Online smoothing of live video transmissions*

Subhabrata Sen Jayanta K. Dey James F. Kurose John A. Stankovic Don Towsley
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Abstract

In this paper, we consider the problem of online smoothing of live video transmissions, where low latency is critical. To address the low latency requirement, our approach is to wait for a small *smoothing window* of video frames to be generated, and smooth them prior to their transmission. We present two classes of techniques that adapt an optimal offline algorithm[11] for online use: *jumping window* algorithms, in which an algorithm executes on consecutive non-overlapping smoothing windows, and *sliding window* algorithms, in which an algorithm executes on overlapping smoothing windows. We then evaluate these techniques using MPEG-1 traces. We find that the videos smoothed by these techniques show dramatic reduction in their peak rate, coefficient of variation and effective bandwidth. Furthermore, these reductions can be obtained by using very small window sizes, from 1 s to 5 s, indicating that even live video applications that have extremely tight latency requirements can benefit tremendously from using online smoothing. It is also of interest to note that these reductions in peak rate, coefficient of variation and effective bandwidth are very close to those obtained by the optimal offline algorithm. This implies that most of the smoothing benefits that can possibly be seen by a live video transmission can be attained by online smoothing operating on a very small smoothing window, thus incurring only a very small additional latency.

1 Introduction

Digital video, compressed using techniques such as MPEG, can exhibit significant bit-rate variability, often spanning multiple time scales and in some cases exhibit *self-similar behavior*[4]. This burstiness is inherently at odds with the goals of designing efficient real-time storage, retrieval, network transport and admission control mechanisms capable of achieving high resource utilization. To ameliorate the problem for video that is stored, several researchers have developed techniques to transmit variable-bit-rate (VBR) video in a less bursty (i.e., *smoother*) manner by exploiting buffering capabilities at the client[3, 8, 9, 10, 11]. Results in [11] show a reduction in peak bit rate and standard deviation by 70-85%, when video is smoothed into a 1 MB client buffer. This indicates that smoothing techniques are likely to be deployed for transport of compressed stored video.

In this paper, we consider the problem of smoothing live video transmissions, which have low playback latency requirements. There is a fundamental tradeoff between playback latency and quality of smoothing. Offline smoothing algorithms perform better smoothing over long video traces. For live video transmissions, however, it is necessary to maintain low latency, and therefore it is not desirable to wait for long video sequences

*This work supported in part under National Science Foundation grants NCR-9508274 and CDA-9502639. 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 National Science Foundation.

to be generated. Thus, the dual goals of an online smoothing technique are to achieve a high degree of smoothing while providing low latency.

To address the low latency requirement, our approach is to wait for a small *smoothing window* of live video frames to be generated, and smooth the transmission of these frames prior to transmitting them. Our algorithms, described in Section 2, are based on the offline algorithm developed in [11]. We use this algorithm because, given a stored video, it produces the smoothest possible schedule by minimizing its peak rate, coefficient of variation and effective bandwidth[5, 2].

We evaluate the efficacy of window-based approaches to on-line smoothing by evaluating their performance on MPEG-1 traces in Section 3. We find that the videos smoothed by these techniques show dramatic reduction in their peak rate, coefficient of variation and effective bandwidth. For example, the SLWIN(W/2) algorithm described in the paper reduces the peak and coefficient of variation of a video trace by 63% and 62% respectively, using only a 5 s smoothing window. These significant reductions can thus be realized by using very small window sizes, and live video applications with extremely tight latency requirements can benefit tremendously from using online smoothing. It is also of interest to note that these reductions in peak rate, coefficient of variation and effective bandwidth are very close to those obtained by the optimal offline algorithm. This implies that most of the smoothing benefits that can possibly be seen by a live video transmission, can be attained using the algorithms we investigate in this paper.

Other techniques have been investigated in smoothing of live video. Adas et. al[1] investigate linear prediction techniques for forecasting bandwidth requirements of future frames. Lam et. al[7] present a lossless smoothing algorithm which employs heuristics for predicting future frames based on knowledge of compression patterns in the video. Kanakia et. al[6] have presented adaptive techniques that use congestion control feedback from a network to regulate and smooth the output bit rate within a video encoder.

