A Comparison of Bandwidth Smoothing Techniques for the Transmission of Prerecorded Compressed Video

Wu-chi Feng
Dept. of Comp. and Info. Sci.
The Ohio State University
Columbus, OH 43210
wuchi@cis.ohio-state.edu

Jennifer Rexford
Networking and Distributed Systems
AT&T Labs Research
Murray Hill, NJ 07974
jrex@research.att.com

Abstract

The transfer of prerecorded, compressed video requires multimedia services to support large fluctuations in bandwidth requirements on multiple time scales. Bandwidth smoothing techniques can reduce the burstiness of a variable-bit-rate stream by prefetching data at a series of fixed rates, simplifying the allocation of resources in video servers and the communication network. Given a fixed client-side prefetch buffer, several bandwidth smoothing algorithms have been introduced that are provably optimal under certain constraints. This paper presents a collection of metrics for comparing these smoothing algorithms and evaluating their cost-performance trade-offs. Due to the scarcity of available trace data, we have constructed a video capture testbed and generated a collection of twenty full-length, motion-JPEG encoded video clips. Using these video traces and a range of client buffer sizes, we investigate the interplay between the performance metrics through simulation experiments. The results highlight the unique strengths and weaknesses of each algorithm.

1. Introduction

Many emerging multimedia applications, such as digital libraries and video-on-demand services, rely on the efficient transfer of prerecorded video. Effective compression techniques, such as MPEG [1] and motion-JPEG, can substantially reduce the resource requirements for storing and transmitting video streams. However, compressed video traffic typically exhibits significant burstiness on multiple time scales, due to the frame structure of the compression algorithm as well as natural variations within and between scenes[2-4]. This bursty, variable-bit-rate traffic complicates the effort to allocate server and network resources to ensure continuous playback at client sites, such as workstations and set-top boxes. To reduce the burstiness of the traffic, stored-video applications capitalize on the *a priori* knowledge of the frame sizes in the compressed video.

In particular, the server can smooth the stream by *prefetching* video frames in advance of each burst. By initiating transmission early, the server can send large frames at a slower rate without disrupting the client application. The client system can then retrieve, decode, and display frames at the stream frame rate. The potential benefit of prefetching depends on the size b of the client buffer. The server must limit the amount of prefetching to prevent overflow of this buffer; however, to avoid underflow, the server should transmit enough data to allow the client to drain its buffer at the frame display rate. The server can reduce the burstiness of prerecorded video, while avoiding both underflow and overflow of the client buffer, by employing a *bandwidth smoothing algorithm* [5-9], as discussed in Section 2.

Based on the frame lengths f_i and the buffer size b, these algorithms generate a *transmission plan* that consists of m fixed-rate *runs*. The lengths of these runs depend on the client buffer size and the degree of burstiness in the underlying video stream. For a reasonable client buffer size, bandwidth smoothing can create a transmission plan with a fairly small number of runs and a significant reduction in the stream's peak bandwidth requirement compared to an unsmoothed transfer of the video. As a result, bandwidth smoothing has the potential to greatly decrease the server and network resources required to transport prerecorded video, without introducing playback delay at the client.

This paper presents a comprehensive comparison of bandwidth smoothing algorithms, based on a collection of performance metrics that relate directly to the complexity of the transmission, transport, and playback of compressed, prerecorded video. The existing bandwidth smoothing algorithms produce transmission plans with different lengths m and different rate requirements in each run, depending on what metrics they attempt to optimize. Specifically, we compare four algorithms that create plans that minimize the number of bandwidth increases [5,6], minimize the total number of bandwidth changes [7], minimize the variability of the bandwidth requirements [8], or consist of periodic bandwidth runs [9]. The following section describes these bandwidth smoothing algorithms, with an emphasis on how they strive to optimize these specific metrics.

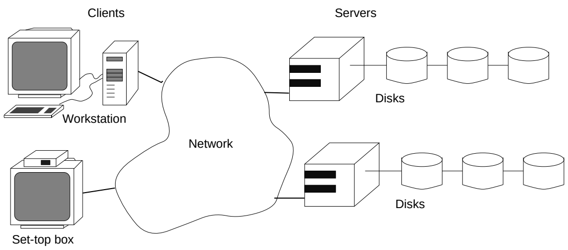


Figure 1: Video-on-Demand Architecture

For an extensive performance comparison, we have generated a library of twenty full-length, motion-JPEG encoded video clips, digitized using a PC-based video capture testbed, as described in Section 3. Based on these compressed video streams, Section 4 compares the smoothing algorithms and investigates the subtle interplay between the performance metrics. In addition to evaluating the bandwidth smoothing algorithms, these experiments also highlight the unique properties of the underlying video clips; in particular, the smoothed video streams show significant differences in the peak-rate requirements and bandwidth variability. These experiments motivate several possible directions for future research on the efficient transmission of prerecorded video, as discussed in Section 5.

2. Bandwidth smoothing

Bandwidth smoothing can reduce the burstiness of compressed video traffic in a video-on-demand architecture, shown in Figure 1. Video servers typically store prerecorded video on large, fast disks [10-13] and may also include tertiary storage, such as tapes or optical jukeboxes, for holding less frequently requested data [14]. A network connects the video servers to the client sites through one or more communication links; the network can help ensure continuous delivery of the smoothed video data by including support for rate or delay guarantees [15,16], based on resource reservation requests from the video server. The client sites, such as workstations or set-top boxes, include a buffer for storing prefetched video frames; in addition, the client may interact with the server to compute video transmission plans, based on the buffer size and frame lengths.

