# A Movie Approximation Technique for the Implementation of Fast Bandwidth Smoothing Algorithms

Wu-chi Feng, Ming Liu, Chi Chung Lam
Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210

***Abstract***

Bandwidth smoothing algorithms have been shown to be effective in reducing the network resource requirements for the delivery of compressed video streams. For stored video, a large number of bandwidth smoothing algorithms have been introduced that are optimal under certain constraints. These algorithms, however, require access to all the frame sizes to achieve their optimal properties, which, in turn, can limit responsiveness of video-on-demand systems especially under VCR functions. In this paper, we introduce a *movie approximation* technique for the representation of the frame sizes, reducing the complexity of the bandwidth smoothing algorithms and reducing the amount of data and computation that must be transmitted prior to the start of playback. Our results show that the proposed approximation technique can accurately approximate the frame sizes with a small number of linear segments *without* affecting the performance measures that the bandwidth smoothing algorithms are attempting to achieve by more than 1%. In addition, we show that implementations of this techniques speed up execution times by 100 to 1400 times, with typical approximated bandwidth smoothing plan calculations for full-length movies requiring less than 20 milliseconds. Evaluation using both a Motion-JPEG and MPEG video clip are provided.

*keywords: bandwidth smoothing, video-on-demand, MPEG, video networking*

## 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 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 [17, 14]. 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 server can reduce the burstiness of prerecorded video, while avoiding both underflow and overflow of the client buffer, by employing a bandwidth smoothing algorithm. For stored video applications, these bandwidth smoothing plans can be created that exhibit optimal properties. Among these include bandwidth smoothing algorithms that minimize the peak bandwidth requirement for the delivery of the video while minimizing the number of increases, total number of changes, the variability of bandwidth requests, or the utilization of the smoothing buffer (to name a few) [4, 6, 18].

To achieve their uniquely optimal properties, these algorithms require a *map* of all the frame sizes that a com-

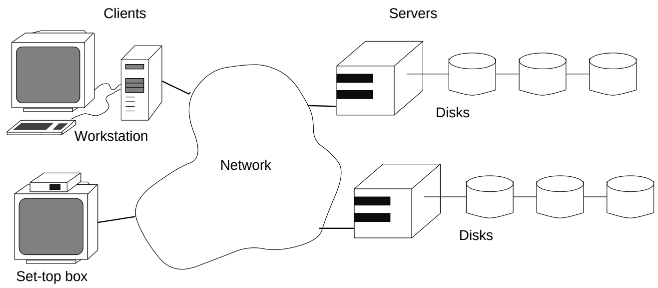



**FIGURE 1: Video-On-Demand Architecture.** This figure shows a basic video-on-demand architecture consisting of video servers, a communication network, and client sites. Possible clients include workstations and set-top boxes that contain hardware to interact with the network and a disk (or RAM) that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

pressed stream consists of in the calculation of the bandwidth plan. For interactive video-on-demand systems, this particular requirement may be difficult to implement for several reasons. First, for client calculated bandwidth plans, it requires the transmission of all the frame of a movie (typically on the order of half a megabyte). Second, it takes time to calculate the bandwidth plan. For a single calculation of the bandwidth done before the start of the movies, this calculation time may be acceptable. For interactive video-on-demand systems, however, VCR invoked functions may require the re-calculation of a new bandwidth plan on the fly, making this delay unacceptable.

In this paper, we introduce a movie approximation technique for modeling the *a priori* information that bandwidth smoothing algorithms require in the calculation of bandwidth smoothing plans. This approximation algorithm creates an approximated frame size curve that the video-on-demand server can use in place of a detailed map of the frame sizes for a particular video, minimizing the video-on-demand resources that these maps require (both computational and memory). Upon a movie request, the video-on-demand server transmits the much *smaller* movie approximation file to the client. The client can then quickly calculate the required bandwidth resource both at the start of the data flow and when VCR functions are invoked. Our results show that we can reduce the amount of data needed to represent a movie by an order of magnitude with very little change (< 1%) in the optimal properties that these bandwidth smoothing techniques strive for. In addition, we have found that implementations of this technique typically run 100 to 1000 times faster, reducing the run times to less than 20 milliseconds for a full-length movie.

In the next section, we briefly review some of the related work and background for bandwidth smoothing techniques. In Section 3, we introduce our movie approximation algorithm, including a discussion of how to implement efficient bandwidth smoothing algorithms with this model. In Section 4, we evaluate the movie approximation algorithm using an MPEG encoded movie and a Motion-JPEG encoded movie. In particular, we highlight how the approximation algorithms affect performance metrics such as peak bandwidth requirements, variability, and number of rate changes. Finally, we summarize our findings in Section 5.

## 2. Background and Related Work

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, 12, 13, 16] and may also include tertiary storage, such as tapes or optical jukeboxes, for holding less frequently requested data [3]. 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, 19], 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.

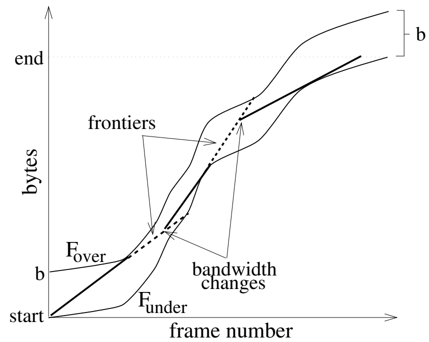



**FIGURE 2: Sample Transmission Plan.** This figure shows the buffer underflow and overflow curves for a sample video stream. The plan consists of three constant-rate runs that serve as a schedule for transmitting video frames from the server.

## 2.1. Bandwidth Plans

A compressed video stream consists of $n$ frames, where frame $i$ requires $f_i$ bytes of storage. To permit continuous playback 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

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

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.

Given these functions, bandwidth smoothing techniques can minimize several performance metrics. In particular, most bandwidth smoothing algorithms *minimize* the peak bandwidth requirements in addition to minimizing some other metric such as:

- the number of bandwidth increases [4, 5],
- the total number of bandwidth changes [6, 8],
- the variability of bandwidth requests [18], or
- the smoothing buffer utilization [9].

We refer readers interested in bandwidth smoothing techniques for the delivery of stored video to a comparison paper [10]. We have graphed several example bandwidth smoothing plans in Figure 3.

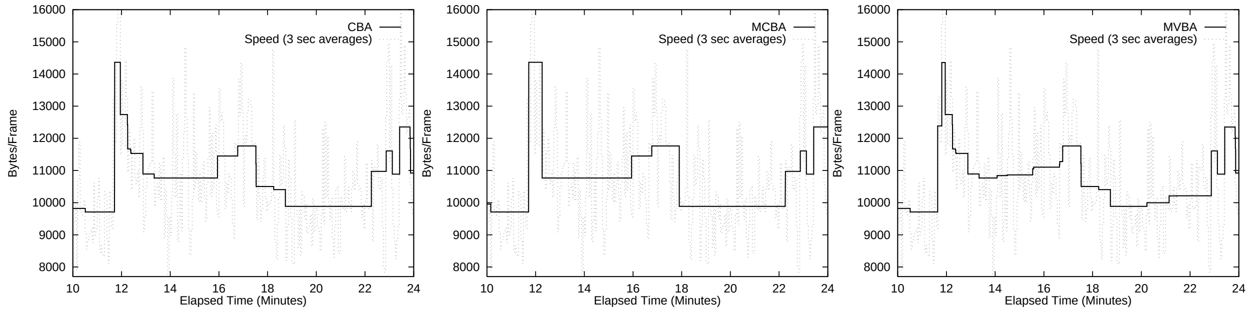



**FIGURE 3: Bandwidth Plans.** These graphs show the transmission plans generated by four different bandwidth smoothing algorithms, applied to the movie *Speed* and a *1* megabyte client prefetch buffer.

