A Hysteresis Based Approach for Quality, Frame Rate, and Buffer Management for Video Streaming Using TCP

Nagasuresh Seelam, Pankaj Sethi, and Wu-chi Feng

The Ohio State University
Department of Computer and Information Science
Columbus, OH 43210
{seelam, psethi, wuchi}@cis.ohio-state.edu

Abstract. In today's Internet, the primary transport mechanism for video streams is the UDP protocol, either congestion sensitive or not. In this paper, we propose a mechanism that supports the high quality streaming and adaptation of stored video across best-effort networks using the TCP transport protocol. Our proposed approach has a number of useful features. First, it is built on top of TCP, which effectively separates the adaptation and streaming from the transport protocol. This makes the question of TCP-friendliness, the behavioral property of a flow that allows fair-sharing of bandwidth with other flows, much easier to answer. Second, it does not rely on any special handling or support from the network itself, although any additional support from the network itself will indirectly help increase the video quality. Finally, we show through experimentation that this approach provides a viable alternative for streaming media across best-effort networks.

1 Introduction

With the rapid advances in the "last mile" networking bandwidth, such as the digital subscriber loop (DSL) and cable modems, the ability to achieve higher quality video networking is becoming more feasible than ever. As a result of moving from 28.8kbps or 56kbps coded streams to megabits per second, streaming of higher quality MPEG video is now possible. As the bit-rates for the video streams increases, the bandwidth requirements for the stream become more diverse over time, requiring more intelligent buffering and rate adaptation algorithms to be put in place.

One of the key issues in creating highly scalable networks is the notion of TCP-friendliness, which measures how fairly transport-layer flows are sharing network bandwidth [7]. While various definitions of fairness have been proposed, the key idea is that all the flows share the bandwidth and that they avoid sending the network into congestion collapse. The manifestation of TCP-friendliness in video applications has resulted in techniques such as congestion-sensitive UDP-based flows [2, 8], TCP-flows without retransmission [1], or limited retransmission UDP-flows [11].

Another key design issue is *how* video streams can be adapted to fit within the available resources of the network. For video conferencing applications and live video applications, the adaptation typically occurs along either adapting the quality of the video, adapting the frame rate of the video, or using a mixture of the two [9, 14]. For the delivery of *stored* video streams, the adaptation really occurs over three parameters: quality, frame, rate, and buffering (greater than that needed to remove

E.S. Al-Shaer and G. Pacifici (Eds.): MMNS 2001, LNCS 2216, pp. 1–15, 2001.

delay jitter). All of these can be used in isolation or combined together, however, efficient techniques need to be developed for the algorithms to deliver the highest quality video over the network.

In this paper, we propose a system for the delivery of *stored* video streams across best-effort networks that has a number of unique properties. One of them is that it uses the TCP protocol for flow and congestion control. Thus, the question of TCP friendliness is not a problem. In this work, we will show how to effectively deliver stored video content over best-effort networks using TCP. To show the efficacy of this approach, we have conducted simulations of the algorithms and have implemented the algorithms for MPEG streaming applications. Our results show that we can effectively combine frame rate, frame quality, and buffering into a single coherent framework.

In the rest of the paper, we will first describe the background and related work necessary for the rest of the paper. In Section 3, we describe our proposed approach as well as an evaluation metric called the *effective frame rate*. Section 4 provides the experimental results of our research, including a simulation-based study of the algorithms and an exploration of the effective frame rate measure. Finally, a conclusion and directions for future work is provided.

Contributions of this work: We believe that there are two main contributions to this work. First, we believe that this work uniquely demonstrates the effectiveness of frame quality, frame rate, and buffering for streaming of stored video streams across best-effort networks. Second, we introduce an evaluation metric for streaming video protocols called the *effective frame rate* measure which provides better insight into streaming quality than using just the average frame rate and the variation in frame rate.

2 Background and Related Work

There are a large number of related research ideas that dove-tail to the techniques proposed here. In this section, we briefly highlight some of this work.

2.1 Content Distribution Networks and Proxy-Based Delivery Mechanisms

There are a number of different approaches to the wide-scale distribution of video information. They are distinguished primarily by the way that the data is managed between the server that created the video data and the clients.

Content Distribution Networks (CDNs) focus on distributing the video information, in whole, to caches and proxies that are close to the end clients that require the information. For companies that provide access to web data, such as Inktomi or Akamai, this involves creating large distributed data warehouses throughout the world and using high-capacity, peer-to-peer networks to make sure the distribution infrastructure is up to date. The video data that is "streamed" is streamed from the cache to the end host.

Proxy-based dissemination architectures focus on actively managing streaming video content through the proxy. Such techniques include efforts from the University of Southern California (RAP) [3], the University of Massachusetts (Proxy Prefix Caching) [13], and the University of Minnesota (Video Staging) [17]. Unlike the

CDNs, these proxy-based mechanisms partially cache video data as it is streamed from the server to the proxy and through to the client. Additional requests to retrieve a video stream from the proxy will result in missing data being filled in at the proxy.

