Performance Evaluation of Smoothing Algorithms for Transmitting Prerecorded Variable-Bit-Rate Video*

Wu-chi Feng
Computer and Information Sciences
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 comprehensive performance evaluation of bandwidth smoothing algorithms, based on a collection of metrics that relate directly to the server, network, and client resources necessary for the transmission, transport, and playback of prerecorded video. 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 bandwidth smoothing algorithm and pinpoint promising directions for future research.

1 Introduction

Many emerging multimedia applications, such as distance learning and entertainment services, rely on the efficient transfer of prerecorded video. Video-on-demand servers typically store video on large, fast disks [1–4], as shown in Figure 1; the server may also include tertiary storage, such as tapes or optical jukeboxes, for holding less frequently requested data [5]. A network connects the video servers to the client sites through one or more communication links. The network can help ensure the continuous delivery of the video data by including support for rate or delay guarantees [6, 7], based on resource reservation requests from the video server. Client sites include workstations and set-top boxes that have a playback buffer for storing one or more video frames. In this paper, we evaluate techniques that capitalize on this buffer space to reduce the server and network overheads for streaming stored video.

*An earlier version of this paper will appear in *Proc. IEEE INFOCOM, April 1997.*

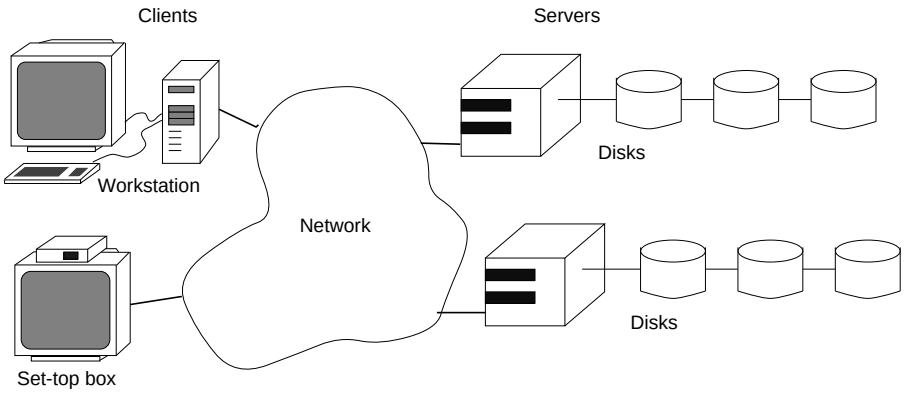


Figure 1: **Video-On-Demand Architecture:** This figure shows a basic video-on-demand architecture consisting of video servers, a communication network, and client sites. Possible clients include workstations and set-top boxes that contain hardware to interact with the network and a disk (or RAM) that serves as a playback buffer.

Effective compression algorithms, such as MPEG [8] and JPEG [9], 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 [10–15]. This bursty, variable-bit-rate traffic complicates the effort to allocate server and network resources to ensure continuous playback at the client sites. To reduce the burstiness of the traffic, stored-video applications can capitalize on the *a priori* knowledge of the frame sizes in the compressed video stream. In particular, the server can smooth the stream by *prefetching* video frames into the client playback buffer 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. Given the frame lengths f_i and the buffer size b, a *bandwidth smoothing algorithm* generates a transmission plan consisting of m constant-bit-rate runs that avoid both underflow and overflow of the client buffer [16–21]. For a reasonable client buffer size, bandwidth smoothing can create a server plan with a fairly small number of runs and a significant reduction in both the peak and standard deviation of the transmission rates.

In response to the increasing interest in stored video services, several smoothing algorithms have been introduced during the past few years. These algorithms produce transmission plans with diverse performance characteristics, depending on what metrics they attempt to optimize. 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 prerecorded variable-bit-rate video. Specifically, we evaluate six algorithms that create plans that

- minimize the number of bandwidth increases [16],
- minimize the total number of bandwidth changes [17],
- minimize the variability of the bandwidth requirements [18],
- minimize the utilization of the prefetch buffer [19],
- consist of periodic bandwidth runs [20],
- minimize cost metrics through dynamic programming [21]

An effective comparison of these algorithms requires experiments that evaluate a wide range of video traces and client buffer sizes.

For an extensive performance evaluation, we have generated a library of twenty full-length, constant-quality video clips[1], digitized using a PC-based, motion-JPEG video capture testbed, as described in Section 2. Section 3 describes the six bandwidth smoothing algorithms, with an emphasis on how they strive to optimize specific performance metrics. Drawing on the motion-JPEG video traces, Section 4 compares the smoothing algorithms and investigates the subtle interplay between five performance metrics

- peak bandwidth requirement
- variability of transmission rates
- number of rate changes
- variability of run lengths
- client buffer utilization

that relate directly to the server, network, and client resources required for transmitting the smoothed video stream. In addition to evaluating the bandwidth smoothing algorithms, these experiments also highlight unique properties of the underlying video clips. These results motivate several possible directions for future research on the efficient transmission of prerecorded variable-bit-rate video, as discussed in Section 5.

[1] The video traces are available on the web at `http://www.cis.ohio-state.edu/~wuchi/Video/index.html`.

2 Compressed Video Sources

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 encoded video traces [22, 23].

2.1 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-90 processor with 32 megabytes of memory. The MiroVideo Capture board is a motion-JPEG compression board containing the C-Cube Microsystems' motion-JPEG chip, the CL550. The motion-JPEG algorithm compresses each video frame using the JPEG still-picture standard [9]. The MiroVideo board digitizes the frames at 640×480 pixels and then subsamples them to 320×240 pixels with guaranteed VHS picture quality. By setting options on this board, the testbed can vary the quality factor used in compressing 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. While the quality factor can be adjusted, each sequence that is captured must have a fixed quality factor, resulting in the bursty variable-bit-rate video sources.

Since motion-JPEG compresses each video frame independently, the traces do not capture the effects of inter-frame dependencies that would exist in MPEG-encoded streams [8, 15]. 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. 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 produces constant-quality, full-length, motion-JPEG video streams.

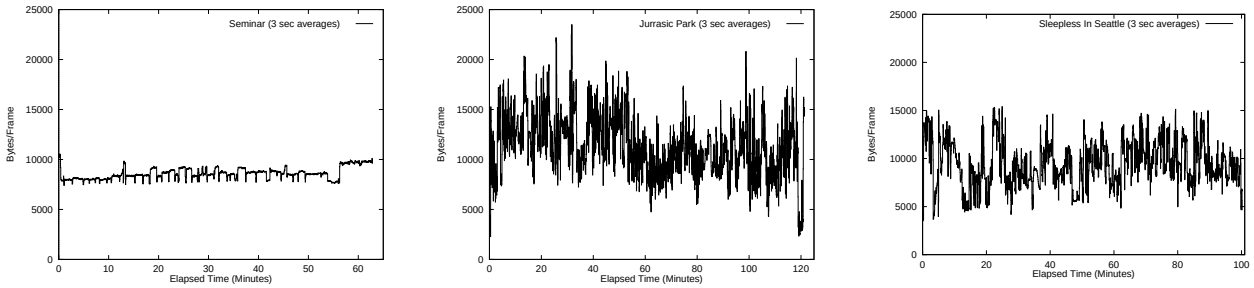
	Resources			Frame Sizes			
	Size (GB)	Time (min)	Rate (Mbps)	Avg (bytes)	Max (bytes)	Min (bytes)	Std (bytes)
Beauty and the Beast	1.82	80	3.04	12661	30367	2701	3580
Big	2.26	102	2.96	12346	23485	1503	2366
Crocodile Dundee	1.82	94	2.59	10773	19439	1263	2336
E.T. (quality 75)	1.24	110	1.51	6305	14269	1511	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

Table 1: **Video Movie Library Statistics**: This table shows the statistics for the clips in our video movie library. All sources are motion-JPEG encoded with a quality factor of 90, except for two versions of *E.T.* with quality factors of 75 and 100. In total, the traces represent 31 hours of video and 38.5 gigabytes of compressed data.