2.1. Bandwidth plans

A compressed video stream consists of n frames, where frame i requires f_i bytes of storage. To permit continuous playback, 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 $k = 0,1, ..., n\text{-}1$. Similarly, the client should not receive more data than

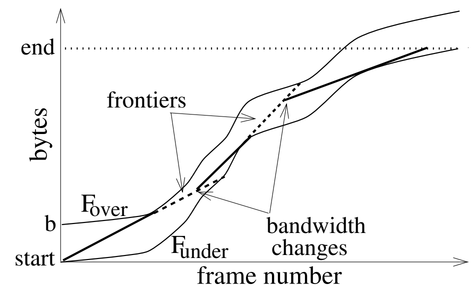


Figure 2: Sample Transmission Plan

$$F_{over}(k) = b + \sum_{i=0}^{k} f_i$$

by frame k, to prevent overflow of the playback buffer (of size b). Consequently, any valid server transmission plan should stay within the river outlined by these vertically equidistant functions, as shown in Figure 2. That is,

$$F_{under}(k) \le \sum_{i=0}^{k} c_i \le F_{over}(k)$$

where c_i is the transmission rate during frame slot i of the smoothed video stream.

Creating a bandwidth plan involves generating m consecutive *runs* each with a constant bandwidth allocation r_j and a duration t_j, where time is measured in discrete frame slots; at time i the server transmits at rate $c_i = r_j$, where slot i occurs during run j. Together, the m bandwidth runs must form a monotonically-increasing, piecewise-linear path that stays between the F_{under} and F_{over} curves. For example, Figure 2 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. Bandwidth smoothing algorithms typically select the starting point for run $j+1$ based on the trajectory for run j. By extending the fixed-rate line for run j, the trajectory eventually encounters either the underflow or the overflow curve, or both, requiring a change in the server transmission rate.

2.2. Smoothing algorithms

Given a starting point for run $j+1$, most smoothing algorithms attempt to select a trajectory that extends as far as possible, to limit the number of bandwidth changes during the remainder of the plan. As a result, the trajectory for each run must eventually reach both the overflow and the underflow curves, generating a *frontier* of possible starting points for the next run, as shown in Figure 2. The various bandwidth smoothing algorithms differ in how they select a starting point for run $j+1$ on rate increases and decreases, resulting in transmission plans with different performance

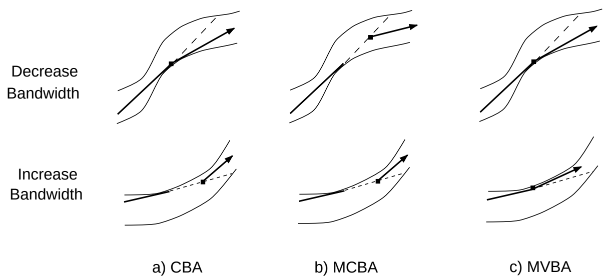


Figure 3: Run Boundary Examples

properties. For example, the critical bandwidth allocation (CBA) algorithm [5,6] starts a rate decrease at the leftmost point on the frontier, where the trajectory for run *j* hits the F_{under} curve; for rate increases, the CBA algorithm performs a search along the frontier to locate the starting point that allows the next trajectory to extend as far as possible, as shown in Figure 3.

For any rate change, the CBA algorithm determines the longest trajectory for run *j+1*, based on the selected starting point and initial buffer occupancy (vertical distance from F_{under}). This results in a transmission plan that has the smallest possible peak bandwidth requirement (minimizes $\max_j\{r_j\}$) and the minimum number of bandwidth *increases* [6]. A CBA plan does not necessarily have the minimum number of bandwidth *decreases*, since the algorithm always selects the leftmost starting point, independent of shape of the underflow and overflow curves. To minimize the number of rate decreases, the *minimum changes bandwidth allocation* (MCBA) algorithm [7] extends the CBA scheme to perform the linear search operation on *all* rate changes. This results in a transmission plan with the smallest possible number of rate changes (minimizes *m*) and the minimum peak bandwidth requirement.

Instead of minimizing *m*, a bandwidth smoothing algorithm can strive to reduce the variability in the rate requirements across the lifetime of the transmission plan [8]; for the remainder of the paper, we refer to this approach as the *minimum variability bandwidth allocation* (MVBA) algorithm. To adjust to variations in the underlying video stream, the MVBA algorithm initiates bandwidth changes at the leftmost point along the frontier, for both rate increases and rate decreases. As a result, an MVBA transmission plan *gradually* alters the stream's rate requirement, sometimes at the expense of a larger number of small bandwidth changes, as shown in Figure 5. The MVBA and CBA algorithms handle bandwidth decreases in the same manner, while an CBA plan more closely resembles an MCBA plan on rate increases. The worst case complexity of the MVBA algorithm is $O(n^2)$ compared to $O(n^2 \log n)$ for the MCBA and CBA algorithms, which perform a binary search along the frontier.

