Online Smoothing for Delayed Transmission of Live Video

Jennifer Rexford
Networking and Distributed Systems
AT&T Labs Research; Murray Hill, NJ
http://www.research.att.com/~jrex

Wu-chi Feng
Computer and Information Sciences
Ohio State University; Columbus, OH
http://www.cis.ohio-state.edu/~wuchi

1 Introduction

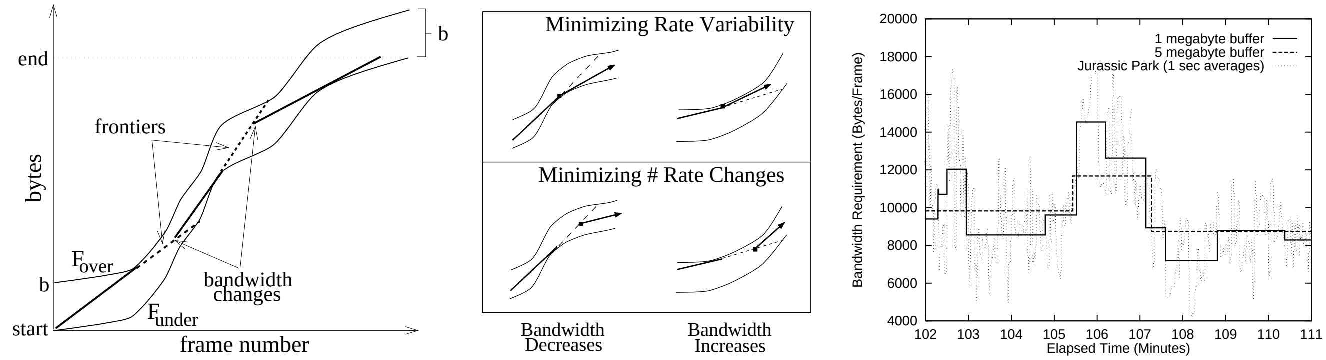
Many emerging applications rely on the efficient transmission of live or pre-recorded video. Although effective compression techniques can substantially reduce the storage and bandwidth requirements, compressed video streams typically exhibit significant burstiness that complicates the provisioning of resources along the path from the video server to the client site. To reduce the burstiness and the peak bandwidth requirements, a multimedia server can smooth the outgoing stream by transmitting a sequence of video frames at an averaged rate. Recently, several bandwidth smoothing techniques have been introduced that reduce the server and network resource requirements for transmitting pre-recorded video [1–5]. 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 [6].

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 averaged rate for a small set of frames, based on the lengths of existing frames and estimates of future frame sizes [7, 8]; 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 exchange 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, the video server 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[1]. Although these clients cannot contribute to the live stream, they enjoy a low-rate transmission without requiring the sender or the receivers to store the entire video.

Drawing on techniques for smoothing pre-recorded video, this paper presents a smoothing algorithm that can reduce the bandwidth requirements of delay-tolerant live video streams, where the server does not have complete

[1] In a similar vein, recent work considers the benefits of delaying video playback to give receivers extra time to recover lost packets by interacting with a retransmit server [9].



(a) Underflow/overflow curves (b) Selecting rate changes (c) *Jurassic Park* plans

Figure 1: Constructing server transmission plans

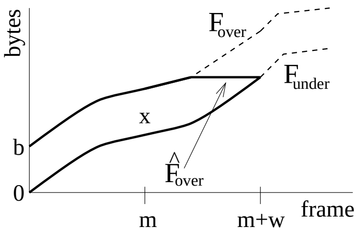
knowledge of the frame sizes. As part of developing an efficient online smoothing algorithm, Section 2 briefly reviews a framework for transmitting pre-recorded video, based on a sequence of frame lengths and the size of the prefetch buffer. Section 3 extends this model to include practical constraints on both knowledge of frame sizes and the availability of prefetch data. Simulation experiments with full-length MPEG and motion-JPEG traces demonstrate that the proposed smoothing algorithm is effective at reducing the bandwidth requirements for distributing compressed video. Section 4 concludes the paper with a discussion of future research directions.