2.2 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



(a) *Seminar1* (b) *Jurassic Park* (c) *Sleepless in Seattle*

Figure 2: **Compressed Video Traces:** These plots show the frame sizes, averaged over three-second intervals, for a departmental seminar and two movies.

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 Table 1 have 0.94 bits-per-pixel, which corresponds to "excellent" picture quality [9].

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 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 departmental 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, as shown in Figure 2. For the seminar video, the trace shows that a few smaller frames occur every minute or two, corresponding to brief intervals with an empty screen, when the speaker changes transparencies. In contrast to the seminar videos, the movies *Jurassic Park* and *Sleepless in Seattle* are more representative examples of the videos in our library. The compressed *Jurassic Park* stream in Figure 2(b) exhibits fairly uniform burstiness throughout the video clip, although the first hour of the movie has larger frames on average than the second hour. The compressed *Sleepless in Seattle* stream exhibits larger bursts of large and small frames, as shown in Figure 2(c).

3 Bandwidth Smoothing

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

3.1 Overflow and Underflow Constraints

A compressed video stream consists of n frames, where frame i requires f_i bytes of storage. 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 data than

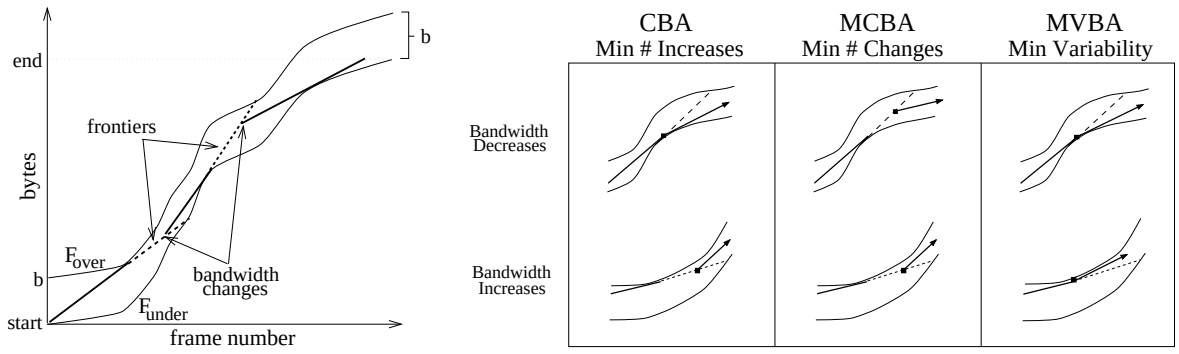
$$F_{over}(k) = F_{under}(k) + b$$

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 3(a). That is,

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

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 3(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. 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.



(a) Underflow and overflow curves (b) Selecting bandwidth changes

Figure 3: **Computing Transmission Plans:** This figure shows the buffer underflow and overflow curves for a sample video stream. The resulting transmission plan consists of three constant-bit-rate runs that serve as a server schedule for transmitting video frames. A smoothing algorithm can optimize various performance metrics depending on how it selects the starting point and slope for each run.

3.2 Smoothing Based on F_{over} and F_{under}

Several different smoothing algorithms have been introduced that use both the F_{over} and F_{under} curves in computing the bandwidth runs in the transmission plans. Given a starting point for run j+1, these algorithms attempt to select a trajectory that extends as far as possible to produce a smooth plan with a limited number of bandwidth changes. 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 3(a). 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 properties:

- **CBA:** For example, the *critical bandwidth allocation* (CBA) algorithm [16] 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(b). 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*.

- **MCBA:** To minimize the number of rate decreases, the *minimum changes bandwidth allocation* (MCBA) algorithm [17] 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), as well as the minimum peak bandwidth requirement. The MCBA and CBA algorithms have a worst-case complexity of $O(n^2 \log n)$, where the $\log n$ term arises from performing a binary search along the frontier of each run; on average, the algorithms run in $O(n \log n)$ time.

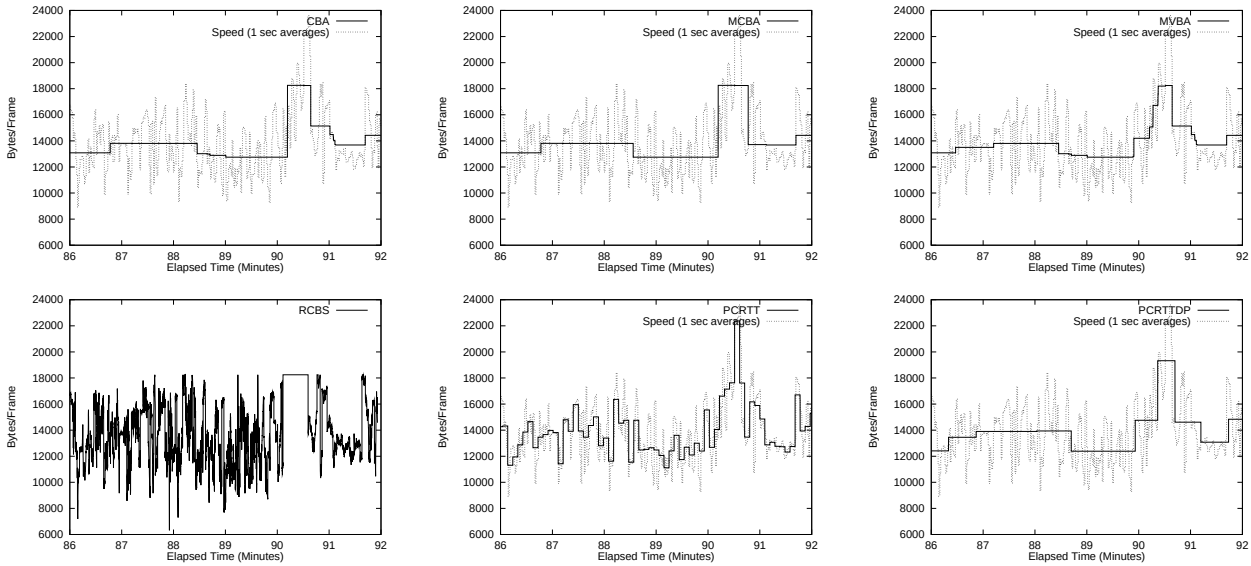


Figure 4: **Bandwidth Plans:** These graphs show the transmission plans generated by four different bandwidth smoothing algorithms, applied to the movie *Speed* and a 1 megabyte client prefetch buffer. For the PCRTT algorithm, the graph shows the plan with the largest possible interval size that would not overflow a 1 megabyte buffer.

- **MVBA:** Instead of minimizing m, a bandwidth smoothing algorithm can strive to reduce the *variability* in the rate requirements [18]; 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. By avoiding a binary search along the frontier, the MVBA algorithm can have a worst-case complexity of $O(n^2)$; this initial algorithm can be modified to be strictly $O(n)$ [18].

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, as shown in Figure 3. For a given client buffer size, the CBA, MCBA, and MVBA bandwidth smoothing algorithms result in transmission plans that minimize the peak bandwidth and maximize the minimum bandwidth. Still, these algorithms differ in terms of rate variability, the frequency of rate changes, and client buffer utilization, as discussed in Section 4.