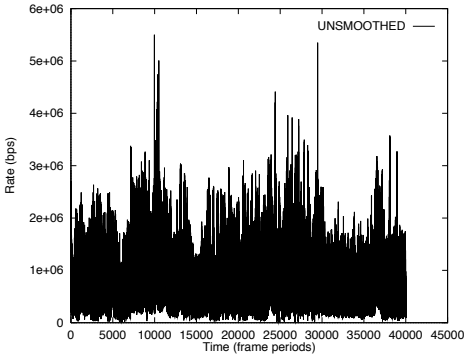
2 Problem setting and algorithm description

Given a video trace and client buffer size information, the smoothing algorithm presented in [11] analyzes the entire trace offline and produces a transmission schedule that is *smoother*, i.e., has a much lower peak rate as well as lower rate variability, subject to the constraint that the client buffer must not overflow or underflow. The technique does work-ahead transmission into the client buffer, using the buffer to absorb the rate variability in the video. To amortize the transmission of the first frame of the video, particularly for MPEG compressed videos in which it is a large *I* frame, a small playback *startup latency* can be used. This playback startup latency is defined to be the amount of time the client waits, after receiving the first frame of the video, to begin playback. In [11] it is shown that a small 0.5 s startup latency is often enough to ameliorate this "first frame" effect.

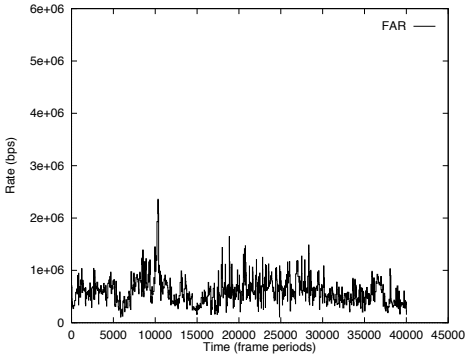
For live video transmission, however, the entire video is not available to perform optimal smoothing. The fundamental idea behind our approach to online smoothing is to adapt the offline smoothing algorithm to execute on small segments of a live video as they are generated. These segments over which smoothing is performed are referred to as *smoothing windows*. Thus the online algorithm is executed repeatedly during the video transmission, each time on a different smoothing window. We investigate the viability of this heuristic approach, by comparing the peak rate and rate variability of the video smoothed by window-based online algorithms with the unsmoothed video and the smoothed video generated by the optimal offline approach.

Without loss of generality, we consider a discrete time model where data is transmitted at time slots, each slot corresponding to a video frame period. If the smoothing window size is W, any smoothing algorithm has to wait for W time units after a live video transmission begins, in order that W frames are generated. Thus the effective latency for the video as seen at the client, ignoring propagation delay, is $W + l$ when the playback startup latency is l[1].

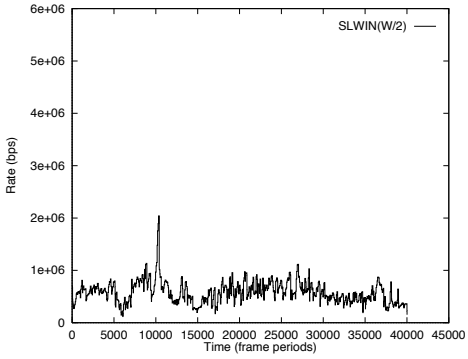
[1] We assume that the execution time of the smoothing algorithm is negligible. The algorithm is $O(N)$ and takes less than 40 μs on the average to smooth one frame on a MIPS R4400 processor with 32 MB of memory, i.e., it takes less than about 12 ms to smooth 1 s worth of 30 frame/s video.



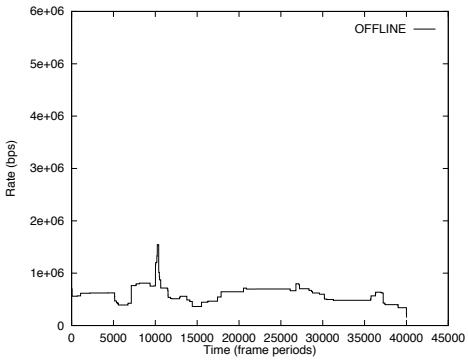
(a) UNSMOOTHED



(b) FAR



(c) SLWIN(W/2)



(d) OFFLINE

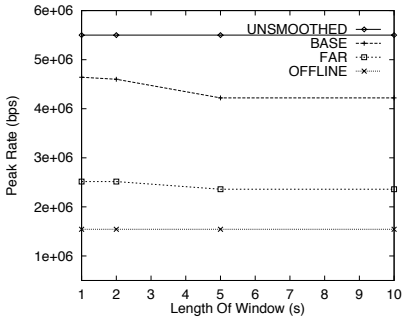
Policy	Normalized Peak	Normalized COV
UNSMOOTHED	100.00	100.00
FAR	42.89	42.55
SLWIN(W/2)	37.10	38.30
OFFLINE	28.07	26.60