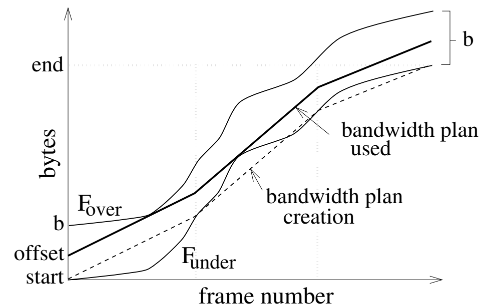


Figure 4: PCRTT Plan Creation

In contrast to the CBA, MCBA, and MVBA algorithms, the *piecewise constant rate transmission and transport* (PCRTT) algorithm [9] creates bandwidth allocation plans by dividing the video stream into fixed-size intervals. The algorithm generates a single run for each interval by connecting the intersection points on the F_{under} curve, as shown in Figure 4; the slopes of these lines correspond to the rates r_j in the resulting transmission plan. To avoid buffer underflow, the PCRTT scheme vertically offsets this plan until all of the runs lie above the F_{under} curve. Raising the plan corresponds to introducing an initial playback delay at the client; the resulting transmission curve also determines the minimum acceptable buffer size to avoid overflow given the interval size, as shown in Figure 4. Because the PCRTT algorithm determines the buffer size from the bandwidth plan, it can be difficult to calculate an appropriate interval size given a fixed buffer *b*.

3. Compressed video streams

An effective comparison of bandwidth smoothing algorithms requires a large collection of video trace data to represent the diverse traffic in emerging multimedia services. However, most studies of video transmission techniques evaluate a small number of compressed video clips, due to the scarcity of publicly-available traces. To facilitate a more comprehensive evaluation of the smoothing algorithms, we have generated a library of full-length, motion-JPEG video traces[1] using a PC-based video-capture testbed [17].

3.1. Video capture testbed

Efficiently gathering large amounts of video trace data requires hardware support for video compression. The video capture testbed consists of a Pioneer Laser Disc player, a MiroVideo DC1tv capture board, and a Pentium-processor with 32 megabytes of memory. The capture board is a motion-JPEG (MJPEG) compression board containing the C-Cube Microsystems' MJPEG chip, the CL550. The

1. A world-wide-web page for this trace data is currently available at http://www.cis.ohio-state.edu/~wuchi

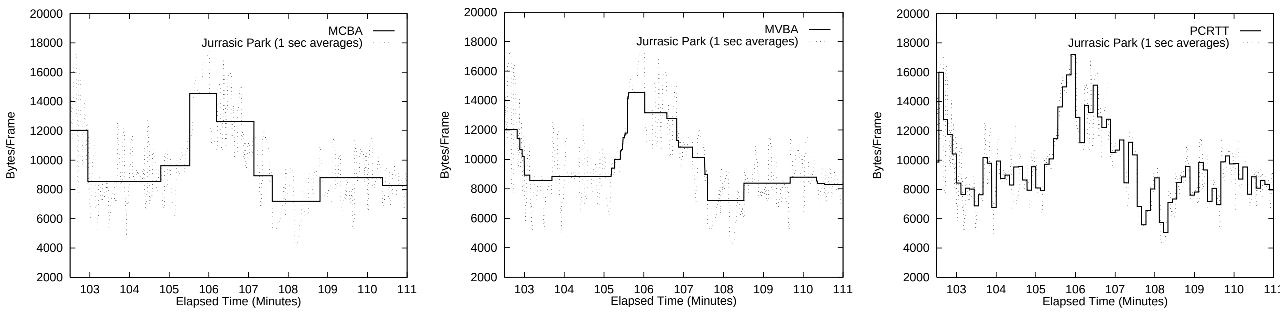


Figure 5: Bandwidth Plan Examples

MJPEG algorithm compresses each video frame using the JPEG still-picture standard [18], resulting in a *constant* quality, variable-bit-rate video stream. By setting options on this board, the testbed can vary the quality of the video encoding. In particular, the overall picture quality can be changed by scaling the quantization levels for the frequency components produced by the DCT (discrete cosine transform) in the JPEG encoding algorithm.

Since MJPEG compresses each video frame independently, the traces do not capture the effects of inter-frame dependencies that would exist in MPEG-encoded streams. For a typical video source, a MPEG encoding would have smaller average frame sizes and larger short-term burstiness, due the mixture of interpolated (I), predictive (P), and bidirectional (B) frames [1]. A video server could limit the effects of this short-term variation through modest prefetching into a small client buffer. Consequently, the relative performance of bandwidth smoothing algorithms is more sensitive to the medium-term and long-term burstiness in the underlying video stream, particularly for a larger client buffer. Since a real-time MPEG encoder would not significantly affect the performance trends, except perhaps under small buffer sizes, the testbed employs the (order of magnitude) cheaper hardware that implements MJPEG.

3.2. Video library

Using this PC-based testbed, we have constructed a video library with twenty video clips, consisting of 31 hours of video and a total of 38.5 gigabytes of motion-JPEG compressed data. A separate script post-processes each video clip to generate a sequence of frame sizes, which drive the simulation experiments in Section 4. The video library includes clips with different lengths and subject matters, to represent the diversity of compressed video sources in emerging multimedia services. For example, the *Beauty and the Beast* video is an animated Walt Disney film that has scenes with a large number of high-frequency components as well as scenes with large areas of constant color. The rest of the movies are a mixture of conventional entertainment with a wide range of scene content, including digital effects and animations.