3.3 Smoothing Based on F_{under}

Given the different starting points on the *frontiers*, the CBA, MCBA, and MVBA algorithms all attempt to select trajectories that extend as far as possible before reaching both the F_{under} and F_{over}

curves. Other smoothing algorithms focus on the F_{under} curve in constructing a schedule; if necessary, these algorithms can iterate to compute a schedule that also satisfies the buffer constraint b for the F_{over} curve:

- **RCBS:** Given a maximum bandwidth constraint r, the *rate-constrained bandwidth smoothing* (RCBS) algorithm generates a schedule with the smallest buffer utilization by prefetching frames as late as possible [23]. In addition, given the rate r, this algorithm minimizes the maximum buffer size required for the particular rate. This $O(n)$ algorithm starts with the last frame of the movie and sequences backwards toward the first frame. Any frame that exceeds the rate constraint is modified to the maximum rate constraint and then prefetched earlier. As shown in Figure 4, the RCBS plan follows the actual data rate for the movie rather closely, particularly for small buffer sizes. To minimize the peak rate requirement, as well as buffer utilization, the RCBS algorithm first invokes the $O(n)$ MVBA algorithm to determine the smallest possible peak rate for a given buffer size; then, this rate constraint is used in computing the RCBS schedule.

- **PCRTT:** In contrast to the four previous algorithms, the *piecewise constant rate transmission and transport* (PCRTT) algorithm [20] creates bandwidth allocation plans by dividing the video stream into fixed-size intervals. This $O(n)$ algorithm generates a single run for each interval by connecting the intersection points on the F_{under} curve, as shown in Figure 5; 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 site; the resulting transmission curve also determines the minimum acceptable buffer size to avoid overflow given the interval size, as shown in Figure 5.

- **PCRTT-DP:** Instead of requiring a rate change for each time interval, a recent extension to the PCRTT algorithm employs dynamic programming (DP) to calculate a minimum-cost transmission plan that consists of m runs [21]. Although dynamic programming offers a general framework for optimization, we focus on the buffer size b as the cost metric to facilitate comparison with the other smoothing algorithms. The algorithm iteratively computes the minimum-cost schedule with k runs by adding a single rate changes to the best schedule with $k-1$ rate changes. However, an exact solution, permitting rate changes in any time slot, would introduce significant computational complexity, particularly for full-length video traces. To reduce the computational overhead, a heuristic version of the algorithm [21] groups frames into intervals, as in Figure 5, when computing each candidate schedule; then, the full frame-level information is used to determine how far to raise the schedule to avoid buffer underflow. This algorithm has a computational complexity of $O(n^3)$.

As shown in Figure 4, the resulting PCRTT-DP algorithm, using a group size of 60 frames, produces bandwidth plans that are somewhat similar to MCBA plans, since both try to limit the number of rate changes. In contrast, the original PCRTT algorithm produces a schedule with a larger number of short runs, since the algorithm uses a single time interval throughout the video; in this example, a small interval size is necessary to avoid overflow of the client buffer. The RCBS plan changes

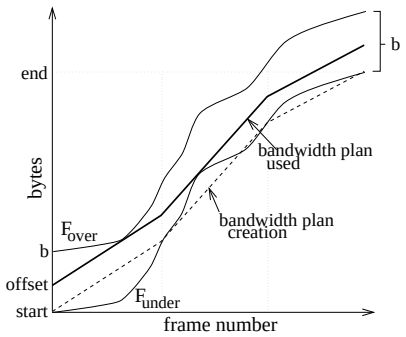


Figure 5: **PCRTT Plan Creation:** This figure shows the creation of a PCRTT bandwidth allocation plan. First, the algorithm calculates the average frame size for each interval (dashed line). Then, the algorithm raises the plan to avoid buffer underflow. Based on the offset plan, the minimum buffer requirement is the maximum distance above the underflow curve.

the transmission rate in almost every time unit, except when large frames introduce smoothing at the peak rate. The next section compares the smoothing algorithms across a range of client buffer sizes, video clips, and performance metrics to evaluate these trade-offs in transmitting prerecorded, variable-bit-rate video.

4 Performance Evaluation

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 variability of the bandwidth allocations, the number of bandwidth changes, the variability of the time between bandwidth changes, and the utilization of the prefetch buffer. By applying these metrics to server transmission plans, across a wide range of realistic client buffer sizes, the simulation experiments show cost-performance trends that affect the transmission, transport, and playback of compressed video. Since some of the algorithms implicitly introduce playback delay, we permit each algorithm to prefetch data on the first bandwidth run for a fair comparison.

For the PCRTT and PCRTT-DP algorithms, which determine the buffer size as a byproduct of computing the bandwidth plan, we vary the window size (for PCRTT) and the number of rate changes (for PCRTT-DP) to generate a collection of plans, each with a corresponding buffer size. The fixed window size in the PCRTT algorithm can result in fluctuations in the performance metrics as the buffer size increases, since a smaller window size can sometimes result in a larger buffer requirement. To remove these artificial fluctuations, we ensure that each data point corresponds to the PCRTT plan that has the lowest peak rate requirement for that buffer size (or smaller). The PCRTT-DP heuristic computes bandwidth plans based on groups of 60 frames to reduce computational complexity;

sample experiments with smaller group sizes resulted in similar values for the performance metrics. However, the frame grouping does limit the ability of the algorithm to compute bandwidth plans for small buffer sizes; for small buffer sizes, a more exact (and computationally expensive) version of the PCRTT-DP heuristic should produce statistics that resemble the MCBA results, since both algorithms compute transmission plans than limit the number of rate changes. The frame-grouping and rate-change parameters both limit the algorithm's ability to compute valid plans for small buffer sizes, since smoothing into a small buffer requires bandwidth changes on a very small time scale.

For a typical two-hour video ($n = 216,000$ frames), the CBA, MCBA, MVBA, RCBS, and PCRTT algorithms require a few seconds of computation time on a modern workstation. The RCBS algorithm generally executes in the smallest amount of time (after determining the rate constraint), followed by the PCRTT, MVBA, CBA, and MCBA algorithms (in that order). The PCRTT-DP algorithm, using a group size of 60 frames and allowing up to 1000 rate changes, requires about an hour to execute. Because the PCRTT-DP algorithm start with the number of rate changes $K = 1$ and iteratively calculates the minimal cost of each successive bandwidth change, calculating a plan that has 1000 bandwidth changes requires the calculation of all plans with fewer bandwidth changes. To speed this algorithm up, we calculated all the costs (in terms of the buffer size) for each sequence of frames (i, j), $0 < i < j \leq N$. This reduces the computational complexity of each bandwidth change to $O(n^2)$.

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

$$\max_j \{r_j\}$$

to increase the likelihood that the server, network, and the client have sufficient resources to handle the stream. This is especially important if the service must reserve network bandwidth based on the peak rate, or if the client has a low-bandwidth connection to the network. In addition, reducing the maximum bandwidth requirement permits the server to multiplex a larger number of streams. Figure 6 plots $\max\{r_j\}$ for each of the six smoothing algorithms, as a function of the client buffer size. The CBA, MCBA, MVBA, and RCBS algorithms all result in the minimum peak bandwidth requirement given a fixed buffer size, as discussed in Section 3.