(e) Comparison of different policies

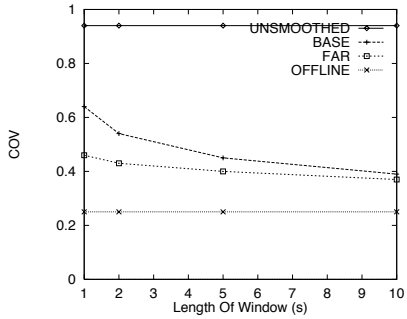
Figure 1: Smoothing of *MTV* trace. For OFFLINE and SLWIN(W/2), the startup latency is 0.5 s and the client buffer size is 512 KB. The smoothing window size for SLWIN is 5 s.

We refer to the first class of algorithms we investigate as *jumping window* algorithms, since they execute on consecutive non-overlapping smoothing windows, each of length W. They execute every W frame periods, smoothing the W frames that are generated in that time period. For MPEG traces, we choose W to be an integer multiple of a group-of-pictures(GOP). The baseline algorithm, called *BASE*, simply applies the offline smoothing algorithm[11] on consecutive segments of the video. In MPEG, every GOP begins with a large *I* frame, and 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. We thus consider a variation called FAR, which places the first I frame farther within a smoothing window. We denote the size of a GOP as G. FAR chooses the first smoothing window size to be $(W + 1)$, thus keeping 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.

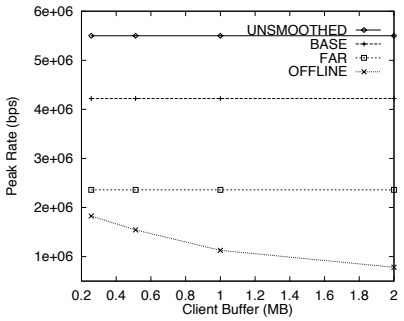
The second class of algorithms are called *sliding window* algorithms. They execute on consecutive overlapping (i.e., sliding) smoothing windows of size W. $SLWIN(\alpha)$ denotes a sliding window algorithm that slides over α frames in a video and smoothes W frames starting from that position. In this paper, we report the performance of $SLWIN(W/2)$. The sliding window algorithms are expected to perform better than the moving window algorithms for the following reason. Consider $SLWIN(\alpha)$ executing with a smoothing window size W. At time t, it smoothes a video segment from frame position p to $p + W$. 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 $(p + \alpha + W + 1)$. 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. This occurs for each and every window of data smoothed by the sliding window algorithms, and thus they are expected to perform better than the jumping window algorithms in reducing the peak rate and the rate variability in the live video. However, this improved



(a) Peak rate vs. Smoothing window size



(b) Coefficient of variation vs. Smoothing window size



(c) Peak rate vs. Client buffer size

Figure 2: Comparison between performance of different jumping window policies using *MTV* trace.

performance does come at a cost. $SLWIN(\alpha)$, operating on a smoothing window size W, smoothes each frame W/α times on the average. This effectively increases the scheduling and execution overhead of the smoothing algorithm, which could become a factor in this real-time live transmission scenario, particularly when the server is heavily loaded.

3 Trace-based Evaluation

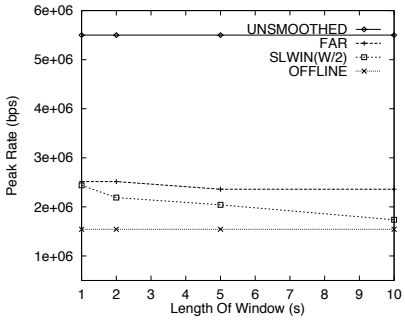
In this section, we examine the impact of online smoothing algorithms on video stream characteristics. To evaluate the performance of the smoothing algorithms, we use the metrics of peak rate, coefficient of variation (standard deviation divided by the mean rate) and effective bandwidth of the video stream. The second two metrics gives us a measure of the rate variability present in a smoothed stream. The effective bandwidth of a video trace with frame sizes $X_1, X_2, \ldots, X_N$, is computed as follows. Assume that the video is passing through a network link with buffer B and the tolerable loss rate is Γ. Defining $\theta = -\log\Gamma/B$, the effective bandwidth estimate of the video is expressed as $\log(\sum_{i=1}^{N} e^{\theta X_i}/N)/\theta$[2].