The library includes three versions of the film *E.T. - The Extra Terrestrial* with different quantization levels; the quality factors of 75, 90 and 100 correspond to 0.66, 0.94, and 1.64 bits-per-pixel, respectively, in the compressed video stream. With a coarser representation of the frequency components, the video stream has smaller average and peak frame sizes, resulting in a lower bandwidth requirement for transmitting and receiving the movie; for example, the movie requires 3.78 megabits/second for a quality factor of 100, whereas 1.51 megabits/seconds suffices for a quality level of 75. The three versions of *E.T.* permit the simulation experiments in Section 4 to study the effectiveness of bandwidth smoothing techniques as a function of video quality. The remainder of the video traces in Figure 6 have 0.94 bits-per-pixel, which corresponds to "excellent" picture quality [18].

To broaden the collection of traces, the library also includes a handful of sources from video cassette tapes. The *NCAA Final Four* video is a documentary describing the 1993 NCAA Final Four basketball tournament, resulting in many scenes with a large amount of detail. As a result, this trace has a higher average bit rate than the other sources, by a fairly significant margin. In addition, the library includes three seminars to study the effects of compression and bandwidth smoothing on "educational" video. These presentations were filmed with a single, stationary camera focusing on the screen for displaying the speaker's transparencies. This results in small bandwidth requirements and low variation in the frame sizes relative to the other videos. Plots for the other video traces are shown in reference [17].

4. Performance comparison

Using the traces from the video library, this section compares bandwidth smoothing algorithms based on a collection of performance metrics. These metrics include the peak rate requirements, the number of bandwidth changes, the variability of the bandwidth allocations, and the variability of the time between bandwidth changes. By applying these metrics to server transmission plans, across a range of realistic client buffer sizes, the simulation experi-

	Resources			Frame Sizes			
	Size (GB)	Time (min)	Rate (Mbps)	Avg (bytes)	Max (bytes)	Min (bytes)	Std (bytes)
Beauty and the Beast	1.82	80	3.04	12661	30367	2701	3580
Big	2.26	102	2.96	12346	23485	1503	2366
Crocodile Dundee	1.82	94	2.59	10773	19439	1263	2336
E.T. (quality 75)	1.24	110	1.51	6305	14269	1511	1840
E.T. (quality 90)	1.78	110	2.17	9022	19961	2333	2574
E.T. (quality 100)	3.11	110	3.78	15749	30553	6827	3294
Home Alone 2	2.35	115	2.73	11383	22009	3583	2480
Honey, I Blew Up the Kid	2.12	85	3.32	13836	23291	3789	3183
Hot Shots 2	1.92	84	3.06	12766	29933	3379	3240
Jurassic Park	2.50	122	2.73	11363	23883	1267	3252
Junior	2.71	107	3.36	14013	25119	1197	3188
NCAA Final Four	1.21	41	3.95	16456	29565	2565	4138
Rookie of the Year	2.22	99	2.98	12435	27877	3531	2731
Seminar1	0.98	63	2.07	8604	10977	7181	592
Seminar2	1.08	68	2.12	8835	12309	1103	608
Seminar3	0.88	52	2.26	9426	11167	7152	690
Sister Act	2.06	96	2.86	11902	24907	1457	2608
Sleepless in Seattle	1.72	101	2.28	9477	16617	3207	2459
Speed	2.46	110	2.97	12374	29485	2741	2707
Total Recall	2.34	109	2.88	11978	24769	2741	2692

Figure 6: Video Movie Library Statistics

ments show cost-performance trends that affect the transmission, transport, and playback of compressed video. A more in-depth discussion of these algorithms can be found in [19], including additional results using the CBA algorithm, which typically generate results that fall between the MCBA and MVBA algorithms.

4.1. Peak bandwidth requirement

The peak rate of a smoothed video stream determines the worst-case bandwidth requirement across the path from the video storage on the server, the route through the network, and the prefetch buffer at the client site. Hence, most bandwidth smoothing algorithms attempt to minimize

$$\max\{r_j \mid j = 0,1, ..., m - 1\}$$

to increase the likelihood that the server, network, and client have sufficient resources to handle the stream; in addition, a low peak rate may reduce the total cost of the data transfer. Figure 7 plots the peak bandwidth requirements for the four smoothing algorithms, as a function of the client buffer size. The CBA, MCBA, and MVBA algorithms all result in the minimum peak bandwidth requirement given a fixed buffer size, as discussed in Section 2. For a fair comparison with the PCRTT algorithm, we permit all four algorithms to prefetch data on the first bandwidth run, which results in an initial playback delay at the client site.

Under small buffer sizes, the movies with the largest variations in frame sizes also tend to have the largest peak bandwidth requirements, due to the limited ability to smooth large peaks. In all four plots, the *Beauty and the Beast*, *E.T. (quality 100)*, and *NCAA Final Four* videos are the top three curves, while the *Seminar* videos have the lowest peak bandwidth requirements for buffer sizes less than one megabyte. For the three *E.T.* videos, the lower-quality encodings have lower peak rate requirements, due to the smaller frame sizes at each point in the video. In fact, under larger buffer sizes, the *E.T. (quality 75)* video actually has a *lower* peak bandwidth than the *Seminar* videos. For large buffers, prefetching removes nearly all of the burstiness in the stream, yielding a plan that stays very close to the mean frame size of 6305 bytes; the *Seminar* videos, digitized with a 90 quality factor, have larger average frame sizes. Thus, for small buffer sizes, the peak bandwidth rate is generally driven by the *maximum* frame sizes, while for larger buffer sizes, the peak rate is driven mostly by the *average* frame size and the long-term burstiness.