2 Bandwidth Smoothing of 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.

Buffer underflow/overflow curves: To formalize the problem, consider a compressed video stream that consists of n frames, where frame i is f_i bytes long. To permit continuous playback at the client site, the server must always transmit quickly enough to avoid buffer underflow, where $F_{under}(k) = \sum_{i=0}^{k} f_i$ indicates the amount of data consumed at the client by frame k, where $k = 0, 1, \ldots, n-1$. Similarly, the client should not receive more than $F_{over}(k) = F_{under}(k) + b$ bytes of data by frame k, to prevent overflow of the prefetch buffer (of size b bytes). 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 $F_{under}(k) \leq \sum_{i=0}^{k} c_i \leq F_{over}(k)$. 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.

Computing bandwidth runs: For example, Figure 1(a) shows a plan with $m = 3$ runs, where the second run increases the transmission rate to avoid 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. 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)–(b). Starting each new run at the *leftmost* point along the frontier results in a transmission schedule that minimizes the *variability* in the rates $\{c_i\}$ [3]. Alternatively, a smoothing algorithm can *search* along the frontier for the starting point with the longest trajectory for the next run; this results in a transmission schedule that minimizes the *number* of rate changes [2]. Both approaches minimize the peak rate $\max_i\{c_i\}$. Figure 1(c) shows a smoothed transmission schedule for a



(a) Modified F_{under} and $\hat{F}_{over}$

```
// Project trajectories across the sliding window
c_m^min = 0; c_m^max = ∞;
for (δ = 1; δ ≤ w and c_m^min ≤ c_m^max; δ = δ + 1) {
    c_m = c_m^min;
    c_m^min = max(c_m^min, (F_under(m + δ − 1) − F_sent(m − 1))/δ);
    c_m^max = min(c_m^max, (F̂_over^m(m + δ − 1) − F_sent(m − 1))/δ);
}
```

(b) Pseudocode for window-based smoothing

Figure 2: Computing the transmission rate c_m for time slot m

motion-JPEG encoding of *Jurassic Park*[2] for two different prefetch buffer sizes.

3 Sliding-Window Smoothing Algorithm

Existing techniques for transmitting pre-recorded video serve as a useful foundation for developing new algorithms for online smoothing under client buffer constraints. However, operating with a limited knowledge of future frame sizes requires extensions to the model in Figure 1(a).

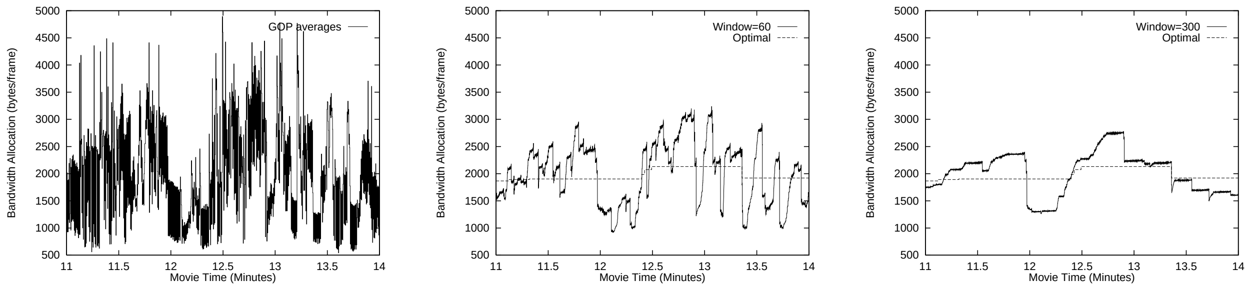
Smoothing constraints: Consider a server that transmits a "live" video stream with a start-up window of w time units; hence, the server must include sufficient buffer space to store up to w frames of the live feed. Ultimately, a larger value of w facilitates a smoother transmission of the video, at the expense of a larger buffer requirement at the server and a longer playback delay at the client(s). At transmission time slot m, the server knows the frame sizes $\{f_i\}$ up to $i = m + w - 1$. Since the server cannot transmit data beyond frame $m + w - 1$, the degree of prefetching is limited by both the client buffer size b and the window size w. In particular, at time m, the server can conservatively deduce that $F_{over}(k) \geq \hat{F}^m_{over}(k)$, where