For our work, we assume that the proxies are far enough from the client to achieve a reasonable hit rate to the video data. As an example, the proxy could serve an entire campus (such as Ohio State). The streaming mechanism that we have developed can be used to deliver data from the data warehouses in the CDNs to the clients or can be augmented to include the techniques that have been developed for proxy-based video distribution networks.

2.2 Video Streaming Protocols

There are a large number of streaming protocols that have been developed for point-to-point video distribution. For sake of brevity, we will purposely omit discussion of techniques developed for the distribution of video over multicast networks such as [15]. For point-to-point video streaming, there are a tremendous number of streaming algorithms including the Windows Media Player, RealPlayer, the Continuous Media Toolkit (CMT) [12], Vosaic [2], the OGI Media Player[16], and the priority-based streaming algorithm [4].

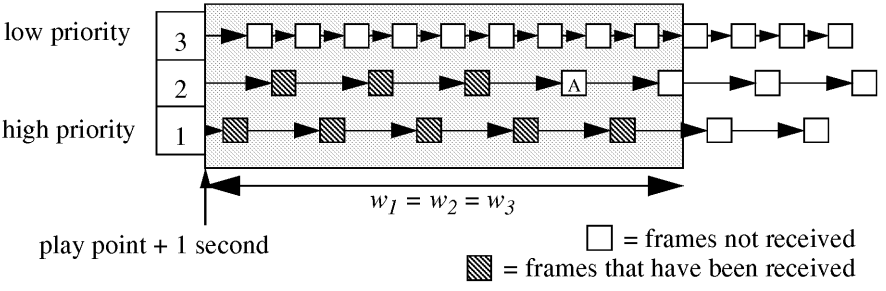


Fig. 1. This figure shows an example of the priority-based streaming algorithm. The window keeps track of which frames are buffered in the client. Using this window, the server always attempts to transmit the highest priority frame within the window that has not yet been sent. In this example, the first unsent frame in priority level 2 is transmitted next (labelled A)

These algorithms are for the most part network congestion sensitive, with some being arguably more TCP-friendly than others. In addition, most of these streaming algorithms focus on local adaptation of video sequences to network bandwidth availability and have either no or limited retransmissions of missing data.

The work proposed here uses buffer management uniquely by controlling the frame rate and quality based on a hysteresis loop which leads to a stabilized frame rate while using TCP as its transport protocol. We briefly describe the streaming mechanism here as it will be needed in the next section. The proposed algorithm prioritizes all the frames within a given video. This assignment can be assigned to gracefully degrade the frame rate as bandwidth becomes scarce. It does so by assigning frames to various priorities within a window (just ahead of the playpoint). This window is used to “look

ahead" into the movie at least a minute into the future. Using this window, it then delivers all frames in the window at the highest priority before delivering the lower priority frames within the window. As a result it is able to gracefully degrade the quality of video during times of congestion. An example of the windowing mechanism is shown in Figure 1.

3 Proposed Adaptive Streaming Mechanism for TCP Delivery of Stored Content

In this section, we propose a mechanism for the delivery of stored video content across best-effort networks. Before describing the algorithm, we first describe some basics of our approach including (i) some of the underlying assumptions that we make, (ii) a discussion of the trade-off between quality, frame rate, and buffering, and (iii) a metric for the evaluation of streaming media.

3.1 Basics

3.1.1 Underlying Assumptions

There are a number of unique aspects of our approach that we believe are important for the delivery of stored video content over best-effort networks. These characteristics are described below (some more contentious than others):

- **Moving towards more bursty video sources** - As the bandwidth over the network increases, the ability to stream higher-bit-rate MPEG streams becomes possible. This increase in bandwidth will create streams of greater bandwidth diversity than we currently see on the Internet.
- **Buffering at the Receiver** - We believe that buffering limitations in the client end systems are becoming a non-issue. We believe this because disk and memory capacity is outpacing the growth in bandwidth available to a single flow (not in aggregate). Thus, we can focus on getting the video data into the buffer without worry of buffer overflow.
- **TCP transport layer** - We believe that stored video streaming should happen over the TCP protocol. This is perhaps the most contentious part, but there are several reasons we believe this. First, TCP is a widely deployed transport protocol. This allows applications that are built on this architecture to be deployed rapidly over a large scale. Second, it allows us to resolve the TCP-friendliness issue rather easily: *it is TCP-friendly*. Third, we do want flow/congestion control *as well as retransmissions*. Flow/congestion control are fairly obvious. The reason we want retransmissions for stored video is that we are planning the delivery of the stored sources well in advance (even for B-frames), thus, when we transmit a frame we want to receive it in whole at the receiver.

3.1.2 Quality and Frame Rate Adaptation

For the delivery of stored video content, there are number of options available for adaptation over best-effort networks. Using these, we can control the quality of the video stream, the frame rate of the video stream, and the ordering in which the video data is streamed across the network. In fact, each of these can be adapted in isolation

or combined together in the delivery of video content. It is our belief that while any quality video can be used at any frame rate, we believe that there are *operating points* that will be used to correlate frame rate and frame quality.