The CBA, MCBA, and MVBA algorithms have monotonically decreasing peak bandwidth requirements as the buffer size grows; the PCRTT plots are somewhat unstable, since the algorithm smooths bandwidth requirements based on fixed-sized intervals. To generate the plot for the PCRTT algorithm, a script creates bandwidth plans for a sequence of different interval sizes; based on the plan for each interval size, the script computes both the peak bandwidth requirement and the client buffer size b to create a single data point in the plot. Under medium-sized buffers, the CBA, MCBA, and MVBA plans have smaller peak rates due to their ability to prefetch data as far in advance as the

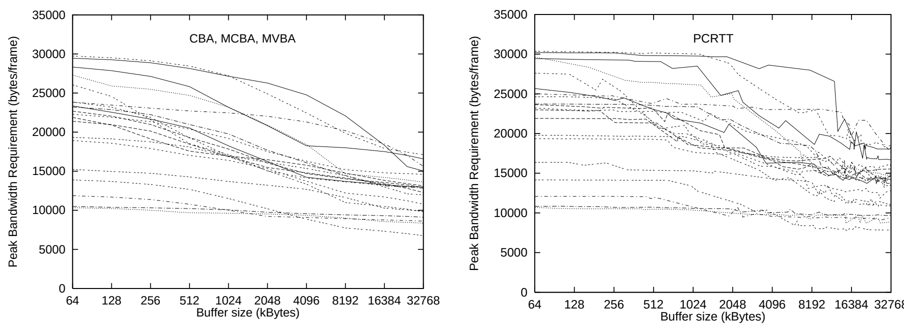


Figure 7: Peak Bandwidth Requirements

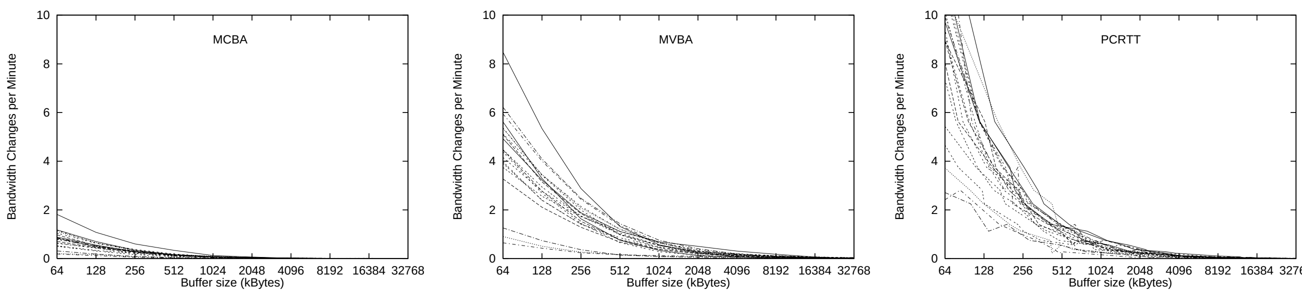


Figure 8: Rate of Bandwidth Changes

buffer will allow, while the PCRTT is limited by the interval size. Still, for most buffer sizes, the PCRTT plans do not have significantly larger peak requirements, suggesting that the algorithm makes fairly good use of the smoothing buffer. The PCRTT algorithm has the most difficulty with video clips that have areas of sustained large frames followed by areas of small frames (or vice-versa), which require small interval sizes to avoid buffer overflow and underflow; these small intervals limit the algorithm's ability to employ prefetching to smooth the large frames in the underlying compressed video stream.

4.2. Number of bandwidth changes

To reduce the complexity of the server and client sites, a bandwidth smoothing algorithm could strive to minimize m, the number of runs in the transmission schedule [7], since each bandwidth change requires the sites to modify the I/O rate for the stream, allowing server and network resources to be allocated on the order of minutes instead of by frame or several frames. Minimizing the number of rate changes can also limit the cost of negotiating with the network [4] to reserve link bandwidth for transporting the video stream[2]. Figure 8 compares the CBA, MCBA, MVBA, and PCRTT algorithms based on the number of bandwidth changes in the server transmission plan for each of the clips in the video library. Since the video clips have different lengths, varying from 41 minutes to 122 minutes, the graphs plot the *rate* of bandwidth changes

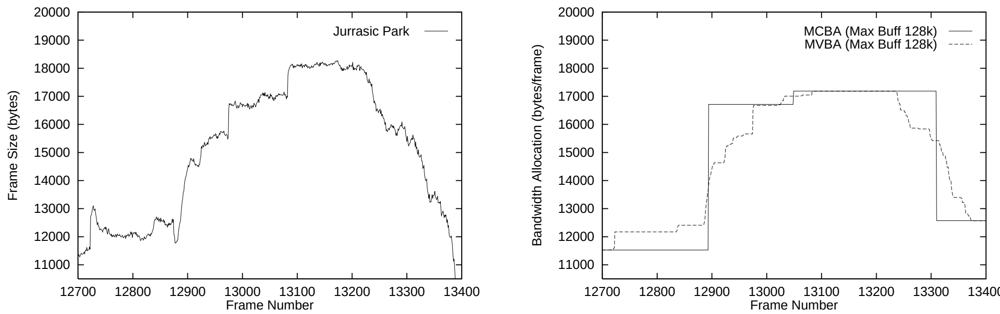
$$\frac{m}{\sum_{j=0}^{m-1} t_j}$$

across a range of client buffer sizes.

