement equal to the rate r for the entire movie. If the rate is large relative to the frame sizes, the RCBS algorithm will deliver the frames at the bandwidth requirement for each frame with little prefetching of data. For a more detailed discussion of this algorithm, see [7].

2.3. Assumptions in This Paper

To make the algorithms more understandable, we have made several assumptions throughout the paper. First, we restrict our attention to a single constant bandwidth allocation. Solving the video delivery problem for a single constant bandwidth allocation allows us to solve the more general case of fluctuating bandwidth requirements. In the general case, it is expected that the lower layer transport protocols will help provide a more stable stream of data or that a statistical process control technique can be used to help determine the bandwidth availability [3]. Second, we assume that sufficient client-side buffer has been set aside for the removal of delay jitter that the lower layer protocols have. Third, we focus our attention to the problem of maximizing the minimum frame rate of video delivered. It is expected that in an implementation of these algorithms that the user would have the ability to set the parameters for the relative importance of the frame rate versus the quality of the video received. To show the impact of quality of video, we will show the difference in frame rate for three different encodings of the movie E.T. Fourth, we restrict our focus to video compression algorithms with non-dependent frame types. In particular, we will be using M-JPEG encoded video clips (similar to all I-frame encoded MPEG video streams). Using M-JPEG allows the discussion of the algorithm to be simplified. Extensions required for support of MPEG video streams are presented in Section 3.3. Finally, recognizing that the full frame rate of the video can vary, we assume that the full frame rate is 30 frames per second to make the discussion simpler.

3. A Proactive Buffer Management Scheme for Stored Video Delivery

In this section, we describe our proposed approach for the streamed delivery of stored video. After establishing the framework for proactive buffer management, we describe some extensions that are necessary to support frame-patterned video compression such as that found in MPEG. Finally, we describe some of the alternatives to delivering video in best effort networks.

3.1. The Proactive Buffer Management Scheme

In the streamed delivery of compressed stored video streams over best effort networks, all the *a priori* information about a video stream is still available. The main difference, however, is that all the data may not be actually viewed by the user. In the *proactive buffer management scheme* we propose to use the *a priori* information to maximize the frame rate by proactively monitoring the data that is buffered and the current bandwidth rate that is achievable through the network. We then make a bandwidth plan that attempts to minimize the fluctuations in frame rate. In particular, we are interested in maximizing the minimum frame rate that the user sees.

To make the description of our proactive buffer management scheme simpler, we need to define several terms:

- We call a region defined by frames i and j ($i < j$) *feasible* given a maximum bandwidth rate, r (in bytes/frame) if:

$$\frac{\sum_{k=i}^{j} frame_k}{(j-i)} \le r$$

  where $frame_k$ is the size in bytes of frame k. We use the notation $feasible(i, j, rate, fps)$ to denote the test of feasibility which returns either true or false. If the frame rate, fps, is less than the full frame rate, the test temporarily alters the video stream by setting the frame sizes to 0 to simulate dropped frames. The details of frame dropping can be found in Section 3.2.

- We call a region defined by frames i and j ($i < j$) *buffer feasible* given a maximum bandwidth rate, r (in bytes/frame), and a maximum buffer size, b (in bytes), if:

$$\forall i \le k \le j \qquad \sum_{l=k}^{j} (frame_k - r) \le b$$

  where $frame_k$ is the size in bytes of frame k. We use the notation *buff_feasible*($i, j, rate, b, fps$) to denote the test of feasibility which returns either true or false. If the frame rate, *fps*, is less than the full frame rate, the test temporarily alters the video stream by setting the frame sizes to 0 to simulate dropped frames. The details of frame dropping can be found in Section 3.2.

- The feasibility test determines whether an area can be played without any prefetch at the given bandwidth rate. The buffer feasibility test determines whether or not it is possible to play frames i to j with no greater prefetch than b bytes and no buffer build up greater than b bytes between i and j. Thus, in order to determine whether it is possible to playback frames i to j with no prefetch, a buffer of size b, and a maximum rate of r, we must have both *feasible(i, j, r)* and *buff_feasible(i, j, r, b)*. Note, the *feasible(i, j, r)* test determines whether or not it is possible to playback the video with no prefetch and the *buff_feasible(i, j, r, b)* test ensures that the buffer constraint b is not violated.