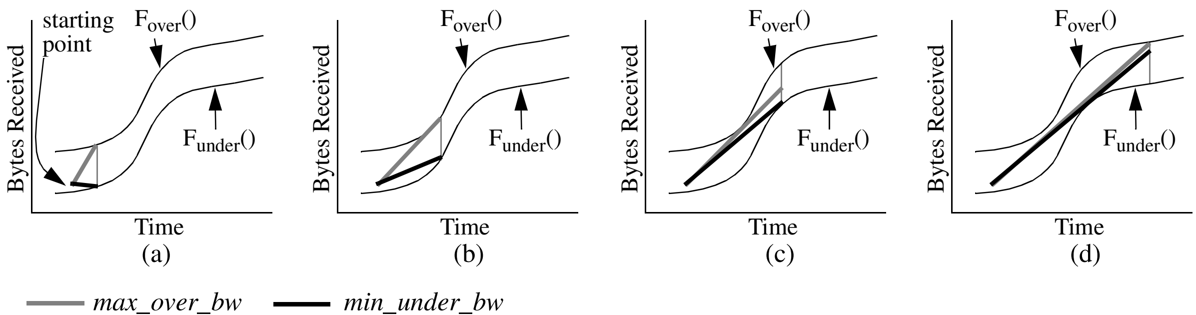



**FIGURE 4: Critical Bandwidth Run Calculation.** This figure shows an example critical bandwidth allocation run calculation. Figure (a) shows the early calculations where a large set of possible bandwidth runs exist (between *min_under_bw* and *max_over_bw*). As the calculation advances through the frames, *min_under_bw* and *max_over_bw* continue to converge (Figures (b) and (c)). Figure (d) shows the state of the calculation just prior to the end of the critical bandwidth calculation, and, hence, the selected bandwidth requirement.

## Creating Bandwidth Smoothing Plans

Many of the bandwidth smoothing algorithms create bandwidth plans by selecting critical bandwidth runs (as presented at this conference in 1995) [4]. Given a fixed size buffer, the movie functions $F_{under}()$ and $F_{over}()$, a starting point, and an initial buffer occupancy, the critical bandwidth algorithm selects the bandwidth such that the run extends as far into the future as possible. In the calculation of a bandwidth plan, the algorithm keeps track of two variables:

- *min_under_bw* - the minimum bandwidth required to avoid buffer underflow
- *max_over_bw* - the maximum bandwidth that will avoid buffer overflow

Given the starting point and the initial buffer occupancy (distance from $F_{under}()$), the critical bandwidth based algorithms examine the frames in the future one at a time, until some frame *j* which causes

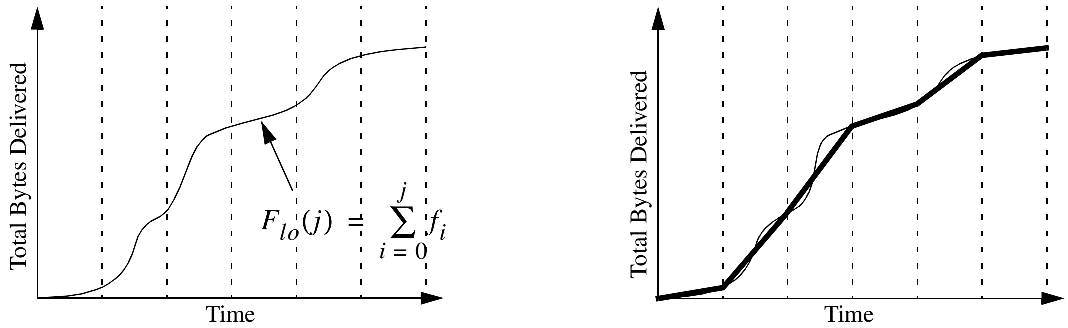
$$max\_over\_bw < min\_under\_bw \quad .$$

This frame *j* is then *unreachable* given the buffer requirement and the initial starting point. The critical bandwidth based algorithms then differ on how they select the starting point for the next run, giving them each their unique optimal properties.

As an example, we have graphed a sample critical bandwidth algorithm calculation in Figure 4. We will refer back to this figure in the implementation subsection for the movie approximation algorithm.

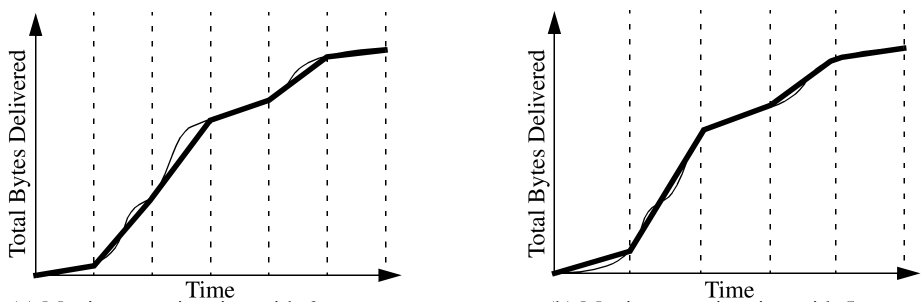
## 3. A Movie Approximation Algorithm

In this section, we introduce a movie approximation algorithm and show how to create fast implementations of bandwidth smoothing algorithms using this movie approximation. As previously mentioned, our approach is to




(a) Actual *a priori* knowledge (b) Approximated *a priori* knowledge

**FIGURE 5: Modeling Frame Sizes via Averaging.** Figure (a) shows the accumulated bandwidth required for the delivery of the compressed video stream. We can approximate this function by using a window of average frame size requirements. Here, the movie is split into 6 groups and is modeled with 6 discrete line segments (Figure (b)).




(a) Movie approximation with 6 segments (b) Movie approximation with 5 segments

**FIGURE 6: Averaging and Errors in Movie Approximations.** While one would expect the amount of error introduced by movie approximating algorithms to increase with increasing window sizes (or decreasing number). Figure (b) shows one such example when this trend is not true.

model the data deterministically (rather than statistically) and then generate bandwidth smoothing plans based on the resulting model. The model is obviously going to be a trade-off between accuracy and the amount of data required to model the video. At the extremes, a model with one point for each frame in the movie results in an *exact* duplicate of the video frame sizes, while a model with two points for the entire movie results would yield a very poor model of the video.

In order to approximate the movie data, several simple techniques may be used. The most obvious technique would be to break the movie into many small fixed size groups and approximate the movie using the averages for these frame. A sample approximation curve is shown in Figure 5. Because the data is now approximated, some error in calculating the bandwidth plan is now introduced. As an example, the third segment of Figure 5 (b) has relatively the largest error in the approximation. The use of averaging for approximating the data, however, can result in some difficulties. When modeling the data, one would expect that the number points used in the approximation to be *inversely* proportional to the error introduced, but it is not always the case. As an example consider the example from Figure 5. In Figure 6, we have graphed the same bandwidth plan with one fewer segments, which results in a model that has less overall error than the approximation from Figure 5. This particular phenomena makes it difficult to translate maximum error tolerance into a fixed number of segments in a model.

To alleviate this affect, we chose to use the *minimum changes bandwidth allocation* algorithm for the creation of the movie model. When used as a bandwidth smoothing algorithm for the delivery of a compressed video stream, it has the unique property that, for a given fixed size buffer, results in a plan for the delivery of video that has the minimum number of rate changes for continuous delivery. To model a video stream, we select a maximum amount of error that the user is willing to tolerate. (We will discuss the error and its effect on network performance metrics until the Evaluation section). We then use the functions:

$$F_{under}(j) = \sum_{i=0}^{j} f_i \qquad \text{AND} \qquad F_{over}(j) = \sum_{i=0}^{j} f_i + e$$

where $f_i$ is the size of frame $i$ in bytes and $e$ is the maximum tolerable error. We then run the *minimum changes bandwidth allocation* (MCBA) algorithm to create a movie approximation function. The resulting plan creates a delivery plan that stays within the river outlined by $F_{under}$ and $F_{over}$.