Operating points define interesting sets of combinations of frame rate and quality. The streaming algorithms we develop are constrained to move from pre-determined operating points to other pre-determined operating points. Switching between operating points (as the bandwidth availability changes over the network) is guided by the principle of *equal priority* to frame-rate and quality. In other words, we choose not to compromise one parameter with respect to the other. Therefore any degradation/improvement in the streaming status would trigger comparable degradation/improvement in both the parameters. This is a fairly common approach as found in the Quasar streaming model as well as industrial models such as the Real Networks SureStream mechanism [10]. For frame rate adaptation, we allow the frame rate to change within a small set of values (e.g., 14-17 frames per second) for a fixed video quality level. Any further deviation from this frame rate will trigger the system to move to a different quality stream.

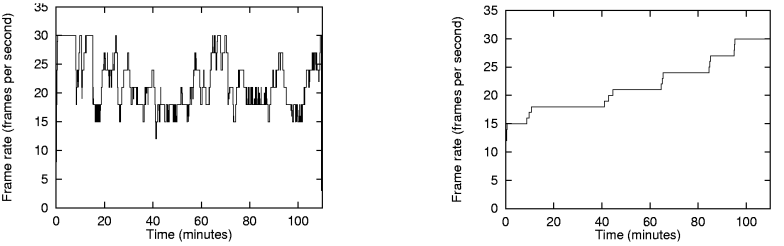
As an example of our philosophy, the operating point of (30 frames per second, low video quality) is not a reasonable operating point since we are severely sacrificing quality to achieve a high frame rate. Instead streaming at (15 frames per second, medium video quality) is a better choice in that it attaches equal importance to frame-rate and quality. For the rest of this work, we assume that multiple copies of the video stream are available at the encoded bit-rates, similar to the delivery mechanisms in the RealNetworks SureStream and the Oregon Graduate Institute streaming player. One could imagine with appropriate computing power that a transcoder could be implemented to do the same thing on-the-fly.

3.1.3 Effective Frame Rate

Our proposed video streaming algorithm attempts to maximize the frame rate displayed to the client, while minimizing the variability or the "jerkiness" in the display frame rate. While we believe that it is ultimately up to the end user to decide how aggressive or conservative the streaming algorithm is (while still being TCP compliant!), we believe it is important to define a metric that adequately measures this objective function in order to evaluate the performance of the algorithm.

In the past, researchers have often measured the average and variation in the frame rate delivered to the client as metrics for performance. While there is typically a strong correlation between standard deviation and jerkiness, this may not necessarily hold true. For example, consider a stream displayed at 10 frames per second for half the duration of the video, and at 20 frames per second for the remaining half. Additionally, consider a stream displayed alternately at 10 and 20 frames per second throughout the length of the video. The average frame rate and standard deviation in both cases is the same, but the perceived "jerkiness" in the former is much less than that in the latter.

As another example, we have graphed the frame rate delivered for a video sequence using the *priority based streaming algorithm* from reference [4] in Figure 2 (a). In Figure 2 (b) we have merely sorted the frame rate points making up figure (a). Again, it is clearly evident that the user would perceive much more "jerkiness" with the frame rate delivered in figure (a) than with (b). However, the standard deviation in both cases is the same (4.62 frames).



(a) (b)

Fig. 2. (a) shows an example of the frame rate delivered by the priority-based streaming algorithm. (b) shows the same frame rate points but sorted in ascending order. Both figures exhibit the same frame rate and frame rate variation.

We propose a metric that accounts for the jerkiness in the display frame rate more appropriately. We define *Effective Frame Rate* (*EFR*) as follows

$$EFR = FPS_{avg} - \frac{W}{n-1} \sum_{i=2}^{n} \left| framerate_i - framerate_{i-1} \right|^P$$

where FPS_{avg} is the average frame rate, $framerate_i$ is the number of frames displayed in the i^{th} second of the video clip, and W and P weighting factors that we will describe shortly. The idea behind *EFR* is that it first measures the average frame rate delivered to the user and then *penalizes* the *EFR* for variations in the frame rate delivered.

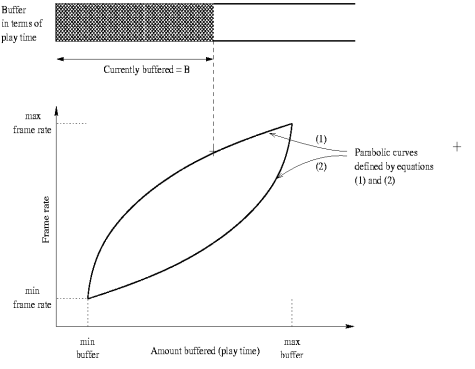


Fig. 3. Pictorial representation of the hysterisis scheme. Two parabolas shown determine how frame-rate should change when buffer level changes. Parabola P1 has the property that the frame-rate drops rapidly with buffer when it is close to the min buffer mark. On the other hand when the buffer level is close to the max buffer mark and dropping, the frame-rate does not drop as rapidly, thereby taking advantage of buffer. Parabola P2 has the property that when buffer level is close to the min buffer mark and rising, the frame-rate does not increase as rapidly, thereby building the buffer. On the other hand when the buffer level is close to the max buffer mark and rising the frame-rate increases at a higher rate thereby greedily utilizing excess bandwidth.