Given these tests, we can describe our algorithm for the proactive buffer management of streamed stored video delivery. The user must first determine the amount of prefetch data that he/she is willing to wait for. The algorithm first determines whether it is possible to playback the movie at the *full frame rate* adhering to the buffer constraint (if any) and the prefetch requirement. That is, it checks *buff_feasible(0, N-1, rate, prefetch, 30)*. If buff_feasible returns FALSE, then the frame rate or the quality of the video stream must be reduced. To reduce the frame rate of the video, the algorithm performs several steps. We first reduce the frame rate by 1. Implementation details of how to reduce the frame rate are described in Section 3.2. Starting at the end of the movie, we work toward the beginning of the movie looking for the first frame, i, where

$$feasible(i, N-1, rate, 29) = TRUE$$

and

$$buff\_feasible(i, N, rate, b, 29) = TRUE \quad .$$

If no such i exists, then we cannot reach the end of the movie given our prefetch and buffer requirements at 29 frames per second so we continue reducing the frame rate, *fps*, by one until we find a frame, i, where

$$feasible(i, N-1, rate, fps) = TRUE$$

and

$$buff\_feasible(i, N, rate, b, fps) = TRUE \quad .$$

The frame rate, *fps*, and frame, i, for which this occurs will allow us to play from i to the end of the movie without buffer overflow and underflow while still adhering to the network bandwidth rate constraint. We then "break" the movie into two parts [0, *i-1*] and [*i*, *N-1*]. We then reset the frame rate to 30 frames per

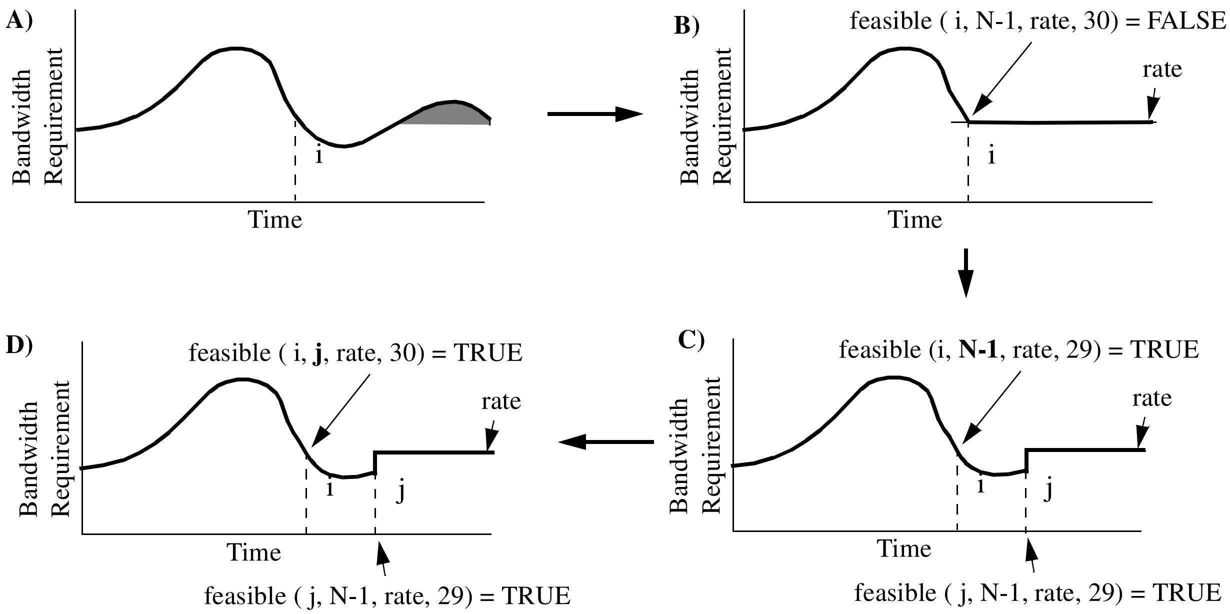

Figure 1: Proactive Buffer Management Example. Figure A shows the bandwidth requirements to deliver the video to the user. The shaded region shows the data that will be prefetched in the previous valley in bandwidth requirements. Figure B shows the *partial* search for a feasible region from *N-1* downto *i*. Continuing the search will result in buff_feasible(0, N-1, rate, prefetch) returning FALSE, requiring the frame rate to be reduced. Figure C shows the *partial* calculation of the feasibility tests at 29 frames per second. For frame *j*, it is possible to play from *j* to the end of the movie at 29 fps so we "break" the movie at frame *j*. As shown by Figure D, returning back to 30 frames per second for the region [0, *j-1*] will allow us to play from *i* to *j* at 30 frames per second. Thus, returning to the full frame rate after every "break" will allow us to play at 30 frames per second where the frame sizes are smaller relative to the bandwidth rate. In this example, the search will continue with the segment [0, *i-1*] at 30 frames per second.

