Online Smoothing of Live, Variable-Bit-Rate Video*

J. Rexford[1], S. Sen[2], J. Dey[2], W. Feng[3], J. Kurose[2], J. Stankovic[4], and D. Towsley[2]

[1] Network Mathematics, AT&T Labs – Research, Murray Hill, NJ 07974

[2] Dept. of Computer Science, University of Massachusetts, Amherst, MA 01003

[3] Comp. & Info. Sciences, Ohio State University, Columbus, OH 43210

[4] Dept. of Computer Science, University of Virginia, Charlottesville, VA 22903

Abstract

Bandwidth smoothing techniques are effective at reducing the burstiness of a compressed, pre-recorded video stream by prefetching frames into the client playback buffer in advance of each burst. In contrast to stored video, live applications typically have limited knowledge of frame sizes and often require bounds on the delay between the source and the client(s). This paper addresses bandwidth smoothing for a growing number of live video applications, such as videocasts of courses or television news, where many clients may be willing to tolerate a playback delay of several seconds or minutes in exchange for a smaller throughput requirement. Extending techniques for smoothing pre-recorded video, we develop online, window-based smoothing algorithms for these delay-tolerant applications. Experiments with MPEG traces demonstrate that the new algorithms significantly reduce the peak rate, coefficient of variation, and effective bandwidth of variable-bit-rate video streams using fairly small window sizes (1–10 seconds), closely approximating the performance of the optimal offline algorithm [1].

1 Introduction

Many emerging applications rely on the efficient transmission of live or pre-recorded video. Although effective compression techniques can substantially reduce the storage and bandwidth requirements, compressed video streams typically exhibit significant burstiness on several time scales, due to natural variations within and between scenes [2–6]. This variability in the bandwidth requirements complicates the provisioning of resources along the path from the video source to the client site.

To reduce the burstiness and the peak bandwidth requirements, a multimedia source can smooth the outgoing stream by transmitting a sequence of video frames at an average rate. Recently, several bandwidth smoothing techniques have been introduced that reduce the server and network resource requirements for transmitting *pre-recorded* video [1, 7–10]. By capitalizing on prior knowledge of the frame sizes for the entire video, these techniques can smooth traffic on a large time scale by prefetching frames into a buffer in advance of bursts of large frames. These algorithms, given a fixed-sized client buffer, can minimize the peak bandwidth of the video stream, while also optimizing other important performance metrics [11].

In contrast, interactive video applications, such as video conferencing, typically have limited knowledge of frame sizes and require strict bounds on the delay between the server and 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 [12, 13]. Such smoothing is especially effective at removing the short-term burstiness in live MPEG video, where a small group of pictures may have widely different frame lengths. Still, techniques for transmitting interactive video typically cannot capitalize on a large client buffer to smooth the stream on a larger time scale. As a result, the live video streams can still exhibit significant burstiness.

An important set of emerging multimedia applications, such as live videocasts of course lectures or television news, lie between the two extremes of interactive and pre-recorded video. In these applications, many (or all) of the clients may be willing to tolerate a playback delay of several seconds or minutes in ex-

*The work at the University of Massachusetts was supported in part under National Science Foundation grants NCR-9508274 and CDA-9502639 and by Mitsubishi Electric Corporation. This work was also partially funded by the Ohio State University Research Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

change for a smaller bandwidth requirement. This is especially true for receivers with low-rate connections to the network. In the absence of more aggressive smoothing, these low-bandwidth clients may experience excessive packet loss or be forced to settle for a lower-quality version of the video stream.

Instead, we propose that the video source could delay transmission of the stream to permit prefetching over a larger interval of frames, based on the buffer space available at the client site(s). Alternatively, a special server in the network could delay the live transmission to smooth video for a subset of the receivers, such as clients within a single company or university. Depending on the playback delay, these clients may not be able to contribute to the live stream; however, these sites enjoy a low-rate transmission without requiring storage of the entire video or degradation in picture quality.

Drawing on techniques for smoothing pre-recorded video, this paper presents two smoothing approaches that can reduce the bandwidth requirements of delay-tolerant live video streams, where the server does not have complete knowledge of the frame sizes. As part of developing an efficient online smoothing algorithm, Section 2 briefly reviews a framework for transmitting pre-recorded video, based on a sequence of frame lengths and the size of the prefetch buffer. Section 3 extends this model to include practical constraints on both knowledge of frame sizes and the ability to prefetch data. In Section 4, simulation experiments with MPEG traces demonstrate that the proposed online smoothing algorithms are effective at reducing the bandwidth requirements for distributing compressed video. Section 5 concludes the paper with a discussion of future research directions.