## 3.1. Making the Movie Model

As just described the MCBA algorithm creates a plan that has the fewest number of rate changes for a given error buffer size. We then store the plan as a set of pairs:

```
<run length (in frames) , bandwidth requirement>
```

Using the run length allows us to save on the space required to represent the movie because the run lengths are small relative to the number of frames within a movie. The movie model is then stored at the video server and used instead of all the individual frame sizes for a particular movie, reducing the control traffic that the video-on-demand server must deliver to the client. In addition, the client memory requirements can be reduced and the time to calculate bandwidth plans can be reduced.

## 3.2. Implementing Bandwidth Smoothing Algorithms Using the Movie Approximation Function

Having a movie model requires a method for implementing bandwidth smoothing algorithms. In this subsection, we show how to create fast approximations for bandwidth smoothing plans using the movie models. In particular, we describe how to calculate a single critical bandwidth allocation run as described in the background section. Let:

- *pt*[i] be the starting point of the *ith* run in the movie approximation model
- *sum*[i] be the total amount of data that should have been received by the start of the *ith* run in the movie approximation model.
- *b* be the client buffer size
- *e* be the maximum error introduce in the movie approximation model

For a run that starts at point *j*, let $bytesRec_j$ equal the total amount of data received up to point *j*. That is $bytesRec_j$ and *j* tells us where in the river we are when we start the critical bandwidth run calculation. Let *k* be the first *pt*[*k*] that is greater than *j*. We then calculate the critical bandwidth allocation run by finding the largest value of *m* such that:

$$max\_over(j, k, m) \geq min\_under(j, k, m)$$

where *max_over(j,k,m)* and *max_under(j,k,m)* are defined as:

$$max\_over(j, k, m) = \min_{k \leq l \leq m} \left( \frac{sum[i] + b - e - bytesRec_j}{pt[l] - j} \right)$$

$$min\_under(j, k, m) = \max_{k \leq l \leq m} \left( \frac{sum[l] - bytesRec_j}{pt[l] - j} \right)$$

These functions simply define the maximum bandwidth that will not cause overflow of the buffer and the minimum bandwidth that will not cause buffer underflow, respectively.

Once an appropriate *m* has been identified, if *max_over(j,k,m+1)* is less than *max_over(j,k,m)*, then a bandwidth decrease is required in the next run. The critical bandwidth for the current run is then *min_under(j,k,m)* and the critical point is the *l* that caused *min_under(j,k,m)*. For algorithms like the MCBA that require the point where this run would cause buffer overflow, we can calculate the end as follows. Let the *start_value* defined as:

$$start\_value = bytesRec_j + min\_under(j, k, m)(pts[m] - j)$$

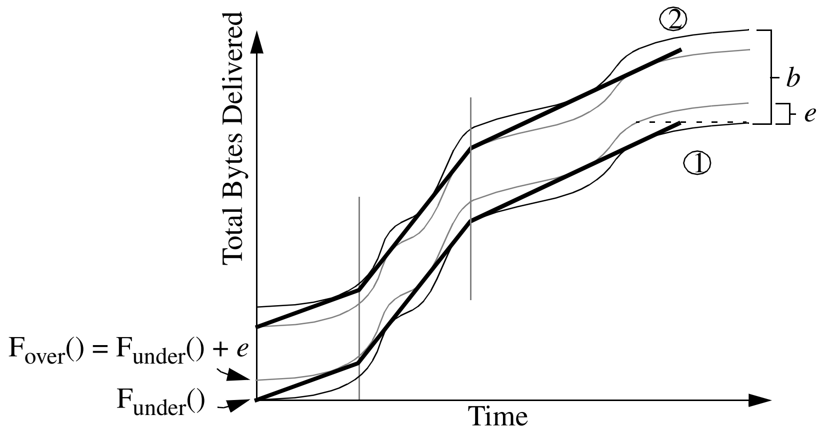



**FIGURE 7: Bandwidth Smoothing Plans with the Movie Model.** This figure shows the creation of a bandwidth smoothing plan where the data has been modeled with our movie model. (1) shows the bandwidth curves that are used in the creation of the movie approximation model with error buffer *e*. (2) shows the displacement that is necessary by the bandwidth smoothing plan in order to avoid buffer underflow and overflow due to the approximation process for the movie model. The grey vertical lines represent the evaluation points that are used in the calculation of the bandwidth smoothing plans.

Then the point where the run crosses the movie approximation function is:

$$end\_value = \frac{sum[m] + b - e - start\_value}{min\_under(j, k, m) - bw[m]} + pt[m]$$

Otherwise, we need a bandwidth increase in the next run (That is, *max_under*(*j,k,m+1*) > *max_under*(*j,k,m*)). In this case, the critical bandwidth for the run is *max_over*(*j,k,m*) and the critical point is the *l* that caused *max_over*(*j,k,m*). For the calculation of the end point (for algorithm such as the MCBA or CBA algorithms), the *start_value* is:

$$start\_value = bytesRec_j + min\_over(j, k, m)(pts[m] - j)$$

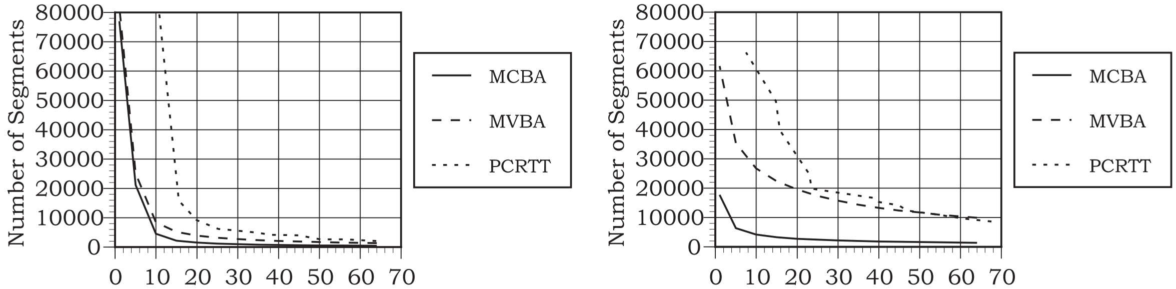
and the end value is:

$$end\_value = \frac{start\_value - sum[m]}{bw[m] - min\_over(j, k, m)} + pt[m]$$

There are a couple of points worth mentioning about the critical bandwidth run creation. First, in the *max_over* function, the upper curve is offset from the lower curve (*sum*[*l*]) by *b-e*. This particular point is best described by example. In Figure 7, we have drawn the approximation function with heavy solid lines. We then note that as the approximation gets farther away from the $F_{under}()$ function, it gets closer to the $F_{over}()$ function. As a result, the total error between $F_{under}()$ and $F_{over}()$ is always exactly *e*. Second, th calculation of the bandwidth run is O(*n*) in the number of points in the approximation, whereas the critical bandwidth allocation algorithm is O(*n*) in the number of frames in the movie. As a result, instead of evaluating the underflow and overflow curves on every frame as shown in Figure 4, we need to evaluate the underflow and overflow points at the points stored in the movie approximation file.

## 4. Evaluation

In the evaluation of our approximation technique, we are interested in answering three questions: *How accurate is the approximation algorithm in the number of points used in the approximation?, How does the approximation algorithm affect current bandwidth smoothing algorithms?,* and *What improvement in performance does the approximation buy us?* In answering the second question, we will evaluate the algorithms based on the peak bandwidth requirement, number of bandwidth changes, and the variability that the approximated bandwidth plans result in. We will evaluate both the *minimum changes bandwidth allocation* (MCBA) algorithm and the *minimum variability bandwidth allocation* (MVBA) algorithm, using the *Star Wars* MPEG-encoded video clip from Bellcore [11] and the *Speed* Motion-JPEG encoded video clip from Ohio State[1] [7].