second for the segment [0, *i-1*] and start the whole process over again. An example of this procedure is shown in Figure 1. The caption of Figure 1 describes why it is necessary to return the frame rate to 30 frames per second for segment [0, *i-1*] through a simple example.

The pseudo-code for this algorithm can be found in Figure 2. Note, if we are not concerned about the buffer limitation, the buffer_feasible check in the middle of the nested loops is replaced with a check to feasible instead (using the same parameters). An example of this algorithm applied to the movie *E.T - The Extra Terrestrial* (Quality 90) is shown in Figure 3. From this figure, we see that the proactive buffer management scheme attempts to keep the frame rate steady over relatively long periods of time. In particular, the algorithm will build a sizeable buffer of data throughout the playback of the data.

3.2. Implementation Details

In this subsection, we describe some of the implementation details of the algorithm, allowing readers interested in the high level concepts to skip to the next section. To implement the proactive buffer management scheme, two basic sub-algorithms are required, a method to reduce the frame rate and a method for

```
END = N-1;
STOP = FALSE;
if (buff_feasible(0,END,rate,b)&&(!feasible)
   STOP = TRUE;

While (!STOP){
   feasible = FALSE;
   for (fps = 30; (fps > 1)&&(!feasible) ; ){
      /*** Try and find a place to 'split' the movie ***/
      for (find_start = END ; (find_start > 0)&&(!feasible) ; ){
         /* zero out some frames temporarily to make the frame rate */
         reduce_frame_rate(find_start, END, fps);

         if (buffer_feasible (find_start,END,rate)){
            BEGIN = find_start;
            feasible = TRUE;
            }
         else
            find_start--;
         }
      /* Reached the beginning of the movie or have a feasible plan */
      if (buff_feasible(0,END,rate,b)&&(!feasible)
         /*** check to see if the prefetch amount is ok ***/
         STOP = TRUE;
      else if (!feasible)
         fps--;
      }

   if (!stop)  /*** we've broken the movie into two parts ***/
      set the frame rate from find_start to END to fps;
   END = BEGIN-1;
  }
set the frame rate from 0 to END to fps;
```

Figure 2: Buffer Management Pseudo-Code. This figure shows the pseudo-code for the proactive buffer management scheme. Optimizations have been omitted.

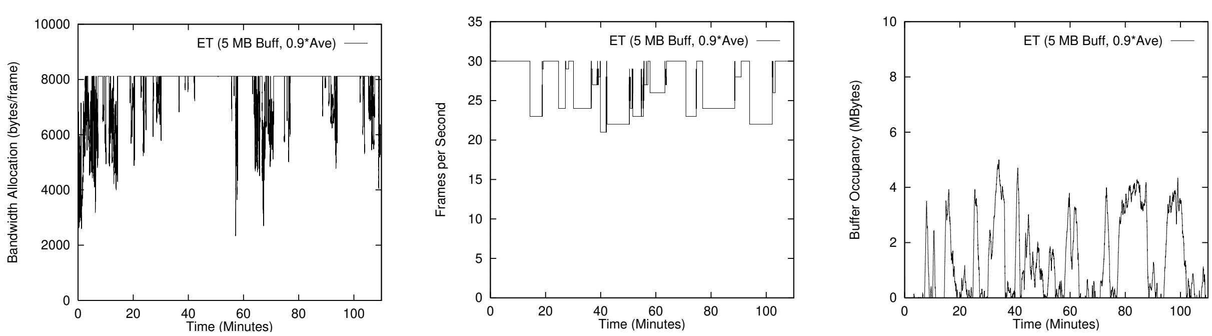

Figure 3: Proactive Buffer Management Example. These figures show an example application of the proposed proactive buffer management scheme for the movie *ET* (Quality = 90) for a 5 MByte buffer using a maximum rate of 90% of the average bandwidth requirement. The middle figure shows the frame rates (averaged over one second intervals) that are delivered to the client. The right graph shows the buffer occupancy of the client buffer.

calculating the feasibility tests.