For all of the smoothing algorithms and traffic patterns, the client prefetch buffer is effective in reducing the frequency of rate change operations. In each graph, the bottom three curves correspond to the *Seminar* videos, which do not require many rate changes due to their small frame sizes and the low variability in their bandwidth requirements. *The NCAA Final Four* video requires the highest rate of bandwidth changes, due to the large frame sizes and long-term variations in scene content; for a 64 kilobyte buffer, this stream requires an average of 1.8, 4.9, 8.5, and 16.8 rate changes per minute under the MCBA, CBA, MVBA, and PCRTT plans, respectively. Across all buffer sizes, the *E.T. (quality 100)* video has more rate changes than the version with a quality factor of 90, which had more changes than the encoding with a quality factor of 75; still, the three versions of *E.T.* have bandwidth plans with a similar shape, due to the same long-term burstiness from scene content.

In Figure 8, the MCBA graph shows the smallest rate of bandwidth changes across all of the movies, since the MCBA algorithm minimizes m; the CBA, MVBA, and

2. To further reduce interaction with the network, each video stream could have a separate *reservation plan* for allocating network resources along the route to the client. This reservation plan could have fewer rate changes than the underlying transmission plan, at the expense of reserving excess link bandwidth[4,8].



(a) Segment of *Jurassic Park* (b) MCBA and MVBA Plans

Figure 9: MCBA and MVBA Comparison

PCRTT plans result in approximately 3, 6, and 8 times as many changes, respectively. For some movies and buffer sizes, the MVBA plans have up to 14 times as many bandwidth changes as the corresponding MCBA plans. This occurs because the MVBA algorithm introduces a larger number of small rate changes to minimize the variability of bandwidth requirements in the server transmission plan. As an example, Figure 9(a) shows a short segment (in frames) from the movie *Jurassic Park*, while Figure 9(b) shows the corresponding portion of the MCBA and MVBA plans for a 128 kilobyte client buffer. The MVBA algorithm introduces 104 rate changes (55 increases and 49 decreases), while the MCBA plan has just three bandwidth changes.

A closer look at Figure 9(a) suggests a possible explanation for this phenomenon. During the first 400 frames of the video segment, the frame sizes gradually increase over time, from a size of 11273 bytes to 18133 bytes. On this long stretch of increasing bandwidth requirements, the MVBA algorithm tends to follow the "curve" of the increase by generating a sequence of small rate increases. A similar effect occurs during the gradual decreases in frame sizes for the remainder of the *Jurassic Park* segment. In Figure 9(b), note that the area between the two plans, in the range of frames 12720 to 12900, is approximately equal to the size of the smoothing buffer. This suggests that the MVBA plan has filled the client buffer, requiring a more gradual response to the rate increases in the video segment; in contrast, the MCBA plan has a nearly empty buffer, giving the algorithm greater latitude in adjusting the server transmission rate[3].

The PCRTT plan results in more bandwidth changes than the CBA and MCBA plans for all buffer sizes. Under moderate buffer sizes, however, the PCRTT algorithm sometimes generates fewer rate changes than the MVBA plan. For these buffer sizes, the PCRTT algorithm is effective at combining several bandwidth runs of the MVBA algorithm into a single rate interval. The PCRTT algorithm also results in very few changes for the *Seminar* videos because the transmission plan can employ a large interval size without creating large buffer deficits.

3. Referring to Figure 3, this case is a situation where the MCBA algorithm selects a starting point at the *rightmost* point along the frontier whereas the MVBA algorithm selects the *leftmost* point.

4.3. Variability in bandwidth requirements

In transmitting a smoothed video stream, the ability to change the bandwidth requirement may depend on the *size* of the rate adjustment ($|r_{j+1} - r_j|$), particularly on rate *increases*. If the system does not support advance booking of resources, the server or the network may not be able to acquire enough bandwidth to start transmitting frames at the higher rate[4]. To improve the likelihood of successful rate-change operations, the smoothing algorithm can minimize the *variability* of the bandwidth requirements in each time slot (standard deviation of $\{c_0, c_1, ..., c_{n-1}\}$), resulting in a smoother plan [8]. Since the video clips have different average rate requirements, varying from 1.51 to 3.95 megabits/seconds, Figure 10 plots the *coefficient of variation*

$$\frac{\mathrm{stdev}\,\{c_0, c_1, \ldots, c_{n-1}\}}{\frac{1}{n}\sum_{i=0}^{n-1} c_i}$$

to normalize the variability metric across different streams.

In Figure 10, the MVBA plans have the smallest variability in bandwidth allocations, since the algorithm optimizes this metric. Since the CBA algorithm is the same as the MVBA algorithm for bandwidth decreases, it has nearly the same variability across the various videos. For small buffer sizes, the MCBA algorithm has nearly the same variability in bandwidth requests since all four algorithms have to perform rate changes on a small time scale; for larger buffer sizes, the MCBA algorithm has more latitude in

4. If the system cannot reserve resources for the higher bandwidth r_{j+1}, the video stream may have to adapt to a smaller rate to avoid terminating the remainder of the transfer. For example, with a *layered* encoding of the video stream, the server could reduce the transmission rate by sending only the higher priority components of the stream [20,21]. To limit the degradation in video quality at the client site, the server can raise the stream's rate as close to r_{j+1} as possible.