The CBA, MCBA, MVBA, and RCBS algorithms have monotonically decreasing peak bandwidth requirements as the buffer size grows. Under medium-sized buffers, these algorithms produces smaller

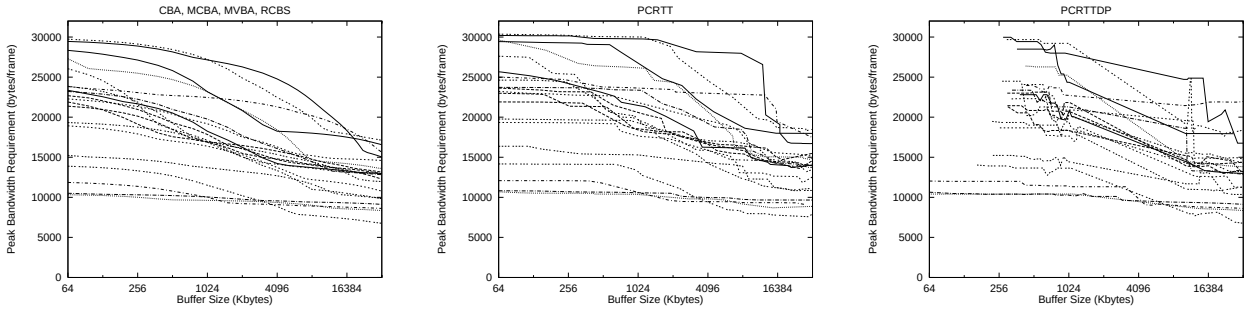


Figure 6: **Peak Bandwidth Requirement:** These graphs plot the peak rate requirements for the various smoothing algorithms as a function of the client buffer size. The CBA, MCBA, MVBA, and RCBS algorithms generate transmission plans with the same peak rate.

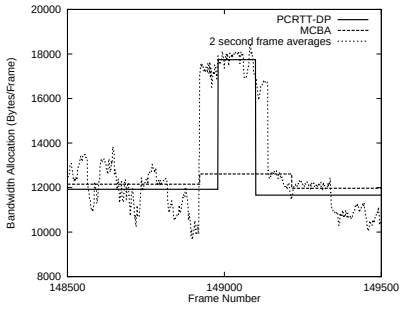
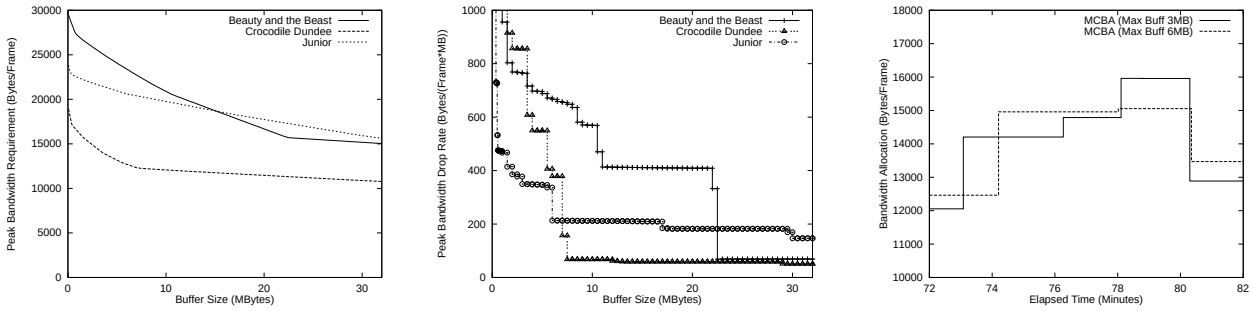


Figure 7: **PCRTT-DP Frame Grouping:** This graph plots the PCRTT-DP and MCBA plans for the movie *Speed* with a 1-megabyte prefetch buffer. Using a group size of 60 frames, the PCRTT-DP plan has a larger peak-rate run because the region of large frames extends part way into adjoining groups. In contrast, the MCBA algorithm can reduce the peak rate by transmitting more of the large frames during a portion of the previous time interval without exhausting the prefetch buffer.

peak rates than the two PCRTT algorithms by prefetching data as far in advance as the buffer allows. In contrast, 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.

The PCRTT-DP plans have smaller peak bandwidth requirements than the PCRTT algorithm and similar to the other algorithms. In fact, an exact version of PCRTT-DP algorithm, using a group size of 1 frame, would generate transmission plans that minimize the peak bandwidth. However, the grouping of frames can sometimes inflate the peak rate when a sequence of large frames fall within a single group, as shown in Figure 7. In this particular example, smoothing the peak over the previous



(a) Peak bandwidth vs. b (b) Peak rate derivative (c) Buffering Example

Figure 8: **Reducing the Peak Rate:** These plots highlight the relationship between the peak bandwidth metric and the client buffer size. In (a), we has selected 3 representative movies *Beauty and the Beast*, *Crocodile Dundee*, and *Junior*. In (b), the derivative (slope between peak bandwidth, buffer points) are shown, emphasizing the linear reduction in peak bandwidth requirement.

group of 60 frames would result in a smaller peak bandwidth requirement at the expense of a larger buffer size. The MCBA algorithm is able to adjust the bandwidth requirement at any frame, allowing it to suitably adjust the start and end of the run at frame 149,000. Referring to Figure 7, note that this high-bandwidth run starts and ends less than one group of frames away from the corresponding run in the PCRTT-DP plan. Still, in most cases, the PCRTT-DP heuristic can produce plans with nearly optimal peak bandwidth requirements.

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 of the 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 1 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 client 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 three *Seminar* videos, digitized with a quality factor of 90, have larger average frame sizes (8604, 8835, and 9426 bytes). Thus, for small buffer sizes, the peak bandwidth requirement is generally driven by the *maximum* frame size, while for larger buffer sizes, the peak rate is driven mostly by the *average* frame size.

Although a large prefetch buffer can substantially reduce the peak rate, the addition of more buffer space offers diminishing returns beyond a certain point. Figure 8(a) further illustrates this effect by plotting the peak-bandwidth curves for *Beauty and the Beast*, *Crocodile Dundee*, and *Junior* with a linear scale on the x-axis. Each curve consists of a sequence of linear segments that become less steep

as the buffer size increases, as shown in Figure 8(b) which plots the magnitude of the slopes. Typically, the peak bandwidth in a transmission plan is determined by a region of large frames in a small portion of the movie. As the buffer size grows, the peak-rate run becomes shorter and wider due to more aggressive prefetching, as shown in Figure 8(c). Eventually, the buffer size grows large enough for this run to reach a region of small frames that can allow a more dramatic reduction in the bandwidth requirement; at this point, the peak rate may occur in a different part of video, also resulting in a new linear segment in the peak-bandwidth curve. Based on this observation, the server could characterize the buffer-bandwidth trade-off as a small number of linear curves, allowing efficient admission control policies that balance the use of network/server bandwidth and client buffer space.

4.2 Variability in Bandwidth Requirements

In addition to minimizing the peak bandwidth, a smoothing algorithm should reduce the overall *variability* in the rate requirements for the video stream [18]. Intuitively, plans with smaller rate variation should require fewer resources from the server and the network; more precisely, smoother plans have lower *effective bandwidth* requirements, allowing the server and the network to statistically multiplex a larger number of streams [24]. Even under a deterministic model of resource reservation, the server's ability to change a stream's bandwidth reservation may depend on the *size* of the 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 be unable to acquire enough bandwidth to start transmitting frames at the higher rate[2]. Since the video clips have different average rate requirements, varying from 1.51 to 3.95 megabits/second, Figure 9 plots the *coefficient of variation*

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

to normalize the variability metric across the different streams.

In Figure 9, 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 combining bandwidth requests, resulting in greater rate variability

[2] 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 [25, 26]. 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.