To reduce the frame rate of the video, we specify the frame rate (in integer frames between 1 and 30). Using this frame rate, we make sure that enough frames are dropped over a single second to meet the frame rate. To make the frame rate as steady as possible, we spread the frames that are dropped as much as possible. For frame drop rates that are divisors of the maximum frame rate, the frame dropping results in evenly spaced dropped frames. As an example at 28 frames per second, the first and fifteenth frames are dropped in each second. For frame drop rates that are not evenly divisible into the frame rate, we spread the frame drops as evenly as possible. For example at 26 frames per second, the first, seventh, fifteenth, and 22nd frames are dropped.

To implement a single feasibility test, we use the *rate-constrained bandwidth smoothing* algorithm with the appropriate frame rate. To achieve the frame rate, we simply set the frame size to 0 in the algorithm. Thus, a frame drop results in the previously played frame to last twice as long assuming it wasn't dropped. While using evenly spaced frame drops may not result in the largest reduction in data dropped, it ensures that the frame rate is constant (over periods of one second). In the calculation of the feasibility test, each frame is examined. If it is larger than the rate, the difference between the rate and the frame size is added to the prefetched data required. If the prefetched data required exceeds the maximum buffer, then the segment is infeasible given the rate, buffer, and starting and end points. On frames that have been dropped (set to 0), the frame is used to reduce the prefetch data required.

3.3. Extensions to Support MPEG Video Streams

In order to support MPEG encoded video streams in our framework, several considerations need to be taken into account. Because MPEG has potentially many frame dependencies, the frame dropping algorithm must be modified to account for this. In particular, B-frames should be dropped in preference to P and I frames, since dropping the P and I frames may result in other frames being unplayable. To implement frame dropping for MPEG streams, it helps to have a frame rate where only B frames will be dropped. This will result in frame drops that may not be evenly spaced out.

3.4. Alternatives to Proactive Buffer Management

In this section, we describe an algorithm that uses a buffer to aid in the delivery of video data across networks. At a high-level, the alternative method uses a client-side buffer to buffer data when the frame rate is smaller than the bandwidth available. If the buffer fills up, the server "waits" for buffer space to free up and then transmits the frame. When the buffer is empty, the client has no choice but to drop the frame. To model an "optimal" drop policy for frames, we employ the following. If the buffer contains less data than the frame we want to playback, we mark the frame as "dropped" and "save" the data as part of the next frame instead of discarding a partial frame. For each frame, we add the maximum bandwidth available into the buffer, assuming that there is room.

An example of this algorithm is shown in Figure 4. There are several interesting points to mention concerning this figure. First, compared to the proactive buffer management scheme, the simple algorithm is *greedy* in that it uses all the available bandwidth until the buffer gets completely full (see the buffer occupancy graph in the 0 to 10 minute range). Once the frames get to be larger, the buffer drains and will

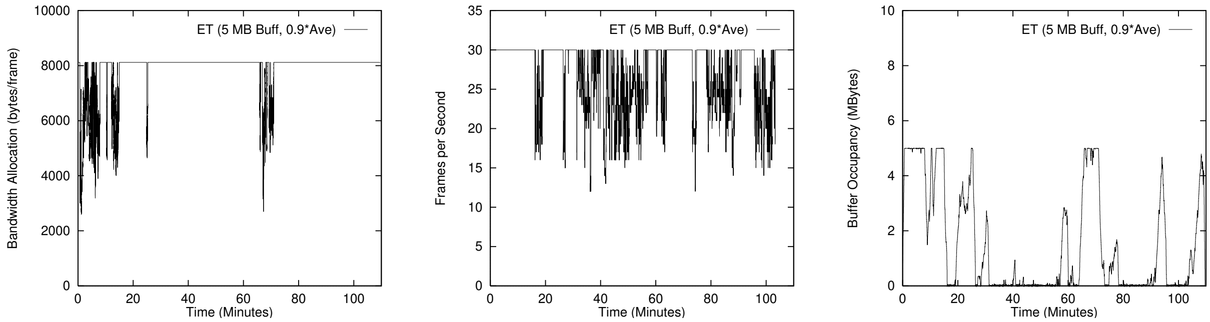

Figure 4: Simple Buffer Management Example. These figures show an example application of the simple buffer management scheme for the movie *ET* (Quality = 90) for a 5 MByte buffer using a maximum rate of 90% of the average bandwidth requirement. The middle figure shows the frame rates (averaged over one second intervals) that is delivered to the client. The right graph shows the buffer occupancy of the client buffer.

continue to be drained (from time 30 to 50 minutes) until an area of small frames allows the buffer to be filled again (time 62 to 70 minutes). It differs from the proactive buffer management scheme in that the proactive scheme will sacrifice frame rate to build the buffer for areas of large frames (see time 40 to 60 minutes in Figure 3).