2 Smoothing Pre-Recorded Video

A multimedia server can substantially reduce the rate requirements for transmitting pre-recorded video by prefetching frames into the client playback buffer. A class of bandwidth smoothing algorithms capitalize on *a priori* knowledge of the pre-recorded stream to compute a server transmission schedule, based on the size of the prefetch buffer.

2.1 Smoothing Constraints

To formalize the problem, consider a compressed video stream that consists of N frames, where frame i is ℓ_i bytes long. To permit continuous playback at the client site, the server must always transmit enough data to avoid buffer *underflow*, where $L_k = \sum_{i=1}^{k} \ell_i$ indicates the amount of data consumed at the client by time k, $k = 0, 1, \ldots, N$. Similarly, to prevent *overflow* of the B-byte playback buffer, the client should not receive more than $U_k = L_k + B$ bytes by time k. Any valid server transmission plan should stay within the river outlined by these vertically equidistant functions, as shown in Figure 1(a). Hence, during time unit i, the server transmits at rate c_i, where $L_k \leq \sum_{i=1}^{k} c_i \leq U_k$.

Creating a bandwidth plan involves generating M consecutive runs, where run j has slope r_j, $j = 0, 1, \ldots, M-1$; at time i the server transmits at rate $c_i = r_j$, where slot i occurs during run j. For example, Figure 1(b) shows a smoothed transmission schedule for an MPEG encoding of *Star Wars* [4] for two different buffer sizes (256 and 512 kilobytes, respectively). With a larger client prefetch buffer, the server can further reduce the peak and variability of the rates c_i, as well as the number of bandwidth runs M.

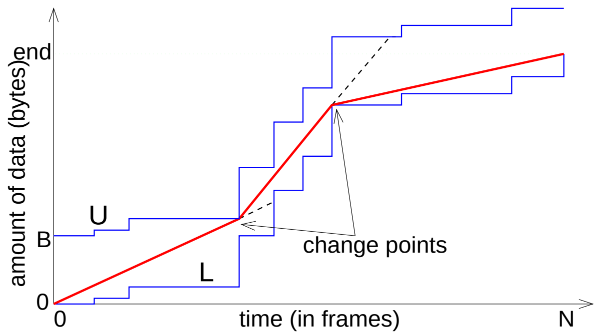
2.2 Computing Transmission Schedules

The constraints L and U typically result in multiple feasible transmission schedules with different performance properties [11]. For example, Figure 1(a) shows a plan with $M = 3$ runs, where the second run increases the transmission rate to avoid an eventual buffer underflow at the client prefetch buffer; similarly, the third run decreases the rate to prevent overflow. To limit the peak rate $\max_i\{c_i\}$ in the transmission schedule, a bandwidth smoothing algorithm can select trajectories that extend as far as possible between the L and U curves. In this framework, the trajectory for run j must eventually encounter both the overflow and the underflow curves, resulting in a frontier of possible starting points for run $j+1$, as shown by the dashed lines in Figure 1(a).

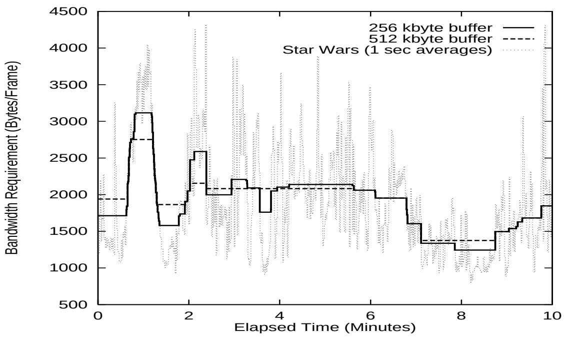
Starting each new run at the *leftmost* point along the frontier results in a transmission schedule that minimizes the peak and variability of the rates $\{c_i\}$, as well as the video's effective bandwidth requirement [1]. This optimal offline algorithm has $O(N)$ complexity [1] and serves as our starting point for developing effective techniques for online smoothing of live video. In the next section, we extend the algorithm to operate on a window of W video frames by modifying the smoothing constraints L and U. The new online algorithms periodically recompute the transmission schedules as more frames arrive, based on either a hopping or sliding window. Smoothing with a small window can remove the short-term burstiness in interactive video, while larger window sizes can further reduce the bandwidth requirements at the expense of longer playback delay.