We evaluated the performance of the algorithms on MPEG-1 traces of an *MTV* (27 min. 46 sec long with mean rate 590.5 kbps) and a *CNN* (91 min 32 sec long, mean rate 1.2 Mbps) telecast. Here we present and analyze results for *MTV*. We depict characteristics of the original unsmoothed trace (labeled UNSMOOTHED in the graphs) to represent a lower bound on the performance. The performance of the optimal smoothing algorithm if it ran offline (labeled OFFLINE in the graphs), i.e., if the entire video were available to it *a priori*, represents an upper bound on the performance.

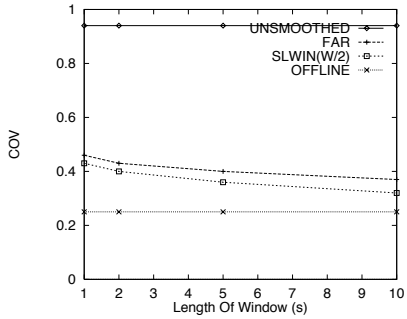
We begin by considering the results obtained by executing SLWIN(W/2) on *MTV*, transmitted with a 0.5 s playback startup latency into a 512 KB client buffer, using a 5 s smoothing window. Figure 1 demonstrates the transmission rates over the entire video for UNSMOOTHED (Figure 1(a)), FAR (Figure 1(b)), SLWIN(W/2) (Figure 1(b)), and the OFFLINE algorithms (Figure1(d)). The figures visually demonstrate the tremendous benefit of online smoothing in reducing the rate variability in the unsmoothed stream. It also shows that by using only a 5 s smoothing window, the performance of FAR and SLWIN(W/2) are very close to that of the OFFLINE algorithm.

Figure 1(e) presents the peak rate and coefficient of variation (labeled COV) for MTV, normalized w.r.t. UNSMOOTHED, using FAR, SLWIN(W/2) and OFFLINE algorithms, when smoothed into a 512 KB client buffer using a 5 s smoothing window. Again we note that all the algorithms reduce the peak rate as well as the coefficient of variation dramatically, producing a much smoother live video. The peak rate reduction is in the range of 57-62% and the coefficient of variation reduction is in the range of 57-61%.

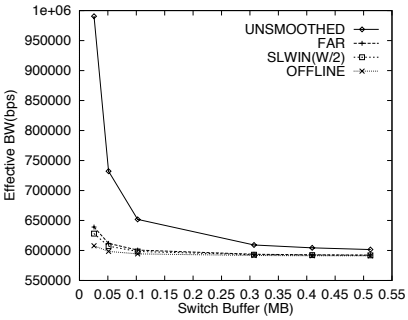
We now focus on examining the performance of the different policies across parametric variations. Focusing on the jumping window policies, Figure 2 plots the performance of BASE and FAR. Figures 2(a) and 2(b) graph the variation in peak rate and the coefficient of variation against the smoothing window size respectively. First



(a) Peak rate vs. Smoothing window size



(b) Coefficient of variation vs. Smoothing window size



(c) Effective bandwidth comparison

Figure 3: Comparison between performance of different policies.

of all, note that there is a significant difference between the lower bound (OFFLINE) and the upper bound (UNSMOOTHED) indicating the benefits of smoothing. Both the policies BASE and FAR reduce the peak rate and the coefficient of variation significantly, although FAR performs substantially better than BASE. FAR reduces the peak rate by 54% over UNSMOOTHED by using just a 1 s smoothing window, while the OFFLINE algorithm reduces it by 71%. This indicates that even very low-latency live applications can benefit significantly from online smoothing of video traffic. We also note that although the peak rate reduces with increasing smoothing window size, the reduction is small in this range. However, the coefficient of variation reduces steadily with increasing window size, indicating that although the peak rate may not reduce substantially, the overall traffic is smoother for larger smoothing windows. In fact, for a 10 s smoothing window, the coefficient of variation reduces from 0.94 in UNSMOOTHED to 0.37 in FAR. The corresponding coefficient of variation in OFFLINE is 0.25, indicating the proximity in smoothness of the FAR and OFFLINE algorithms.

Figure 2(c) plots the variation in peak rate when the client buffer is varied from 256 KB to 2 MB, for a smoothing window of 5 s. It is interesting to note that although the performance of the offline algorithm improves with increasing buffer size, the performance of the online algorithms are insensitive to the buffer size. The offline algorithm has the entire video trace available to it, enabling it to perform better smoothing into larger client buffers. But, for low-latency live video transmissions, the smoothing window size is too small for these online smoothing algorithms to take full advantage of the client buffer sizes. This suggests that for these video applications, smaller client buffers are likely to be sufficient.