4. Experimentation

In this section, we show the results of the experiments we conducted. In particular, we highlight the advantages that our proactive buffer management techniques have over a simple buffering technique that "optimally" delivers video frames at as high a rate as possible. We have chosen to use the movie *E.T. - The Extra Terrestrial* that is Motion-JPEG encoded using three different quality encodings. This will allow us to show the effect of quality on the reduction in frame rate. The *E.T.* video streams have the following statistical properties:

Quality	Max Frame Size (bytes)	Min Frame Size (bytes)	Ave Frame Size (bytes)	StdDev of Frames (bytes)
75	14269	1511	6305	1840
90	19961	2333	9022	2574
100	30553	6827	15749	3294

In the rest of this section, we show how different bandwidth availability affects the two algorithms and show the effect of varying buffer sizes on the minimum frame rate in delivery of the video.

4.1. Effect of Bandwidth Availability

The availability of bandwidth will greatly affect the frame rate of the video playback that can be achieved. In particular, a very large rate relative to the frame sizes will typically allow for the playback of video at the full frame rate. Alternatively, a very small rate relative to the frame sizes will result in a larger reduction in playable frames at the client. To highlight the difference of the algorithms in a relatively low

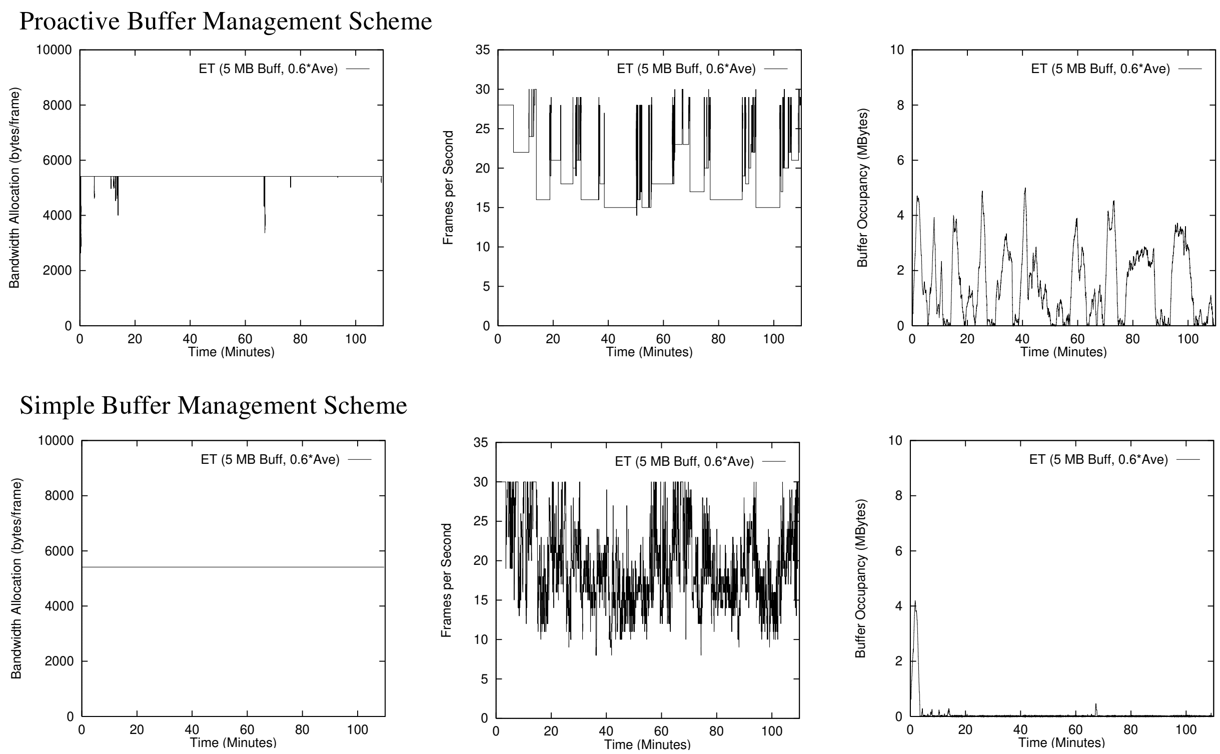

Figure 5: A Comparison of Buffering Techniques. These figures show a comparison of the proactive and simple buffering schemes for the movie *ET* (Quality = 90) using a maximum rate of 60% of the average frame rate. The left figures show the bandwidth allocation curves, the middle figures show the frame rates (averaged over one second intervals), and the right curves show the buffer occupancy curves.