3 Window-Based Smoothing

Existing techniques for transmitting pre-recorded video serve as a useful foundation for developing new algorithms for online smoothing under client buffer



(a) Underflow/overflow curves



(b) *Star Wars* plans

Figure 1: Constructing server transmission plans

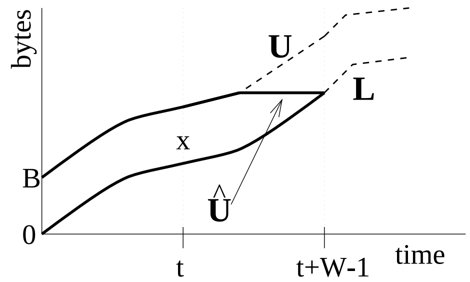


Figure 2: Online smoothing constraints L and $\hat{U}$

constraints. However, operating with a limited knowledge of future frame sizes requires extensions to the model in Figure 1(a).

3.1 Online Smoothing Constraints

For a live video transmission, the multimedia server cannot perform bandwidth smoothing over the entire video stream. Instead, our approach to online smoothing is to apply the existing techniques for offline smoothing to small segments of the live video as the frames are generated. Thus, the online algorithm is executed repeatedly during the video transmission, each time on a different smoothing window.

To develop the algorithm, we consider a model where the video source generates one frame of data in each time unit. A smoothing window of W frames introduces a playback delay of W time units at the client; in addition, the server must include sufficient buffer space to store up to W frames of the live video feed. Ultimately, a larger value of W facilitates a smoother transmission of the video, at the expense of a larger buffer requirement at the server and a longer playback delay at the client(s).

For online smoothing, the amount of prefetching is limited by the video data available in the W-frame window, as well as the size B of the client prefetch buffer. At transmission time slot t, the server knows the frame sizes $\{\ell_i\}$ up to $i = t+W-1$. Hence, at time t, the server can conservatively deduce that $U_k \geq \hat{U}_k^t$, where

$$\hat{U}_k^t = \min\left(L_k + B, \sum_{i=1}^{t+W-1} \ell_i\right)$$

for $k = t, t+1, \ldots, t+W-1$, where $L_k = \sum_{i=1}^{k} \ell_i$. Depending on the values of B and W, prefetching can be limited by the data available at the server or the space available at the client, as shown in the example in Figure 2.

The server can use the conservative estimate $\hat{U}$ to compute the online transmission schedule. Building on the intuition from Section 2, the server can smooth the video by selecting trajectories that stay between the constraint curves as long as possible. However, the estimate of U_k can gradually improve as more frames arrive; by time t, the server can transmit up to $\hat{U}_t^t$ bytes of data. Still, to limit the computational requirements of online smoothing, the server may choose not to change the transmission schedule in every time slot; in fact, in many cases, the arrival of new frames does not change the optimal transmission rate anyway.

Drawing on the modified constraint curves and the optimal offline algorithm, we propose a collection of online smoothing schemes that differ in when they choose to recompute the server schedule. Initially, we consider smoothing algorithms that compute transmission rates based only on the existing video frames; these algorithms could be extended to perform more aggressive prefetching by incorporating effective heuristics [12–14] for predicting future frame sizes.

3.2 Hopping-Window Smoothing

As an initial approach to online smoothing, we consider *hopping-window* algorithms that operate on consecutive non-overlapping smoothing windows, each of length W. These algorithms execute every W

