Online Smoothing of Variable-Bit-Rate Streaming Video

Subhabrata Sen*[1], Jennifer Rexford[2], Jayanta Dey[3], James F. Kurose[1], and Don Towsley[1]

[1] Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
(413) 545-4753
{sen,kurose,towsley}@cs.umass.edu

[2] Network Mathematics
AT&T Labs – Research
Florham Park, NJ 07932
(973) 360-8728
jrex@research.att.com

[3] Interactive Multimedia Services
GTE Laboratories
Waltham, MA 02254
(781)-466-2066
dey@gte.com

Abstract

Bandwidth smoothing techniques for stored video perform end-end workahead transmission of frames from the video source into the destination client video playback buffer in advance of their display times, and are very effective in reducing the bursty transmission bandwidth requirements of compressed, stored video. This paper addresses online bandwidth smoothing for a growing number of streaming video applications such as newscasts, sportscasts and distance learning, where many clients may be willing to tolerate a playback delay of a few seconds in exchange for a smaller bandwidth requirement. The smoothing can be performed at either the end-point source of the videocast or at special *smoothing server(s)* (e.g., proxies or gateways) within the network. A key difference with smoothing stored video at the source is that, in the online situation, the smoothing server has limited knowledge of frame sizes and access to only a segment of the video at a time. This is either because the feed is live or because it is streaming past the server. We derive an online smoothing model which incorporates playback delay, client and server buffer sizes, server processing capacity, and frame-size prediction techniques. Our model can accomodate an arbitrary arrival process. Using techniques for smoothing stored video at the source as a starting point, we develop an online, window-based smoothing algorithm for these delay-tolerant applications. Extensive experiments with MPEG-1 and M-JPEG video traces demonstrate that online smoothing significantly reduces the peak rate, coefficient of variation, and effective bandwidth of variable-bit-rate video streams. These reductions can be achieved with modest playback delays of a few seconds and moderate client buffer sizes, and closely approximate the performance of optimal offline smoothing of stored video [1]. In addition, we show that frame-size prediction can offer further reduction in resource requirements, though prediction becomes relatively less important for longer playback delays. However, the ability to predict future frame sizes affects the appropriate division of buffer space between the server and client sites. Our experiments show that the optimal buffer allocation shifts to placing more memory at the server as the server has progressively less information about future frame sizes.

EDICS: 5-SEND, 5-BEEP

1 Introduction

Many multimedia applications, such as videocasting and video-based entertainment services [2–4], rely on the efficient transmission of live or stored video. However, even when compressed, high-quality video can consume a significant amount of network bandwidth, ranging from 1–10 megabits/second. In addition, compressed video often exhibits significant burstiness on a variety of time scales, due to the frame structure of the encoding scheme and natural variations within and between scenes [5–10]. The transmission can become even more bursty when one or more video sources are combined with text, audio, and images as part of an orchestrated multimedia stream. Variable-bit-rate (VBR) traffic complicates the design of efficient real-time storage, retrieval, transport, and provisioning mechanisms capable of achieving high resource utilization.

One possible solution to this problem is for the encoder to reduce burstiness by adjusting the quantization level of the frames across time, particularly during scenes with high bandwidth requirements. However, a constant-bit-rate encoding reduces the picture quality, particularly at scene changes or intervals with significant detail or motion, precisely when the effects are likely to be most noticeable to users. For the same average bandwidth, a variable-bit-rate encoding offers higher quality and more opportunities for statistical multiplexing gain than would be possible for a constant-bit-rate encoding [11, 12]. However, transmission of variable-bit-rate video requires effective techniques for transporting bursty traffic across the underlying communication network.

In this paper, we present and evaluate techniques for *online smoothing* to reduce the burstiness of VBR streaming video, without compromising on the quality of the encoding. This smoothing can be applied at the source or at another node (e.g., proxy or gateway) along the path to the client(s). In addition to reducing resource requirements on high-speed backbone links, bandwidth smoothing is particularly valuable for lower-speed broadband access links that have significantly lower opportunity for statistical multiplexing. Using knowledge of frame sizes, the server can schedule the transmission of frames into the client playback buffer in advance of each burst. This approach to smoothing is particularly appropriate for a growing number of videocast applications, such as newscasts, sportscasts, and distance learning, where many (or all) of the clients may be willing to tolerate a playback delay of several seconds or minutes in exchange for a lower bandwidth transmission.

Previous work on bandwidth smoothing has focused on the two extremes of *interactive* and *stored* video. In interactive video applications, such as video teleconferencing, the sender typically has limited knowledge of frame sizes and must impose strict bounds on the delay to the client(s). As a result, smoothing for interactive video typically requires dynamic techniques that can react quickly to changes in frame sizes and the available buffer and bandwidth resources. For example, the server can smooth the video on a small time scale by transmitting at an average rate for a small set of frames, based on the lengths of existing frames and estimates of future frame sizes [13, 14]. Such smoothing is especially effective at removing the short-term burstiness of streaming MPEG video, where a small group of pictures (GOP) may consist of widely different frame lengths. Since the tight latency constraints limit smoothing to a few frames, interactive video applications often require the encoder to lower the frame quantization level when the source rate exceeds the network capacity on a larger time scale. Techniques for smoothing interactive video cannot remove the medium timescale burstiness within and between scenes without degradation in picture quality.

While interactive applications have limited knowledge of future bandwidth requirements, smoothing algorithms for stored video [1, 15–18] can use prior knowledge of the frame sizes for the entire stream. In particular, since the video frames are stored in advance at the server, these bandwidth smoothing techniques can reduce rate variability on a large time scale through workahead transmission of frames into the client playback buffer, as discussed in Section 2. Given a fixed client buffer size, these smoothing algorithms can minimize the peak bandwidth of the video stream, while also optimizing other important performance metrics [19], such as rate variability, effective bandwidth, and the number of rate changes. As an example, experiments with MPEG-1 video traces show that a one-megabyte playback buffer can reduce the peak and standard deviation of the transmission rates by a factor of of 5–10 [1]. With the decreasing cost of high-speed memory, video set-top boxes and personal or network computers can easily devote a few megabytes of buffer space to video smoothing, in exchange for these substantial reductions in network resource requirements.

Bandwidth smoothing of stored video serves as the starting point for the development of an effective *online* window-based smoothing algorithm for non-interactive videocast applications. We propose delaying the playback of the video to permit the source to perform workahead transmission over a larger interval (window) of frames, based on the buffer space available at the client site(s). Alternatively, a special proxy server in the network could delay the transmission in order to smooth video destined for a subset of the receivers, such as clients within a single company or university, as shown in Figure 1. Similar approaches have been proposed for placing “retransmission servers” inside the network for efficient recovery from packet loss, and for transcoding servers [20, 21]. Throughout the paper, the term *smoothing server* is used to denote the node at which smoothing is performed. A stream may cross multiple network domains, and the client for the smoothed video, called the *smoothing client* could be either the receiver end-host, or an intermediate node in the network (e.g., an egress node in some network domain).

A key difference between online smoothing of streaming video and smoothing stored video at the source is that in the online case, the smoothing server does not have access to the entire video beforehand. The server only has limited knowledge of frame sizes and access to only a segment of the video at a time. This is because either the feed is live or it is streaming past the server. In Section 3, we present the framework for online smoothing of streaming video which incorporates practical constraints on the knowledge of future frame sizes, the ability to perform workahead transmission, and the playback delay, as well as limitations on client and server buffer sizes and processing capacity. The model can accomodate an arbitrary arrival process, which may be the original video stream or its transformation after travelling through the network from the source to the smoothing server.

In addition to busrtiness within an MPEG group-of-pictures, video streams can exhibit substantial burstiness at the time scale of scene changes [10], since scenes often differ in their bit rates. In Section 4, we demonstrate that 1–10 seconds of playback delay can reduce resource requirements by a factor of two or more beyond the gains of techniques for smoothing interactive video. In addition, the smoothing benefits with such delays are also very close to that of optimal offline smoothing, for finite as well as infinite client buffers. This suggests that most of the benefits of offline smoothing can be gained in the online scenario, with smoothing windows (and therefore playback delays) which are of the order of scene lengths, and that larger windows have limited utility. Experiments also show that knowledge of future frame sizes improves the effectiveness of online smoothing, though prediction becomes less important for longer playback delays.

In addition, even for small playback delays, knowledge of future frame sizes is of limited utility, as the amount of workahead smoothing is limited by the amount of data actually available to the server. However, the ability to predict future frame sizes still affects the appropriate division of buffer space between the server and client sites. Our experiments show that the optimal buffer allocation (i.e., the allocation to the server and client buffers that produces maximal smoothing gains) shifts to placing more memory at the server as the server has progressively less information about future frame sizes. We also investigate the processing requirements for deploying our smoothing scheme at a video source or at a proxy server inside the network. We show that, instead of computing the transmission schedule in every frame time slot, the server can compute a new schedule only twice in each smoothing window with very little degradation in performance. This permits a workstation or personal computer to compute transmission schedules simultaneously for multiple video streams. We conclude the paper in Section 5 with a discussion of future research directions.

2 Smoothing Stored Video

A multimedia server can substantially reduce the bandwidth requirements of stored video by transmitting frames into the client playback buffer in advance of each burst. The server controls workahead smoothing by computing upper and lower constraints on the amount of data to send, based on *a priori* knowledge of frame sizes and the size of the client playback buffer. Then, the bandwidth smoothing algorithm constructs a transmission schedule that minimizes burstiness subject to these constraints.

2.1 Smoothing Constraints

Consider an N-frame video stream, where frame i is ℓ_i bits long, $i = 1, 2, \ldots, N$. The entire stream is stored at the server, and is transmitted across the network to a client which has a B_C-bit playback buffer. Without loss of generality, we assume a discrete time-model where one time unit corresponds to the time between successive frames; for a 30 frames/second full motion video, a frame time corresponds to $1/30^{th}$ of a second. To permit continuous playback at the client site, the server must always transmit enough data to avoid buffer *underflow* at the client, where

$$L(t) = \sum_{i=1}^{t} \ell_i$$

specifies the amount of data consumed at the client by time t, $t = 1, \ldots, N$ ($L(0) = 0$). Similarly, to prevent *overflow* of the B_C -bit playback buffer, the client should not receive more than

$$U(t) = \min(L(t-1) + B_C, L(N))$$

bits by time t. Any valid server transmission plan which results in lossless, starvation-free playback, should stay within these two constraints, as shown in Figure 2(a). Hence, during time unit i, the server transmits at rate s_i, where $L(t) \leq \sum_{i=1}^{t} s_i \leq U(t)$.