(a) *Star Wars* (b) *Speed*

**FIGURE 8: Model Approximation Size Versus Accuracy in Model.** This graph shows the number of segments required for a given fixed size error buffer. The error buffer is the maximum deviation from the movie data that is allowable.

The *Star Wars* video clip is MPEG compressed using the frame pattern *IBB PBB PBB PBB IB...,* with 12 frames between each *I*-frame. It also has a maximum frame size of 23,160 bytes and a mean frame size of 1950 bytes. The video sequence consists of 174136 frames. The *Speed* Motion-JPEG compressed video stream is very similar to an all I-frame encoded video stream (with no inter-frame compression). The video consists of 198745 frames, has a maximum frame size of 29485, and has an average frame size of 12,374 bytes (nearly 10 times larger than *Star Wars*).

## 4.1. Number of Points in Approximation Model Versus Accuracy in Buffer

One of the primary concerns with using a movie approximation model instead of the actual movie frame data is the trade-off that the approximation makes in terms of accuracy. If the approximation requires just as much data as the original movie size but is not very accurate (in terms of error), then the model would be pretty useless. Alternatively, if the model can greatly reduce the amount of data required *without* sacrificing much accuracy, then it would be a good thing to use.

To determine the error buffer versus accuracy trade-off, we selected three algorithms to use for the model: the piece-wise constant transport and transmission (PCRTT), the minimum variability bandwidth allocation (MVBA) algorithm, and the minimum changes bandwidth allocation (MCBA) algorithm. Using these bandwidth allocation algorithms, we calculated bandwidth allocation algorithm given a fixed error buffer size and determined the number of points that would be necessary to represent the plan. For our purposes, the PCRTT algorithm is essentially the averaging algorithms offset vertically to avoid buffer underflow [15]. As mentioned in Section 3, averaging algorithms are somewhat unstable in terms of the buffer requirements, given a fixed window size. To make the PCRTT graph, we ran the algorithm for window sizes of 2 to 200. We then filtered out points where a larger window size with smaller error was available.

As shown in Figure 8, the MCBA algorithm results in the smallest number of segments for a given error buffer size, as expected. An interesting trend in these graphs is that the movie *Star Wars* has a much steeper curve for very small buffer sizes and relatively flatter curve for larger buffer sizes than the movie *Speed.* The main reason for this phenomena is that the MPEG video stream consists of three frame types (I, P, and B). As a result, on a much smaller time scale, the MPEG video stream exhibits significantly more burstiness due to the frame pattern type. Once this frame burstiness has been removed, additional buffering is used to remove medium and long term burstiness. The asymptotic slope is also flatter for the movie *Star Wars* than the movie speed due to the difference in the average bit rate of the stream. With an error buffer of approximately 8 kilobytes, the MCBA and MVBA algorithms achieve an

1. A Motion-JPEG video library consisting of 20 full-length movies is available via the web at: http://www.cis.ohio-state.edu/~wuchi

order of magnitude fewer required points than representing the data by frame (recall, *Star Wars* consists of approximately 174000 frames.

For the movie *Speed*, the MVBA and PCRTT algorithms result approximately 6 to 8 times the number of required segments than the MCBA algorithm. Using the MCBA algorithm for the movie approximation model gives us an order of magnitude reduction in the number of points required to represent the movie with a maximum error of only 1 kilobyte. We can achieve two orders of magnitude reduction using a 40 kilobyte error buffer. For very small buffer sizes, the PCRTT algorithm results in more segments than the MVBA algorithm, due to the fact that the PCRTT algorithm is driven by one worst case window (recall, the PCRTT algorithm uses fixed sized segments). For very large buffers, the PCRTT algorithm behaves similarly to the MVBA algorithm.

### 4.2. Effect of the Movie Approximation Model on Bandwidth Smoothing Algorithms

As previously mentioned, another important performance measure of the movie approximation model is its affect on the metrics for which the bandwidth smoothing algorithms were created. Among these are the peak bandwidth requirements, the number of bandwidth changes required, and the variability of bandwidth requests. In particular, the goal is to have the movie approximation function have little affect on the performance metric for which each of the bandwidth smoothing algorithms was defined.

To evaluate the performance of the movie approximation model, we used the *minimum changes bandwidth allocation* (MCBA) algorithm and the *minimum variability bandwidth allocation* (MVBA) algorithm. To get the various performance metrics using the movies *Star Wars* and *Speed*, we let the error buffer range from 1 kilobyte to 64 kilobytes. In addition, we let the smoothing buffer size range from a half megabyte to 16 megabytes. We then took all the combinations of <movie, algorithm, smoothing buffer, and error buffer> to use as input to the bandwidth smoothing / movie modeling simulator we built. The basic steps used were:

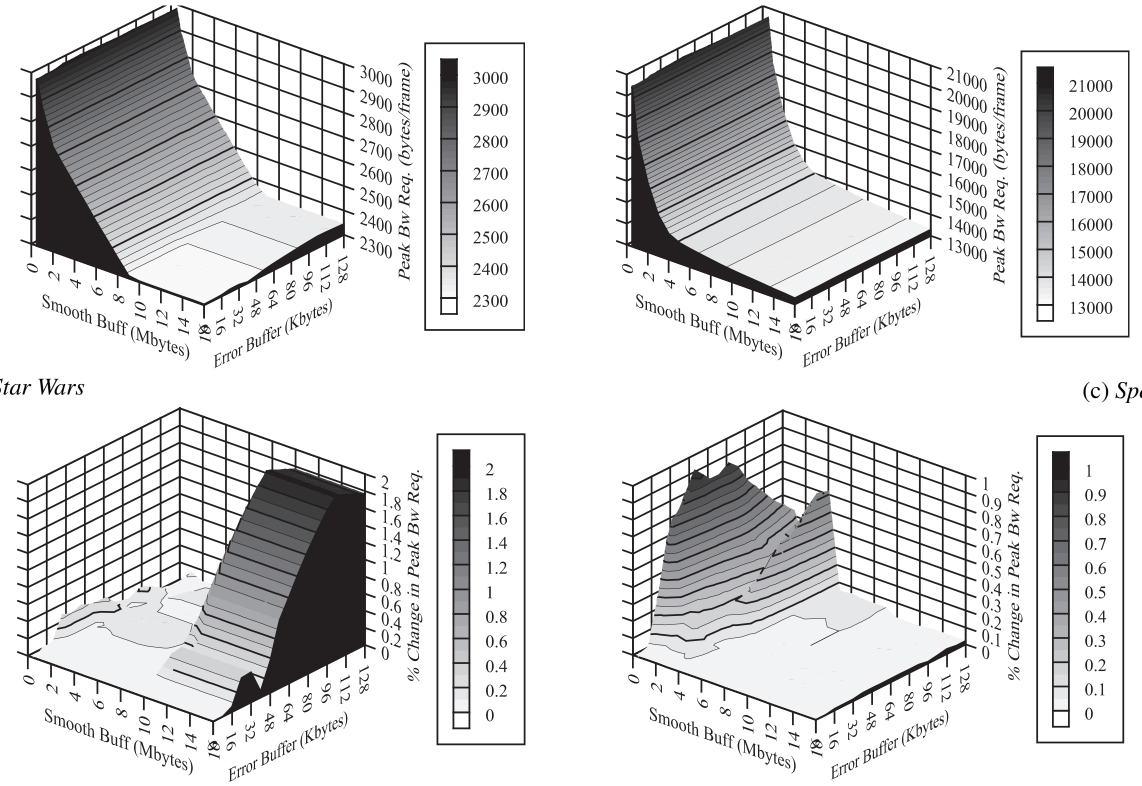
- Create the movie approximation model using the MCBA algorithm and the error buffer
- Using the chosen bandwidth algorithm and chosen movie, calculate the bandwidth allocation plan using the chose smoothing buffer size and the movie approximation curves
- Calculate the statistics from the resulting bandwidth allocation plans

The resulting plans we will refer to as the *approximated* bandwidth plans. Thus, *approximated MVBA* refers to the case using MCBA for the movie model and MVBA for the creation of the bandwidth plan, while *approximated MCBA* refers to the case of using MCBA for both the movie model as well as the creation of the bandwidth plan. For comparison purposes, we have also taken all the combinations of {movie, algorithm, smoothing buffer} and used these as input into the original MCBA and MVBA algorithms for comparison. That is, we calculated the MVBA and MCBA plans using the actual frame size data. We refer to these plans as the *actual* bandwidth smoothing plans.