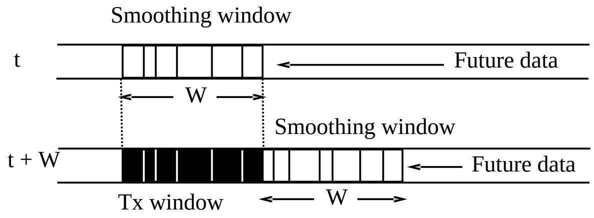


Figure 3: Hopping-window smoothing

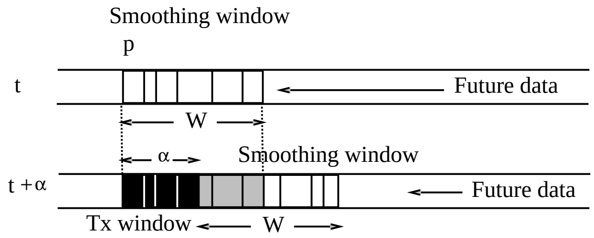


Figure 4: Sliding-window smoothing

time units, smoothing the W most recently generated frames; for example, the dark shaded region in Figure 3 represents the smoothed schedule available for transmission due to smoothing performed at time t. For MPEG encoded videos, we choose W to be an integer multiple of the number of frames G in a group-of-pictures (GOP) to ensure that each window has a similar proportion of I, P, and B frames. If the first I frame is large, the hopping-window smoothing schemes can incorporate a modest amount of extra playback delay to amortize the cost of transmitting this frame.

The baseline hopping-window algorithm, called $BASE$, simply applies the offline smoothing algorithm [1] on consecutive segments of the video stream. However, since I frames are typically larger than P and B frames, placing an I frame at the start of a window can require a high transmission rate since the server cannot "prefetch" the data for this large frame across the other $W-1$ time slots. Hence $BASE$ is likely to perform poorly because it will attempt to smooth an I frame at the beginning of every smoothing window except in the first window, where the transmission of the first I frame is amortized over a startup latency period.

Instead, our next hopping-window algorithm, called FAR, aligns the consecutive windows with an I frame at the *end* of each window; this can be achieved by starting the smoothing algorithm with a initial window of $W+1$ frames before using the size W for the remainder of the video. The first I frame in every other smoothing window $G-1$ frames away, which is the farthest an I frame can be from the beginning of any window. Hence, the FAR algorithm maximizes the time for the server to "prefetch" the largest frames in the video stream.

3.3 Sliding-Window Smoothing

Although the hopping-window algorithm can smooth across a set of W/G GOPs, the non-overlapping windows do not permit the server to prefetch data across window boundaries. In contrast to the hopping-window approach, *sliding-window* smoothing executes on overlapping windows of size W. To incorporate the incoming video frames, the algorithm SLWIN(α) computes the transmission schedule every α time units over the next W frames, where $\alpha \leq W$. At time t the algorithm smooths a video segment from frame position p to $p+W$, as shown in Figure 4. At time $t+\alpha$, when α additional video frames are generated for transmission, the algorithm starts to smooth W frame positions starting from frame position $(p+\alpha+1)$ to $(p+\alpha+W+1)$. The current smoothing window at $t+\alpha$ consists of a initial $(W-\alpha)$ frame slots (grey shaded area) which have been part of one or more past smoothing windows, and α new frame positions (unshaded region in the window). The dark shaded area corresponds to the smooth schedule available for transmission due to smoothing performed at time t.

Since the offline algorithm uses work-ahead transmission, whenever possible, it sends more data into the client buffer than needed by the client. Thus several frames in the range $[p+\alpha+1, p+W]$ may be completely or partially transmitted by time $t+\alpha$, leaving fewer than W frames worth of data to be smoothed over a transmission time period of W. Figure 5 compares the sliding-window algorithm to the hopping-window schemes. If a window boundary occurs at position XY, then, under a hopping-window algorithm, the first few large frames in the window starting at XY would inflate the peak rate of the smoothed schedule. In contrast, the sliding-window algorithm reduces the peak rate by prefetching part of this data along with earlier frames in the stream.

Figure 6 demonstrates the benefits of smoothing an MPEG encoding of the movie *Star Wars* [4], which has a maximum frame size of 23,160 bytes and a mean frame size of 1950 bytes. As shown in Figure 6(a), the video exhibits significant burstiness even after averaging across the 12-frame GOP; averaging over this small time scale reduces the peak rate from 23160 bytes/frame-slot to 10670 bytes/frame-slot. In contrast, the sliding-window algorithm with a 2-second