Figures 3(a) and 3(b) plot the performance of the sliding-window policy SLWIN(W/2) along with FAR, the best performing jumping window policy. Figures 3(a) and 3(b) graph the variation in peak rate and the coefficient of variation against the smoothing window size respectively. In Figure 3(a), the performance of the two policies are very similar for a 1 s smoothing window, but the sliding window algorithms perform better than FAR for larger smoothing windows. This demonstrates the superiority of sliding window algorithms[2]. In fact, using just a 10 s smoothing window, the performance of SLWIN(W/2) approaches that of OFFLINE. This demonstrates that even for small latencies, the peak rate reduction of sliding window smoothing is very close to optimal. Figure 3(b) indicates that SLWIN(W/2) produce video that have lower coefficient of variation than the jumping window algorithms. The coefficient of variation decreases, i.e., the video is smoother, with increasing smoothing window sizes. As in the case of the jumping window algorithms, the performance of SLWIN(W/2) does not change with buffer sizes.

Figure 3(c) graphs the effective bandwidth estimates of FAR and SLWIN(W/2), along with UNSMOOTHED and OFFLINE, for a client buffer size of 512 KB and a startup latency of 0.5 s. The video MTV is assumed to pass through a network link and the loss rate is fixed at 0.001. Figure 3(c) plots the effective bandwidth of the algorithms as the link buffer is varied from 25.6 KB to 512 KB. As expected, the effective bandwidth of all the four curves decay exponentially with linear increase in buffer size. However, for small network buffer sizes, the

[2]The performance of SLWIN(W/2), SLWIN(W/3) and SLWIN(W/4) are very similar. We analyze only SLWIN(W/2) here due to lack of space.

effective bandwidth of FAR and SLWIN(W/2) is significantly lower than that of the UNSMOOTHED trace, a corroboration of the rate variability reduction performed by these algorithms. Once again, the performance of FAR and SLWIN(W/2) are comparable to that of OFFLINE. The decay rate of the effective bandwidth is also highest for UNSMOOTHED and the least for OFFLINE, and they all converge with increasing buffer size. Once again, this corroborates the efficacy of our online smoothing algorithms.

References

[1] ADAS, A. Supporting real time VBR video using adaptive linear prediction. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)* (San Franscisco, CA, Mar. 1996).

[2] CROSBY, S., AND ET AL. Predicting bandwidth requirements of ATM and Ethernet traffic, Feb. 1996.

[3] FENG, W.-C., AND SECHREST, S. Smoothing and buffering for delivery of prerecorded compressed video. In *IS&T/SPIE Multimedia Computing and Networking* (San Jose, CA, Feb. 1995), IS&T/SPIE, pp. 234–242.

[4] GARRETT, M., AND WILLINGER, W. Analysis, modeling and generation of self-similar VBR traffic. In *SIGCOMM Symposium on Communications Architectures and Protocols* (London, UK, Aug. 1994), ACM, pp. 269–280.

[5] HUI, J. Y. Resource allocation for broadband networks. *IEEE Journal on Selected Areas in Communications 6* (June 1988), 1598–1608.

[6] KANAKIA, H., MISHRA, P. P., AND REIBMAN, A. An Adaptive Congestion Control Scheme for Real-Time Packet Video Transport. In *ACM SIGCOMM 93 Proceedings* (Ithaca, NY, Aug. 1993), ACM, pp. 20–30.

[7] LAM, S. S., CHOW, S., AND YAU, K. Y. An algorithm for lossless smoothing of MPEG video. In *SIGCOMM Symposium on Communications Architectures and Protocols* (1994), ACM.

[8] MCMANUS, J. M., AND ROSS, K. W. Video-on-demand over ATM: Constant rate transmission and transport. *IEEE Journal on Selected Areas in Communication 14*, 6 (August 1996), 1087–1098.

[9] REIBMAN, A. R., AND BERGER, A. On VBR videoconferencing over ATM networks. In *Proc. IEEE GLOBECOMM* (1992), IEEE, pp. 314–319.

[10] REIBMAN, A. R., AND BERGER, A. Traffic descriptors for VBR video teleconferencing over ATM networks. *IEEE/ACM Transactions on Networking 3*, 3 (June 1997), 329–339.

[11] SALEHI, J., ZHANG, Z., KUROSE, J., AND TOWSLEY, D. Supporting stored video: Reducing rate variablity and end-to-end resource requirements through optimal smoothing. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems* (Philadelphia, PA, May 1996), ACM.