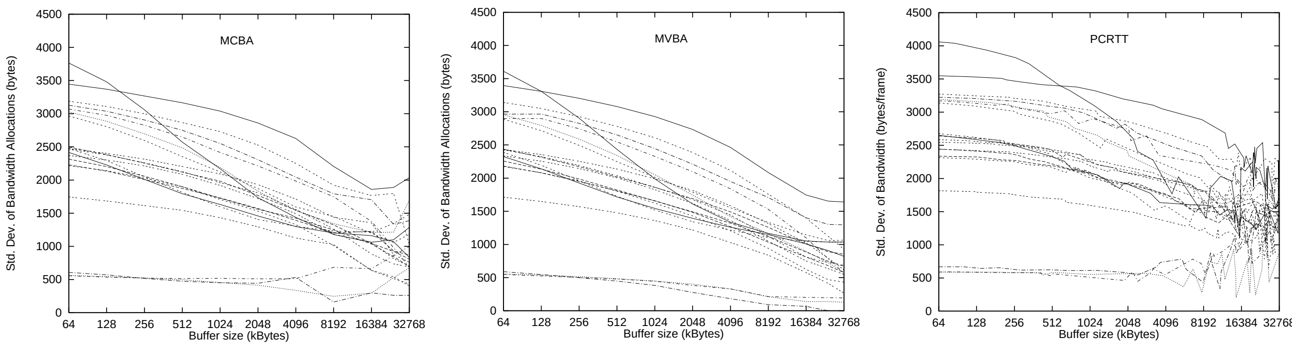


Figure 10: Bandwidth Variability

combining bandwidth requests, resulting in greater rate variability than MVBA and CBA. The PCRTT algorithm had the largest variability in bandwidth allocations. Because the PCRTT algorithm smooths bandwidth requests based on fixed interval lengths, it cannot smooth burst of large frames beyond the size of the interval, resulting in higher peaks and lower valleys. As with the other metrics, the PCRTT algorithm is somewhat unstable because of the partitioning of frames at fixed intervals. In particular, as buffer sizes get larger, the partitioning of the frames into fixed intervals plays a heavy role in the minimum amount of buffering required to have continuous playback.

4.4. Periodicity of Bandwidth Requests

To efficiently handle a large number of smoothed video streams, a server may need to limit the *total* number of rate change operations that can occur during an interval of time. As a result, a bandwidth smoothing algorithm may try to enforce a lower bound on the minimum time between rate changes for each stream to improve the scalability of the server architecture. In addition, for systems that have in-advance reservations, having periodic bandwidth allocation intervals simplifies the reservation algorithms. Since the smoothing algorithms have different average run lengths, we measure

$$\frac{\text{stdev}\,\{t_0, t_1, \ldots, t_{m-1}\}}{\frac{1}{m}\sum_{i=0}^{m-1} t_i}$$

to compare the "periodicity" of bandwidth requests. Figure 11 plots this normalized metric for the four algorithms as a function of the client buffer size.

Since PCRTT plans consist of fixed-size intervals, the coefficient of variation is *0* for all of the video streams; in addition, the PCRTT algorithm can produce plans with a meaningful lower bound on the minimum time between rate changes ($\min_j\{t_j\}$). In contrast, the CBA, MCBA, and MVBA plans have a wide variation in run lengths; for most buffer sizes less than *2* megabytes, $\min_j\{t_j\}$ is just one or two frame slots. For the most part, the MCBA algorithm results in less variability than the CBA and MVBA algorithms, which can introduce a large number of small bandwidth changes, even under large buffer sizes. Under small buffer sizes, the top three curves for the MCBA algorithm correspond to the *Seminar* videos. These videos have a small number of short frames, followed by a large number of long frames; under a small prefetch buffer, the server cannot continually transmit frames at a high rate when the client is consuming the short frames. As a result, the bandwidth plan consists of a mixture of short and long run lengths, causing greater variability than the other videos; with larger buffers, the plans become more "periodic."

5. Conclusions and future work

In this paper, we have presented a comprehensive comparison of bandwidth smoothing algorithms for the delivery of compressed video streams in video-on-demand systems. By capitalizing on the *a priori* knowledge of frame lengths, these algorithms can significantly reduce the burstiness of resource requirements for the transmission, transfer, and playback of prerecorded video. For small buffer sizes, the PCRTT algorithm is useful in creating plans that have near optimal peak bandwidth requirements while requiring very little computation time to calculate. For larger buffer sizes, however, the PCRTT algorithm limits the ability of the server to prefetch frames across interval boundaries. The CBA, MCBA, and MVBA algorithms exhibit similar performance for the peak rate requirement and the variability of bandwidth rates. The MCBA algorithm, however, is more effective at reducing the total number of rate changes.

Future work can consider new smoothing algorithms that enforce a lower bound on the time between rate changes. The PCRTT algorithm serves as an initial approach to this problem, with some limitations in exploiting the prefetch buffer. In our experiments, we attempted to find the best interval size to use for the PCRTT algorithm given a fixed buffer size by calculating many interval lengths. Creating an efficient algorithm to find the best interval and interval offset, given a fixed buffer size, is also an avenue of research. In addition, the server may have additional latitude in smoothing video streams if the client