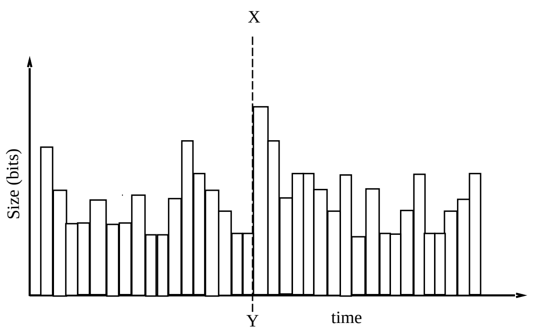


Figure 5: Hopping vs. sliding windows

(60-frame) window, a 0.5-megabyte client buffer, and a slide length of $\alpha = 1$, reduces the peak bandwidth to 5800 bytes/frame-slot. Figure 6(b) shows part of the resulting bandwidth plan, compared to the offline algorithm that has complete knowledge of the frame sizes and unlimited prefetching within the 0.5-megabyte buffer constraint. Figure 6(c) shows the plan for a 30-second (900-frame) window, which nearly converges with the offline schedule.

3.4 Cost/Performance Trade-Offs

For a given window size, the sliding-window algorithm should outperform the hopping-window algorithm by prefetching across overlapping windows. However, this improved performance does come at a cost. The computation overhead for the sliding-window approach is higher than that for a hopping-window approach, for a given window size, and depends on how often the server recomputes the transmission schedule. With a window of size W, the sliding-window algorithm smooths each frame potentially W/α times on the average, whereas the hopping-window approach smooths each frame only once.

Smaller values of α allow the server to gradually refine the schedule based on new frames, at the expense of additional computational load[1] on the video server. Fortunately, though, the server does not always have to determine the schedule for the entire W-frame interval, since the schedule is recomputed every α time units anyway. In fact, the server can stop computing after determining the first run that proceeds up to (or beyond) α time units in the future. Still, in a real-time environment, the server must balance the trade-off between computational load and the reduction in the bandwidth requirement. In fact, a flexible video server could allow the parameter α to vary across the video transmission, depending on the prevailing constraints on processing and network resources.

[1] The running time of the offline smoothing algorithm is linear in the number of frames. Using a MIPS R4400 processor with 32 megabytes a memory, the algorithm consumes just under 12 msec to smooth one second of 30 frames/second video.

4 Performance Evaluation

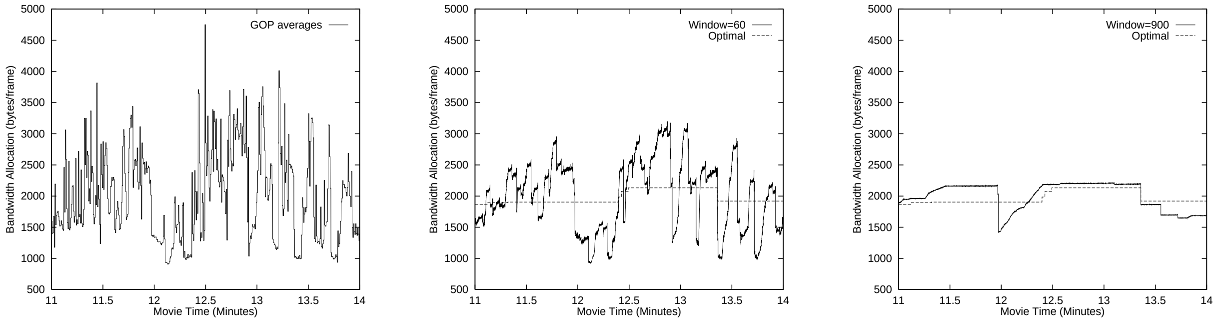
Drawing on MPEG-1 trace data, this section examines the performance of the online smoothing algorithms. The experiments focus on the server and network resource requirements by measuring the *peak rate*, *coefficient of variation* (standard deviation divided by the mean rate), and the *effective bandwidth* of the smoothed video stream. The effective bandwidth [15, 16] statistically estimates the network throughput required to transmit the video through a b-byte buffer with a tolerable loss rate of γ. For a video with transmission rates rates $c_1, c_2, \ldots, c_N$, the effective bandwidth is computed as $\log(\sum_{i=1}^{N} e^{\theta c_i}/N)/\theta$ where $\theta = -\log(\gamma/b)$.

We explore how these three bandwidth metrics change as a function of the window size W, the client buffer size B, and the slide length α. To show the lower and upper bounds for the performance metrics, the graphs also plot results for the unsmoothed video stream and the optimal offline schedule. The experiments evaluate a 28-minute MPEG trace of *MTV* with a mean rate of 590 kilobits/sec. Experiments with other compressed video sources show the same trends.