The penalty in the calculation of the frame rate is determined by two parameters in the metric, *W* and *P*. These two terms determine the nature of the penalty to be applied when the frame rate changes. The super linear term (*P*) is added to penalize

large changes in the frame rate more then small changes which in some cases may even be imperceptible. The linear function(W) is used to weigh the penalty for changes in frame rate. These two terms can be set to user preferences to allow one to evaluate the algorithms based upon their own idea of what makes a good streaming algorithm.

3.2 Hysteresis Based Video Adaptation

In this section, we describe our proposed approach. Our model first assigns each of the frames in the video stream to a priority level for streaming. As a simple case, one might assign all the I-frames in an MPEG sequence to the highest priority level, all P-frames to the second highest priority, and all B-frames to the lowest priority. Thus, no B-frame will be delivered before it's P-frame has been delivered. Also note that because we are using TCP as the transport protocol, we are guaranteed that the B-frame will have its reference frames in at the client. While we have described a very simple mapping here, our system is able to assign arbitrary priorities to frames depending upon the user's preferences. For example, one might map every alternating B-frame to a higher priority than the rest of the B-frames. In this case, during times of low bandwidth availability, every other B-frame will be delivered, spacing out the loss of frames. Finally, we create static priorities for all operating points for the movie that is stored.

3.3 Proposed Approach

Once we have the priority assignments, the next goal is to determine a plan for the delivery of the stored video. We observe that buffer occupancy is a measure of the frame rate and quality that can be supported. A shrinking buffer indicates either a decrease in average bandwidth and/or increase in frame sizes in the video. Similarly expanding buffer indicates increase in average bandwidth and/or decreasing frame sizes in the video.

The proposed approach is based on a hysteresis model. The streaming itself consists of two phases. First one is the prefetch phase wherein sufficient video is prefetched. In the second phase, the remaining video's play-time is divided into fixed length intervals. At the beginning of each time interval the streaming parameters, frame-rate and quality, are dynamically determined. For this a hysteresis based buffer monitoring mechanism is used.

The hysteresis model is pictorially represented in Figure 3. The algorithm tries to maintain the frame-rate constant when the buffer size lies between the two parabolas. Initially we use the parabola P1 to determine the frame-rate. Subsequently for each interval we check the buffer level to determine the next set of streaming parameters. For increasing buffer levels we use parabola P2 and for decreasing buffer levels parabola P1 is used. The parabolas P1 and P2 are defined as follows

$$f = f_{min} + (f_{max} - f_{min}) \bullet \sqrt{(B - b_{min} / b_{max} - b_{min})} \qquad \text{(P1)}$$

$$f = f_{min} + (f_{max} - f_{min}) \bullet (B - b_{min} / b_{max} - b_{min})^2 \qquad \text{(P2)}$$

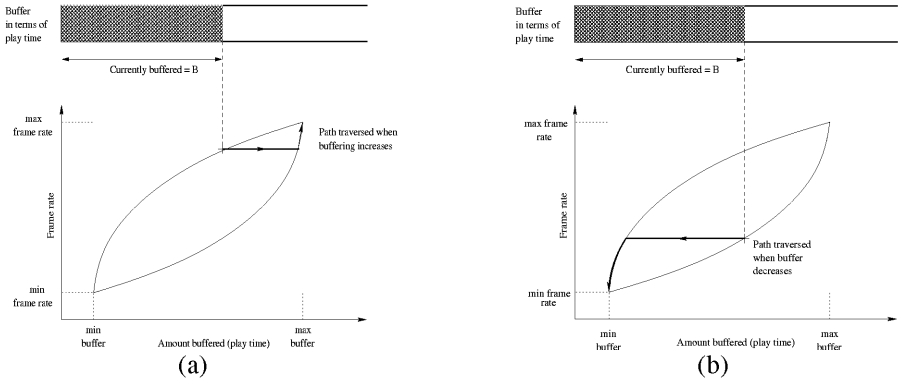


Fig. 4. Using parabola P1: For increments in buffer level the frame-rate is held constant at the previous value until the buffer occupancy hits parabola P2. Then P2 is used to recalculate frame-rates from this point when-ever buffer level increases.Using parabola P2: For drops in buffer level the frame-rate is held constant at the previous value until the buffer occupancy hits parabola P1. Then P1 is used to recalculate frame-rates from this point whenever buffer level decreases.

where

f is the new operating point. The new operating point depends on the following parameters.

B is the current buffer occupancy.

b_{min} is the minimum buffer occupancy allowed. This is same as the lower threshold of the hysteresis loop. It's typical value might be half a minute of video.

b_{max} is the maximum buffer occupancy allowed. This is usually the maximum amount of buffer available to the application. This is same as the upper threshold of the hysteresis loop. It's typical value is three to 5 minutes of video.

f_{max} is the maximum frame rate. Typically this is set to maximum which is 30 frames per second.

f_{min} is the minimum frame rate permitted. This is a user configurable parameter and indicates minimum frame rate which is considered reasonable by the user. Algorithm tries to maintain atleast this frame rate.

