Rate-constrained bandwidth smoothing for the delivery of stored video

Wu-chi Feng
Department of Computer and Information Science
Ohio State University
Columbus, Ohio 43210
wuchi@cis.ohio-state.edu

ABSTRACT

Bandwidth smoothing techniques for the delivery of compressed prerecorded video have been shown effective in removing the burstiness required for the continuous playback of stored video. Given a fixed client-side buffer, several bandwidth smoothing algorithms have been introduced that are provably optimal under certain constraints. These algorithms, however, may be too aggressive in the amount of data that they prefetch, making it more difficult to support VCR functions that are required for interactive video-on-demand systems. In this paper, we introduce a *rate-constrained bandwidth smoothing* algorithm for the delivery of stored video that, given a fixed maximum bandwidth rate minimizes both the smoothing buffer requirement as well as the buffer residency requirements. By minimizing the buffer residency times, the clients and servers can remain more tightly coupled making VCR functions easier to support. A comparison between the rate-constrained bandwidth smoothing algorithm and other bandwidth smoothing algorithms is presented using a compressed full-length movie.

Keywords: bandwidth smoothing, video-on-demand, video retrieval techniques

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[18] and motion-JPEG, can substantially reduce the resource requirements for storing and transmitting video streams. However, constant-quality 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[23, 19, 16]. 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 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 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 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 [7, 8, 10, 21, 24].

For interactive video-on-demand systems, two mutually opposite goals are required. In order to provide interactive delivery, where the consumption rate of the client may be altered, the clients and servers need to be tightly coupled so that changes in consumption in the client can be reflected in the transmission rate of the server. However, for systems that allocate bandwidth based on the *peak* bandwidth requirement for the duration of the video, minimizing the peak bandwidth is necessary, requiring that the client's consumption point and the server's transmission point be decoupled. Given some fixed buffer size, bandwidth smoothing techniques so far have typically attempted to maximize the utilization of the client-side buffer by minimizing the number of changes or variability of bandwidth requirements while also minimizing the peak band-

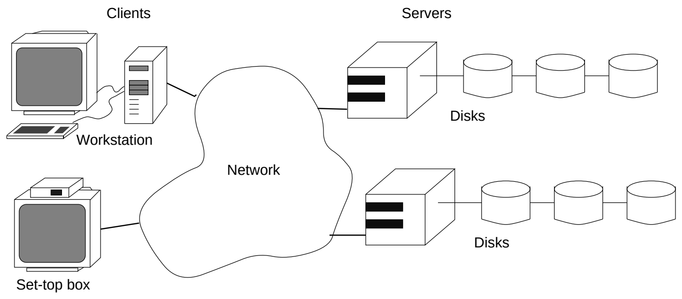


Figure 1 - Video-On-Demand Architecture: This figure shows a basic video-on-demand architecture consisting of video servers, a communication network, and client sites. Possible clients include workstations and set-top boxes that contain hardware to interact with the network and a disk (or RAM) that can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

width requirements. For interactive video-on-demand systems, these bandwidth smoothing techniques may be too aggressive in their prefetching strategies

In this paper, we introduce a *rate constrained bandwidth smoothing* (RCBS) algorithm that minimizes the buffer required for smoothing (given a maximum rate constraint *r*), while also minimizing the distance that the consumption point and transmission point stray from each other. Unlike algorithms that have been presented in the literature, this algorithm only prefetches video data when the rate constraint will be violated. Thus, the RCBS algorithm does not prefetch data when the frames are well below the rate constraint *r.* By prefetching data in this manner the client and server can remain the most tightly coupled, given the rate constraint *r.* Our results show that by using the RCBS algorithm, the buffer can store approximately 50% more data for rewind and examine operations and result in much smaller amount of resynchronization data for random accesses (such as those found in long fast-forwards).

In the next section, we present some of the work on bandwidth smoothing that has been introduced in the literature as well as discuss its impact on providing VCR functionality. In Section 3, we describe the rate constrained bandwidth smoothing algorithm. A comparison of several bandwidth smoothing techniques with the RCBS algorithm is presented in Section 4. Finally some conclusions are presented in Section 5.

2 Background

2.1 Bandwidth Smoothing

Bandwidth smoothing can reduce the burstiness of compressed video traffic in a video-on-demand architecture. The basic video-on-demand architecture that we assume is shown in Figure 1. Video servers typically store prerecorded video on large, fast disks[1, 14, 17, 22] and may also include tertiary storage, such as tapes or optical jukeboxes, for holding less frequently requested data[6]. 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[2, 26], 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.

2.1.1 Bandwidth Plans

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

$$F_{under}(k) = \sum_{0}^{k} f_i$$

indicates the amount of data consumed at the client by frame *k*, where $k = 0,1,...,n-1$. Similarly, the client should not receive more data than

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

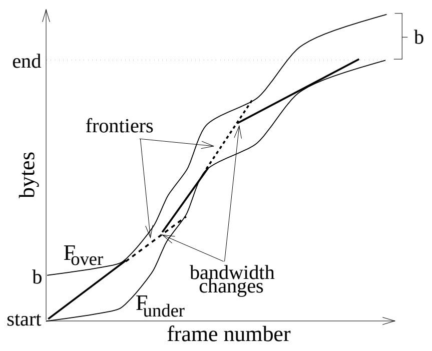


Figure 2 - Sample Transmission Plan: This figure shows the buffer underflow and overflow curves for a sample video stream. The plan consists of three constant-rate runs that serve as a schedule for transmitting video frames from the server.