## Peak Bandwidth Requirements

The peak bandwidth requirements determine the worst case resource requirements required to play back the video. For guaranteed delivery of video, lower peak bandwidth requirements will allow for more streams to be supported. Because the MCBA and MVBA algorithms both find the minimum peak bandwidth requirements, the statistics for the peak bandwidth requirement graphs will be the same.

In Figure 9, we have graphed the approximated, actual, and percentage change curves for the movies *Star Wars* and *Speed.* We see that for very small buffer sizes (0.5 megabytes), the MCBA and MVBA result in large reductions in the bandwidth requirements required for both the approximated and original algorithms ( Figure 9(a) ). In this graph, the error buffer of size 0 is the original algorithm, while the other points (error buffer > 0) are for the implementation of the algorithm using the approximated data. As shown by Figures 9 (b) the main difference occurs for large smoothing buffers and large error buffers (ex. smoothing buffer 16 MB and error buffer 128 kbytes). While the difference of approximately 1.8% is not particularly large, this phenomena can be described as follows. Under small




(a) *Star Wars*

(c) *Speed*

(b) *Star Wars* - Percentage Change

(d) *Speed* - Percentage Change

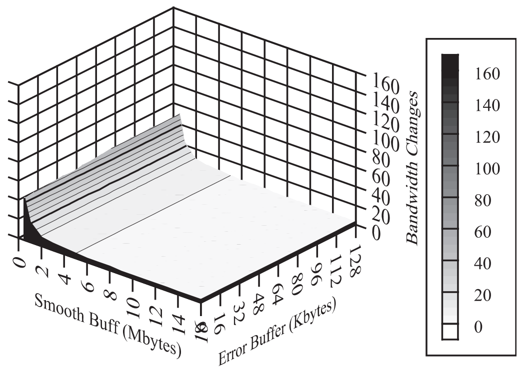
**FIGURE 9: Peak Bandwidth Requirements.** These figures show the performance (in terms of the peak bandwidth requirements) of the MCBA and MVBA algorithms for the movies *Star Wars* (on the left) and *Speed* (on the right). The MCBA and MVBA have the same peak bandwidth requirements and have the same exact curves for the respective movies.

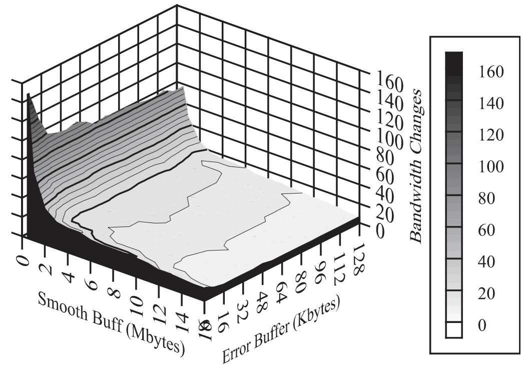
smoothing buffers and large error buffers (ex. smoothing 2 MB and error 128 kbytes), the smoothing buffer is not as effective at removing burstiness compared to the error buffer. As a result, it is somewhat immune to the error introduced by the error buffer, with an approximate change in bandwidth of less than 0.2%! For the larger smoothing buffer and large error buffer, the smoothing buffer is much more effective relative to the error buffer sizes. As a result, errors in the movie model are magnified in these regions. It is important to remember that the error buffers less than 40 kbytes yielded an order or magnitude reduction in the number of segments needed to represent the data so we would expect that these larger regions of deviation would be avoided. Nonetheless, it is important to determine where these regions are.

In figures (c), and (d) of Figure 9, we see that the resulting bandwidth plans for the movie *Speed* have very similar peak bandwidth requirements. The main difference occurring with small smoothing buffers and large error buffers. (See the left wall of figure (d), where the line approaches 21000 bytes/frame). As shown in Figure 9 (d), the approximation algorithm result in a maximum deviation of the peak bandwidth requirements of less than 1% and having most of the differences less than 0.1% (i.e. < 20 bytes/frame).

## Number of Bandwidth Changes

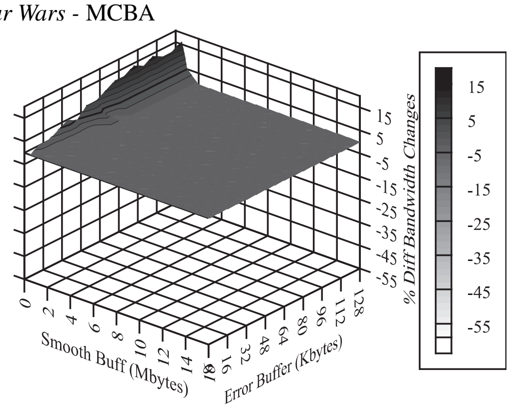
Limiting the number of bandwidth changes can limit the cost of renegotiating bandwidth requirements across the network. Reducing the number of bandwidth changes also allows network and server resource to be allocated for longer periods of time without having to update the resource allocations. We have graphed the MCBA and MVBA approximated bandwidth smoothing algorithm performance for the movie *Star Wars* in Figure 10 and for the movie *Speed* in Figure 11.

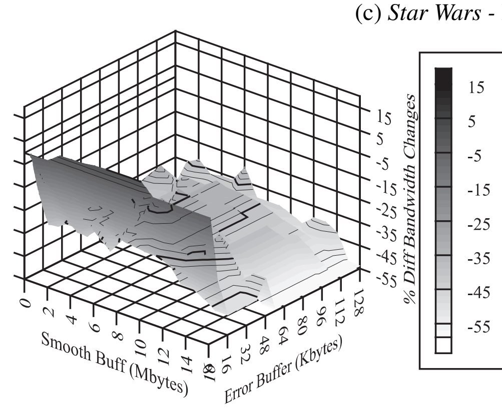



(a) *Star Wars* - MCBA

(c) *Star Wars* - MVBA

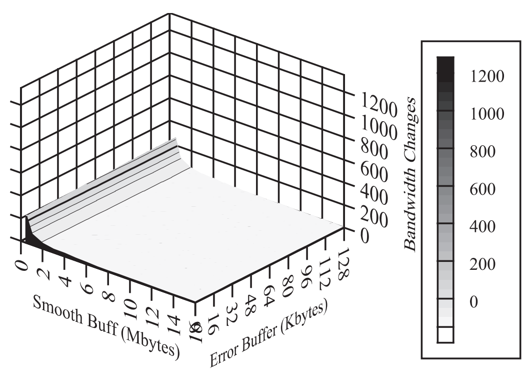


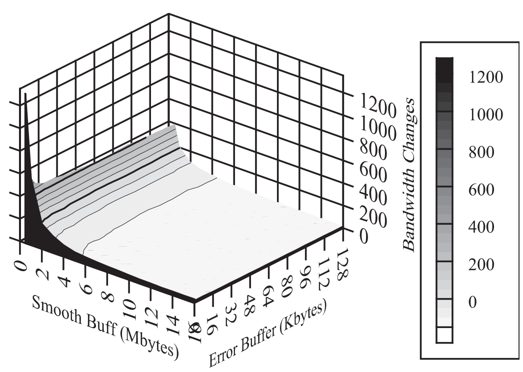




(b) *Star Wars* - % Change MCBA

(d) *Star Wars* - % Change MVBA

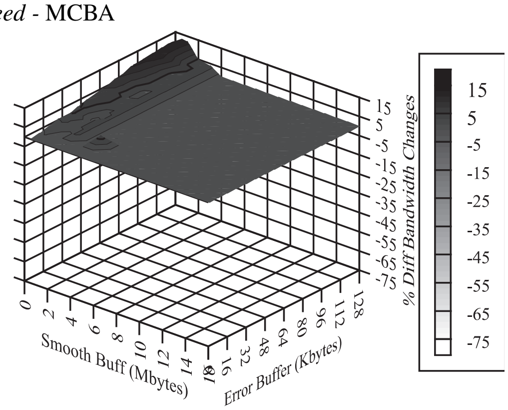
**FIGURE 10: *Star Wars* Bandwidth Changes.** These graphs show the number of bandwidth changes that are required for the MCBA and MVBA algorithms for the movie *Star Wars.*