Assume we are currently using parabola P1 to make the streaming decisions. Figure 4(a) shows this scenario. Depending on the direction in which buffer occupancy changes, there can be two possible cases for the next interval.

- If the buffer level decreases then we use the current parabola-in-use (P1) to trace it downward.
- If buffer level does not decrease then we continue with the current frame-rate for the future intervals as long as the buffer level does not hit the parabola P2.

When the buffer occupancy hits parabola P2, this becomes the active parabola and the next streaming frame-rate parameter is decided based on this curve. Figure 4 (b)

shows this scenario. Here again, depending on the changes in buffer occupancy, there can be two possible cases for the next interval.

- If the buffer level increases we use the current parabola-in-use (P2) to climb upwards.
- If the buffer level does not increase then we hold to the current frame-rate as long as the buffer levedoes not hit parabola P1.

The motivation behind this is to achieve a constant frame-rate and quality video as long as the buffer occupancy remains between upper and lower thresholds of the hysterisis loop determined by parabolas P1 and P2. Frame rate is changed only when the buffer occupancy improves or degrades substantially. The hysteresis mechanism takes care of bursts of frames with large variation in sizes in the video by not changing the frame rate and the quality immediately. The maximum buffer limit and the minimum frame-rate are the user defined parameters. A higher maximum buffer limit allows more amount of video to be buffered thereby allowing a higher degree of smoothing of frame rate. The algorithm aims at building the buffer in advance to deal with a potential burst of larger sized frames in the future. The algorithm does not build too much of buffer. When the buffer occupancy increases beyond a threshold it increases the frame rate and/or quality of the video. The algorithm also provides cushion to the variations in bandwidth, immunity to short gaps in bandwidth and variations in the round trip times of the underlying transmission route by appropriately setting the lower and upper thresholds for the buffer occupancy.

4 Experimentation

In this section, we describe some of the performance results of our streaming algorithm compared with simple streaming algorithms that perform rate changes based on per-second adaptability and the priority-based streaming algorithm which uses windows that are in the size of minutes. We will also explore the use of the *effective frame rate* metric for measuring a streaming algorithm's performance. Before we describe our experimental results, however, we will describe our experimental simulation environment and the video data and network traffic traces that we captured.

4.1 Experimental Environment

We have captured and compressed two full-length movies (*Matrix* and *E.T. - the Extra Terrestrial*) into the MPEG compressed video format for use in the simulations[1]. For the movie *Matrix*, we used Future Tel's PrimeWare hardware digital video compression board. This board captured the analog output of a DVD player and compressed the movies into MPEG-2 video format. Using this board, we captured four streams at varying qualities for the simulations. For the movie *E.T. - the Extra Terrestrial*, we already had an MPEG-1 video stream compressed into three different

[1] The video and network traces will be made available via the web site: http://www.cis.ohio-state.edu/~wuchi/ Videos.

quality levels and are using those streams here. The statistics for the video trace data are shown in Table 1.

Table 1. This figure shows the statistics for the video traces that were used in the experimentation of the various algorithms

Movie	Ave. Bit Rate (Mbps)	Frames	Length (Minutes)	Ave. Frame Size	Ave. I-frame Size	Ave. P-frame Size	Ave. B-frame Size
Matrix	3.5	244301	129	13562	38341	16175	10039
Matrix	5	244301	129	19335	48686	24275	14424
Matrix	6.5	244301	129	24724	139733	28439	11736
Matrix	8	244301	129	30049	139733	39706	15217
E.T.	0.233	204679	113	965	4408	695	172
E.T	0.755	204679	113	3126	12883	2435	860
E.T	1.559	204679	113	6459	24591	6152	2004

To provide deterministic results of the various algorithms, we captured 2 two-hour long TCP traces. One was captured from the University of Michigan to Ohio State and the other was captured locally between two machines on different sub-networks. Thus, the comparisons that we draw in this paper are based upon the TCP traces. For actual streaming algorithms that just use UDP with or without flow control and with or without retransmissions may be slightly different than those presented here. We also note that comparing the performance of streaming algorithms that receive partial data (as in a UDP stream) is extremely difficult as it is hard to compute what the relative impact of receiving half a frame of video is for any streaming algorithm.

For comparison purposes, we use two streaming algorithms for comparison. The first, which we will refer to as the *naive* algorithm, simply calculates the bandwidth transmitted over the last several seconds to calculate the closest operating point for which to stream. This algorithm is very similar to the algorithms found in the CMT toolkit out of Berkeley or the Oregon Graduate Institute player. The *Priority-Based Streaming algorithm* is an extension to the basic *naive* algorithms that uses a much larger lookahead window. Here, the lookahead was set to two-minutes, meaning that all high-priority (I-frames) must be received before the algorithm starts delivering lower priority-frames (P and B). Thus, we expect the priority-based algorithm to provide substantially smoother operation than the *naive* algorithm. These two algorithms are explained in more detail in reference [6]. Finally, we note that while we are using simulation-based experimentation, all algorithms were not allowed to "look" at the future bandwidth availability from the network trace.