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,

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

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

The bandwidth smoothing plans that have introduced in the literature create plans that have *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.

2.1.2 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 properties. For example, the *critical bandwidth allocation* (CBA) algorithm [7, 8] starts a rate decrease at the left-most 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 and the minimum number of bandwidth increases [8]. A CBA plan does not necessarily have the minimum number of bandwidth decreases, since the algorithm always selects the left-most 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[9,10] 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.

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 [24]; 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 algo-

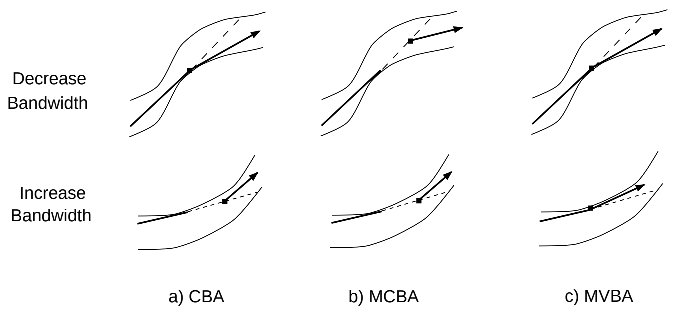


Figure 3 - Run Boundary Examples: These figures highlight the differences in how the CBA, MCBA, and MVBA algorithms choose the starting point for a bandwidth run, based on the trajectory for the previous run. For each bandwidth change, the MCBA algorithm searches along the frontier (dotted line) to maximize the distance reachable in the next run. In contrast, the CBA algorithm searches the frontier only on bandwidth increases; bandwidth decreases always start at the leftmost point along the frontier, where the previous run intersects the underflow curve. The MVBA algorithm selects the leftmost point along the frontier for both increases and decreases.

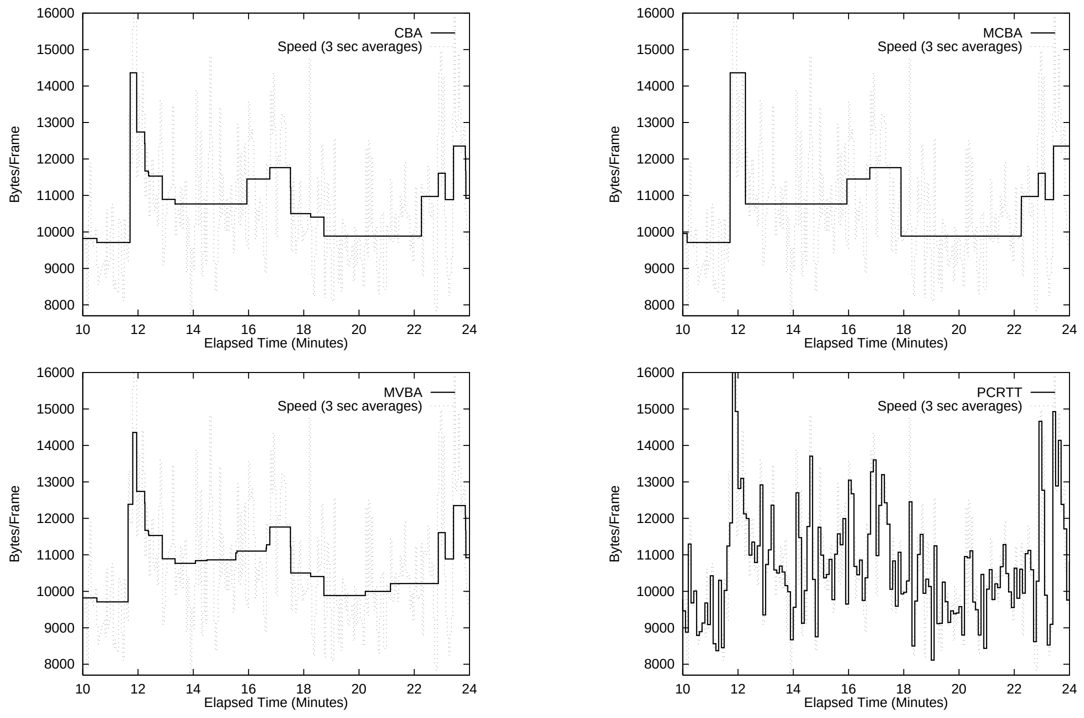


Figure 4 - Bandwidth Plans: These graphs show the transmission plans generated by four different bandwidth smoothing algorithms, applied to the movie *Speed* and a *1* megabyte client prefetch buffer. For the PCRTT algorithm, the graph shows the plan with the largest possible interval size that would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

rithm initiates bandwidth changes at the left-most 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 4. 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.