In order to smooth out transmissions at the beginning of the video, a playback delay of w time units can be introduced. The client underflow (consumption) and overflow curves are shifted to the right by w time units, and are given by

$$L^w(t) = \begin{cases} 0, & 0 \leq t \leq w-1, \\ L(t-w), & w \leq t \leq N+w, \end{cases}$$

and

$$U^w(t) = \min(L^w(t-1) + B, L(N)).$$

2.2 Computing Transmission Schedules

Creating a *feasible* transmission schedule involves generating a monotonically non-decreasing path that does not cross either constraint curve. Figure 2(b) shows a smoothed transmission schedule for an MPEG encoding of *Star Wars* [7] for two different client buffer sizes (256 and 512 kilobytes, respectively). The video exhibits significant burstiness, even at the one-second timescale. Performing workahead transmission into the client playback buffer significantly reduces the peak and variability of the transmission rates s_i, particularly for larger buffer sizes. In general, the constraints L and U typically result in multiple feasible transmission schedules S with different performance properties [19]. In this context, we focus on an $O(N)$ algorithm that computes the shortest-path schedule S, subject to the constraints L and U [1]. Figure 2(a) shows an example schedule, where each change in the transmission rate occurs at the leftmost point along the longest trajectory from the previous change point.

The algorithm generates a schedule that minimizes the peak and variability of the transmission rates s_i, as well as the effective bandwidth requirement [1]. This algorithm serves as our starting point for developing effective online smoothing techniques. In the rest of the paper, we refer to this as the *optimal offline algorithm*.

In the next section, we derive an online smoothing model which incorporates an arbitrary arrival process, playback delay, client and server buffer sizes, server processing capacity, and frame-size prediction techniques. We generalize the shortest path algorithm to operate on a window of w video frames by modifying the smoothing constraints L and U. The new online smoothing algorithm periodically recomputes the transmission schedule as more frames arrive, based on a sliding window of frames. Smoothing with a small window can remove the short-term burstiness in video stream, while larger window sizes can further reduce the bandwidth requirements by smoothing across scenes with different bit rates, at the expense of longer playback delays.

3 Window-Based Online Smoothing

Existing techniques for smoothing stored video at the source serve as an useful foundation for developing new algorithms for online smoothing of streaming video under client buffer constraints. However, operating with limited available data, and limited knowledge of future frame sizes requires important changes to the model in Figure 2(a). We begin by formulating the online smoothing model. Our approach involves computing the online smoothing constraints corresponding to short segments of the video stream as frames become available at the smoothing server, and using the existing shortest-path algorithm in this context to compute (and recompute) transmission schedules for finite durations into the future. This online smoothing model accounts for the constraints placed by the server and client buffers, as well as the playback delay, knowledge of future frame sizes, and the overhead for computing a transmission schedule.

3.1 Online Smoothing Model

Without loss of generality, we consider a discrete time model at the granularity of a frame time. The smoothing server has a B_S-bit buffer and smooths the incoming video into a B_C-bit client buffer. The server introduces a w-frame playback delay (beyond the transmission delay) between the two sites. The smoothing client plays out the stream according to the original unsmoothed schedule. As shown in Figure 3, the video arrives according to a *cumulative arrival vector* $\mathbf{A} = (A_0, A_1, \ldots, A_{N+w})$, where A_i is the amount of data which has arrived at the smoothing server by time $i, 0 \leq i \leq N + w$. The corresponding *arrival vector* is $\mathbf{a} = (a_0, a_1, \ldots, a_N, \ldots, a_{N+w})$, where $a_i = A_i - A_{i-1}$ is the amount of data which arrives in time $(i-1, i)$. Although the server does not receive more than A_i bits by time i, we assume that it has knowledge of future frame sizes up to a certain time in the future. At any time τ, the smoothing server has knowledge of the next P consecutive elements in the arrival vector, $(a_{\tau+1}, \ldots, a_{\tau+P})$, where $P \geq 0$. P is refered to as the *lookahead interval* in the rest of the paper.

The server receives the entire video by time N, with $A_j = A_N$ for $j = 1, \ldots, N + w$. Transmission can continue until time $N + w$, since playback at the client is delayed by w frame times to enable smoothing. The *cumulative playback vector* is **D(w)** $= (D_0, D_1, \ldots, D_{N+w})$, where D_i is the amount of data which must be transmitted from the smoothing server by time $i, 0 \leq i \leq N+w$. For example, if the smoothing client is the actual end user, the playback vector should satisfy $D_j = 0$ for $0 \leq j < w$ and $D_j = \sum_{i=0}^{j-w} \ell_i$ for $w \leq i \leq N+w$. We assume that $A_i \geq D_{i+w}, 0 \leq i \leq N+w$, and $A_N = D_{N+w}$. The combined server and client buffer space must be enough to accommodate the difference in the cumulative arrival and playback amounts at any instant; that is, $B_S + B_C \geq \max(A_i - D_{i-1})$ for $i = 0, 1, \ldots, N+w$.

Based on the buffer and delay constraints, the smoothing server computes a *cumulative transmission vector* **S(w)** $= (S_0, S_1, \ldots, S_{N+w})$, where S_i is the amount of data which must be transmitted from the smoothing server by time i, $0 \leq i \leq N+w$. The corresponding *transmission vector* is **s(w)** $= (s_0, s_1, \ldots, s_N, \ldots, s_{N+w})$, where $s_i = S_i - S_{i-1}$. Given this framework, the online smoothing algorithm computes a transmission vector **s(w)** subject to the arrival and playback vectors, as well as the constraints on the buffer sizes and playback delay. The smoothing server generates the transmission schedule in an online fashion.

To generate a transmission schedule, the server could conceivably compute a new schedule at every time unit to incorporate the most recently available frame size information. To reduce computational complexity, the server could instead execute the smoothing algorithm once every α time units ($1 \leq \alpha \leq w$). The parameter α is referred to as the *slide length* throughout this paper, and is an integer number of frame slots. Intuitively, after any invocation to the smoothing algorithm, the server slides over α time units before it invokes the the online smoothing algorithm with a smoothing window starting α time units past the beginning of the smoothing window for the first invocation. Table 1 summarizes the key parameters, which guide our derivation of the smoothing constraints, as well as our performance evaluation in Section 4.

3.2 Online Smoothing Constraints

The parameters in the smoothing model translate into a collection of constraints on the smoothing schedule **S**. At any given time instant $\tau (\tau \leq N+w)$, the server can compute the amount of *workahead* $C^w(\tau)$ that has been sent to the client:

$$C^w(\tau) = S_\tau - D_\tau = C^w(\tau - 1) + s_\tau - d_\tau.$$

The server can also compute the constraints on smoothing from time τ to a time $t > \tau$ in the future, as shown in Figure 4. To avoid underflow and overflow at the client buffer, the server transmission schedule must also obey the lower and upper constraints:

$$L_1^w(\tau, t) = \sum_{i=\tau+1}^{t} d_i = D_t - D_\tau,$$

and

$$U_1^w(\tau, t) = L_1^w(\tau, t-1) + B_C.$$

Similarly, to avoid underflow and overflow at the server buffer, the schedule must obey the upper and lower constraints:

$$U_2^w(\tau,t) = \sum_{i=0}^{\min(\tau+P,t)} a_i - D_\tau = A_{\min(\tau+P,t)} - D_\tau,$$

and

$$L_2^w(\tau,t) = U_2^w(\tau,t+1) - B_S.$$

In determining $U_2^w(\tau,t)$, the parameter $\min(\tau+P,t)$ limits the server to transmitting data that was known by time τ and has arrived by time t. The model can be extended to handle a bounded amount of delay jitter by making the above constraints more conservative, based on the maximium and minimum amount of data that may arrive by any time [1, 22].

Using these constraints, the server can compute a schedule from time τ to time $t_\tau = \min(\tau+w+P, N+w)$. In particular, a feasible schedule $S^w(\tau,t)$ must satisfy

$$\max\left(L_1^w(\tau,t), L_2^w(\tau,t)\right) \leq S^w(\tau,t) \leq \min\left(U_1^w(\tau,t), U_2^w(\tau,t)\right) \quad \text{for } t = \tau+1, \tau+2, \ldots, t_\tau.$$

Given this set of constraints and the workahead $C^w(\tau)$, the smoothing server employs the majorization algorithm described in Section 2 to compute a schedule $\{s_j : j \in (\tau+1,\ldots,t_\tau)\}$ for the next $w+P$ time intervals. As shown in Figure 4, the upper and lower smoothing constraints meet at time $\tau+w+P$, due to limited knowledge and availability of arriving frames. Rather than waiting until time $\tau+w+P$ to generate the next portion of the transmission schedule, the server computes a new schedule every α time units, where $1 \leq \alpha \leq w+P$, at the expense of an increase in computational complexity. Smaller values of α improve the effectiveness of the online smoothing algorithm by refining the conservative estimate $U_2^w(\tau,t)$ of the data available for transmission. Our *sliding-window* smoothing executes on overlapping windows of frames (Figure 5(c)). This reduces the peak rate and variability of the resultant transmission schedule over a *hopping-window* (Figure 5(b)) approach (where $\alpha = w+P$). In Section 4.5, we evaluate the impact of varying this slide length α.

3.3 Cost/Performance Trade-Offs

The overhead associated with computing the shortest path schedule at each invocation of the online smoothing algorithm is $O(w+P)$. The running time of the offline smoothing algorithm is linear in the number of frames. On a 300 MHz Pentium PC, the online smoothing algorithm consumes about $1-2$ milliseconds to smooth 30 seconds of 30 frames/second video, suggesting that a commercial off-the-shelf workstation configuration could easily offer smoothing services for tens of video streams.

Given a sliding window of α, the total computation overhead for the sliding-window algorithm for smoothing a N-frame video is given by $O((w+P)N/\alpha)$. Smaller values of α allow the server to gradually refine the schedule based on new frames, at the expense of additional computational load at the server. Fortunately, though, the server does not always have to determine the schedule for the entire $(w+P)$-frame interval, since the schedule is recomputed every α time units anyway. In fact, the server stops computing after determining the schedule for the next α time units in the future.