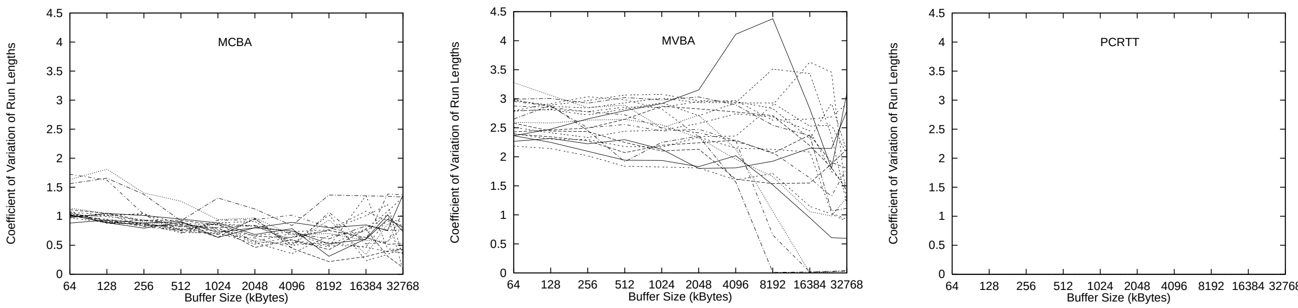


Figure 11: Bandwidth Length Variability

is willing to tolerate some loss in quality; for example, the server could avoid rate change operations by occasionally dropping frames, particularly for layered stream encodings. Finally, a comparison with several newer bandwidth smoothing algorithms that have been recently published can be added to the comparison.

Future work can also consider new algorithms for constructing network *reservation plans* that minimize the total cost of the video transmission for the server, the network, and the client. Ultimately, the construction of transmission and reservation plans should depend on the actual configuration of the server and client sites, as well as the degree of network support for resource reservation and performance guarantees. Consequently, future work can consider the integration of bandwidth smoothing algorithms with the server and client architectures, to ensure efficient delivery of prerecorded video from the storage subsystem on the server to the playback buffer at the client.

6. References

[1] D. Gall, "MPEG: A video compression standard for multimedia applications," *CACM*, vol. 34, pp. 46-58,Apr. 1991.

[2] E. Rathgeb, "Policing of realistic VBR video traffic in an ATM network," *Int'l Jour. of Digital and Analog Comm. Sys.*, vol. 6, pp. 213-226, Oct. 1993.

[3] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Trans. on Networking*, vol. 2, pp. 1-15, Feb. 1994

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

[5] W. Feng and S. Sechrest, "Smoothing and buffering for delivery of prerecorded compressed video," in *Proc. of the IS&T/SPIE Symp. on Multimedia Comp. and Networking*, pp. 234-242, Feb. 1995.

[6] W. Feng and S. Sechrest, "Critical bandwidth allocation for delivery of compressed video," *Comp. Comm.*, vol. 18, pp. 709-717, Oct. 1995.

[7] W. Feng, F. Jahanian, and S. Sechrest, "Optimal buffering for the delivery of compressed prerecorded video," in *Proc.of the IASTED/ISMM Int'l Conf. on Networks*, Jan. 1995.Extended version to appear in *ACM/Springer-Verlag Multimedia Systems*

[8] J. Salehi, Z. Zhang, J. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," in *Proc. of ACM SIGMETRICS*, pp. 222-231, May 1996.

[9] J. McManus and K. Ross, "Video on demand over ATM: Constant-rate transmission and transport," in *Proc. of IEEE INFOCOM*, pp. 1357-1362, Mar. 1996.

[10] D. Anderson, Y. Osawa, and R. Govindan, "A file system for continuous media," *ACM Transactions on Comp. Sys.*, pp. 311-337, Nov. 1992.

[11] D. Gemmell, J. Vin, D. Kandlur, P. Rangan, and L. Rowe, "Multimedia storage servers: A tutorial," *IEEE Comp. Magazine,* vol. 28, pp. 40-49, May 1995.

[12] D. Kandlur, M. Chen, and Z. Shae, "Design of a multimedia storage server," in *Proc. of the IS&T/SPIE Symp. on Elec. Imaging Sci. and Tech.*, Feb. 1994.

[13] P.Lougher and D. Shepard, "The design of a storage server for continuous media," *The Computer Journa*l, vol. 36, pp. 32-42, Feb. 1993.

[14] C. Federighi and L. Rowe, "A dist. hierarchical storageman-ager for a video-on-demand system," in *Proc. of the IS&T/SPIE Symp. on Elec. Imaging Sci. and Tech.*, Feb. 1994.

[15] C. Aras, J. Kurose, D. Reeves, and H. Schulzrinne, "Realtime communication in packet switched networks," *Proc. of the IEEE*, vol. 82, pp. 122-139, Jan. 1994.

[16] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proc. of the IEEE*, vol. 83, pp. 1374-1396, Oct. 1995.

[17] W. Feng, Video-On-Demand Services: Efficient Transportation and Decompression of Variable-Bit-Rate Video. PhD thesis, University of Michigan, Apr. 1996.

[18] G. K. Wallace, "The JPEG still picture transmission standard," *Comm. of the ACM*, vol. 34, pp. 30-44, Apr. 1991.

[19] W. Feng, J. Rexford, "A Comparison of Bandwidth Smoothing Techniques for the Transmission of Prerecorded Compressed Video," Ohio State Univ. Tech. Report, *OSU-CISRC-12/96-TR64.*

[20] L. Rowe, K. Patel, B. Smith, and K. Liu, "MPEG video in software: Representation, transmission, and playback," in *Proc. of the High Speed Networking and Multimedia Comp. Symposium*, Feb. 1994.

[21] P. Pancha and M. E. Zarki, "Prioritized transmission of variable bit rate MPEG video," in Proc. of IEEE GLOBECOM, pp. 1135-1139, 1992.