4.1.1 Experimental Results

In the rest of this section, we present several simulation results that demonstrate the effectiveness of the hysteresis-based streaming algorithm and that show how the *effective frame rate* metric that we propose work. Figure 5 shows the frame rate achieved by the naive, priority-based, and hysteresis-based streaming algorithms for the movie *Matrix* and the bandwidth trace shown in Figure 5 (a).

We see here that because the bandwidth availability drastically changes over time and the compressed video streams' network requirements vary significantly over time, the naive algorithm (as shown in Figure 5 (b)) has a very hard time sustaining a smooth frame rate for any significant period of time. This is because planning does not occur on anything greater than second-level in time making the frame rate delivered highly sensitive to the bandwidth requirements and network availability. We also note that the frame rate delivered by the naive algorithms is actually somewhat smooth over very small time scales but because we are displaying over 100 minutes in the figure, it appears significantly more bursty. In comparison, the priority-based streaming algorithm with a window of one-minute attempts to very conservatively change the frame rate over time by making sure a minute's worth of video at higher priority layers has been delivered before transmitting the lower priority frames. As shown in Figure 5 (c), the frame rate does not vary as drastically over time but is still somewhat bursty on a medium-term time-scale. We also see that the priority-based streaming algorithm is more conservative than the other algorithms in that it very slowly increases the frame rate of the video stream even when a higher frame-rate is possible. The main reason that the frame rate is still somewhat bursty is that the bandwidth requirements and bandwidth availability are changing over time, making it extremely difficult to track the available bit-rate that needs to be sent. Finally, the hysteresis-based streaming algorithm (as shown in Figure 5 (d)) does a much better job of smoothing out the frame rate delivered to the client. Buffering requirements are bounded by upper and lower thresholds and they also smooth out the changes in network bandwidth while keeping the bandwidth demand from the application somewhat more constant.

Applying our *effective frame rate* measurement to the algorithms, we have listed the various EFR measurements for a variety of W and P values in Table 2. By doing so, we can describe how the metric works with an example streaming session shown in Figure 2. For the parameters ($W=0$, $P=1$), no penalty is incurred by the algorithms for bursty frame behavior. As a result, the metric is a measure of how many frames were delivered. We see that under these parameters, the naive algorithm does the best. This is somewhat intuitive as the naive algorithm is very greedy. When there are small frames that are currently being transmitted, the algorithm fetches as many of them as it can, while the other approaches attempt to prefetch some of the larger frames in the future knowing that they need to prepare for these regions. We also see in Figure 2 that the priority-based algorithm is extremely conservative in its frame rate delivery to the client, with a consistently lower average frame rate. Under ($W=1$, $P=1$), the algorithms are penalized linearly for having bursty frame rates over time. The penalty here is the average amount of rate change between consecutive second intervals. It is important to note that simply achieving a constantframe rate is not the goal as it will undoubtedly result in a very low average frame rate in the first term of the EFR. Finally, we see that as the weight penalty increases the hysteresis-based approach does even better.

Figure 6 shows the delivery of the *Matrix* using the second bandwidth trace where the bandwidth availability is greater than in the previous experiment. We see here that the naive algorithm is able to achieve the full 30 frames per second delivery quite a number of times. We also see the same type of response as previously from the various algorithms. That is the priority-based algorithm is somewhat conservative,

while the hysteresis-based approach is able to smooth out the frame rate over considerable time-scales. The EFR measurement for the algorithms shown in Table 3.