The values of w, α, and P determine how closely our online smoothing algorithm can approach the performance of the optimal offline algorithm. The client buffer size B_C limits the amount of workahead data $C^w(\tau)$, while the server buffer size B_S limits the server's ability to store and smooth a sequence of large frames. The sizes of the two buffers interact with the window size w. In general, a large value of w improves performance, since it permits the server to smooth over a larger interval of frames. However, if w grows too large, relative to $B_C + B_S$, then the buffer space can limit and adversely impact smoothing gains.

Larger values of P improve performance by allowing the server to predict future bursts in frame sizes, particularly when the window size w is small. Varying P allows us to determine whether or not frame-size prediction is necessary for online smoothing of non-interactive video. Finally, varying α allows us to determine how often the server needs to compute transmission schedules. The next section shows that relatively large computations periods, such as $\alpha = w/2$, are sufficient in practical examples. Similarly, modest values of w, B_C, B_S, and P allow online smoothing to achieve most of the performance gains of the optimal offline algorithm.

4 Performance Evaluation

The performance evaluation in this section studies the interaction between the parameters in Table 1 to help in determining how to provision server, client, and network resources for online smoothing. The study focuses on bandwidth requirements, in terms of the peak rate, coefficient of variation (standard deviation normalized by the mean rate), and the effective bandwidth of the smoothed video stream. The effective bandwidth [23, 24] statistically estimates the network throughput required to transmit the video through a intermediate switch having a b-byte buffer, with a tolerable loss rate of γ. For a video with transmission rates rates $s_1, s_2, \ldots, s_N$, the effective bandwidth is computed as $\log(\sum_{i=1}^{N} e^{\theta s_i}/N)/\theta$ where $\theta = -log\gamma/b$. For upper and lower bounds on the performance metrics, we consider results for the unsmoothed video stream and the optimal offline schedule. The results reported here are based on a 23-minute MPEG trace of *Wizard of Oz* with a mean rate of 1.25 megabits/sec, a 97-minute MPEG trace of *Star Wars* with a mean rate of 0.5 megabits/sec, and a 20-minute M-JPEG encoding of *Beauty and the Beast* with a mean rate of 3.4 megabits/sec. Experiments with other compressed video sources show similar behavior.

4.1 MPEG Coding Order

MPEG encoding uses both *intra*-frame and *inter*-frame compression, and the compressed video stream consists of a series of groups of pictures (GOPs). Each GOP consists of three types of frames. The I frames are compressed using intra-frame coding and can be encoded (decoded) by themselves. The P frames are coded using motion compensation from the preceding I (or P) frame. A B frame is also inter-frame coded, using the preceding and succeeding I (or P) frame. The resulting inter-frame dependencies impose constraints on the compression/decompression order of the different frames, and the *coding order* (the order in which the frames are decoded by the decoder) is different from the *playback order* or natural temporal order of the frames [25, 26]. Figures 6(a) and (b) illustrate these different orders. Note that to guarantee starvation-free playback at the client, both the preceding and succeeding I (or, P) frame must arrive before the display deadline of a B frame. The client consumption curve should reflect this constraint. The cumulative client consumption curve corresponding to the example in Figure 6(b) is shown in Figure 6(c).

For our experiments, we assume that the video streaming into the smoothing server is played back at the client with a w-frame time lag. The arrival vector $\mathbf{A}$ therefore is the consumption vector shifted w time units to the left. Another valid service model would assume that the arrival vector is some smoothed transmission schedule, but we do not consider that here.

4.2 Basic Performance Trends

Considering each of the smoothing parameters in isolation, we first explore the case where the smoothing server has no knowledge about future frame sizes. Figure 7 demonstrates the benefits of smoothing an MPEG-1 encoding of the movie *Star Wars* [7], which has a peak rate of 5.6 Mbps and a mean rate of 0.5 Mbps. As shown in Figure 7(a), smoothing over a small time scale of 4 frames reduces the peak rate to 2.9 Mbps. However, the video exhibits significant burstiness even after using a smoothing window of 4-frames, which can be removed using larger smoothing windows. For example, the sliding-window algorithm with a 1-second (30-frame) window, reduces the peak bandwidth by an additional 100% to 1.4 Mbps. Figure 7(b) shows the corresponding schedule which is much less bursty. Figure 7(c) shows the transmission schedule for a 30-second (900-frame) window, which nearly converges with the offline schedule for the same startup delay and client buffer size. The corresponding peak rate is reduced by another factor of 2 to 0.75 Mbps. Hence, non-interactive live applications which can tolerate latencies of a few tens of seconds can realize almost all of the potential benefits of smoothing. Note that the window size here is of the same order as scene length distributions in videos [10]. The above trends clearly demonstrate the utility of using a smoothing window size which is sufficiently large to enable smoothing across different scenes in the video.

We next consider the other extreme situation in online smoothing when the smoothing server has prior knowledge of all frame sizes in the entire video (i.e., $P = \infty$). This can occur, for example, when prerecorded video is streaming into the server from the storage end-host at the edge of the network. Intuitively, this represents the most favorable scenario for online smoothing gains. Figures 8(a) and (b) respectively plot the peak rate and effective bandwidth of the online smoothed schedule for this situation, as a function of the window size w, for different client buffer sizes. Note that the effective bandwidth decays as a function of the buffer size. However, a large switch buffer reduces the effective bandwidth at the cost of introducing longer delays. For continuous media transmission, a reasonable operating region would limit this delay to a few tens of packets or ATM cells. This corresponds to switch buffer sizes of $1 - 3$ kilobytes. For the evaluations presented in this paper, we use a 3-Kbyte switch buffer.

Having complete knowledge of future arrivals enables the server to smooth future bursts by aggressive workahead transmission. However, since the server does not have access to the actual future video frames, workahead at any instant is limited by the amount of data present at the server. The graphs in Figure 8 illustrate the dramatic benefits of online smoothing even over small window sizes, and small client buffers. For example, a client buffer of only 46 Kbytes (the size of the largest frame in the video) and a small 4-frame smoothing window reduces the peak rate and effective bandwidth by 71.67% and 76.47%, respectively, over the unsmoothed video. Increasing the smoothing window increases the startup delay at the client. This has two important implications for smoothing. A larger window gives the server more time to transmit any frame. A larger window also allows the server to perform workahead transmission more aggressively, as it has access to a larger number of frames. Because of this, for a given client buffer size, as the length of the smoothing window increases, the peak rate and effective bandwidth decrease at first. However, increasing w beyond a certain point (e.g., beyond 2 sec for $B_C = 512$ KB) does not help as the client buffer becomes the limiting constraint on smoothing. The curves flatten out in this region.

For a small window, the main constraint on smoothing is the window size and so the performance is similar across different client buffers. As the window size increases, the benefits of a larger client buffer become apparent. For example, for a 10-second (300-frame) window, a 2-megabyte client buffer results in a peak rate which is just 1.8% larger than the peak rate under an infinite client buffer, while the corresponding figure for a 46-Kbyte buffer is 69.5% larger. For a given w, the performance improves initially as the client buffer size increases, until the client buffer becomes sufficiently large that it does not constrain the achievable smoothing. As the buffer size continues to increase further, the window size becomes the main constraint on smoothing. For a 15-frame window, increasing the client buffer beyond 186 Kbytes (the size of the largest consecutive 15 frames in the video) has no effect.

We next explore the performance impact of the availability of the actual video frames at the server. This is graphed by the bottom two curves in Figures 8(a) and (b), where the client buffer size does not impose any limitation on workahead transmission. For comparison purposes, the corresponding offline schedule is computed assuming a startup delay of w frames. The main difference between the two situations (online and offline) is that, in the offline case, the server has access to all the video frames beforehand. In the online case, the server has access to only the next w frames at any time. The graphs indicate that, for small window sizes, the limited availability of data at the server considerably impacts online smoothing. The difference diminishes with increasing w. The effective bandwidth of the online smoothed schedule is more than 3.7 times larger than the effective bandwidth of the optimal offline schedule, when the smoothing window is just two frames. For a larger five-second smoothing window, the effective bandwidth is within 34.1% of the corresponding optimal offline value.

4.3 Knowledge of Future Frame Sizes (P)

In this subsection, we investigate the performance impact of knowledge of P future frame sizes beyond the current smoothing window. Figures 9(a) and (b) plot the peak rate and effective bandwidth for $P = 0$ (i.e., no lookahead) and $P = \infty$ (full lookahead), as a function of the smoothing window w. For comparison, we also plot the statistics for the unsmoothed and offline smoothed schedules. For this experiment, the client buffer is large enough to accommodate the entire video; that is, the upper constraint curve in Section 3 depends only on U_2. The graphs show that lookahead offers, at best, only moderate reductions in the peak rate and effective bandwidth. In comparison, for small to moderate window sizes, there is a significant difference between the online schedule with $P = \infty$ and the offline schedule, indicating that the limited data availability at the server is the primary factor limiting smoothing performance in this region.

Figure 10(a) plots the effective bandwidth as the lookahead interval P increases, for different smoothing window sizes. Initially, to isolate the effect of lookahead, we assume that the client buffer size $B_C = \infty$, that is, it is sufficiently large that it does not constrain the smoothing. The x-axis represents the lookahead P as a fraction of the smoothing window size w. The behavior of the peak rate curves is similar and is therefore not graphed here. The curves indicate that lookaheads which are very small compared to the window size hardly impact the effective bandwidth. Larger lookaheads which are comparable to the window sizes are the most effective in decreasing the effective bandwidth. For most of the plots, the incremental benefits of lookaheads beyond one or two times the window size (i.e., $P/w \geq 1 \quad \text{or} \quad 2$) are minimal. For a given smoothing window size w, a larger lookahead interval P provides information further into the future and, therefore, improves awareness of future bursts in frame sizes. This information is used by the online smoothing algorithm to smooth out transmissions by working ahead, i.e., scheduling transmissions ahead of time. However, the actual amount of workahead transmission that can be performed at any time i is limited by the amount of data that has already arrived at the server by that time (the client buffer is assumed to be large enough that it does not constrain the workahead).