$$\hat{F}^m_{over}(k) = \min\left(F_{under}(k) + b, \sum_{i=0}^{m+w-1} f_i\right)$$

for $k = m, m+1, \ldots, m+w-1$, where $F_{under}(k) = \sum_{i=0}^{k} f_i$. Note that the estimate of $F_{over}(k)$ gradually improves as more frames arrive; by time m, the server can transmit up to $\hat{F}^m_{over}(m)$ bytes of data. The server can use the estimate $\hat{F}_{over}$ to compute the online transmission schedule, as shown in Figure 2(a).

Selecting long trajectories: Drawing on the intuition from Section 2, the server should limit the peak rate $\max_i\{c_i\}$ by selecting trajectories that stay between the F_{under} and F_{over} curves as long as possible. However, the online algorithm only has a conservative estimate of F_{over}, making it impossible to determine an "optimal" rate for each bandwidth run. Instead, we propose a heuristic scheme for computing the transmission rate c_m for interval m, as shown in Figure 2(b). Based on the amount of data sent before time m (where $F_{sent}(m-1) = \sum_{i=0}^{m-1} c_i$), the algorithm first computes the set of trajectories $[c_m^{\min}, c_m^{\max}]$ that proceed as far as possible without crossing F_{under} or $\hat{F}_{over}$. This computation draws directly on the basic approach in [2,3] for computing minimum and maximum rates, using the modified overflow constraints and considering at most w time units in the future. If the $\hat{F}_{over}$ curve encounters the F_{under} curve, as in Figure 2(a), the rates $c_m^{\min}$ and $c_m^{\max}$ converge to a single trajectory by the end of the time window. Otherwise, if multiple trajectories can proceed beyond w time units, the algorithm in Figure 2(b) selects the smallest rate $c_m^{\min}$, in the hope of limiting the peak rate and the utilization of the prefetch buffer; more generally, the server could compute the rate c_m as a weighted average of $c_m^{\min}$ and $c_m^{\max}$. Note that,

[2] Our library of 20 constant-quality, motion-JPEG video traces is publicly available at `http://www.cis.ohio-state.edu/~wuchi`. We are in the process of expanding this library to include a collection of MPEG traces.



(a) GOP frame sizes (b) 2-second smoothing (c) 10-second smoothing

Figure 3: Sample bandwidth smoothing plans for the *Star Wars* MPEG video

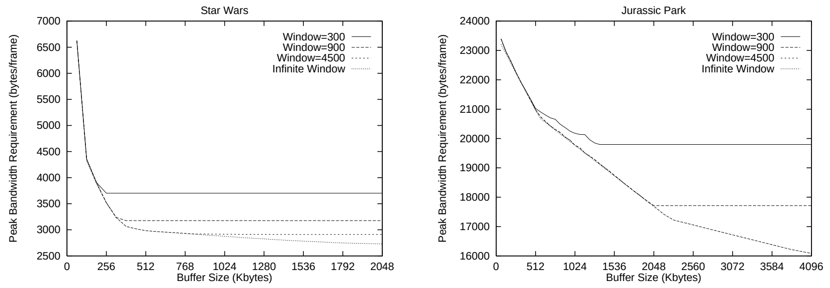
unlike the schemes in Section 2, the algorithm in Figure 2(b) computes a new transmission rate in each time unit m, to draw on any new information about frame sizes. In the worst case, this incurs $O(w)$ work in each time unit; future research can investigate ways to reduce this algorithmic complexity.