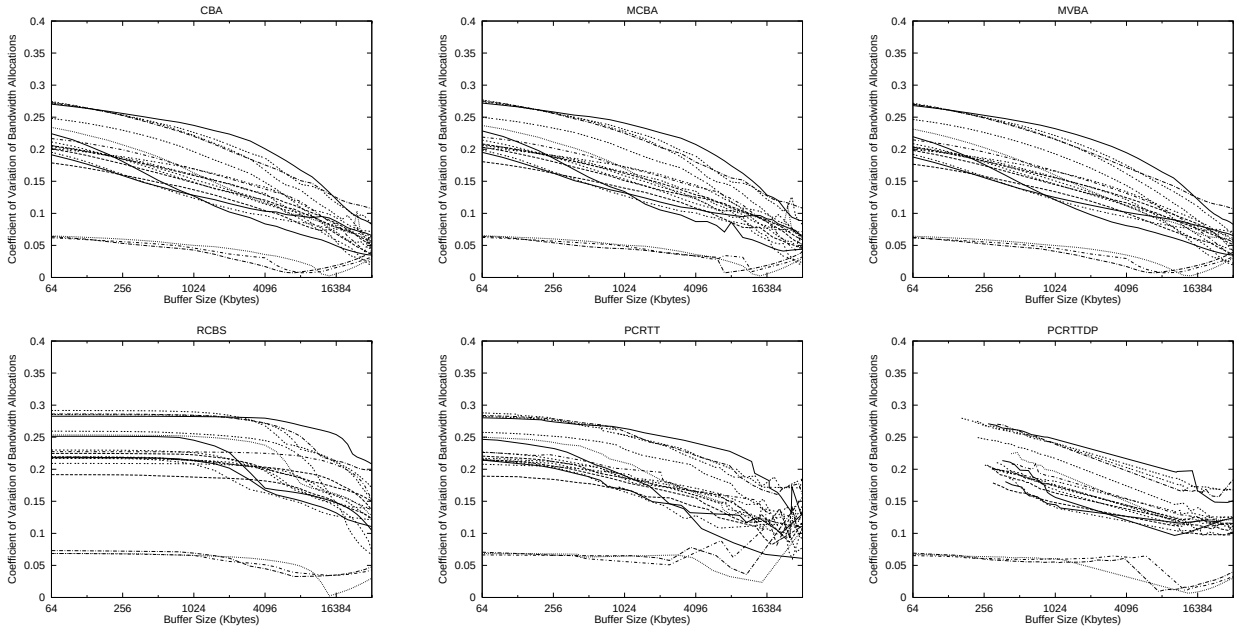
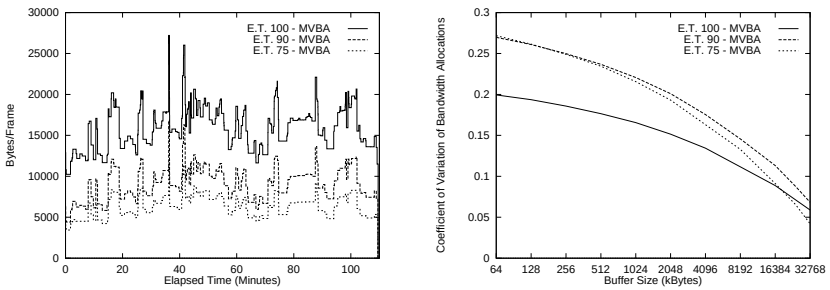


Figure 9: **Bandwidth Variability:** This figure shows the normalized standard deviation of bandwidth allocations (on a per frame basis) for the various smoothing algorithms, as a function of the client buffer size.

than MVBA and CBA. For the videos in Figure 9, the MCBA plans have rate variations that are approximately 5% larger than the corresponding MVBA plans. In videos where the MCBA algorithm has much fewer bandwidth changes than the MVBA, the MCBA algorithm results in higher variability of bandwidth requests. Still, the MVBA, CBA, and MCBA plans have similar variability metrics, while also minimizing the peak rate and maximizing the minimum rate [16–18], so all three algorithms should produce transmission plans with low effective bandwidth requirements.

In contrast, the RCBS plans have greater rate variability, particularly under larger buffer sizes, since the algorithm limits prefetching unless it is necessary to avoid increasing the peak rate. As a result, an RCBS plan often transmits small frames at a low rate, resulting in a much lower minimum bandwidth than the MVBA, CBA, and MCBA algorithms. Hence, the modest increase in rate variability under the RCBS algorithm stems from the *small* frames, instead of the large frames as one would expect from an unsmoothed transfer of the video. The PCRTT algorithm has 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 buffer sizes get larger, the partitioning of the frames into fixed intervals plays a large role in determining the minimum amount of buffering required to have continuous playback of the video.



(a) Transmission plans (b) Coefficient of variation

Figure 10: **Rate Variability of *E.T.* Encodings:** These graphs evaluate the three traces of *E.T.* (with quality levels of 100, 90, and 75), using the MVBA algorithm and a 1 megabyte buffer. Despite reducing the bandwidth requirements, the coarser encodings exhibit greater rate variability, relative to the average frame size.

For all the algorithms, the *Beauty and the Beast*, *E.T. (quality 75)*, and *E.T. (quality 90)* videos exhibit the most bandwidth variability. Interestingly, the *E.T.* streams with *lower* quality factors have greater variability in the bandwidth requirements, as shown in Figure 10. Although a coarser encoding reduces the *average* frame size, some frame lengths decrease more than others, depending on the scale of detail in the scene; from the information in Table 1, the coefficient of variation for frame sizes is 0.29, 0.28, and 0.20 for quality factors of 75, 90, and 100, respectively. Under small buffer sizes, the larger variability in frame sizes translates into larger variability in the bandwidth requirements. For larger buffer sizes, the three versions of *E.T.* have similar bandwidth variability, due to the common long-term variation in scene content; the variability of the *E.T. (quality 75)* clip decreases more quickly, as a function of buffer size, since a larger prefetch buffer can absorb most of the variation in frame sizes for the lower quality stream.

4.3 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 [17]. A rate change alters the amount of data that the server must read from the disk in each time interval, which can have implications for disk layout and scheduling policies, particularly when the server must multiplex a large number of video streams. Also, smaller values of m reduce the storage requirements for the bandwidth plans, although this is typically small in comparison to the size of the actual video data. Minimizing the number of rate changes can also limit the cost of negotiating with the network [14] to reserve link bandwidth for transporting the video stream[3]. Figure 11 compares the algorithms based on the number

[3]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 [14, 18].