bandwidth situations, we have graphed a comparison of the algorithms (similar to those found in Figure 3 and Figure 4) using a maximum bandwidth of 60% of the average frame size in Figure 5. As shown by this figure, the simple buffer scheme has frame rates that vary between 10 and 30 frames per second. In addition, we see that the buffer occupancy once the small frames in the beginning of the movie have passed is essentially 0 for the rest of the movie, highlighting the fact that the average bit rate is rather low. For the proactive buffering scheme, we see that it sustains constant frame rates for relatively long periods of time and will have buffer occupancy measurements that are nearly full. Also shown in these figures are the bandwidth allocation curves. For the simple buffer management scheme, we see that it uses *all* the bandwidth that it is given. For the proactive approach, there are times where the bandwidth is lower than the maximum rate. These inverted spikes are due to the transition between one region of constant frame rates to another and usually have duration of exactly one frame (due mainly to the way we implement frame dropping).

4.2. Minimum and Average Frame Rate

Figure 6 shows the minimum and average frame rates for the proactive buffer management scheme and the simple buffering algorithms. We see two main trends. First, the simple buffering algorithm results in a higher frame rate compared to the proactive approach. Second, the simple buffering algorithm has a smaller minimum frame rates than the proactive approach. These trends can be explained as follows. During

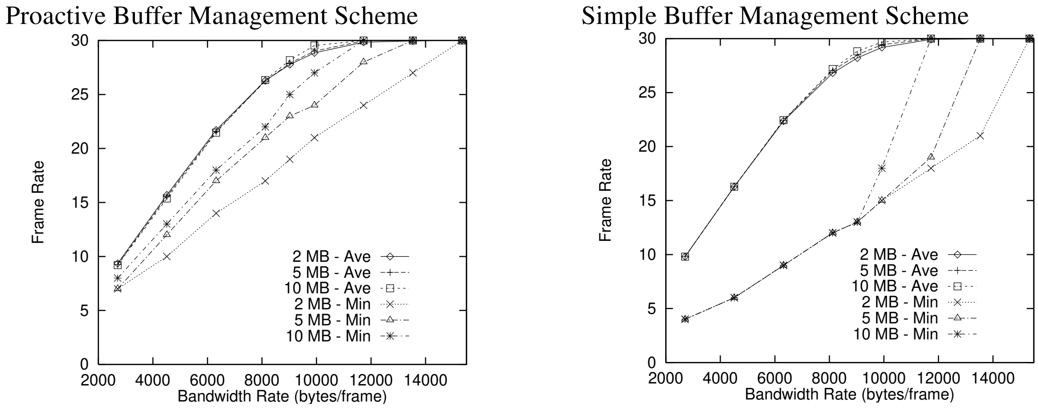

Figure 6: Minimum and Average Frame Rate. This figure shows the minimum and average frame rates for the two algorithms that we are comparing using 2, 5, and 10 MByte buffers for the movie *ET* (Quality=90).