In contrast to the CBA, MCBA, and MVBA algorithms, the *piecewise constant rate transmission and transport* (PCRTT) algorithm[21] 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 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 ini-

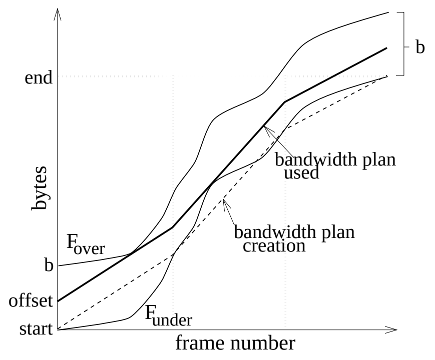


Figure 5 - PCRTT Plan Creation: This figure shows the creation of a PCRTT bandwidth allocation plan. First, the algorithm calculates the average frame size for each interval (dashed line). Then, the algorithm raises the plan to avoid buffer underflow. Based on the offset plan, the minimum buffer requirement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

tial 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 5. 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*. A more detailed comparison of these algorithms can be found in an upcoming comparison paper[13].

2.2 VCR-Functionality

In video-on-demand systems, we believe that users will typically change their access patterns during the playback of a video that fall into one of the four categories below:

- *Pause/Stop* - user stops the movies for a short time to answer a phone call, etc.
- *Rewind* - user rewinds the video to play back part of the video that they did not understand
- *Examine* - user stops the VCR to examine more closely a portion of the video. As an example, a user may be watching a football game and wants to see a certain play a couple of times in slow motion to see why it did or didn't work
- *Fast-forward* - user scans past parts of the video such as commercials in the program.

We expect that users may want all of these functions from a VOD system, although the actual distribution of access patterns within these categories may change. For example, consider the operation fast-forward which is frequently used to fast-forward through commercials. If all users are going to fast-forward scan past commercials, then it would defeat their purpose. We would expect that in such an environment that commercial messages would become a service, whereby, the video providers allow users to access commercials by companies on demand. Nevertheless, we should not omit fast-forward scans all together.

2.2.1 Supporting VCR Functionality

VCR interactions can be handled in one of several possible ways depending on the locality of the access when using the VCR functions. For accesses that are local to the point of playback (i.e. the consumption point), the client can service the request through the data that is sitting in the client-side smoothing buffer. For rewinds and examines that can be serviced through the smoothing buffer, the client can simply stop the flow of data from the server when the rewind function is initiated. Let this point be the *VCR initiation Point* (VIP). Then, the client is free to examine the buffered data at his or her own pace. When the client is finished, the flow of data from the server is restarted when the VIP is reached. Because the stop and start of the flow of data happen in the same state, only two interactions are required with the server without requiring the delivery of any additional data or having a larger peak bandwidth requirement. (Note that stops and pauses can be handled in the same way). We refer to the set of buffered frames for rewinds and examines as the *VCR-window*[11]. For systems where a large portion of request are stops, pauses, rewinds, and examines, maximizing the number of frames within the VCR-window is useful.

For accesses that are longer in nature, such as long fast-forwards or rewinds that exceed the bounds of the VCR-window, more complex interactions are required to support these functions. Much work has focused on supporting these VCR interactions from the video-on-demand servers [3, 5, 15, 25], where the clients and servers are tightly coupled. For long fast-for-

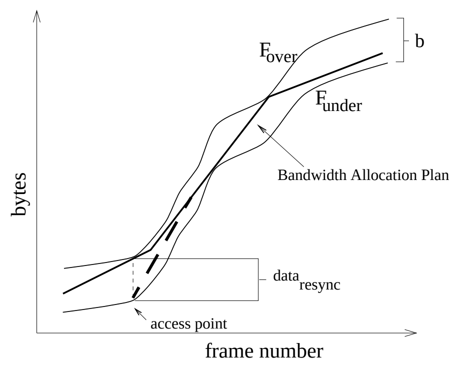


Figure 6 - Supporting VCR Functionality: This figure shows the result of a scan to "access point". With no overlap of data in the smoothing buffer, the distance between the bandwidth allocation plan and F$_{under}$ at the access point will need to prefetched before playback begins. As sample plan for making up the data$_{resync}$ at the access point is shown by the heavy dashed line.

wards and rewinds, the clients must always rely on these mechanisms. During the actual fast-forward or rewind, the video frame rate can be modified to fit within the channel that has been reserved for the client. Bandwidth smoothing environments, however, have an additional constraint in that they must handle the resynchronization of the new consumption point with the original bandwidth allocation plan. Consider the bandwidth allocation plan shown in Figure 6. Suppose that a long fast-forward has been used to move the consumption point to "access point". Let $data_{resync}$ be the difference between the bandwidth allocation plan and F$_{under}$ at the access point. Then, if playback is to start at the "access point", either the client will have to make up $data_{resync}$ before playback begins or will have to use excess channel capacity, such as contingency channels [4], to make $data_{resync}$. If the later approach is used, the excess channel capacity allocated must be large enough to avoid buffer underflow. More on this can be found in a recent Ph.D. thesis[12]. The main point, however, is that minimizing $data_{resync}$ minimizes the delay or excess channel capacity that is required to resynchronize the client with its original bandwidth plan. Note that resynchronization is necessary so that the original peak bandwidth requirements may be preserved.