For very small window sizes, the data available at the server is the main constraint on workahead transmission. Hence, increasing the lookahead does not help since the server does not have enough data to take advantage of this knowledge. This explains the behavior of the curves for a two-frame window. For a larger window size, the server has access to a larger amount of data and, as such, is better able to take advantage of lookahead to workahead more aggressively. If the window w is large, and if the lookahead P is very small compared to w, then the effect of this added information is marginal, and most of the smoothing gains come from having the large window. Thus, for the 9000-frame (five-minute) window, the effective bandwidth decreases very slowly for small lookahead values, and to achieve any notable improvement, large lookaheads are required. As the lookahead increases, it becomes increasingly effective, until a point is reached where w, and hence the limited data available for transmission becomes the main constraint. Increasing the lookahead beyond this point has little effect on smoothing.

We now consider the case where the client buffer size is limited. In Figure 10(b), we plot the effective bandwidth as a function of P/w for a 512-Kbyte client buffer. We find that, for small windows, the client buffer is still large enough that it does not constrain the allowable workahead and so the behavior of the curves is identical to that in the infinite buffer case. For larger windows, however, the client buffer limits the amount of workahead; the server has enough data but is constrained by the relatively small client buffer. In addition, for such buffer sizes, large smoothing windows allow the online smoothing algorithm to achieve peak rates and rate variabilities which are close to those of the optimal offline schedules. For example, for a 512-Kbyte client buffer and a 5-second window, the effective bandwidth of the smoothed schedule with no lookahead is just 13% larger than the effective bandwidth in the offline schedule; the performance of online and offline smoothing converge for a window of 30 seconds. As such, lookahead is not expected to reap significant benefits for large window sizes, as there is very little room for improvement.

So far, our evaluations assumed that the lookahead information consisted of actual future frame sizes. The corresponding performance serves as an upper bound on the performance of any prediction scheme which may be used to estimate future frame sizes. We have also explored how a practical prediction scheme fares when used in conjunction with our online smoothing algorithm. Our results [27] indicate that for short lookahead intervals of up to a couple of GOPs, a simple scheme such as the GOP-based prediction for MPEG video (the estimated sizes of the next x frames is identical to the sizes of the last x frames) in [13] performs remarkably well, resulting in peak rate and rate variability measures which are very close to those for perfect prediction.

4.4 Client Buffer Size (B_C)

So far, our experiments have suggested that with a client buffer size of only a few megabytes, and smoothing window on the order of a few seconds, the online algorithm is extremely effective in removing the short and medium-term burstiness in the underlying video stream. Removing any remaining longer-term burstiness requires both a larger window size and a larger workahead buffer. However, increasing the size of the client workahead buffer offers diminishing returns, as illustrated in Figure 11. To highlight the interplay between the smoothing window size w and client buffer size B_C, these graphs plot the peak rate for online smoothing (with $\alpha = 1$, $P = 0$ and windows of 1, 5, 20, 60 seconds and 5 minutes) across a range of client buffer sizes, for *Wizard of Oz* and *Beauty and the Beast*. Plots of effective bandwidth (not shown) exhibit similar trends.

For very small values of B_C, the performance of the online algorithm for small window sizes is similar to that for much larger windows. This is because the client buffer size imposes the main constraint on workahead transmission in this region. These results suggest that small windows are sufficient to reap most of the benefits of smoothing for low-latency applications with relatively small client buffers. For a given window size, the bandwidth requirements for the video stream initially decrease as the client buffer size grows. In this region, the client buffer size is still the main constraint on workahead transmission. Therefore, a larger client playback buffer size allows the online smoothing algorithm to schedule transmissions into the buffer more effectively. Beyond some critical buffer size, the performance is largely independent of the buffer size, since the smoothing window becomes the main limitation on the ability to perform workahead transmission. The curves flatten out in this region. For example, for *Beauty and the Beast*, for a 20-second window, increasing the client buffer beyond 4 megabytes offers no discernible performance benefits.

A larger window allows the server to further reduce the longer-term burstiness in the underlying video stream. As such, the performance of the online algorithm for different window sizes diverges as the buffer size gets larger. In order to remove burstiness on a larger time scale, the window size w should grow along with the buffer size B, based on a rough estimate of the frame sizes in the video stream. As w grows relative to B, the algorithm can effectively utilize the client buffer to smooth the stream at a larger time scale. A larger smoothing window size, in the range of 1–5 minutes, would be appropriate for a client with a low-bandwidth connection to the network. Such a client can benefit from having a larger buffer. In addition, as shown in Figure 11(b), such large windows are especially appropriate for videos such as *Beauty and the Beast*, which exhibits substantial burstiness on the 20-second to 5-minute time scale. For example, the peak rate drops by more than a factor of two when w is increased from 20 seconds to 5 minutes.

4.5 Computation Period (α)

Recall that the benefits of online smoothing depend on how often the server recomputes the transmission schedule to incorporate new arriving frames. To investigate the sensitivity of online smoothing to the computation frequency, Figure 12(a) compares the performance of the online algorithm for different values of α across a range of window sizes w, for *Wizard of Oz*, with a 512-Kbyte client buffer and $P = 0$. As expected, for a given window size, the performance improves as the computation period decreases, with the best performance exhibited at $\alpha = 1$. Even a very large computation period of $\alpha = w$ (w is the length of the smoothing window) can result in considerable smoothing gains. For example, for a 10-second smoothing window, with $\alpha = 10$ seconds, the peak rate and effective bandwidth are reduced to 58% and 52% of the corresponding unsmoothed values.

However, there is a significant difference in performance between $\alpha = w$ and $\alpha = 1$. For the same 10-second window, the peak rate for $\alpha = w$ is still 184% larger than the corresponding value for $\alpha = 1$.

When $\alpha = w$, the server effectively performs *hopping-window* smoothing over consecutive non-overlapping windows. As a result, the server only smooths within a window, but cannot reduce burstiness across different windows. This can lead to situations such as that depicted in Figure 5. If a window boundary occurs at position XY, then, under the hopping-window algorithm, the first few large frames in the window starting at XY would inflate the peak rate of the smoothed schedule. This accounts for the relatively poor performance for $\alpha = w$. Any smaller computation period can smooth out bursts at the beginning of a smoothing window by workahead transmitting part of the burst at an earlier time in an earlier window. Although smaller computation periods decrease the burstiness of the video stream, the performance differences between $\alpha = 1, w/8, w/4, w/2$ are not significant. For the 10-second window, the effective bandwidths for $\alpha = w/2$ and $\alpha = 1$ are just 9% and 5% larger than that of the offline smoothed schedule.

Figure 12(b) plots the effective bandwidth measure for the M-JPEG *Beauty and the Beast* trace for different values of α across a range of window sizes w, assuming a 5-Mbyte client buffer. Compared to MPEG video, M-JPEG videos do not exhibit much small time scale rate variability, since each frame is encoded independently. Therefore, for small window sizes, there is very little potential for smoothing and this is reflected in the negligible difference in the effective bandwidth values across different values of α, and their close proximity to the unsmoothed performance for window sizes below a second. However, for larger windows, there is a significant performance difference between the effective bandwidths of the schedule produced with $\alpha = w$ and those produced with smaller values of α. The effective bandwidth values for $\alpha = w/2, w/4, w/8, 1$ are identical to the offline smoothed effective bandwidth by $w = 1$ minute (1800 frames) and the maximum difference between the effective bandwidths is less than 5.3% of the corresponding offline smoothed value for $w \leq 1800$ frames.

Our experiments suggest that online smoothing offers significant smoothing gains even for relatively large values of α. Nearly all of the benefits of online smoothing can be achieved by choosing $\alpha = w/2$. This has major implications, since a larger computation period can substantially reduce the CPU overheads of online smoothing.

4.6 Server and Client Buffers (B_S and B_C)

Until now, our evaluations assumed that the smoothing server has enough buffer to accommodate a window of any w consecutive frames in the video. We now investigate how buffer placement at the server and client impacts online smoothing. The minimum combined buffer requirement at the server and client for a smoothing window of w frame times is the envelope of the largest $w + 1$ consecutive frames in the video (the w frame smoothing window and the frame being currently displayed at the client)

$$B(w) = \max(L(i) - L(i - w - 1)), \forall w + 1 \leq i \leq N.$$

In this context it is necessary to distinguish between two kinds of workahead that can occur in smoothing. *Workahead for smoothing* uses workahead to smooth out burstiness in the video stream. In addition, if the server does not have sufficient buffer to accommodate w frames, it may have to workahead aggressively into the client just to prevent server buffer overflow. This second *workahead to prevent overflow* is detrimental to online smoothing, as it involves increasing the burstiness of the transmission schedule.

4.6.1 No Lookahead ($P = 0$)

We first consider the situation when the smoothing server has no knowledge about future frame sizes (i.e., $P = 0$). Figure 13 plots the peak rate and coefficient of variation of the smoothed schedule with a 30-frame smoothing window, as a function of the buffer allocation at the smoothing server. The largest window of 31 frames consists of 404 Kbytes. Each curve represents statistics for the smoothed schedules for a total buffer budget of $M = 404 * F$ Kbytes, where $F = 1, 1.25, 2, 3$ from top to bottom. The most striking feature in the graphs is the extreme asymmetry with respect to buffer placement. From left to right, the curves (for $F > 1$) have three distinct regions for all values of F. At the left is the region where the server buffer allocation is less than 370 Kbytes, the minimum buffer required to accommodate any 30 consecutive frames in the video. Here the smoothing is extremely sensitive to the buffer allocation, and the peak rate and rate variability decrease dramatically with small increases in the server buffer allocation (i.e., small decreases in the client buffer allocation).

Since the server has no knowledge about future frame sizes, it is constrained to compute a smoothed schedule based on the frames it already possesses, and may not be able to take advantage of the large available client buffer in this region. During regions of consecutive large frames, less aggressive workahead earlier on may result in a situation where the server is forced to transmit large amounts of data at a later time, to prevent buffer overflow, thereby increasing the burstiness of the transmitted schedule. In the worst case, this effect can result in a peak rate equal to that of the unsmoothed trace. Note that in this region, for a given server buffer, the performance metrics are identical across different client buffer allocations, illustrating that the server buffer is the main limiting factor on smoothing. This underlines the importance of allocating at least enough buffer at the server to accommodate a window of w frames. Once the server has enough buffer to accommodate 30 frames, the phenomenon of workahead to prevent overflow disappears, considerably improving the peak rate and rate variability. The performance curves then enter a region where the performance is insensitive to the buffer distribution. The width of this region increases from $F = 1.25$ to $F = 3$. In this region, the peak rate and coefficient of variation are identical across different values of F. For $F = 2$ and $F = 3$, this flat region includes allocations with enough buffer space for both the server and client to accommodate any consecutive 30 frames.