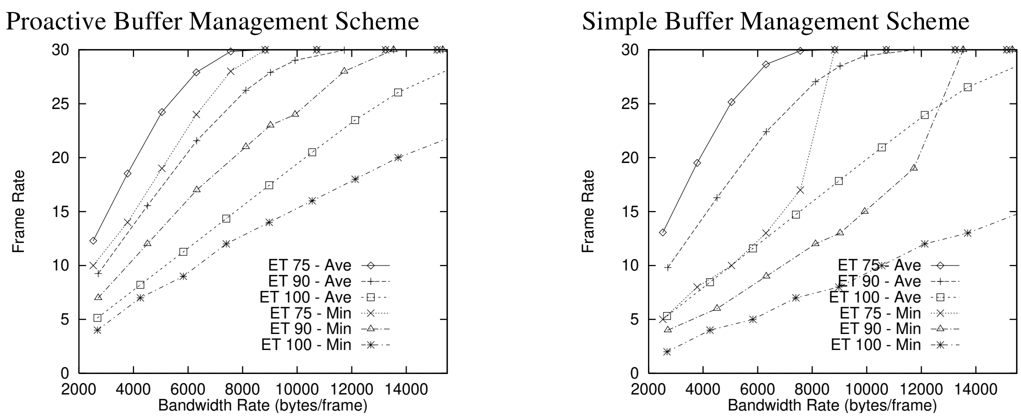

Figure 7: Effect of Quality on Frame Rate. This figure shows for the *E.T.* movies (Qualities 75, 90, and 100, the frame rate that is achievable given the bandwidth requirement. Note, the proactive lines are roughly ordered (from top to bottom) 75 ave, 75 min, 90 ave, 90 min, 100 ave, 100 min, while the simple approach has the ordering 75 ave, 90 ave, 100 ave, 75 min, 90 min, 100 min.

ing times of small frames, the simple buffering scheme is greedily sending small frames at the maximum rate, while the proactive approach is using this time to prefetch large frames. Thus, in the times of large frame sizes, the proactive approach has buffered some data, allowing it to achieve a higher minimum frame rate while the simple approach must drop lots of large frames. On average, the proactive approach produce a minimum frame rate that is 5 frames higher than the simple buffer management scheme. For very low bandwidth rates, the proactive approach typically doubles the minimum frame rate achievable through simple buffering techniques. Finally, both algorithms result in a full frame rate playback at the same points, as expected.

4.3. Effect of Quality on Frame Rate

Figure 7, shows the minimum and average frame rates using a 5 megabyte buffer for three different encoding of the *E.T.* video. Given these graphs and a given bandwidth rate, the user can chose the quality of video/frame rate trade-off that he/she is willing to make. For example, using the proactive approach and a bandwidth of approximately 8000 bytes/frame, the user can choose a minimum frame rate of approximately 20, 15, and 11 for the qualities 75, 90, and 100 respectively. As shown by the right figure, the simple buffering scheme's minimum frame rate is more sensistive to the network capacity than the proactive buffering scheme. For the movie *ET* (Quality = 75) the minimum frame rate nearly doubles as the bandwidth rate changes from 7500 to 8500 bytes/frame. We also note that the minimum frame rate typically occurs several times throughout the movie as shown by our sample simple buffering scheme graphs.

5. Conclusions and Future Work

In this paper, we have proposed a proactive buffer management scheme for the delivery of stored video across best effort networks. While many end-to-end streaming video applications and protocols exist, they typically use a buffer to smooth out network jitter only, continuing to adapt to network bandwidth as it become available or goes away. The one exception to this is the work from Readily/Relived which uses a buffer to prefetch data, but the data is not proactively managed, resulting in periods of very low frame rates and possibly a pause to prefetch some more data.

By taking advantage of the *a priori* information available from stored video streams, it is possible to analyze, given the current network bandwidth availability, whether one should increase the quality of the video stream when more network bandwidth becomes available relative to the video frame sizes or to hold the frame rate steady to ensure a reasonably constant frame rate in the future. By taking advantage of this information, it is possible to *smooth* out the frame rate received from the video server. It is the belief o the authors that users are more tolerable of a lower quality stream as long as the frame rate and quality are held more constant, rather than having large fluctuations in frame rate and video quality.

6. 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] Nelson Manohar, Marc Willebeek-LeMair, “Applying Statistical Process Control to the Adaptive Rate Control Problem”, In *Proceedings of SPIE Multimedia Computing and Networking 1998*, Kevin Jeffay, Dilip Kandlur, Timothy Roscoe, Editors, Proceedings of SPIE Vol. 3310, pp 42-56

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

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

[11]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.

[12]Pramod Pancha, M. El Zarki, "Prioritized Transmission of Variable Bit Rate MPEG Video", *Proceedings of IEEE GLOBECOM 1992*, pp 1135-1139.

[13]S. Ramanathan, P.V. Rangan, H.M. Vin, S.S. Kumar, "Enforcing Application-Level QoS by Frame-Induced Packet Discarding in Video Communications", *Computer Communications*, Vol. 18, No. 10, October 1995, pp 742-754.

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

[15]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.

[16]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.

[17]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.