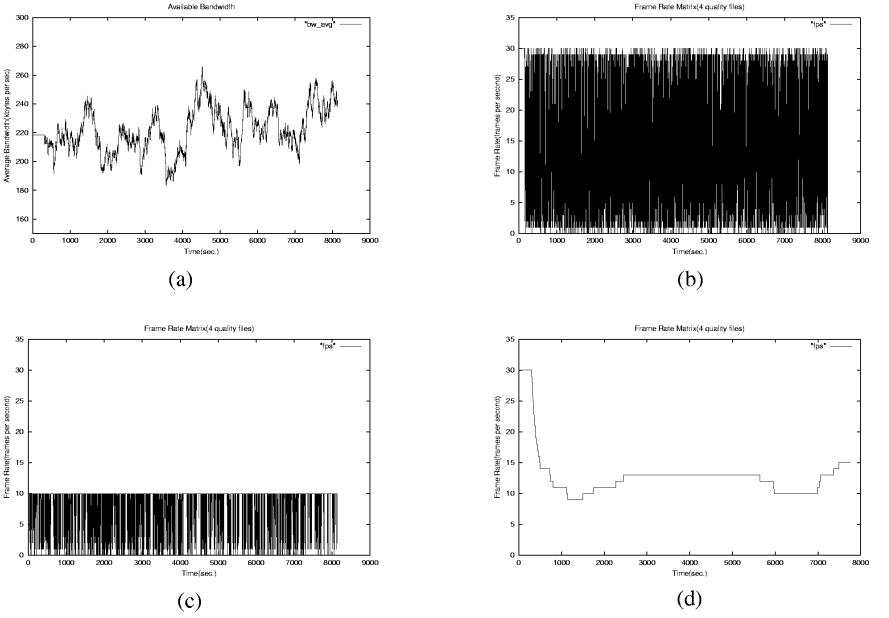


Fig. 5. This figure shows the frames rates obtained from the various algorithm for the movie *Matrix* using the bandwidth trace used in (a). Figures (b) through (d) show the frame rates of the naive algorithm, the simple priority-based algorithm, and the hysteresis-based approach for the entire duration of the movie. The variability in (d) is significantly lower then the other two algorithms.

Table 2. The effective frame rate for *Matrix* for frame rates shown in Figure 5.

Configuration	Naive	Priority-Based	Proposed approach
W = 0,P = 1	13.74	7.6	12.97
W = 1,P = 1	5.15	5.27	12.96
W = 3, P = 1	-12.03	.62	12.95
W = 1, P = 1.5	-18.22	1.84	12.96
W = 1, P = 2	-113.47	-7.42	12.96

As a final example, to show what happens when we change the movie to *E.T.* we have graphed the results in Figure 7. Finally, the EFR figures for the movie *E.T.* (shown in Figure 7) are shown in Table 4.

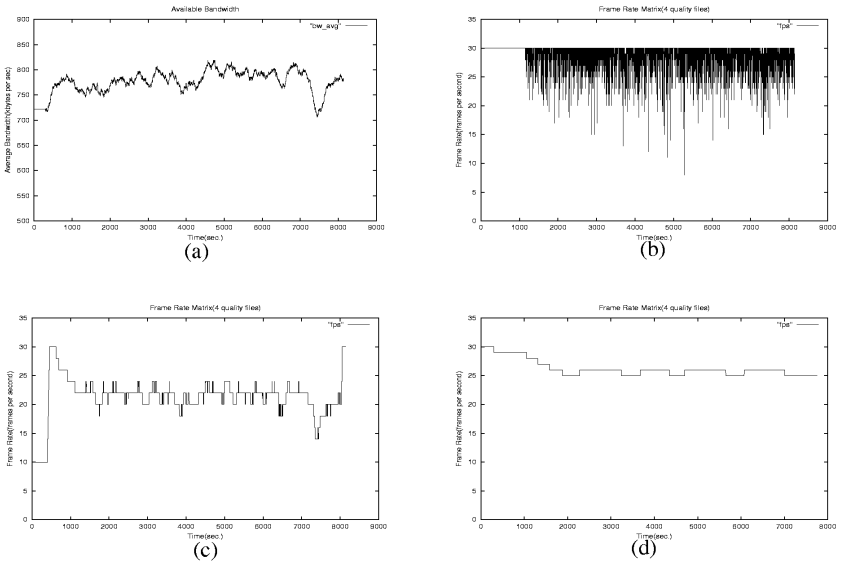


Fig. 6. This figure shows the frames rates obtained from the various algorithm for the movie *Matrix* using the bandwidth trace used in (a). The figures (b) through (d) show the frame rates of the naive algorithm, the simple priority-based algorithm, and the hysteresis-based approach. The variability in (d) is significantly lower then the other two algorithms

5 Conclusion

In this paper, we have introduced a hysteresis-based streaming algorithm that uniquely uses buffer management in the streaming of stored video data over best-effort networks. As we have shown, this approach is much more effective in smoothing out the variability in frame rate delivered to the user. We have also introduced an *effective frame rate* metric to evaluate the effectiveness of various stored video streaming algorithms. The key concept here is that instead of measuring just the average frame rate and the variability in frame rate delivered, it calculates an

Table 3. The effective frame rate for *Matrix* for frame rates shown in Figure 6.

Configuration	Naive	Priority-Based	Proposed approach
W = 0,P = 1	28.75	21.24	26.24
W = 1,P = 1	27.55	21.17	26.23
W = 3, P = 1	25.15	21.01	26.23
W = 1, P = 1.5	26.48	21.14	26.23
W = 1, P = 2	23.84	21.11	26.23