Sample bandwidth plans: Figure 3 demonstrates the benefits of smoothing an MPEG encoding of the movie *Star Wars* [10], which has a maximum frame size of 23,160 bytes and an mean frame size of 1950 bytes. As shown in Figure 3(a), the *Star Wars* video exhibits significant burstiness even after averaging across the 6-frame group of pictures (GOP) for the MPEG stream; averaging over this small time scale reduces the peak rate from 23160 bytes/frame-slot to 10670 bytes/frame-slot. In contrast, using the algorithm in Figure 2(b), with a 2-second (60-frame) window and a 0.5-megabyte client buffer, reduces the peak bandwidth to 5800 bytes/frame-slot. Figure 3(b) shows part of the resulting bandwidth plan, compared to an optimal algorithm [3] that has complete knowledge of the frame sizes and unlimited prefetching within the 0.5-megabyte buffer constraint. Figure 3(c) shows the plan for a 10-second (300-frame) window, which reduces the peak rate to 3700 bytes/frame-slot. With complete knowledge of the video stream, an optimal algorithm could reduce the peak rate to 2980 bytes/frame-slot.

Effects of window size w: Figure 4 compares the online algorithm (with w of 10 seconds, 30 seconds, and 2.5 minutes) to an optimal offline algorithm. The experiments employ an MPEG encoding of *Star Wars* and a motion-JPEG encoding of *Jurassic Park* to highlight the interplay between the window size w and the buffer size b. The online algorithm performs extremely well for small values of b, when the size of the client buffer imposes the main constraint on prefetching. To remove burstiness on a larger time scale, the window w should grow along with the buffer size b, based on a rough estimate of the frame lengths in the video stream. When w approaches the length n of the video stream, the algorithm has full knowledge of the frame sizes and can accurately compute the longest trajectory at time m based on $F_{sent}(m-1)$; as a result, for large w, the algorithm converges to an optimal transmission schedule for pre-recorded video [3]. Still, increasing the buffer size b eventually offers diminishing returns, even for the optimal algorithm. As a result, a window-based scheme can achieve most of the reduction in the peak bandwidth, if the application can tolerate even a few tens of seconds of playback delay.

4 Conclusions

By delaying the transmission of video frames by w time units, the sliding-window smoothing algorithm can substantially reduce the bandwidth requirements for distributing live video streams by "prefetching" frames into the b-byte client playback buffer. The online algorithm extends previous work on techniques for smoothing pre-recorded video by limiting the knowledge of frame sizes and the availability of prefetch data to w frames beyond the current time. Simulation experiments show that the algorithm, 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 algorithm in Figure 2 can further improve performance by using more optimistic estimates of the overflow curve



(a) *Star Wars* (MPEG) (b) *Jurassic Park* (motion-JPEG)

Figure 4: Peak rate $\max_i\{c_i\}$ vs. buffer size b for different window sizes w

$\hat{F}_{over}$ 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 [7].

As ongoing work, we are considering ways to combine the online smoothing algorithm 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 rate control algorithms, layered encoding schemes, playout delay adjustment policies, and packet retransmission protocols. To apply our results, we are investigating the practical issues of incorporating window-based smoothing in multicast video and audio services. Finally, performance evaluation with traces of live-video source would lend greater insight into the relationship between the smoothing delay w and the reduction in the peak bandwidth, as well as the potential for accurately estimating future frame sizes. These experiments can help guide the sizing of the smoothing window and the server/client buffers, to smooth the video stream under modest buffer sizes and a reasonable playback delay.

References

[1] 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.

[2] 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*.

[3] 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.

[4] 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.

[5] 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.

[6] W. Feng and J. Rexford, "A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video." To appear in *Proc. IEEE INFOCOM*, April 1997.

[7] 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.

[8] 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.

[9] N. F. Maxemchuk, K. Padmanabhan, and S. Lo, "A cooperative packet recovery protocol for multicast video." To appear in *IEEE Internet Computing Magazine*, 1997.

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