The client buffer becomes an impediment to smoothing only for small client buffer sizes. This occurs to the extreme right of the graphs in the figures. Note that the client has to have at least enough space to accomodate any one frame in the incoming stream, and so the minimum client buffer allocation in these plots is the size of the largest frame in the video. As we saw in Figure 8, even with this small buffer size, some smoothing gains are possible. The above trends indicate that, in situations where the smoothing server has no knowledge about future frame sizes, given a total buffer budget, sufficient buffering should be alloted to the server for accommodating w consecutive frames in the video, and the remaining allocated to the client. Furthermore, a total buffer allocation of $M = 2B(w)$ is sufficient to achieve all the benefits of online smoothing with a w frame smoothing window.

4.6.2 Impact of Lookahead ($P > 0$)

We next evaluate how knowledge of future frame sizes impacts the server-client buffer allocation tradeoffs. Figures 14(a) and (b) plot the peak rate and coefficient of variation for $F = 1.25$ and $w = 30$ frames for *Wizard of Oz*. From top to bottom the different curves correspond to $P = 0, 1, 2, 7$ frames, where P is the lookahead. The bottommost curve corresponds to infinite lookahead. The graphs show that for any buffer allocation at the server and client, the peak rate and coefficient of variation decrease as the lookahead increases. With more knowledge about future frame sizes, the server is better able to anticipate the space requirements of future frames. By factoring this information into the smoothing algorithm, much before these frames arrive, the server is better able to accommodate future bursts by performing workahead more aggressively (if required) earlier on. This helps avoid the need to transmit a large burst of data to prevent server buffer overflow. In the region where the server has less buffer than needed to accommodate 30 consecutive frames, the improvements are especially dramatic. For example, using a 7-frame lookahead, and equal buffer allocation at the server and client, the peak rate and effective bandwidth respectively reduce to 39% and 36% of the corresponding values for the no lookahead case.

Figures 15(a) and (b) highlight the case where the server has complete knowledge of all frame sizes. Both performance metrics show complete symmetry with respect to the client-server buffer allocation. The minimum buffering at the server or client is the maximum size of any frame in the video. As the client buffer allocation increases, it allows more workahead for smoothing, and the resultant schedule is initially much smoother. However, simultaneously the server buffer allocation is decreasing. A smaller server buffer limits smoothing by forcing early transmission to prevent buffer overflow. With decreasing server buffer allocation, this effect predominates, making the resultant schedule much burstier. The result is a U-shaped performance curve. It has been shown in [28] that for the infinite knowledge case, the peak rate curve is symmetric for $x \in [0, M]$ and has a minimum at $x = M/2$. We find similar behavior for both the coefficient of variation and the effective bandwidth (not shown). These results suggest that an even allocation of buffer space is best when the server has complete knowledge of the video, and that more buffer space should be allocated to the server when the knowledge about future arrivals is limited.

5 Conclusion

In this paper, we have shown that by delaying the transmission of video frames by w time units, our window-based online smoothing algorithm can substantially reduce the bandwidth requirements for distributing streaming video by "prefetching"(workahead transmission of) frames into the client playback buffer. Our online algorithm builds on previous work on techniques for smoothing stored video. Our algorithm incorporates constraints caused by the limited availability of "future" frames in an online setting, the need to recompute the transmission schedule as new frames arrive, and limitations on buffer sizes and processing capacity. Our experiments show that the algorithm substantially reduces the peak transmission rate and effective bandwidth, allowing low-bandwidth clients to receive the video stream with a modest playback delay. The smoothing algorithm achieves these performance gains with modest playback delays up to a few tens of seconds, and reasonable client buffer sizes in the range of hundreds of kilobytes to a few megabytes. Prediction of future frame sizes, coupled with a careful allocation of the server and client buffers, offers further performance improvements.

We have also shown that the algorithm has relatively small processing and memory requirements, making it possible to deploy smoothing servers inside the network. Based on these results, we are developing an architecture and prototype implementation of a smoothing server. We are also considering ways to combine online smoothing with other effective techniques for adjusting video transmission to the delay, bandwidth, and loss properties of the underlying communication network. For example, emerging network services could integrate window-based smoothing with layered encoding schemes and packet retransmission protocols, particularly in the context of multicast video and audio services. We are currently investigating a technique for caching the initial frames of popular video streams at an intermediate proxy server [22]. One of the benefits of such *prefix caching* is that it allows the smoothing server to decouple the smoothing window size from the client-perceived startup latency. Thus all the advantages of using a larger smoothing window can be achieved, without increasing client playback latency.

References

[1] J. D. Salehi, Z.-L. Zhang, J. F. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," *IEEE/ACM Trans. Networking*, vol. 6, pp. 397–410, August 1998.

[2] CNN Videoselect Website. `http://www.cnn.com/videoselect`.

[3] Timecast Website. `http://www.timecast.com`.

[4] TV Interactive Inc. Website. `http://www.tviweb.com`.

[5] E. P. Rathgeb, "Policing of realistic VBR video traffic in an ATM network," *International Journal on Digital and Analog Communication Systems*, vol. 6, pp. 213–226, October–December 1993.

[6] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, “On the self-similar nature of Ethernet traffic (extended version),” *IEEE/ACM Trans. Networking*, vol. 2, pp. 1–15, February 1994.

[7] M. Garrett and W. Willinger, “Analysis, modeling and generation of self-similar VBR video traffic,” in *Proc. ACM SIGCOMM*, September 1994.

[8] A. R. Reibman and A. W. Berger, “Traffic descriptors for VBR video teleconferencing over ATM networks,” *IEEE/ACM Trans. Networking*, vol. 3, pp. 329–339, June 1995.

[9] M. Grossglauser, S. Keshav, and D. Tse, “RCBR: A simple and efficient service for multiple time-scale traffic,” *IEEE/ACM Trans. Networking*, December 1997.

[10] M. Krunz and S. K. Tripathi, “On the characteristics of VBR MPEG streams,” in *Proc. ACM SIGMETRICS*, pp. 192–202, June 1997.

[11] I. Dalgic and F. A. Tobagi, “Performance evaluation of ATM networks carrying constant and variable bit-rate video traffic,” *IEEE J. Selected Areas in Communications*, vol. 15, August 1997.

[12] T. V. Lakshman, A. Ortega, and A. R. Reibman, “Variable bit-rate (VBR) video: Tradeoffs and potentials,” *Proceedings of the IEEE*, vol. 86, May 1998.

[13] S. S. Lam, S. Chow, and D. K. Yau, “An algorithm for lossless smoothing of MPEG video,” in *Proc. ACM SIGCOMM*, pp. 281–293, August/September 1994.
`ftp://ftp.cs.utexas.edu/pub/lam/smooth.ps.Z`.

[14] P. Pancha and M. E. Zarki, “Bandwidth-allocation schemes for variable-bit-rate MPEG sources in ATM networks,” *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 3, pp. 190–198, June 1993.

[15] W. Feng and S. Sechrest, “Smoothing and buffering for delivery of prerecorded compressed video,” *Computer Communications*, vol. 18, pp. 709–717, October 1995.
`http://www.cis.ohio-state.edu/□ wuchi/CompComm.ps.gz`

[16] W. Feng, F. Jahanian, and S. Sechrest, “An optimal bandwidth allocation strategy for the delivery of compressed prerecorded video,” *Springer-Verlag Multimedia Systems Journal*, vol. 5, pp. 297–309, September 1997.
`http://www.cis.ohio-state.edu/□ wuchi/mmsj.ps.gz`

[17] J. M. McManus and K. W. Ross, “Video on demand over ATM: Constant-rate transmission and transport,” in *Proc. IEEE INFOCOM*, pp. 1357–1362, March 1996. Extended version appears in *IEEE J. Selected Areas in Communications*, pp. 1087–1098, August 1996.

[18] J. M. del Rosario and G. C. Fox, “Constant bit rate network transmission of variable bit rate continuous media in video-on-demand servers,” *Multimedia Tools and Applications*, vol. 2, pp. 215–232, May 1996.

[19] W. Feng and J. Rexford, “A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video,” in *Proc. IEEE INFOCOM*, pp. 58–66, April 1997.
`http://www.research.att.com/□ jrex/papers/infocom97aps.Z`.

[20] N. F. Maxemchuk, K. Padmanabhan, and S. Lo, “A cooperative packet recovery protocol for multicast video,” in *Proc. International Conference on Network Protocols*, October 1997.
`http://www.research.att.com/□ nfm/ref.1463.ps`

[21] E. Amir, S. McCanne, and H. Zhang, “An application level video gateway,” in *Proc. ACM Multimedia*, November 1995.

[22] S. Sen, J. Rexford, and D. Towsley, “Proxy prefix caching for multimedia streams,” in *Proc. IEEE INFO-COM*, April 1999.

[23] J. Y. Hui, “Resource allocation for broadband networks,” *IEEE J. Selected Areas in Communications*, vol. 6, pp. 1598–1608, December 1988.

[24] S. Crosby, M. Huggard, I. Leslie, J. Lewis, B. McGurk, and R. Russell, “Predicting bandwidth requirements of ATM and Ethernet traffic,” in *Proc. IEE UK Teletraffic Symposium*, March 1996. Preprint DIAS-STP-9612.

[25] J. Mitchell, W. Pennebaker, C. Fogg, and D. LeGall, *MPEG Video Compression Standard.* 1996.

[26] M.-S. Chen and D. Kandlur, “Stream conversion to support interactive video layout,” *IEEE Multimedia*, vol. 3, pp. 51–58, Summer 1996.

[27] S. Sen, J. Rexford, J. Dey, J. Kurose, and D. Towsley, “Online smoothing of variable-bit-rate streaming video,” Tech. Rep. 98-75, Department of Computer Science, University of Massachusetts Amherst, 1998.