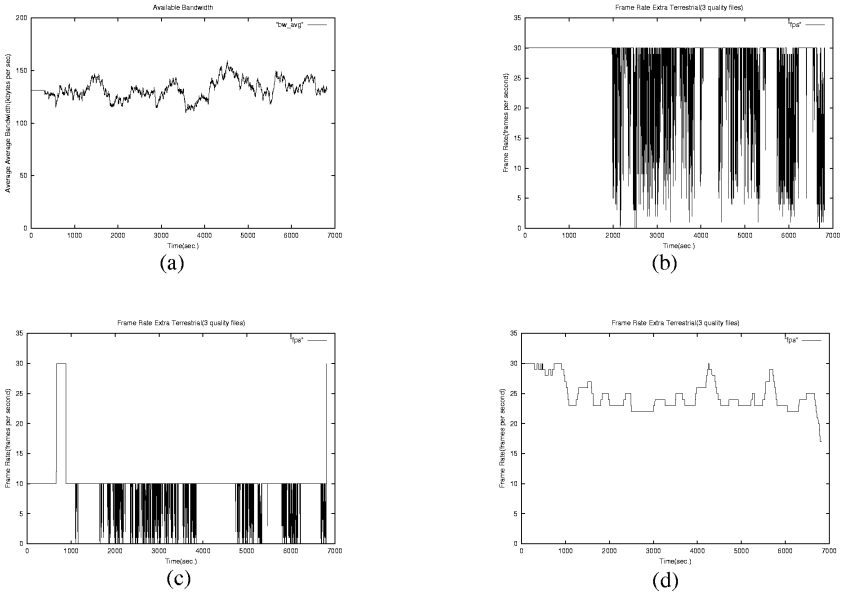


Fig. 7. This figure shows the frames rates obtained from the various algorithm for the movie *Extra Terrestrial* using the bandwidth trace used in (a). The figures (b) through (d) show the frame rates of the naive algorithm, the simple priority-based algorithm, and the hysteresis-based approach. The variability in hysteresis-based approach is significantly lower then the other two algorithms.

average frame rate and then penalizes the average frame rate depending on how quickly the frame rate changes at small time scales. Finally, we have captured several MPEG-1 and MPEG-2 video streams as well as two network bandwidth traces for simulation, all of which will be made available via the Web.

Table 4. The effective frame rate for *Extra Terrestrial* frame rates shown in Figure 7

Configuration	Naive	Priority-Based	Proposed approach
W = 0,P = 1	26.54	9.42	24.75
W = 1,P = 1	24.24	8.29	24.74
W = 3, P = 1	19.65	6.02	24.71
W = 1, P = 1.5	19.12	6.62	24.74
W = 1, P = 2	0.42	2.11	24.74

We are currently working towards incorporating semantic information into this streaming model. That is, all frames are created equal in this approach. One can imagine that if all the semantic information from the stream can be automatically extracted that some scenes within the video sequence might have higher priority than others.

References

1. Shanwei Cen, Jonathan Walpole, Calton Pu, "Flow and Congestion Control for Internet Media Streaming Applications", in *Proceedings of SPIE Multimedia Computing and Networking 1998*, pp 250-26 4.
2. Zhigang Chen, S.M. Tan, R. Campbell, Y. Li, "Real Time Video and Audio in the World Wide Web", in the *Fourth International World Wide Web Conference*, Boston, Massachusetts, December 1995.
3. E. Ekici, R. Rajaie, M. Handley, D. Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet," in *Proceedings of INFOCOM 99*.
4. W. Feng, M. Liu, B. Krishnaswami, and A. Prabhudev, "A Priority-Based Technique for the Best-effort Delivery of Stored Video", in *Proc. of the SPIE Multimedia Computing and Networking*, January 1999.
5. 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, pp. 234-242.
6. 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.
7. J. Mahdavi and S. Floyd. TCP-friendly unicast rate-based flow control. Note sent to end2end-interest mailing list, Jan 1997
8. Steve McCanne, V. Jacobson, "VIC: A Flexible Framework for Packet Video", Proceedings of ACM Multimedia 1995, November 1995.
9. P. Nee, K. Jeffay, and Gunner Danneels, "The Performance of Two-Dimensional Media Scaling for Internet Videoconferencing", in *Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, May 1997.
10. RealNetworks - http://www.real.com
11. I. Rhee, "Error control techniques for interactive low-bit rate video transmission over the Internet", in *Proc. SIGCOMM, 1998*
12. *L.A. Rowe, K. Patel, B.C. Smith, K. Liu, " MPEG Video in Software: Representation, Transmission and Playback", in Proc. of IS&T/SPIE 1994 Int'l Symp. on Elec. Imaging: Science and Technology*, San Jose, CA, February 1994.
13. S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams", in *Proceedings of INFOCOM 99*, April 1999.
14. M. Talley and K. Jeffay, "Two-Dimensional Scaling Techniques For Adaptive, Rate-Based Transmission Control of Live Audio and Video Streams", in *Proceedings of the Second ACM International Conference on Multimedia*, San Francisco, CA, October 1994.
15. Brett J. Vickers, Célio Albuquerque and Tatsuya Suda, "Source-adaptive multilayered multicast algorithms for real-time video distribution", *IEEE/ACM Transactions on Networking*, December 2000
16. Jonathan Walpole, Rainer Koster, Shanwei Cen, et. al, "A Player for Adaptive MPEG Video Streaming Over The Internet", in *Proc. 26th Applied Imagery Pattern Recognition Workshop,* Washington DC, October 15-17, 1997.
17. Zhi-Li Zhang, Yuewei Wang, David H. C. Du and Dongli Shu, "Video staging: a proxy-server-based approach to end-to-end video delivery over wide-area networks", IEEE/ACM Transactions on Networking, Aug. 2000