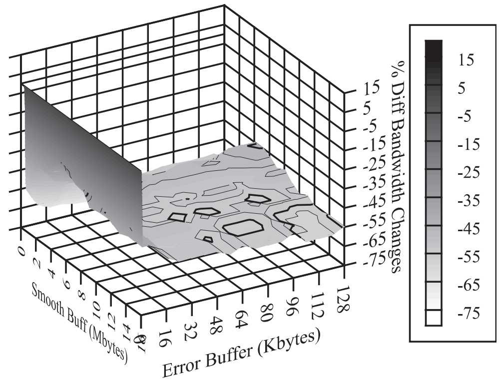



(a) *Speed* - MCBA

(c) *Speed* - MVBA




(b) *Speed* - % Change MCBA

(d) *Speed* - % Change MVBA

**FIGURE 11: *Speed* Bandwidth Changes.** These graphs show the number of bandwidth changes that are required for the MCBA and MVBA algorithms for the movie *Speed.*

In Figure 10 (a), we see that the MCBA algorithm has less than 10 changes in bandwidth for smoothing buffer sizes greater than 2 megabytes. That is, one change for approximately every 10 minutes of video. In the region greater than 2 megabytes, the approximated and actual number of bandwidth changes is the same for both the approximated and the actual graphs. We will come back to this discussion when we discuss the *Speed* results. For the MVBA algorithm, we see that the approximated MVBA algorithms have a much different graph than the actual MVBA bandwidth smoothing plan (Figure 10 (d)). In particular, it results in both increases and decreases in the number of bandwidth requests. These increases and decreases seem attributable to three factors. First, the valleys seem correlated to large drops in the number of segments as the error buffer size increases. Second, for an error buffer size of 128 kbytes, the number of segments in the movie model is 200, three orders of magnitude smaller than the total number of frames. The error introduced relative to the number of bandwidth changes seems to cause the MVBA algorithm to oscillate. Third, the oscillations in the short-term burstiness due to the MPEG frame pattern results in groups of pictures being combined at a time, causing the number of segments in the MCBA to drop more rapidly at some error buffer sizes versus other error buffer sizes.

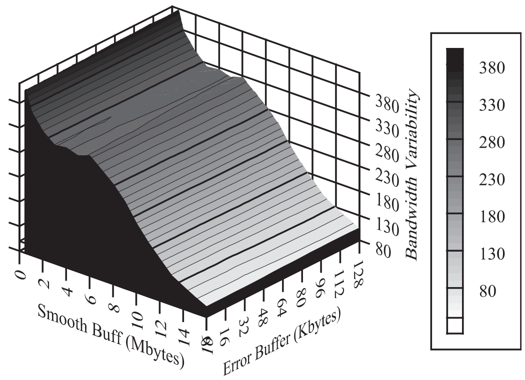
In Figure 11, we have graphed the same figures as in Figure 10, except that the graphs are for the movie *Speed.* Again, the findings are similar to those found in Figure 10 with some notable differences. First, the number of bandwidth changes only decreases for the approximated MVBA algorithm than for the actual MVBA algorithm, where it decreases the total number of bandwidth requests an average of approximately 25%. This highlights the fact that the MCBA algorithm was used as the movie modeling algorithm and the MVBA algorithm as the bandwidth smoothing algorithm. For bandwidth increases and decreases, the MVBA strives to "hug" the corners of the turns, breaking a single bandwidth increase in the MCBA algorithm into several smaller increases. By using the MCBA algorithm as the movie approximation algorithm, the curves get "squared-out" a little, forcing the MVBA algorithm to combine many smaller increases and decreases in bandwidth into larger steps. This helps to explain why there is relatively no change recorded in Figure 10 (b) for the MCBA algorithm. The changes that the movie modeling algorithm removed would also have been removed under the MCBA bandwidth smoothing plans. Finally, there are several spikes in the difference change curves for the MCBA algorithm. These spikes occur on boundary conditions in the MCBA algorithm occasionally where the error buffer size is just great enough to cause the extra bandwidth changes. Because the MCBA algorithm results in very few changes in bandwidth for moderately sized buffers (> 5 MBytes), a single change in bandwidth can result in a fairly large % change in bandwidth requests.

## Variability in Bandwidth Requirements

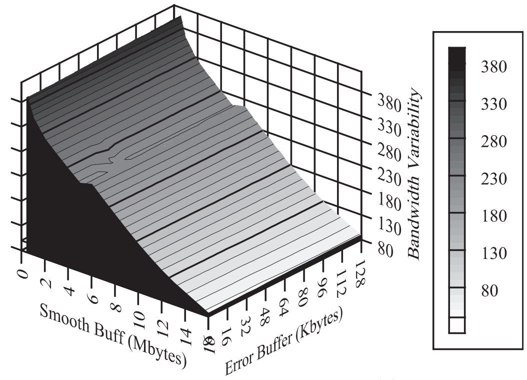
Minimizing the variability of bandwidth requirements can reduce the *effective bandwidth* requirement of the video stream, allowing it to have greater statistical multiplexing gain to be increased. In Figure 12 and Figure 13, we have graphed the bandwidth variability requirements.

In Figure 12, we have graphed the variability in bandwidth measurements for the movie *Star Wars*. The MCBA and MVBA algorithms have very similar graphs for the approximate and original bandwidth algorithms (with the error buffer equal to 0). Of particular interest in the difference curves (Figure 12 (b) and (d)) is the bimodal curve that appears for larger error buffer sizes. The reason for this is the same as discussed in the peak bandwidth requirement section. That is, for large error buffers and large smoothing buffers, the smoothing buffer is more effective relative to the error buffer. As a result, the errors tend to be magnified more.

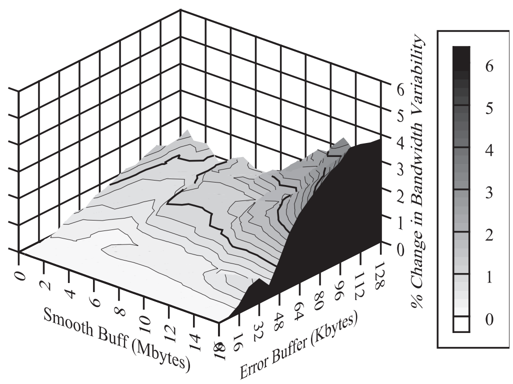
Finally, in Figure 13, the movie approximation model has relatively small effect on the bandwidth variability. For the most part, the variability is changed less than several tenths of one percent. These figures do no suffer from the bimodal changes in variability, mainly due to the fact that much more buffer is needed to smooth out the *Speed* video stream. We would expect that for very large error buffers that we would see a similar phenomena as in the *Star Wars* case.