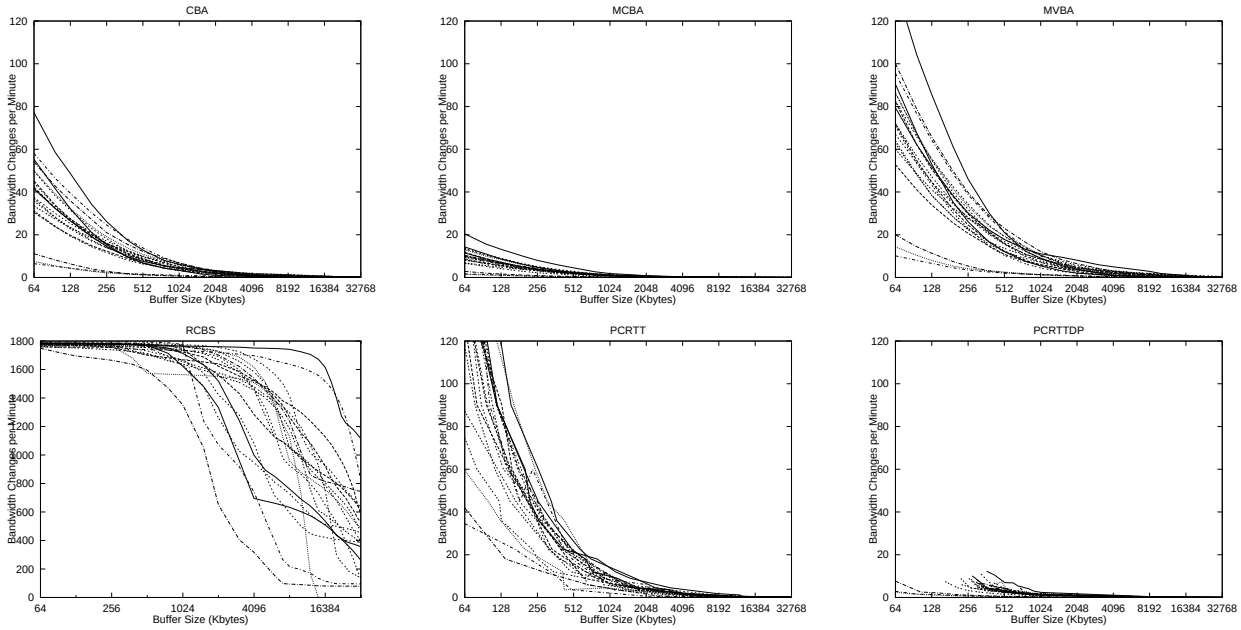


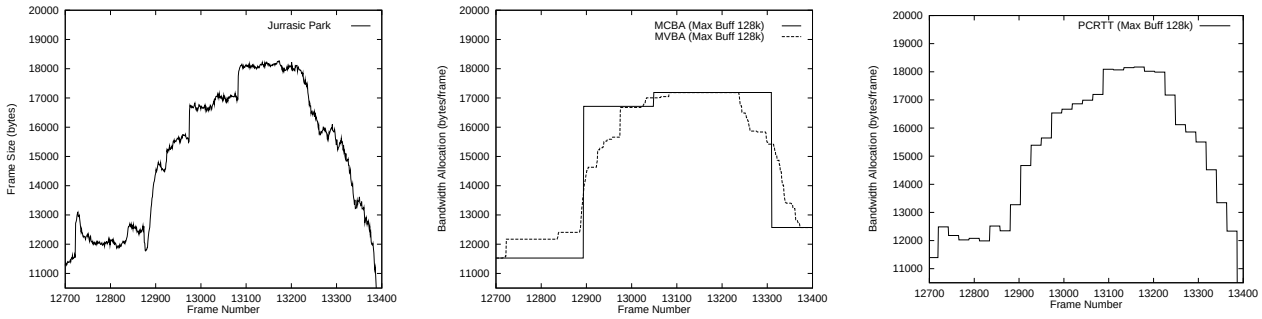
Figure 11: **Rate of Bandwidth Changes:** These graphs show the rate of bandwidth changes required for the various smoothing algorithms as a function of the client buffer size. Note that the RCBS graph has a *much* wider range of values on the y-axis, compared to the plots for the other five algorithms.

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 *frequency* of bandwidth changes

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

in changes per minute across a range of client buffer sizes.

For all of the smoothing algorithms and video traces, 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, and 8.5 rate changes per minute under the MCBA, CBA, and MVBA plans, respectively. In general, the CBA and MVBA plans result in approximately 3 and 6 times as many changes as MCBA algorithm, which minimizes m. 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



(a) Segment of *Jurassic Park* (b) MCBA and MVBA plans (c) PCRTT plan

Figure 12: **Gradual Changes in Frame Sizes:** This figure highlights the differences between the MCBA and MVBA plans for a 23-second segment of *Jurassic Park* under a 128-kilobyte prefetch buffer. The MVBA algorithm performs a large number of small bandwidth changes to track the gradual increases (decreases) in the frame sizes. In contrast, the MCBA plan initiates a smaller number of larger rate changes (3 changes vs. 104 in the MVBA plan). The corresponding PCRTT plan has 31 rate changes.

small rate changes to minimize the variability of bandwidth requirements in the server transmission plan.

As an extreme example, we compare the MCBA and MVBA algorithms on the 23-second video trace shown in Figure 12(a). For a 128 kilobyte buffer, the MVBA algorithm introduces 104 rate changes (55 increases and 49 decreases), while the MCBA plan has just three bandwidth changes, as shown in Figure 12(b). During the first 400 frames of the video segment, the frame sizes gradually increase over time. 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 video segment. In Figure 12(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; referring to Figure 3(b), this is a case where the MCBA algorithm selects a starting point at the *rightmost* point along the frontier whereas the MVBA algorithm selects the *leftmost* point.

Although the MVBA plans typically have fewer rate changes than the corresponding PCRTT plans, the PCRTT algorithm sometimes generates fewer rate changes under moderate buffer sizes. For these buffer sizes, the PCRTT algorithm is effective at combining several bandwidth runs of the MVBA algorithm into a single rate interval. For example, in Figure 3(c), the PCRTT algorithm generates only 31 rate changes, in contrast to the 104 changes in the corresponding MVBA plan. The PCRTT-

DP algorithm produces bandwidth allocation plans that are very similar to the MCBA algorithm, since they both strive to minimize the number of rate changes; however, under smaller buffer sizes, the PCRTT-DP heuristic generate more bandwidth changes due to the frame-grouping factor. In contrast to the PCRTT algorithms, the RCBS plans tend to follow the sizes of the individual frames for most of the stream, except when some prefetching is necessary to avoid increasing the peak rate for transmitting the video. With a small client buffer, the RCBS algorithm requires nearly 1800 rate changes per minute, as shown in Figure 11; in the absence of any smoothing, a 30 frames/second rate would correspond to at most 2400 changes per minute. Although the number of rate changes decreases as the buffer size grows, the RCBS algorithm still generates significantly more bandwidth changes than the other algorithms except for extremely large buffer sizes.

4.4 Periodicity of Bandwidth Requests

In addition to minimizing the frequency of bandwidth changes, the server may also wish to limit the total number of rate changes that can occur during an interval of time. In between rate changes, the server can transmit the smoothed video without altering the underlying transmission schedule. Hence, a bandwidth smoothing algorithm may try to enforce a lower bound on the minimum time between rate changes to reduce the overheads in multiplexing a large number of video streams. In addition, in networks that support advance booking of resources, periodic bandwidth allocation intervals can reduce the complexity of the admission control algorithms by ensuring that reservations change only at a relatively small set of points [27]. While Section 4.3 evaluates the *average* rate of bandwidth changes, the plots in Figure 13 focus on the *variability* of this metric. Since the bandwidth plans have different average run lengths, the graphs plot

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

to compare the "periodicity" of bandwidth requests. Smaller values imply that the transmission plan has run lengths that stay close to the mean values in Figure 11.

Ideally, in fact, the transmission plans should impose a lower bound on the minimum time between rate changes ($\min_j\{t_j\}$). However, only the PCRTT algorithms produce meaningful values for $\min_j\{t_j\}$, since the plans are constructed from fixed-size intervals; in fact, the coefficient of variation is 0 for the PCRTT algorithm. The MCBA, CBA, MVBA, and RCBS algorithms typically have one or more runs with extremely short durations; 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,