4.1 Window Size (W)

To investigate the potential for online smoothing, Figure 7 compares the smoothing algorithms across a range of window sizes W. The simulation experiment evaluates the smoothing algorithms with a 512 kilobyte prefetch buffer, a 0.5-second start-up latency, and a slide length of $\alpha = W/2$ for the sliding-window algorithm; experiments with other video traces and different values of α show similar trends. The effective bandwidth graph in Figure 7(c) corresponds to the minimum network throughput required for a $\gamma = 0.001$ loss probability at a switch buffer with capacity $b = 64$ kilobytes. Comparing the original unsmoothed video stream with the offline schedule illustrates the potentially dramatic benefits of smoothing in reducing network bandwidth requirements.

The two hopping-window algorithms effectively reduce the bandwidth requirements for transmitting the video stream, with FAR consistently outperforming BASE. Even with a 1-second smoothing window, BASE and FAR reduce the peak rate by 16% and 54% over the original video traces, respectively; for comparison, the optimal offline algorithm reduces the peak by 72%. For the same window size, the coefficient of variation drops from 0.94 for the original video



(a) GOP frame sizes (b) 2-second smoothing (c) 30-second smoothing

Figure 6: **Sliding-window smoothing plans for the *Star Wars* MPEG video**

to 0.64 for BASE, 0.46 for FAR, and 0.25 for the offline algorithm. Similarly, effective bandwidth drops by 13%, 16%, and 18% for BASE, FAR, and the offline algorithm, respectively. The superiority of FAR over BASE is most dramatic for the peak-bandwidth metric in Figure 7(a), since FAR explicitly avoids placing large frames at the beginning of a smoothing window.

Both the hopping and sliding window-based approaches significantly reduce the variability of the transmission rates, even for fairly small window sizes. These results indicate that even very low-latency live applications can benefit significantly from online smoothing of video traffic with a modest amount of buffer space at the client site. For larger window sizes, the performance of FAR and SLWIN(W/2) gradually converge to that of the optimal offline algorithm. For example, using a 20-second window, the FAR algorithm reduces the peak rate by 69% over the original unsmoothed video, which is comparable to the performance of the offline algorithm. Hence, non-interactive live applications which can tolerate latencies of a few tens of seconds can receive almost all of the potential benefits of smoothing into the client prefetch buffer. The sliding-window algorithm consistently outperforms FAR by incorporating incoming frames into the smoothing schedule more quickly.

To further highlight the benefits of online smoothing, Figure 8 graphs the effective bandwidth estimates of FAR and SLWIN(W/2), along with the original video and the offline schedule. The experiment employs the same parameters Figure 7, except that we fix the window size at $W = 5$ seconds and vary the switch buffer size b in measuring the effective bandwidth. As expected, the effective bandwidth of all the four curves decays rapidly with linear increases in buffer size. For small switch buffer sizes, the effective bandwidth of FAR and SLWIN(W/2) is significantly lower than that of the original trace. This

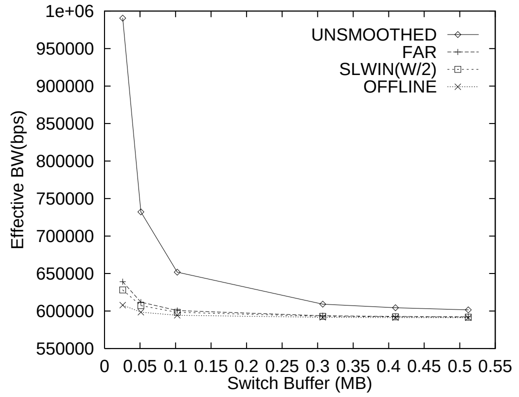


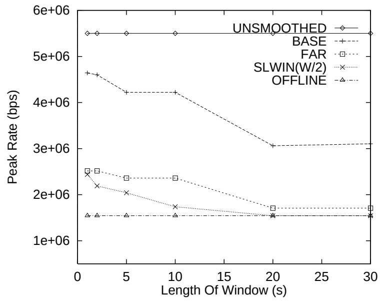
Figure 8: Effective bandwidth vs. switch buffer size

dramatically reduces the network bandwidth requirements for transmitting the video stream, even under a fairly small window size.