3 Rate-Constrained Bandwidth Smoothing

Using the bandwidth smoothing algorithms presented in Section 2 results in plans for the delivery of that optimize the delivery of stored video under certain constraints. The CBA, MCBA, and MVBA algorithms result in plans that have the smallest peak bandwidth requirements, given a fixed buffer size. Using these algorithms, however, results in large buffer occupancy requirements, resulting in smaller VCR-window sizes as well as larger contingency channel allocation requirements. To more tightly couple the client and server, and thereby minimizing the buffer occupancy times, we introduce *rate-constrained bandwidth smoothing* (RCBS) for the delivery of stored video.

The RCBS algorithm results in plans for the delivery of stored video that have the smallest peak bandwidth requirements for a given buffer size as in the CBA, MCBA, and MVBA algorithms, but also maximizes the VCR-window size and minimizes the contingency channel allocation requirements for interactive playback. To implement this algorithm, a maximum bandwidth rate *r* is specified. Using this peak rate constraint, each frame, *i*, in the movie is examined. The bandwidth requirements for the preceding frames leading up to frame *i* are then modified such that the bandwidth requirement for frame *i* is *no greater* than the peak bandwidth requirement. In the event that frame *i* is less than the rate constraint *r*, then no preceding frames are modified.

Formally, let *r* be the maximum rate constraint requirement. Also, let *prefetch(i)* be the maximum frame *j* such that the average frame size from *j* to *i* is less than or equal to *r*. That is,

$$prefetch(i) = \left\{ \max \text{ j: } j \le i \text{ AND } \sum_{k=j}^{i} \frac{b_k}{(i-j+1)} < r \right\}$$

```
for (excess=0,i=n;  i>0  ;  i--){
   if (frame_size[i] > rate)              /*** Build up buffering because ***/
      excess += (frame_size[i]-rate);      /*** rate constraint violated ***/
      allocation[i] = rate;
   else
      if (excess > (rate - frame_size[i]))
         allocation[i] = rate;
         excess = excess - (rate - frame_size[i]);
      else
         allocation[i] = frame_size[i] + excess;
         excess = 0;
   }
```

Figure 7 - Rate-Constrained Bandwidth Smoothing Pseudo-Code: This figure shows the pseudo-code for the PCBS algorithm. The variable *excess* holds the amount of data that needs to be prefetched in order to maintain the rate-constraint *rate*. The second *if* clause does the actual distribution of the prefetched data into previous frame slots.

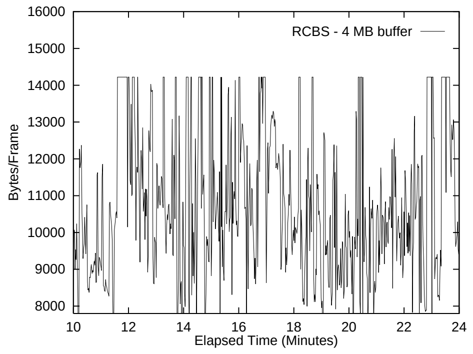


Figure 8 - RCBS Example: This figure shows a sample RCBS bandwidth allocation plan using the same parameters as found in Figure 4 for the movie *Speed*. Note only regions above the minimum peak bandwidth requirement of 14219 bytes/frame (for a 4 megabyte smoothing buffer) have been modified, leaving large portions of the video that are untouched.

Thus, *prefetch(i)* is simply the first frame for which prefetching must be used in order to keep the bandwidth under the maximum rate constraint *r.* Similarly, let *bw(j,i)* be defined as

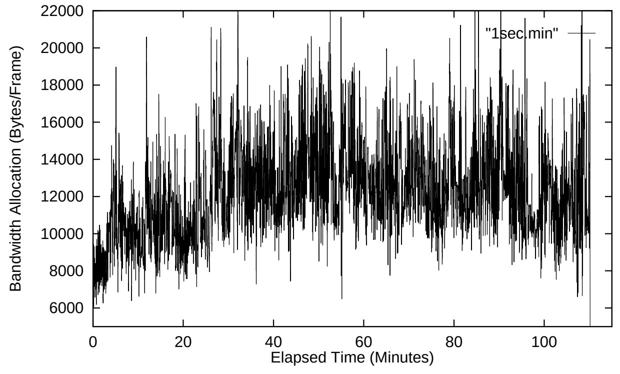
$$bw(j, i) = \begin{cases} 0 & \text{if prefetch(i)>j} \\ rem\left(\sum_{k=j}^{i} b_k, r\right) & \text{if prefetch(i)=j} \\ r & \text{if prefetch(i)<j} \end{cases}$$

This equation simply finds, given frames *j* and *i*, *j<i*, the bandwidth that would be needed on a frame *j* to keep the rate constraint until frame *i.* For the case where prefetch(i) equals j, bw(j,i) is the remainder from using rate *r* for frames *j+1* to *i.* Using these definitions, the bandwidth for any frame in the movie can then be written as:

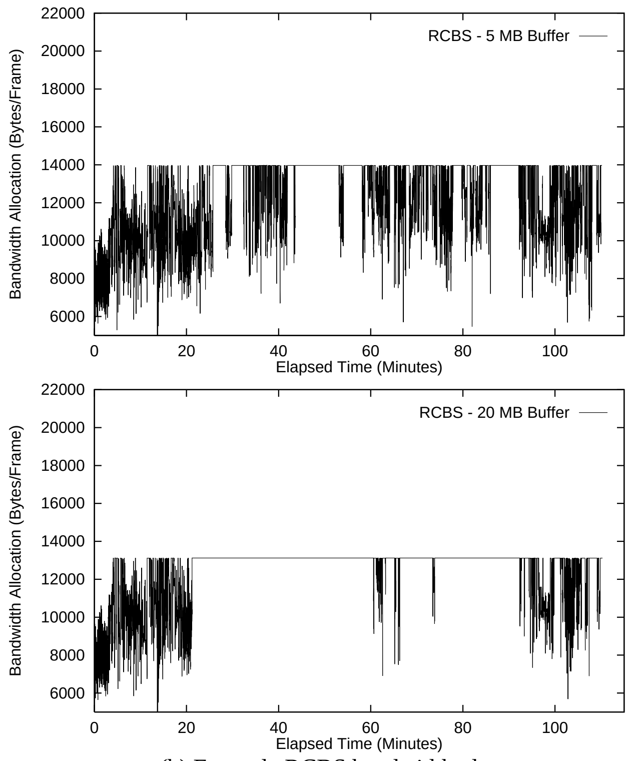
$$c(i) = \max_{i \le k \le n} bw(i, k)$$

From an implementation standpoint, the RCBS plan can be calculated in O(n) by working from the end of the movie to the beginning and keeping track of the excess data that needs to be prefetched in order to maintain the rate constraint. The pseudo-code for this algorithm can be found in Figure 7. An example plan similar to those in Figure 4 is shown in Figure 8. In comparing Figure 4 and Figure 8, we see that the RCBS algorithm has only modified a few regions within the segment shown. In particular, the largest rate-constrained region occurs around the 12 minute area within the movie. Alternatively, using the F_{under} and F_{over} curves, the algorithm can be described as always following the F_{under} unless a left turn (increase in bandwidth) would violate the rate constraint, in which case the plan deviates from F_{under} to *smooth* the delivery.

Using the RCBS algorithm results in several key properties. First, given a maximum rate constraint *r*, the RCBS uses the minimum size buffer. That is, no other plan that delivers all the video data can use less buffer while retaining the same rate



(a) *Speed* - 1 second frame averages



(b) Example RCBS bandwidth plans

Figure 9 - RCBS Examples: Figure (b) shows the RCBS algorithm for both a 5 and 20 MByte smoothing buffer for the M-JPEG encoded movie *Speed.* Figure (a) shows the 1 second frame averages for comparison. Note, because figure (a) is using 1 second frame averages, some smoothing has occurred.

constraint *r*. Second, because the algorithm prefetches only when the rate constraint will be violated, it results in the tightest coupling between the client and server plans. This results in resynchronization times that are the smallest given the maximum rate constraint *r.* A sample RCBS plan can be found in Figure 9. Notice that the algorithm appears to run a knife over the frames of the movie from left to right, filling in valleys of previous frames in order to meet the maximum rate constraint.

4 Experimentation

In this section we compare and contrast the various bandwidth smoothing algorithms. Several measures are important for interactive video-on-demand systems. One of the most important factors in bandwidth smoothing algorithms is the reduction in the peak rate during the video stream life time. By minimizing the peak bandwidth requirements, deterministic admission control is made somewhat simpler. Somewhat related to the peak bandwidth requirements is the variability of data that has to be handled. For interactive video-on-demand systems, maximizing the VCR-window size will limit the interactions that are required for resynchronizations, while tightly coupling the client and server limits the required contingency channel capacity required for longer fast-forward and rewind scans and searches.

To compare and contrast the RCBS algorithm with the MCBA and MVBA algorithms, we have digitized the full-motion picture *Speed.* This video was digitized using the MiroVideo DC1Tv video capture board and a Pentium based PC. The MiroVideo Capture board is a Motion-JPEG compression board containing the C-Cube Microsystems' Motion-JPEG chip, the CL550. The MiroVideo board can digitize frames at 640x480 pixels (at 30 frames per second) and then subsample them to 320x240 with guaranteed VHS picture quality. For the movie *Speed*, the resultant stream has an average bit rate of approximately 3 Megabits per second. Because the JPEG compression algorithm forms the basis of the MPEG I-frame, our Motion-JPEG encoded video stream is similar to an all I-frame encoded MPEG video stream.

In the comparison of the different algorithms, we have chosen to use the MVBA and the MCBA algorithms. The original MVBA and the MCBA algorithms represent two end approaches for bandwidth smoothing. As a result, the CBA algorithm

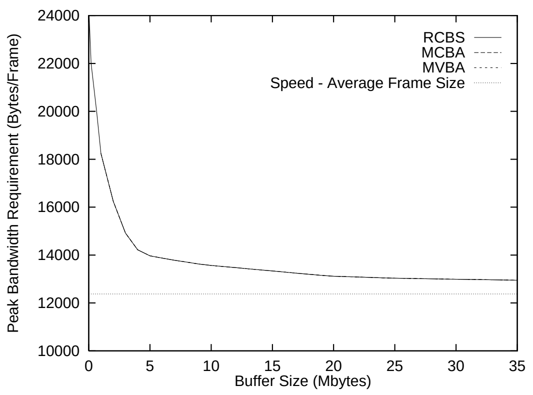


Figure 10 - Peak Bandwidth Requirements: This figure shows the peak bandwidth requirements for a fixed size buffer for the movie *Speed* for the RCBS, MCBA, and MVBA algorithms. The average frame size (12374 bytes) is also shown.