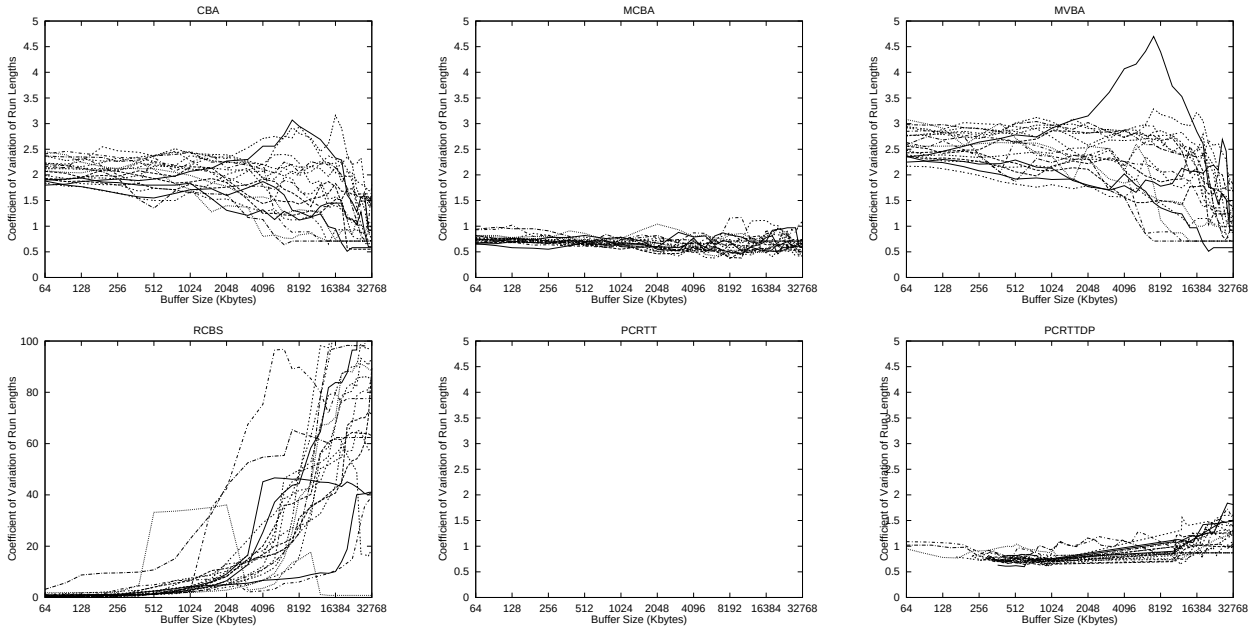


Figure 13: **Bandwidth Length Variability:** This figure shows the coefficient of variation for the bandwidth run lengths as a function of the client buffer size. Note that the RCBS plot has a different y-axis scale than the other graphs.

even under fairly large buffer sizes. The RCBS plans have *much* larger variability, since the transmission plans consists of a mixture of large run lengths (where frames have been smoothed at the peak rate) and small run lengths (where no prefetching occurs). To reduce the frequency of rate changes, generalizations of the RCBS algorithm could transmit frames at averaged rates over small intervals of time by introducing a modest amount of extra prefetching [19]. This hybrid of the PCRTT and RCBS algorithms would produce plans with a reasonable time between consecutive rate changes, while still limiting the amount of prefetching.

Under small buffer sizes, the top three curves for the MCBA algorithm correspond to the *Seminar* videos. As shown in Figure 2(a), 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; under a larger buffer size, the plans become much more "periodic." As shown in the MVBA graph in Figure 13, one movie (*Beauty and the Beast*) exhibits a particularly high variation of run-lengths. Compared to the other videos, this movie has longer sustained regions of large and small frames. Using a larger buffer smooths the *tops* and *bottoms* of the bandwidth plans, combining relatively larger areas into even larger areas while still leaving many smaller rate adjustments in the transitions. As the number

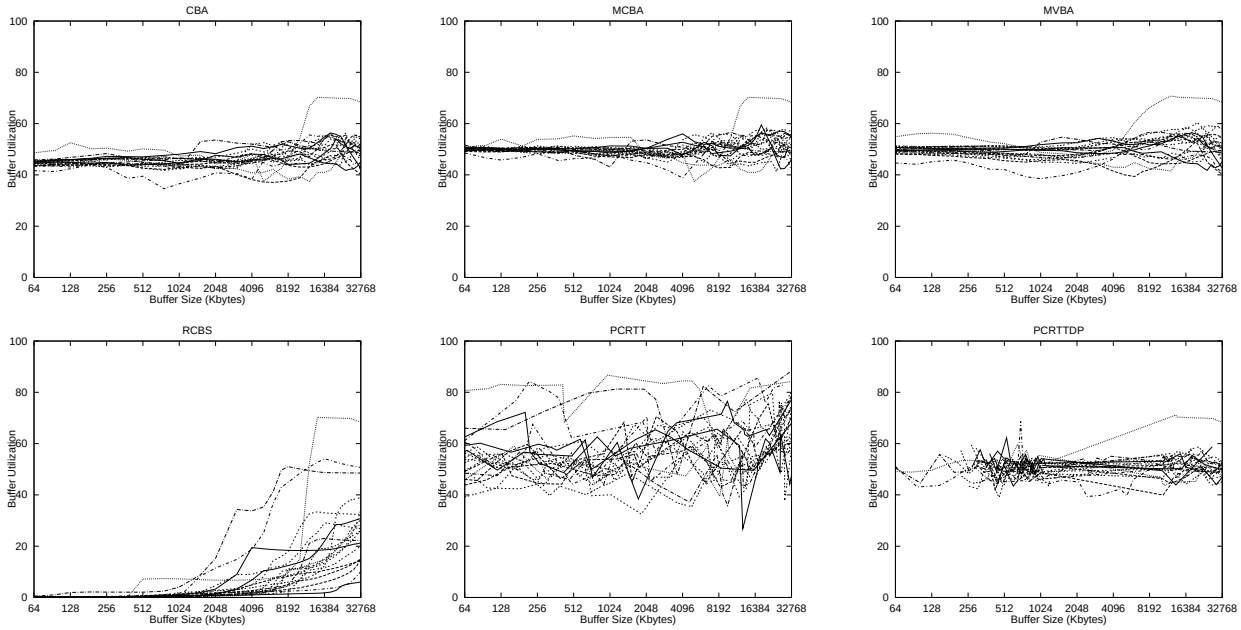


Figure 14: **Buffer Utilization:** This figure shows the average utilization of the client prefetch buffer for the various smoothing algorithms, as a function of the client buffer size.

of changes starts to decrease with large buffer sizes, the plans generated by the MVBA algorithm approach the plans that are generated by the MCBA algorithm.

4.5 Buffer Utilization

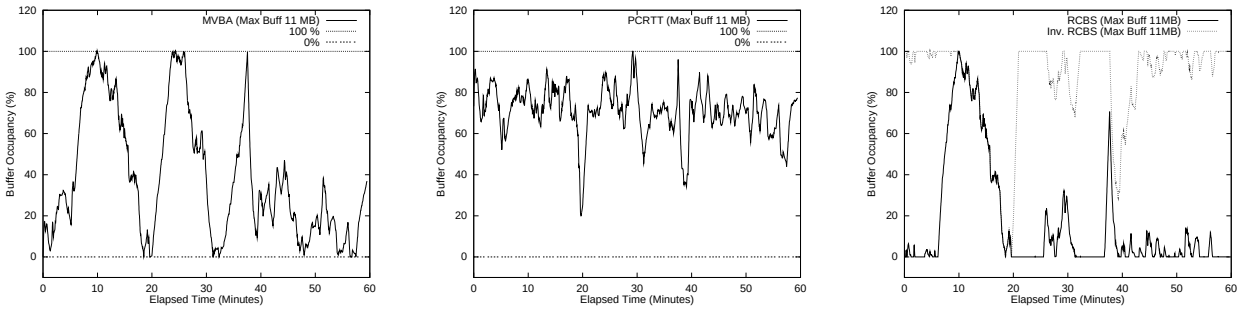
Although bandwidth smoothing reduces the rate requirements for transmitting stored video, prefetching may consume significant buffer resources at the client site. For a given size b for the playback buffer, a smoothing algorithm could strive to limit buffer utilization while still minimizing the peak rate [19]. Reducing the amount of prefetching allows the client to statistically share the playback space between multiple video streams, or even other applications. If the client application can perform VCR functions, such as rewinding or indexing to arbitrary points in the video stream, a bandwidth plan that limits prefetching also avoids wasting server and network resources on transmitting frames ahead of the playback point. With fewer future frames in the prefetch buffer, the client can cache multiple frames behind the current playback point, allowing the service to satisfy small VCR rewind requests directly at the client site. Figure 14 plots the average buffer utilization as a function of buffer size b for each of the smoothing algorithms.