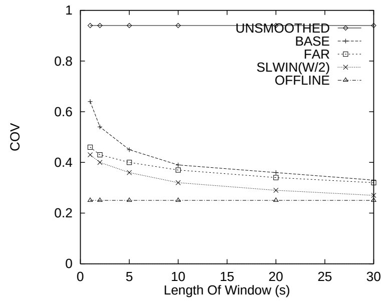
4.2 Client Buffer Size (B)

For a practical buffer size ($B = 512$ kilobytes) and playback delay ($W = 10$ seconds), the the online algorithms are extremely effective in removing the short and medium-term burstiness in the underlying video stream, as shown in Figure 7. Removing any remaining longer-term burstiness would require both a larger window size and a larger prefetch buffer. However, increasing the size of the client prefetch buffer offers diminishing returns, even for the offline smoothing algorithm, as shown in Figure 9. To highlight the interplay between the window size W and the buffer size B, these graphs compare the sliding-window algorithm (with $\alpha = W/2$ and windows of 1, 5, 10 and 60 seconds) to the offline algorithm across a range of client buffer sizes.

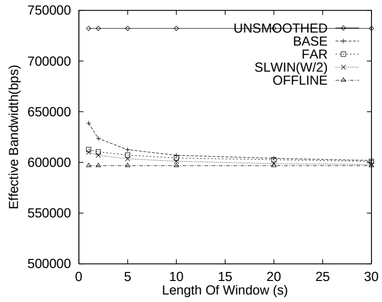
The online algorithm performs extremely well for small values of B, since the buffer size imposes the main constraint on prefetching in this region. These



(a) Peak rate

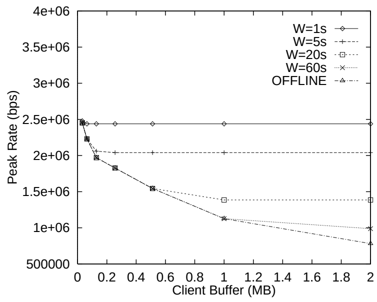


(b) Coefficient of variation

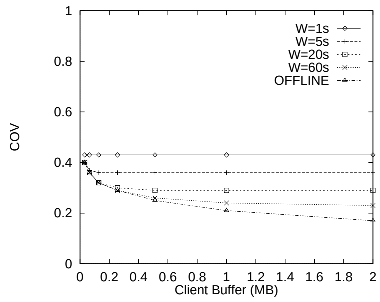


(c) Effective bandwidth

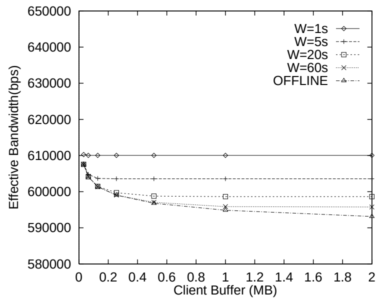
Figure 7: **Comparing smoothing algorithms under different window sizes W**



(a) Peak rate



(b) Coefficient of variation



(c) Effective bandwidth

Figure 9: **Comparing sliding-window sizes W under different buffer sizes B**

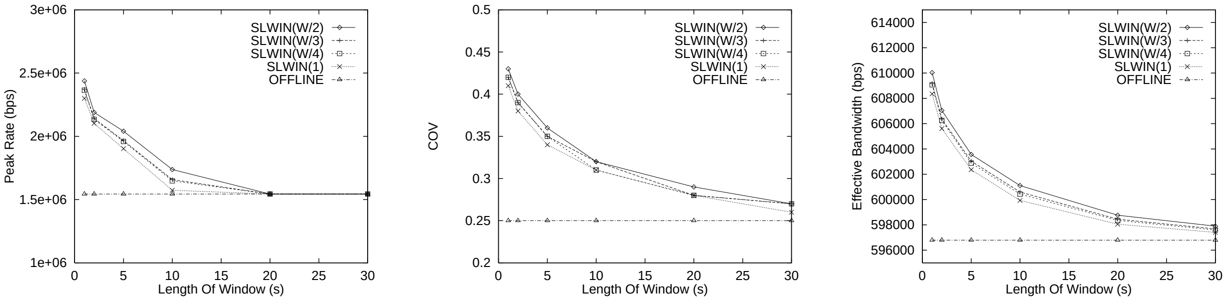
results indicate that small windows are sufficient to reap most of the benefits of smoothing for low-latency applications with relatively small client buffers. For a given window size, the bandwidth requirements for the video stream initially decrease as the client buffer size grows. In this region, the client buffer is still the main constraint on prefetching and so a larger client buffer size allows the smoothing algorithm to prefetch data 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 prefetch data. For example, for a 5-second sliding window, increasing the client buffer beyond 0.256 megabytes has no discernible performance benefits.