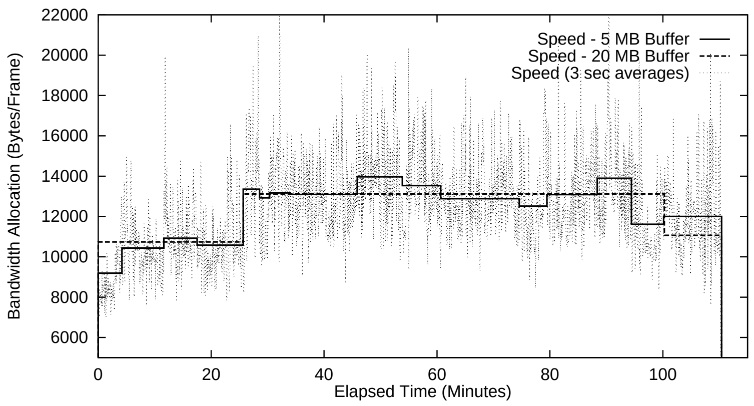


Figure 11 - Bandwidth Smoothing Example: This figure shows the MCBA algorithm for both a 5 and 20 MByte smoothing buffer for the Motion-JPEG compressed video *Speed.*

will always fall in between the MVBA and MCBA algorithms. The PCRTT algorithm does not result in the minimum peak bandwidth requirements because the prefetching of data is fixed by the interval size (in which the rate must be constant). In addition, because the interval sizes are fixed, the results are very unstable for larger buffer sizes.

4.1 Peak Bandwidth Requirements

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. Figure 10 plots the peak bandwidth requirements for the MCBA, MVBA, and RCBS algorithms. As shown by the figure, all three algorithms result in the same peak bandwidth requirements.

Under small buffer sizes (less than 3 MBytes), much of the short-term burstiness is removed, typically prefetching data no more than several scenes in advance. As the smoothing buffer is increased above 4 MBytes, the extra buffer is typically used to smooth only several areas of large bursts of frames. As an example, consider the MCBA graphs shown in Figure 11. With a 5 MB smoothing buffer, the much of the burstiness has been removed. Adding 15 additional MBytes of smoothing buffer is used in the prefetching of 5 large areas of larger frames. The RCBS algorithm results in somewhat similar results as shown in Figure 9. For a 5 MB smoothing buffer, there are approximately three to five areas where the maximum rate *r* is used for any substantial length. As the buffer size is increase, these areas can no longer be used as prefetching (as they are already at the maximum rate).

4.2 Bandwidth Variation

In addition to the peak bandwidth requirements for video playback, the variability of bandwidth requirements can also be important. As shown in Figure 12, the variability of the RCBS algorithm is higher than both the MCBA and MVBA algo-

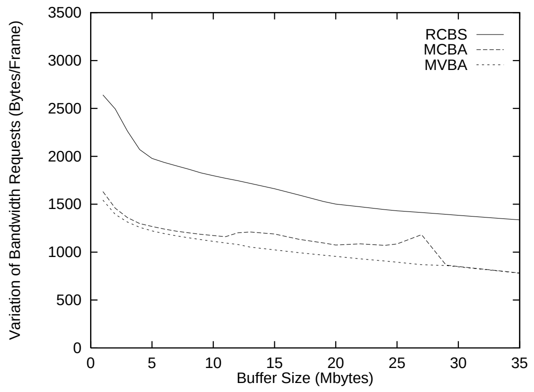


Figure 12 - Bandwidth Variability: This graph shows the variability of bandwidth requirements for the RCBS, MCBA, and MVBA algorithms for the M-JPEG compressed video *Speed.*

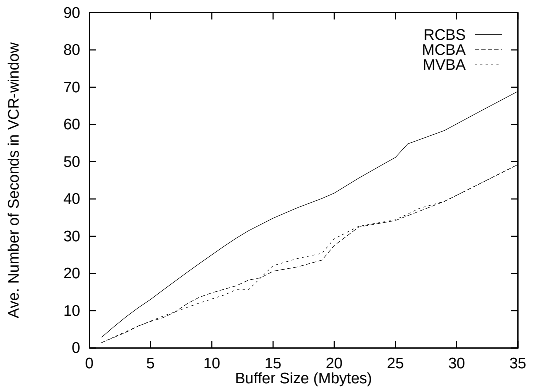


Figure 13 - VCR-window Size: This figure shows the average amount of video that is available in the VCR-window for use in rewinds and examines. The RCBS algorithm typically results in 50% more video in the rewind buffer than using the MCBA and MVBA algorithms.

rithms. This is not entirely unexpected as the RCBS algorithm only smooths large frames. As a result, many frames under the bandwidth constraint *r* do not get smoothed resulting in greater variability of the bandwidth usage. The MCBA algorithm exhibits similar behavior as the MVBA algorithms for very large and very small buffer sizes. For small buffer sizes, both algorithms are limited by the small buffer sizes (which limits the amount of data that can be prefetched) and therefore results in somewhat similar plans. For very large buffer sizes, both algorithms are limited by the amount of data that they can prefetch for only a several areas within the movie, again resulting in plans that are somewhat similar. For medium size buffers (10 to 27 Mbytes), the algorithms have greater latitude in what they can prefetch resulting in slightly different results. In general the RCBS algorithm results in bandwidth variabilities that are 50 to 70% larger than the MCBA and MVBA algorithms. It is important to remember that these algorithms exhibit the same *peak* bandwidth requirements, and that the RCBS algorithm adds variability from not smoothing only the smaller frame sizes.