Buffer utilization corresponds to how far the transmission plan lies above the F_{under} curve, on average. The MCBA and MVBA plans have buffer utilizations of approximately 50% across most of the video traces, since these two algorithms do not differentiate between rate increases and rate decreases

when computing bandwidth runs. Hence, the runs have a nearly even split between trajectories that hit F_{under} and trajectories that hit F_{over}; runs between the two constraint curves experience a progression in buffer utilization from 0% to 100%, or vice versa, as shown in Figure 15(a). In contrast, the CBA algorithm tends to stay closer to the F_{under} curve by behaving like MCBA on bandwidth increases and MVBA on bandwidth decreases, as shown in Figure 3(b). As a result, the CBA plans have lower buffer utilization than the corresponding MCBA and MVBA plans. Although these three algorithms typically have 40–50% buffer utilization, the *Seminar1* video has higher utilization under large buffer sizes, as shown by the lone higher curve in CBA, MCBA, and MVBA plots in Figure 14. The larger buffer sizes permit the algorithms to smooth the *Seminar1* video with a single bandwidth run, so the transmission plan never oscillates back and forth between the F_{under} and F_{over} curves.

The PCRTT plans typically have the highest buffer utilization. Drawing on the example in Figure 5, the PCRTT algorithm generates trajectories based on F_{under} and then raises the plan until no points remain below F_{under}, as shown in Figure 5. As a result, the final plan has very few points that lie close to F_{under}, as shown by the example in Figure 15(b). In contrast, RCBS plans stay as close to the F_{under} curve as possible, without increasing the peak rate requirement. In fact, an RCBS plan only reaches the F_{over} curve during the bandwidth runs that must transmit frames at the peak rate; for example, the RCBS plan has less than 15% buffer utilization for most of the video clip in Figure 15(c). The RCBS algorithm generates transmission plans with much lower buffer usage than the other algorithms, especially under small buffer sizes. Buffer utilization is especially low for small values of b, where very few frames are prefetched; for larger b values, even RCBS performs must perform more extensive prefetching to successfully minimize the peak rate. Still, the RCBS algorithm gives the client sites the greatest latitude in sharing the prefetch buffer amongst multiple streams or in supporting efficient rewind operations.

Although low buffer utilization permits greater resource sharing and efficient VCR operations, some video-on-demand services may benefit from more aggressive prefetching, depending on the characteristics of the network connection between the server and client sites. If packets experience variable delay or significant loss rates, the service may perform more aggressive prefetching to mask the network latency (or retransmission delay). To *maximize* buffer utilization, the server could employ an algorithm that is conceptually similar to RCBS. However, instead of transmitting frames as late as possible, under the constraint on peak bandwidth, this algorithm would transmit frames as *early* as possible to tolerate larger network latency. Hence, the server would transmit frames at the peak rate, except when the transmission rate must be reduced to avoid buffer overflow at the client site. Figure 15(c) compares this new algorithm to the corresponding RCBS plan to highlight the upper and lower limits on buffer utilization; all other bandwidth plans that minimize the peak-rate requirement



(a) MVBA (b) PCRTT (c) RCBS and Inverse RCBS

Figure 15: **Buffer Utilization:** This figure shows the buffer utilization over time for smoothing the movie *Crocodile Dundee* with an 11-megabyte prefetch buffer.

have buffer utilizations between these two curves.

5 Conclusions and Future Work

In this paper, we have presented a comprehensive comparison of bandwidth smoothing algorithms for compressed, prerecorded video. 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 allocations; the MCBA algorithm, however, is much more effective at reducing the total number of rate changes. The RCBS algorithm introduces a large number of rate changes, and a wide variability in the bandwidth requirements, to minimize the utilization of the client prefetch buffer.

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 a possible avenue for research. More generally, the use of dynamic programming in the PCRTT-DP algorithm offers a valuable framework for minimizing "costs" that are functions of multiple performance metrics. Similarly, hybrids of the other smoothing algorithm should be effective in balancing the trade-offs between different metrics. For example, extensions to RCBS (or inverse

RCBS) algorithm could operate over coarser time intervals to reduce the variability in the transmission plans without significantly changing the buffer utilization properties.

Ultimately, the construction of server transmission 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. For example, the server may have additional latitude in smoothing video streams if the client is willing to tolerate some loss in quality; for example, the server could avoid rate change operations by occasionally dropping frames, particularly if the stream has a layered encoding. These new schemes can broaden the family of bandwidth smoothing algorithms to tailor video transmission protocols to delay, throughput, and loss properties along the path from the server, through the communication network, to the client sites.

References

[1] D. Anderson, Y. Osawa, and R. Govindan, "A file system for continuous media," *ACM Transactions on Computer Systems*, pp. 311–337, November 1992.

[2] P.Lougher and D. Shepard, "The design of a storage server for continuous media," *The Computer Journal*, vol. 36, pp. 32–42, February 1993.

[3] D. Kandlur, M. Chen, and Z. Shae, "Design of a multimedia storage server," in *Proceedings of the IS&T/SPIE Symposium on Electronic Imaging Science and Technology*, February 1994.

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

[5] C. Federighi and L. Rowe, "A distributed hierarchical storage manager for a video-on-demand system," in *Proceedings of the IS&T/SPIE Symposium on Electronic Imaging Science and Technology*, February 1994.

[6] C. Aras, J. Kurose, D. Reeves, and H. Schulzrinne, "Real-time communication in packet switched networks," *Proceedings of the IEEE*, vol. 82, pp. 122–139, January 1994.

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

[8] D. L. Gall, "MPEG: A video compression standard for multimedia applications," *Communications of the ACM*, vol. 34, pp. 46–58, April 1991.

[9] G. K. Wallace, "The JPEG still picture transmission standard," *Communications of the ACM*, vol. 34, pp. 30–44, April 1991.

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

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

[12] S. S. Lam, S. Chow, and D. K. Yau, "An algorithm for lossless smoothing of MPEG video," in *Proceedings of ACM SIGCOMM*, pp. 281–293, August/September 1994.

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

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

[15] O. Rose, "Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems," in *Proceedings of Conference on Local Computer Networks*, pp. 397–406, October 1995.

[16] W. Feng and S. Sechrest, "Smoothing and buffering for delivery of prerecorded compressed video," in *Proceedings of 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.

[17] W. Feng, F. Jahanian, and S. Sechrest, "Optimal buffering for the delivery of compressed prerecorded video," in *Proceedings of the IASTED/ISMM International Conference on Networks*, January 1995. Extended version to appear in *ACM/Springer-Verlag Multimedia Systems Journal*.

[18] 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 *Proceedings of ACM SIGMETRICS*, pp. 222–231, May 1996.

[19] W. Feng, "Rate-constrained bandwidth smoothing for the delivery of stored video," in *Proceedings of IS&T/SPIE Multimedia Networking and Computing*, February 1997.

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

[21] J. M. McManus and K. W. Ross, "A dynamic programming methodology for managing prerecorded VBR sources in packet-switched networks." Unpublished report extending September 1995 report, January 1997.

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

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

[24] D. Towsley. Private communication, January 1997.

[25] L. Rowe, K. Patel, B. Smith, and K. Liu, "MPEG video in software: Representation, transmission, and playback," in *Proceedings of the High Speed Networking and Multimedia Computing Symposium*, February 1994.

[26] P. Pancha and M. E. Zarki, "Prioritized transmission of variable bit rate MPEG video," in *Proceedings of IEEE GLOBECOM*, pp. 1135–1139, 1992.

[27] D. Ferrari, A. Gupta, and G. Ventre, "Distributed advance reservation of real-time connections," in *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 15–26, April 1994.