[28] J. Rexford and D. Towsley, “Smoothing variable-bit-rate video in an internetwork,” *IEEE/ACM Trans. Networking*, February 1999.
`http://www.research.att.com/□ jrex/tandem.ps.Z`

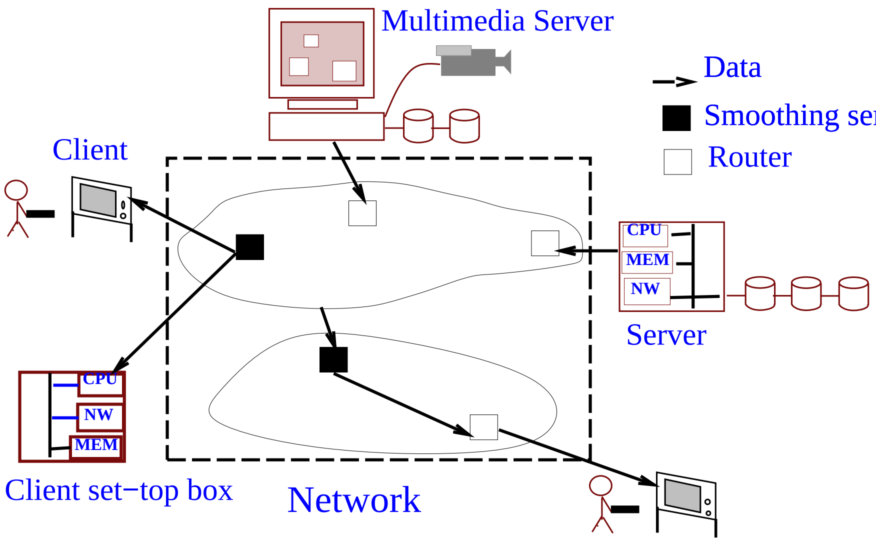
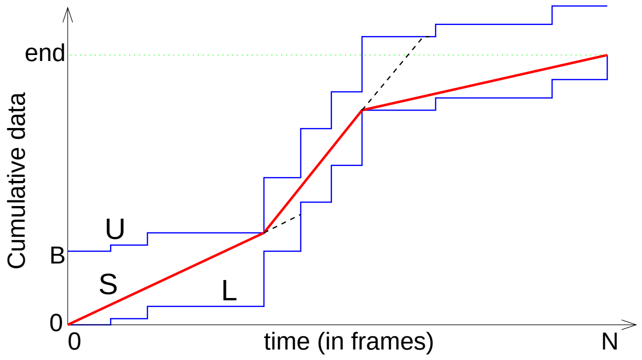
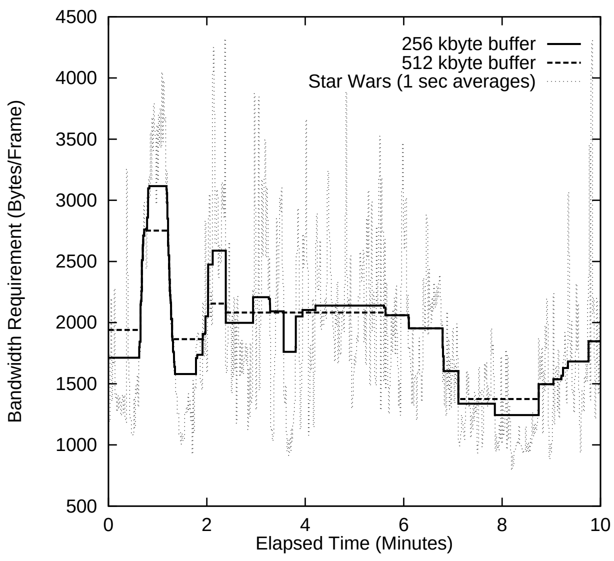


Figure 1: **Bandwidth smoothing in an internetwork:** The live or stored video stream originates from a multimedia server or a video-on-demand system and travels through the network to one or more clients, including workstations and set-top boxes. Smoothing occurs at the video source and/or at servers inside the network.



(a) Underflow/overflow curves



(b) *Star Wars* plans

Figure 2: **Server transmission plans:** The figure on the left shows an example of a transmission plan with three runs that stays between the upper and lower constraint curves. The figure on the right shows transmission plans for a 10-minute clip of an MPEG-1 encoding of *Star Wars* for two different client buffer sizes.

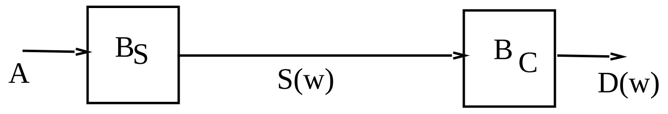


Figure 3: **Online smoothing model:** The smoothing server has a B_S-bit buffer and transmits the video to a smoothing client with a B_C-bit buffer. The video arrives according to an arrival vector A, is scheduled according to a vector S, and is played at the client according to a vector D, with a w-frame smoothing delay.

Parameter	Definition
w	Smoothing delay (in number of frame slots)
B_S	Smoothing server buffer size (in bits)
B_C	Smoothing client buffer size (in bits)
P	Knowledge of future arrivals (in number of frame slots)
α	Slide length for smoothing (in number of frame slots)
A	Arrival vector to smoothing server (bits per frame slot)
D	Playback vector at smoothing client (bits per frame slot)
S	Transmission vector from smoothing server (bits per frame slot)

Table 1: **Parameters in online smoothing model:** This table summarizes the key parameters in the smoothing model.

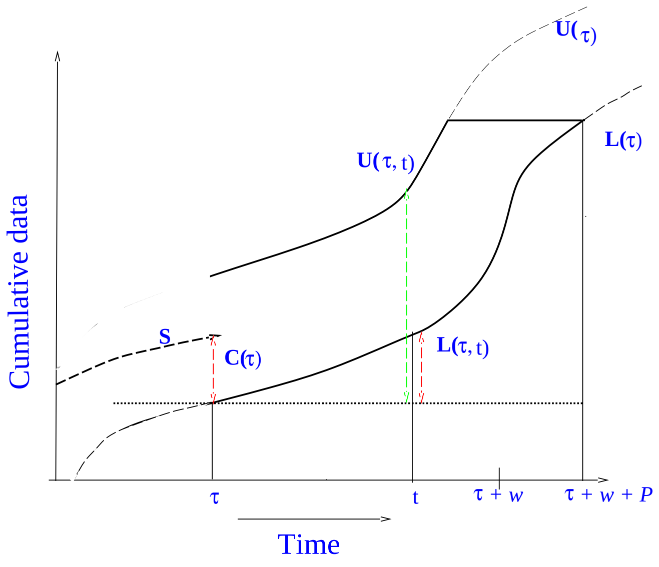
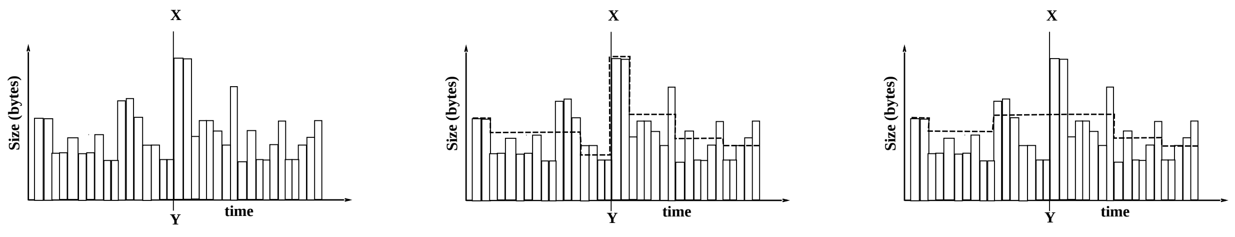


Figure 4: **Online smoothing constraints:** Starting at time τ, the server can compute a transmission schedule based on the *workahead* $C^w(\tau)$, the upper constraint $U^w(\tau, t)$, and the lower constraints $L^w(\tau, t)$.



(a) unsmoothed arrivals (b) hopping window (c) sliding window

Figure 5: **Sliding window smoothing:** This figure shows a sample sequence of arrivals at the smoothing server, with a large amount of data arriving at time Y. The dotted lines in (b) and (c) correspond to the smoothed schedules for hopping and sliding window smoothing respectively. Smoothing with a sliding window ($\alpha < w + P$) permits the server to begin transmitting this data ahead of time. In contrast, a hopping-window smoothing algorithm ($\alpha = w + P$) could be forced to transmit the large frame in a single burst to avoid violating the delay constraint.

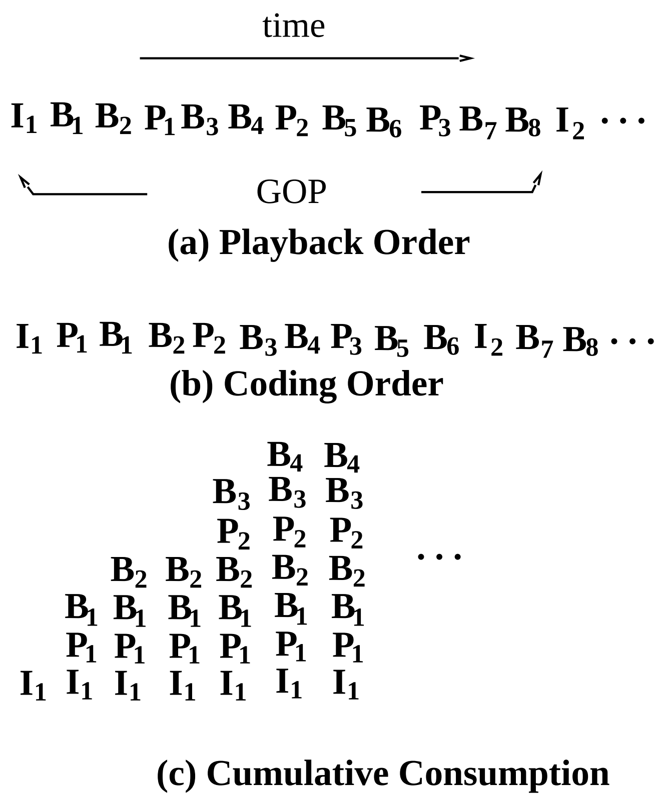
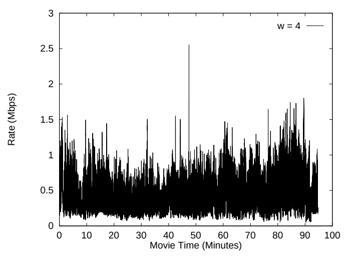
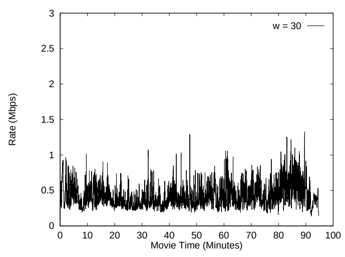

time
I1 B1 B2 P1 B3 B4 P2 B5 B6 P3 B7 B8 I2 · · ·
GOP
(a) Playback Order
I1 P1 B1 B2 P2 B3 B4 P3 B5 B6 I2 B7 B8 · · ·
(b) Coding Order
(c) Cumulative Consumption

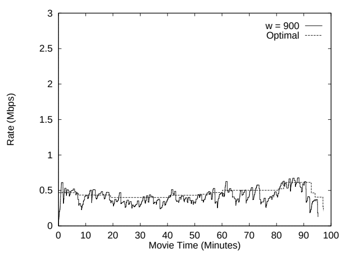
Figure 6: This figure illustrates the differences between the playback and coding orders for MPEG video. The decoding of a B frame depends on the subsequent I or P frame, which impacts the amount of data that must arrive at the client (shown in (c)).