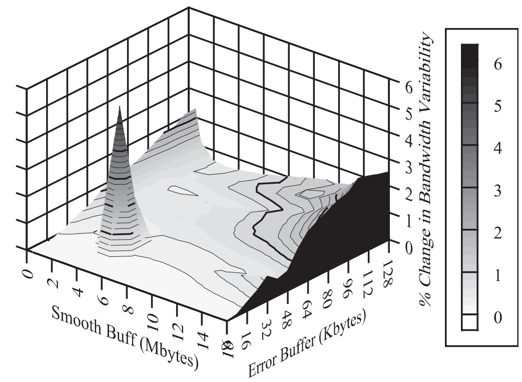

(a) *Star Wars* - MCBA




(c) *Star Wars* - MVBA

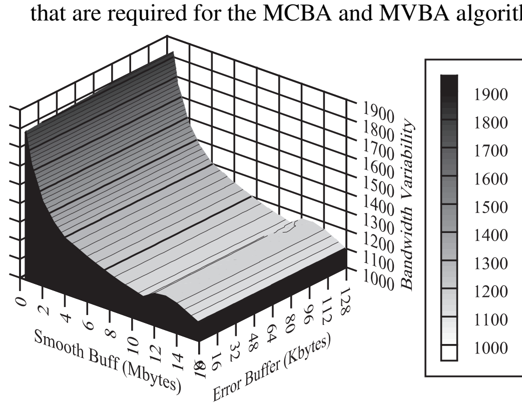



(b) *Star Wars* - % Change MCBA

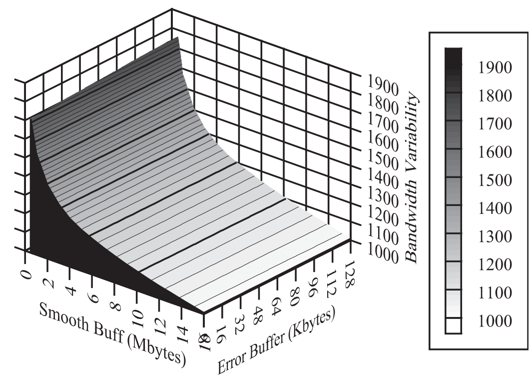



(d) *Star Wars* - % Change MVBA

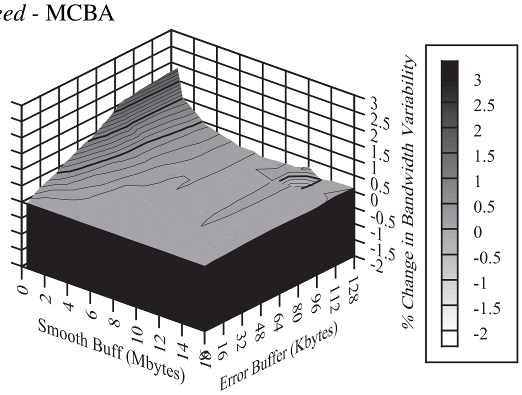
**FIGURE 12: *Star Wars* Bandwidth Variability.** These graphs show the number of bandwidth changes that are required for the MCBA and MVBA algorithms for the movie *Star Wars*.



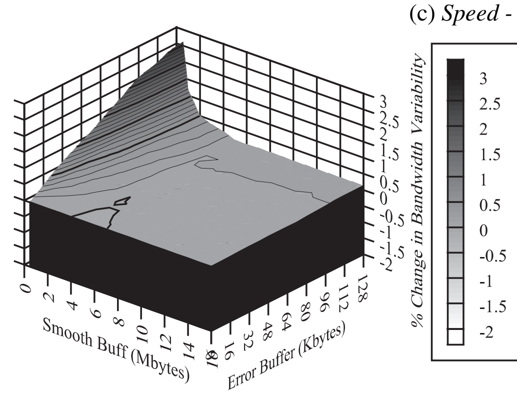

(a) *Speed* - MCBA




(c) *Speed* - MVBA

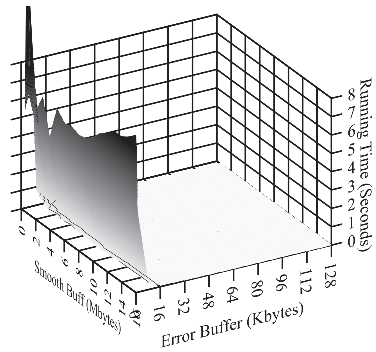



(b) *Speed* - % Change MCBA

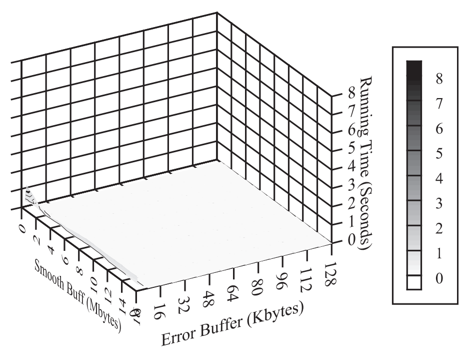



(d) *Speed* - % Change MVBA

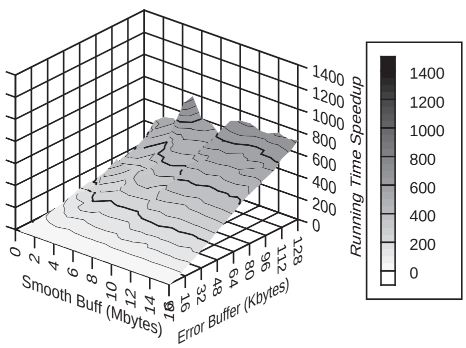
**FIGURE 13: *Speed* Bandwidth Variability.** These graphs show the number of bandwidth changes that are required for the MCBA and MVBA algorithms for the movie *Speed*.



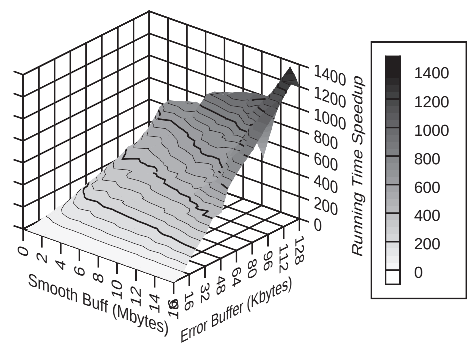

(a) *Star Wars* - MCBA




(c) *Star Wars* - MVBA




(b) *Star Wars* - % Change MCBA




(d) *Star Wars* - % Change MVBA

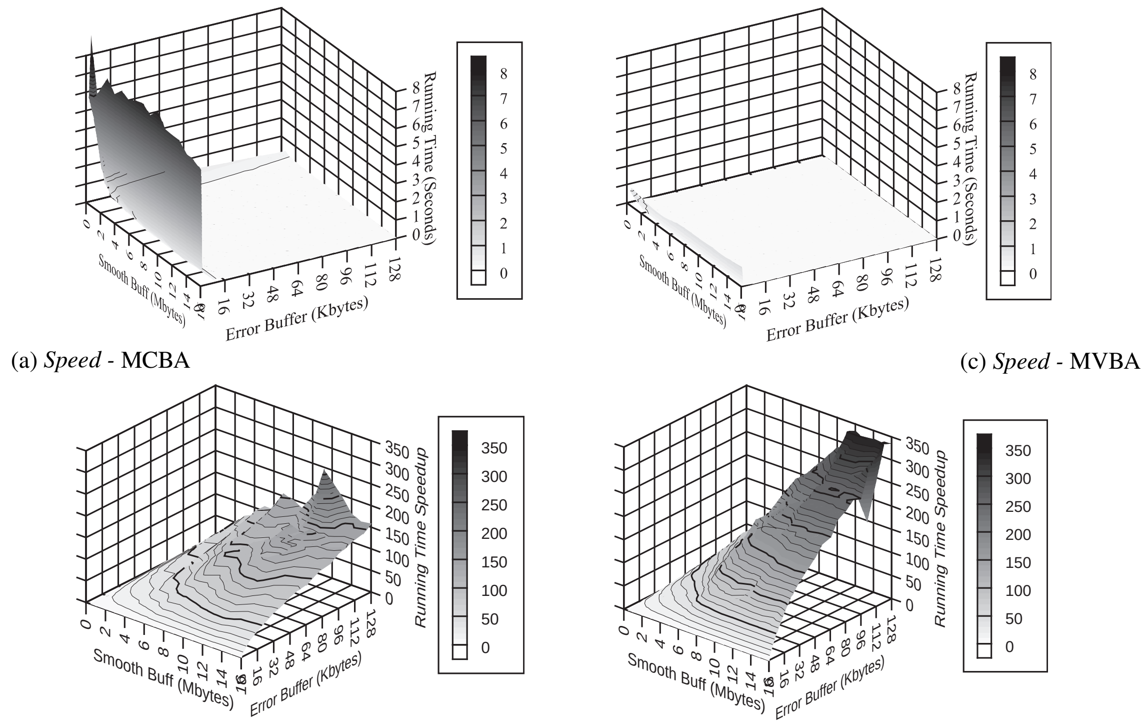
**FIGURE 14: *Star War* Running Time.** These graphs show the running time that are required for the MCBA and MVBA algorithms for the movie *Star War*.