4.3 VCR-window size

For systems where a large portion of the VCR interactions are stop, pause, rewind, and examine, maximizing the size of the VCR-window can be useful. As shown in Figure 13, the RCBS algorithm results in a larger VCR-window than the MCBA and MVBA algorithms. Typically, the RCBS algorithm results in a 50 to 90% more frames in the VCR-window than the MCBA and MVBA algorithms. As examples, with 5 MBytes of buffering, the RCBS algorithm has 80% and 82% more seconds of video in the VCR-window than the MCBA and MVBA algorithms, respectively. With 20 MBytes of buffering, the RCBS algorithm has 51% and 42% more seconds of video in the VCR-window than the MCBA and MVBA algorithms, respectively. The results shown in Figure 12 are for M-JPEG compressed video. If MPEG video compression that takes advantage of inter-frame redundancy is used, these VCR-times are expected to increase by an order of 4 to 7. As a result, using a 5 MByte buffer with MPEG would result in having an average VCR-window size of approximately one minute to one and a half minutes of video instead of 15 seconds.

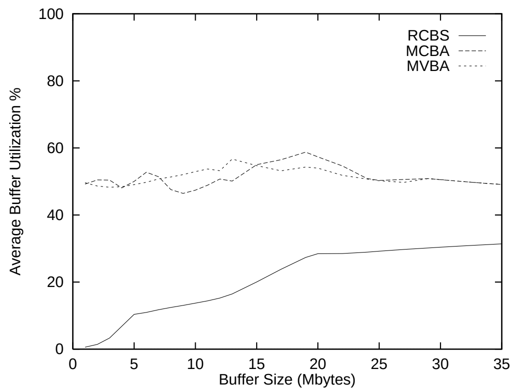


Figure 14 - Buffer Utilization: This figure shows the average buffer utilization of the client-side buffer, a measure of how tightly coupled the client and server are. The graph was generated by using the RCBS, MCBA, and MVBA and calculating the various plans for the M-JPEG compressed movie *Speed.*

4.4 Buffer Utilization

For interactive video that accesses points outside of the VCR window, how tightly coupled the client and server are determines the amount of extra data ($data_{resync}$) that must be made up. One measure of this coupling is the buffer utilization of the client-side buffer. With lower utilizations, the amount of data that is expected to be made up for accesses outside the VCR will decrease. Figure 14 shows the average buffer utilization for the RCBS, MCBA, and MVBA algorithms. As shown in the figure, the MCBA and MVBA algorithms hover around 50% utilization. This merely emphasizes the fact that the MCBA and MVBA algorithms smooth both large and small frame sizes. The RCBS algorithm has very low utilizations for small buffer sizes, mainly due to the large peak bandwidth requirements for small buffer sizes, leading to very little prefetched data. As the client-side buffer is increased, the buffer utilization approaches approximately 30%. As a result, the RCBS algorithm results in very short resynchronization times for small buffer sizes. For larger buffer sizes, the RCBS algorithm results in approximately half the resynchronization time than the MCBA or MVBA algorithms.

5 Conclusion

Bandwidth smoothing techniques are attractive for use in interactive video-on-demand systems because they can minimize the *peak* bandwidth requirements, reducing the burstiness exhibited as well as making admission control simpler. For systems that allocate bandwidth based on just the *peak* bandwidth requirements (to ensure that in the worst case that the user will get all its data), creating bandwidth plans that aggressively prefetch to maximize the utilization of the client-side smoothing buffer may not always be necessary.

In this paper, we introduced the *rate-constrained bandwidth smoothing* algorithm, which takes a less aggressive approach to smoothing the bandwidth requirements. Given a maximum rate constraint, the rate-constrained bandwidth smoothing algorithm minimizes the amount of prefetched data (and hence the client-side buffer size) for the continuous playback of video. By minimizing the amount of prefetched data, the support of more complex VCR functions is possible without long delays or large excess bandwidth requirements for resynchronization. Our results show that the rate-constrained bandwidth smoothing algorithm results in the same peak bandwidth requirements for a given size buffer as the MCBA and MVBA algorithms. By tightly coupling the point of consumption and the point of playback, the amount of data in the VCR-window can be increased on the order of 50 to 90%.

The main drawback of the RCBS algorithm is that it requires the server to send at more variable rates when the frame sizes are less than the rate constraint. To limit the variability, smaller windows of smoothing can be used to minimize the variability at a smaller time scale, making the client and server somewhat more decoupled. Because most video-on-demand servers process data based on periodic intervals (typically 1/30th of a second or less), this variability may not be a severe limitation.