(a) 4-frame smoothing

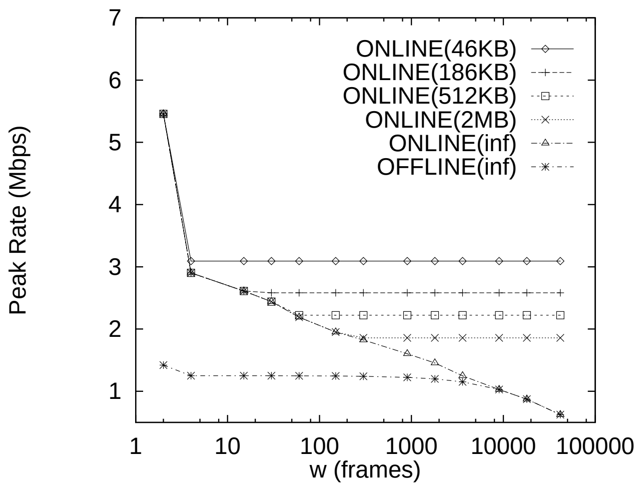


(b) 1-second smoothing

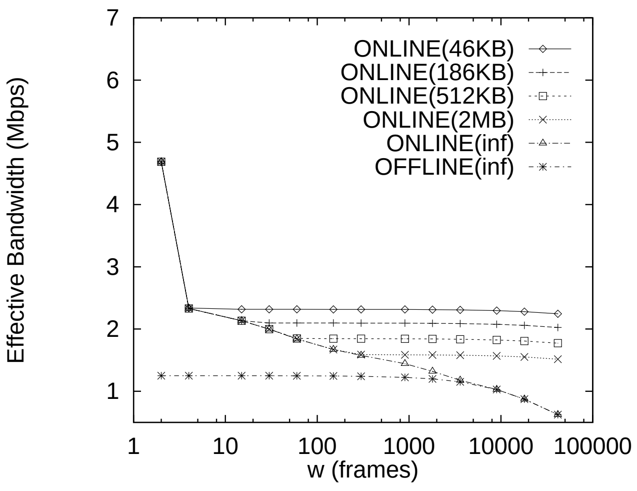


(c) 30-second smoothing

Figure 7: **Online smoothing example:** These graphs plot example transmission schedules for a portion of the MPEG-1 *Star Wars* video for offline and online smoothing with a 5-megabyte client buffer. For the online algorithm, we consider lookahead interval $P = 0$, computation period $\alpha = 1$, and an unconstrained server buffer $B_S = \infty$.

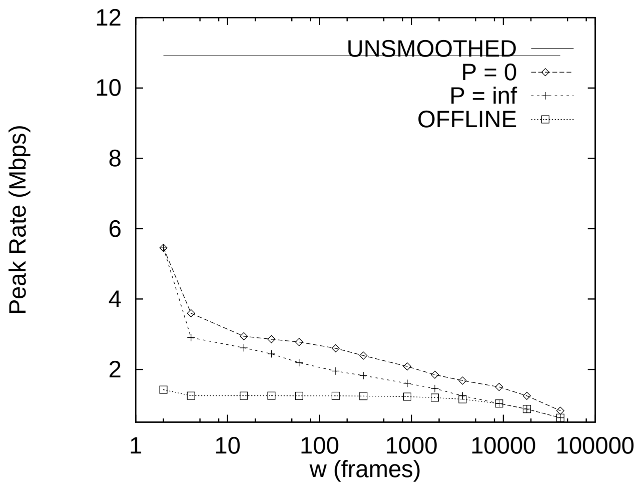


(a) Peak rate

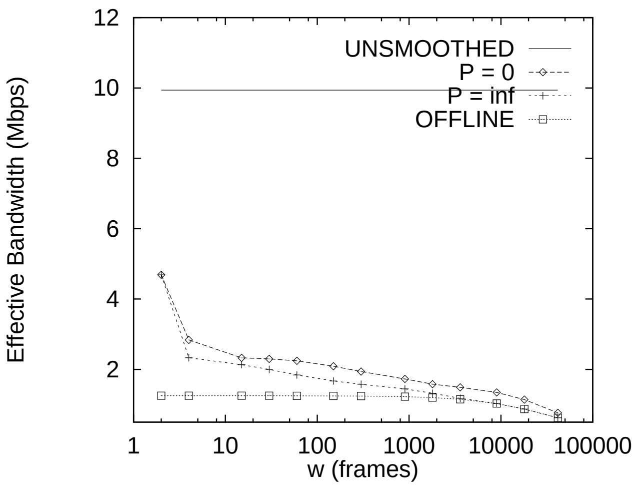


(b) Effective bandwidth

Figure 8: **Performance under full knowledge of future frame sizes:** These graphs plot the peak and effective bandwidth for MPEG *Wizard of Oz* as a function of the window size w for various client buffer sizes B_C, with $\alpha = 1$, $B_S = \infty, P = \infty$, $b = 3$ Kbytes, and $\gamma = 0.001$ (The peak rate and effective bandwidth of the unsmoothed trace are 11 and 10 Mbps respectively).

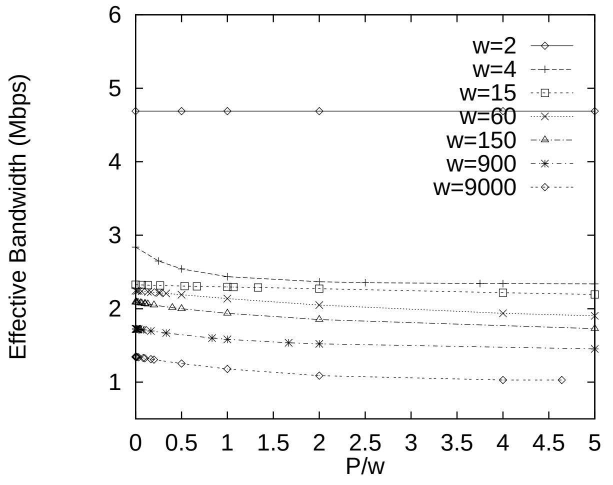


(a) Peak rate

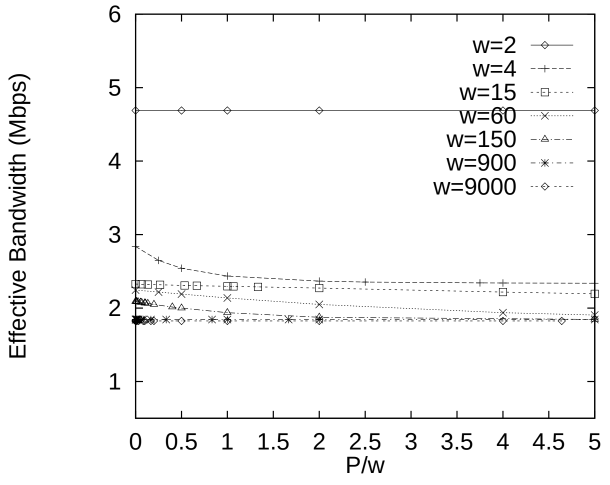


(b) Effective bandwidth

Figure 9: **Benefits of full knowledge of frame sizes:** These graph compare online smoothing with and without knowledge of future frame sizes for *Wizard of Oz*, with $\alpha = 1$, $B_C = B_S = \infty$, $b = 3$ Kbytes, and $\gamma = 0.001$.

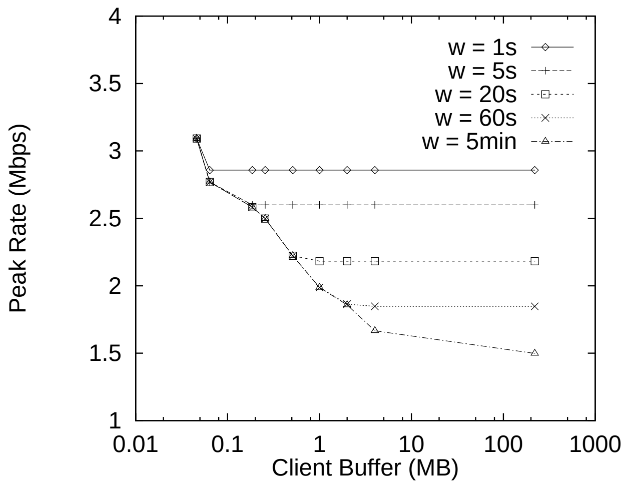


(a) $B_C = \infty$

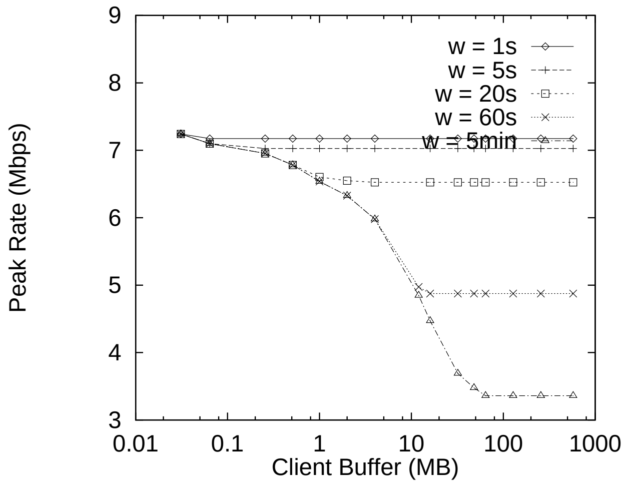


(b) $B_C = 512$ Kbytes

Figure 10: **Effects of lookahead:** The graphs plot effective bandwidth as a function of P/w for various window sizes w, for *Wizard of Oz* with $\alpha = 1$, $B_S = \infty$, $b = 3$ Kbytes, and $\gamma = 0.001$.