## Performance of Movie Approximation Based Bandwidth Smoothing Algorithms

To compare the relative execution times of the approximation algorithm implementation versus the original bandwidth smoothing algorithms, we measured the execution time for just the bandwidth calculation routines, ignoring any file access times to read in the approximation or frame data. The measurements in this section were timed on HP 715s. We have graphed the run times in Figure 14 and Figure 15 for the two movies.

In Figure 14., we have graphed the actual run times and the speed up in execution times compared to the original algorithms for the movie *Star Wars.* The original MCBA algorithm typically finished the creation of the bandwidth plans in 5-6 seconds. Using the approximation algorithms, the bandwidth calculation times typically dropped to around 20 milliseconds, well below what is required for interactive responsiveness. In comparing the approximation implementation with the original, we recorded execution times that are 100 times faster than the original algorithm for error buffer sizes greater than 24 kbytes. The original MVBA typically finished execution in 0.6 seconds to 1 second. Using the approximation algorithms, we recorded execution times that were also 100 times faster than the original for error buffers greater than 16 kbytes. Typically, the execution times were in the 10 millisecond range, well below what is required for interactive responsiveness. For larger error buffers, the MVBA algorithm had greater speed up measurements than the MCBA algorithm, mainly due to the very small number of points in the approximation curve. For very small smoothing buffers (< 0.5 MB) and very small error approximations ( 1 kbyte ), the execution times were actually worse for the MVBA algorithms. We suspect that these were due to the implementation in pointers that we used for the approximation function versus simple arrays for the original algorithm. Comparing the MCBA and MVBA algorithms, the MCBA algorithm performed worse in all experiments because it performs searches on bandwidth increases and decreases to minimize the number of rate changes, as expected.

In Figure 15, we have graphed the execution times for the movie *Speed*., As shown by this figure, we had similar results for the movie *Speed.* The MCBA algorithm tended to do worse with smaller smoothing buffer sizes because




(a) *Speed* - MCBA

(c) *Speed* - MVBA

(b) *Speed* - % Change MCBA

(d) *Speed* - % Change MVBA

**FIGURE 15: *Speed* Running Times.** These graphs show the running times that are required for the MCBA and MVBA algorithms for the movie *Speed.*

the number of rate changes was higher with the Motion-JPEG encoded movie. Overall, the execution times were very similar to the movie *Star Wars*. The speed up measurements were slightly smaller due to the fact that the smoothing buffer is slightly less effective for larger (in bit rate) video streams, resulting in more points in the approximation. Nevertheless, the execution times were in the millisecond range as in the *Star Wars* case.

## 5. Conclusions

In this paper, we have introduced a movie approximation modeling algorithm for representing the actual frame sizes for the stored prerecorded video clips. By storing the movie approximation models instead of the actual frame sizes, several important performance enhancements can be achieved for video-on-demand systems. First, by storing a simple model instead of the actual frame sizes, the video-on-demand server does not have to waste its bandwidth transmitting the frame sizes of the movie to the client. Second, by having a simple model, the clients can then more interactively create bandwidth smoothing plans on the fly because the number of points evaluated in the calculation of the bandwidth plan are greatly reduced. Third, by using a simple model, the memory in the set-top-boxes which create the bandwidth smoothing plans can be reduced, allowing more of it to be used for prefetching data.

We have demonstrated several interesting results. Using the MCBA algorithm for the movie approximation model allows the number of stored points to represent a movie to be reduced by an order of magnitude with relatively small amounts of error. In particular, we have found that using a maximum error of approximately 10 kbytes allows us to reduce the number of points by an order of magnitude, while using a maximum error of 40 kbytes allows us to reduce the number of points by two orders of magnitude! In the experimental section, we have demonstrated that this algorithm does not affect the performance measures that the bandwidth smoothing algorithms are trying to achieve, particularly for error buffers less than 40 kbytes in size. Typically, the results less than a 1% change from the original bandwidth algorithms. The only exception to this was the number of changes that the MVBA algorithm had under the movie approximation model, where the MVBA algorithm was slightly unstable. However, for error buffers of less than 40 kbytes, the MVBA algorithm's variability was not affected by more than 1%. We have also shown that we can improve the execution times by over 90% with even a small amount of error in the movie approximation model.

In summary, we have shown that by modeling the *a priori* knowledge of a stored video stream, we can still achieve the minimal properties that the bandwidth smoothing algorithms were designed for. By using the movie approximation model, we can then simplify the amount of control messages that are required for interactive video-on-demand systems.

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

[4] Wu-chi 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.

[5] Wu-chi Feng, S. Sechrest, "Critical Bandwidth Allocation for the Delivery of Compressed Prerecorded Video", *Computer Communications*, Vol. 18, No. 10, Oct. 1995, pp. 709-717.

[6] 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.

[7] W. Feng,''Video-on-Demand Services: Efficient Transportation and Decompression of Variable Bit Rate Video'', Ph.D. Thesis, University of Michigan, April 1996.

[8] Wu-chi Feng, F. Jahanian, S. Sechrest, "An Optimal Bandwidth Allocation Strategy for the Delivery of Compressed Prerecorded Video", *ACM/Springer-Verlag Multimedia Systems Journal,* 1997.

[9] Wu-chi Feng, "Rate-Constrained Bandwidth Smoothing for the Delivery of Stored Video", in *Proceedings of the 1997 SPIE Multimedia Computing and Networking Conference*, San Jose, CA, Feb. 1997.

[10] Wu-chi Feng, Jennifer Rexford, "A Comparison of Bandwidth Smoothing Techniques for the Delivery of Compressed Prerecorded Video", *Proceedings of IEEE INFOCOM '97*, Kobe, Japan, pp. 58-66, April 1997.

[11] M.W. Garrett, *Contributions Towards Real-Time Services in Packet Switched Networks*, Ph.D. Thesis, Columbia University, May 1993.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] P. Venkat Rangan, H.M. Vin,''Designing File Systems for Digital Video and Audio'', In *Proc. of the 13th ACM Symposium on Operating Systems Principles*, Operating Systems Review, Vol. 25, No. 5, Oct. 1991, pp. 81-94.

[17] 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.

[18] J.D. Salehi, Z.L. Zhang, J.F. Kurose, D. Towsley, "Optimal Buffering for the Delivery of Compressed Prerecorded Video", in *Proceedings of ACM SIGMETRICS*, pp. 222-231, May 1996.

[19] H. Zhang, S. Keshav,"Comparison of Rate-Based Service Disciplines", In *Proceedings of ACM SIGCOMM*, Sept. 1991, pp. 113-121.