6 References

1. D. Anderson, Y. Osawa, R. Govindan, "A File System for Continuous Media", *ACM Transactions on Computer Systems*, Vol. 10, No. 4, Nov, 1992, pp. 311-337.

2. C.M. Aras, J.F. Kurose, D.S. Reeves, H. Schulzrinne, “Real-time Communication in Packet Switched Networks”, *Proceedings of the IEEE*, Vol. 82, No. 1, pp. 122-139, Jan. 1994.

3. M.S. Chen, D.D. Kandlur, P.S. Yu,“Support for Fully Interactive Playout in a Disk-Array-Based Video Server’’, In *Proceedings of ACM Multimedia 1994*, San Francisco, CA, Oct. 1994, pp. 391-398.

4. A. Dan, D. Sitaram, P. Shahabuddin,“Scheduling Policies for an On-Demand Video Server with Batching”, In *Proceedings of ACM Multimedia 1994*, San Francisco, CA, Oct. 1994, pp. 15-23.

5. J. Dey-Sircar, J. Salehi, J. Kurose, D. Towsley,“Providing VCR Capabilities in Large-Scale Video Servers’’, In *Proceedings of ACM Multimedia 1994*, San Francisco, CA, Oct. 1994, pp. 25-32.

6. C. Federighi, L. Rowe,“A Distributed Hierarchical Storage Manager for a Video-on-Demand System”, In *Proceedings of 1994 IS&T/SPIE Symposium on Electronic Imaging: Science and Technology*, San Jose, CA Feb. 1994.

7. W. Feng, S. Sechrest,“Smoothing and Buffering for Delivery of Prerecorded Compressed Video”, In *Proceedings of IS&T/SPIE Multimedia Computing and Networking* , Feb. 1995, San Jose, CA, pp. 234-242.

8. W. Feng, S. Sechrest,“Critical Bandwidth Allocation for the Delivery of Compressed Prerecorded Video”, *Computer Communications*, Vol. 18, No. 10, Oct. 1995, pp. 709-717.

9. W. Feng, F. Jahanian, S. Sechrest,“Optimal Buffering for the Delivery of Compressed Prerecorded Video’’, In *Proc. of IASTED International Conf. on Networks*, Jan. 1996, Orlando, Florida.

10. W. Feng, F. Jahanian, S. Sechrest,“An Optimal Bandwidth Allocation Strategy for the Delivery of Compressed Prerecorded Video”, To appear *ACM/Springer-Verlag Multimedia Systems Journal*.

11. W. Feng, F. Jahanian, S. Sechrest,“Providing VCR Functionality in a Constant Quality Video-On-Demand Transportation Service”, In *Proc. of 3rd IEEE Inter. Conf. on Multimedia Computing and Systems*, Hiroshima, Japan, June 1996.

12. W. Feng,“Video-on-Demand Services: Efficient Transportation and Decompression of Variable Bit Rate Video’’, Ph.D. Thesis, University of Michigan, April 1996.

13. W. Feng, J. Rexford,“A Comparison of Bandwidth Smoothing Techniques for the Transmission of Prerecorded Compressed Video’’, In *Proc. IEEE INFOCOM 1997*, April 1997, Kobe, Japan.

14. D.J. Gemmell, H.M. Vin, D. Kandlur, P.V. Rangan, L.A. Rowe,“Multimedia Storage Servers: A Tutorial”, *IEEE Computer*, Vol. 28, No. 5, May 1995, pp. 40-49.

15. Pawan Goyal, Harrick M. Vin,“Network Algorithms and Protocol for Multimedia Servers”, In *Proceedings of INFOCOM 1996*, San Francisco, CA, March 1996, pp. 1371-1379.

16. 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.

17. D. Kandlur, M. Chen, Z.Y. Shae,“Design of a Multimedia Storage Server” , In *IS&T/SPIE Symposium on Electronic Imaging Science and Technology*, San Jose, CA, Feb. 1994.

18. D.J. LeGall, ‘‘A Video Compression Standard for Multimedia Applications,’’ *Communications of the ACM,* Vol. 34, No. 4, (Apr. 1991), pp. 46-58.

19. 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.

20. P. Lougher, D. Shepherd,“The Design of a Storage Sever for Continuous Media”, *The Computer Journal*, Vol. 36, No. 1, Feb. 1993, pp. 32-42.

21. J. M. McManus, K. W. Ross,“Video on demand over ATM: Constant-rate transmission and transport,” In *Proceedings of IEEE INFOCOM*, pp. 1357-1362, March 1996.

22. P. Venkat Rangan, H.M. Vin,“Designing File Systems for Digital Video and Audio’’, In *Proceedings of the 13th ACM Symposium on Operating Systems Principles*, Operating Systems Review, Vol. 25, No. 5, October 1991, pp. 81-94.

23. 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, December 1993.

24. 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.

25. P. J. Shenoy, H. M. Vin, “Efficient Support for Scan Operations in Video Servers”, In P*roceedings of the 3rd ACM Conference on Multimedia*, October, 1995.

26. H. Zhang, S. Keshav,“Comparison of Rate-Based Service Disciplines”, In *Proceedings of ACM SIGCOMM,* Sept. 1991, pp. 113-121.