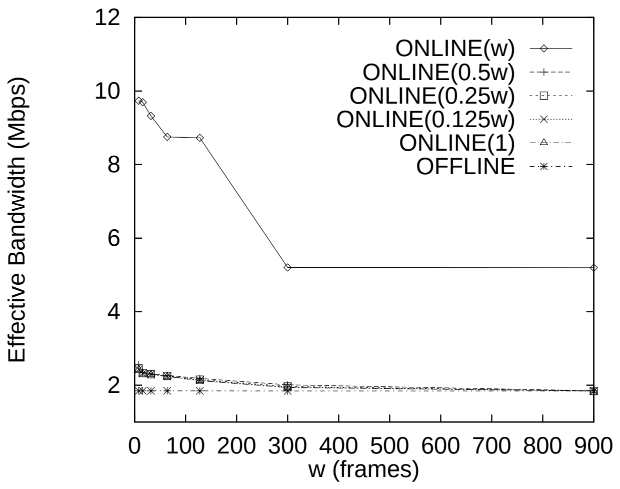


(a) Wizard Of Oz

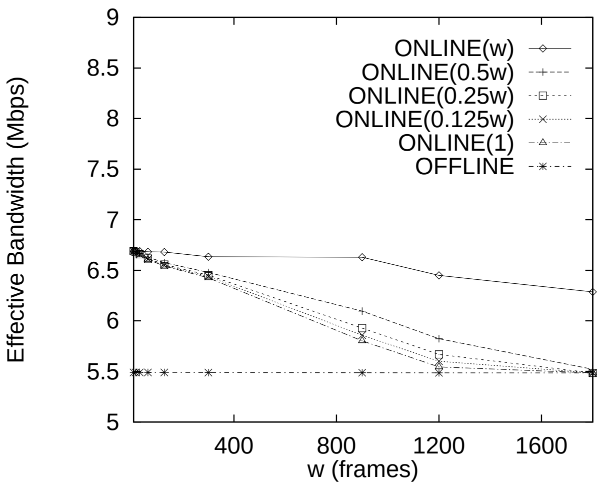


(b) Beauty and the Beast

Figure 11: **Client buffer size:** These graphs plot the peak rate as a function of the client buffer size B_C for various window sizes w, with $\alpha = 1$, $B_S = \infty$, and $P = 0$.

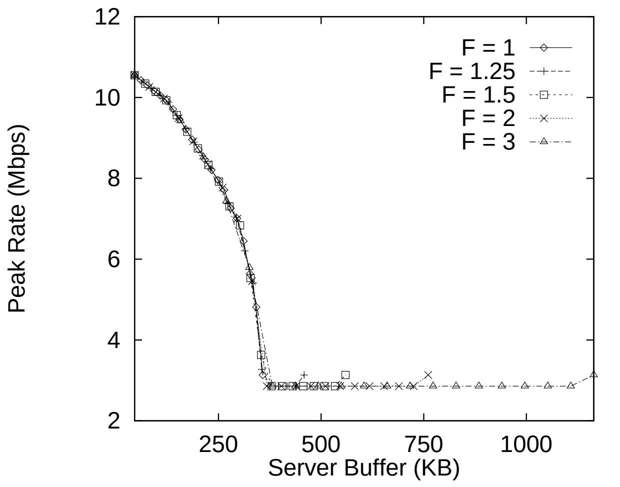


(a) Wizard of Oz

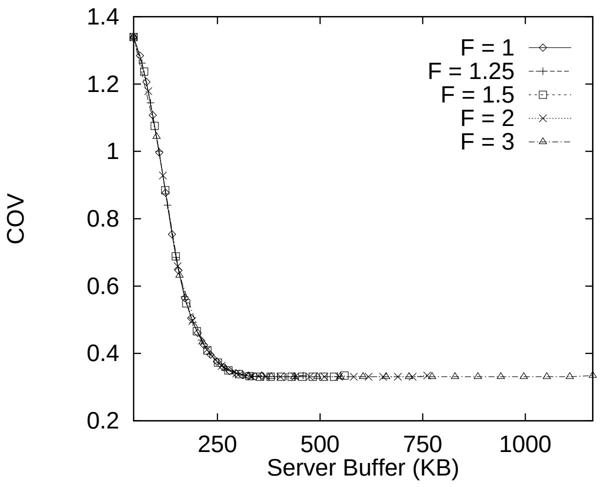


(b) Beauty and the Beast

Figure 12: **Frequency of schedule computation:** These graphs plot the performance of online smoothing for different computation periods α across a range of window sizes w. The online curves from top to bottom correspond to $\alpha = w, 0.5w, 0.25w, 0.125w$ and 1, respectively, with $P = 0$, $B_S = \infty$, $b = 3$ Kbytes, and $\gamma = 0.001$. *Wizard of Oz* has $B_C = 512$ Kbytes, whereas *Beauty and the Beast* has $B_C = 5$ Mbytes.

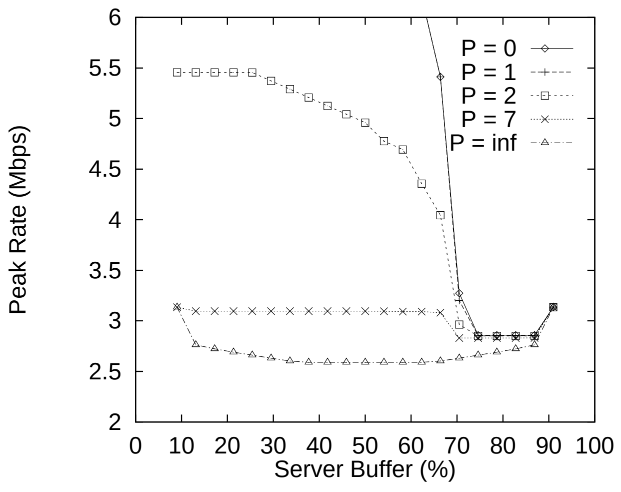


(a) Peak rate

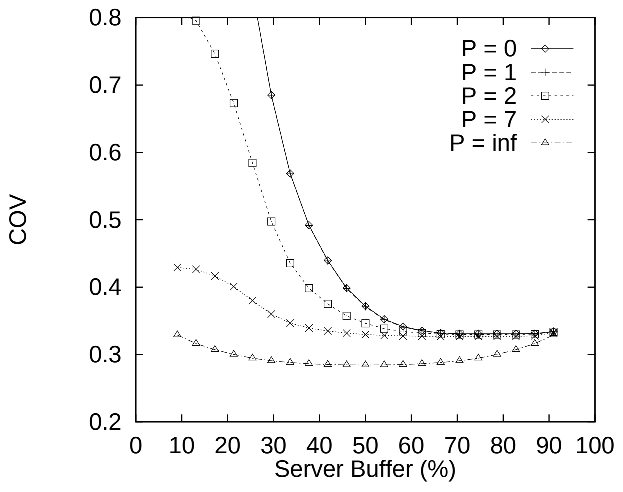


(b) Coefficient of variation

Figure 13: **Buffer allocation with no knowledge of future frame sizes:** These graphs plot the peak rate and coefficient of variation for different allocations of the server and client buffers for *Wizard of Oz* with $w = 30$, $P = 0$, and $\alpha = 1$. The experiment varies the distribution of a total buffer size of $M = F * 404$ Kbytes between an x-bit server buffer and an $(M - x)$-bit client buffer.

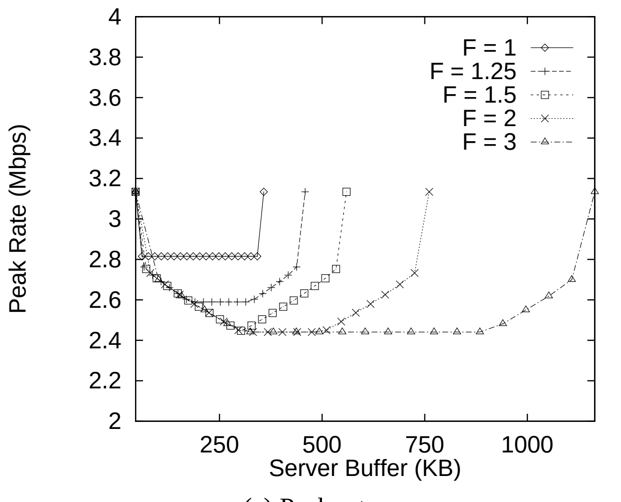


(a) Peak rate

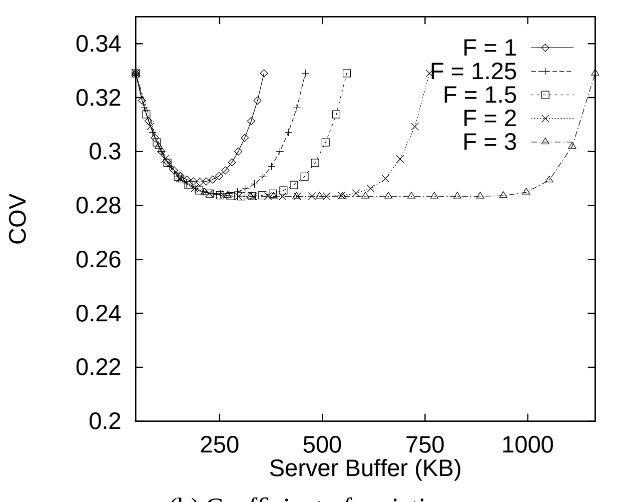


(b) Coefficient of variation

Figure 14: **Buffer allocation with partial knowledge of future frame sizes:** These graphs plot the peak rate and coefficient of variation for different allocations of the server and client buffers for *Wizard of Oz* with $w = 30$ and $\alpha = 1$. The experiment varies the proportion of the total buffer size of $M = F * 404$ Kbytes that is allocated to the smoothing server, where $F = 1.25$.



(a) Peak rate



(b) Coefficient of variation

Figure 15: **Buffer allocation with complete knowledge of future frame sizes:** These graphs plot the peak rate and coefficient of variation for different allocations of the server and client buffers for *Wizard of Oz* with $w = 30$, $\alpha = 1$, and $P = \infty$. The experiment varies the distribution of a total buffer size of $M = F * 404$ Kbytes between an x-bit server buffer and an $(M - x)$-bit client buffer.