The offline algorithm, having access to the entire video, can effectively utilize a larger buffer to remove some of the longer-term burstiness in the underlying video. As such, the performance of the window-based algorithm for different window sizes diverges for larger buffers, with the performance for larger window sizes approaching that of the optimal offline algorithm. 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. Larger window sizes, in the range of 1–5 minutes, would be appropriate for a non-interactive client with a low-bandwidth connection to the network, and these applications can benefit from larger client buffers.

4.3 Slide Length (α)

The benefits of sliding-window smoothing depend on how frequently the video source recomputes the transmission schedule to incorporate new arriving frames. To investigate the sensitivity of online smoothing to the computation frequency, Figure 10 compares the performance of the sliding-window algorithm for different values of α across a range of window sizes W with a 512-kilobyte client buffer. Although smaller slide lengths decrease the burstiness of the video stream, the performance differences between SLWIN(1), SLWIN(W/2), SLWIN(W/3) and SLWIN(W/4) are not significant.

For a 1-second smoothing window, the difference in peak rate and effective bandwidth for SLWIN(W/2) and SLWIN(1) is 2.5% and 0.23% respectively of the corresponding values for the original unsmoothed trace. For a moderately long smoothing window of



(a) Peak rate (b) Coefficient of variation (c) Effective bandwidth

Figure 10: **Comparing slide lengths α under different window sizes W**

10 seconds, the differences are 3% and 0.16% of the corresponding values for the unsmoothed case. The peak rates converge to the optimal offline peak rate for a 20-second window, and the difference in effective bandwidths is less than 0.1% for window sizes above 20 seconds, indicating that the smoothed streams have very similar rate variability. Hence, nearly all the benefits of sliding-window smoothing can be achieved by the SLWIN(W/2), the least computationally intensive of the four algorithms.

5 Conclusions

By delaying the transmission of video frames by W time units, the window-based online smoothing algorithms can substantially reduce the bandwidth requirements for distributing live video streams by "prefetching" frames into the B-byte client playback buffer. The hopping-window and sliding-window algorithms extend previous work on techniques for smoothing pre-recorded video by incorporating the limited availability of "future" frames and the need to recompute the transmission schedule as new frames arrive. Simulation experiments show that the algorithms, while conservative, can reduce the peak transmission rate, allowing low-bandwidth clients to receive the video stream with a modest playback delay.

Extensions to the window-based algorithms can further improve performance by using more optimistic estimates of the overflow curve to compute bandwidth trajectories. For example, the server could project the lengths of future frames based on the underlying video encoding scheme and measurements of the previous frame sizes. Also, for a more flexible approach to online smoothing, the server could select the window size W and computation interval α to balance the use of processing, memory, and network resources.

Finally, as ongoing work, we are considering ways to combine sliding-window 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.

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," in *Proc. ACM SIGMETRICS*, pp. 222–231, May 1996.

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

[3] 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.

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

[5] 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.

[6] M. Grossglauser, S. Keshav, and D. Tse, "RCBR: A simple and efficient service for multiple time-scale traffic," in *Proc. ACM SIGCOMM*, pp. 219–230, August/September 1995.

[7] W. Feng and S. Sechrest, "Smoothing and buffering for delivery of prerecorded compressed video," in *Proc. IS&T/SPIE Symp. on Multimedia Computing and Networking*, pp. 234–242, February 1995. Extended version appears in *Computer Communications*, October 1995, pp. 709–717.

[8] W. Feng, F. Jahanian, and S. Sechrest, "Optimal buffering for the delivery of compressed prerecorded

video," in *Proc. IASTED/ISMM Inter. Conf. on Networks*, January 1995. Extended version to appear in *ACM/Springer-Verlag Multimedia Systems Journal*.

[9] 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*, August 1996, pp. 1087–1098.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] A. Adas, "Supporting real-time VBR video using dynamic reservation based on linear prediction," in *Proc. IEEE INFOCOM*, pp. 1476–1483, March 1996.

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

[16] 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.