An Optimal Bandwidth Allocation Strategy for the Delivery of Compressed Prerecorded Video

Wu-chi Feng, Farnam Jahanian, Stuart Sechrest

Software Systems Research Lab
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, Michigan

E-mail {wuchi, farnam, sechrest}@eecs.umich.edu

Abstract

The transportation of compressed video data without loss of picture quality requires the network to support large fluctuations in bandwidth requirements. Fully utilizing a client-side buffer for smoothing bandwidth requirements can limit the fluctuations in bandwidth required from the underlying network. This paper shows that for a fixed size buffer constraint, the critical bandwidth allocation technique results in the minimum number of bandwidth increases necessary for stored video playback. In addition, this paper introduces an optimal bandwidth allocation algorithm which minimizes the number of bandwidth changes necessary for the playback of stored video and achieves the maximum effectiveness from client-side buffers. For bandwidth allocation plans that allocate bandwidth for the length of a video, the optimal bandwidth allocation algorithm also minimizes the number of bandwidth increases as well as the total increase changes in bandwidth. A comparison between the optimal bandwidth allocation algorithm and other critical bandwidth based algorithms using several full-length movie videos is also presented.

Keywords: video-on-demand, bandwidth allocation, MPEG, video compression, smoothing

1.0 Introduction

Video applications, such as video-on-demand services, rely on both high-speed networking and data compression. Data compression can introduce burstiness into video data streams which can complicate the problem of network resource management. For live-video applications, the problem of video delivery is constrained by the requirement that decisions must be made on-line and that the delay between sender and receiver must be minimized. As a result, live video applications may have to settle for weaker guarantees of service or for some degradation in quality of service. Work on problems raised by the requirements of live video includes work on statistical multiplexing[2,7], smoothing in exchange for delay[4], jitter control[9,11], and adjusting the quality of service to fit the resources available[6]. Stored video applications, on the other hand, can take a more flexible approach to the latency of data delivery. In particular, they can make use of buffering to smooth the burstiness introduced by data compression. Because the entire video stream is known *a priori*, it is possible to calculate a complete plan for the delivery of the video data that avoids both the loss of picture quality and the loss of network bandwidth due to overstatement of bandwidth requirements.

The utility of prefetching is quite simple to explain. Since the bytes for any given frame can be supplied either by the network or by a prefetch buffer, the burstiness of the network bandwidth requirement can be controlled by filling the prefetch buffer in advance of each burst by delivering more bytes across the network than needed, and draining it in the course of the burst. The size of the prefetch buffer, of course, determines the size of burst that can be averaged out in this way. With a small buffer, only a limited amount of data can be prefetched without overflowing the buffer, so the bandwidth required of the network will remain relatively bursty. With a larger buffer, on the other hand, there is the possibility that most of the burstiness of a video clip can be eliminated through prefetching. This, however, requires a plan for prefetching the data that ensures that the large buffer is filled in advance of bursts which place high demand upon the buffer.

In a previous paper, we introduced the notion of *critical bandwidth allocation*, which provides plans for smoothing the network bandwidth by utilizing a buffer of a fixed size[3]. The critical bandwidth algorithm allows for long sequences of monotonically decreasing bandwidth requirements, such that, at any point during playback, the bandwidth allocated is the minimum *constant* bandwidth necessary to play back the video without buffer overflow or underflow. Our hypothesis was that the critical bandwidth algorithm could be used as the basis for minimizing the total number of bandwidth changes necessary for the playback of a video. In this paper, we first show that the critical bandwidth allocation algorithm results in the minimum number of bandwidth increases necessary for video playback. We will then introduce (and prove) an optimal bandwidth allocation algorithm for stored video which minimizes the total number of bandwidth changes as well as the total (magnitude) of bandwidth increase changes necessary for continuous, uninterrupted video playback. That is, the algorithm guarantees that the plan will not cause the available buffer to underflow or to overflow while producing the fewest possible changes in bandwidth.

The optimal bandwidth allocation approach has implications for the design of underlying network delivery services. For test video clips, the approach yields plans that call for constant bandwidth allocations for periods lasting on the order of minutes and requires admission control decisions at intervals of several minutes. This relative constancy makes it possible for the network management to provide bandwidth guarantees without having to waste bandwidth through overstatement of the actual application requirements. Requests to increase network bandwidth are made infrequently. This suggests that network services providing guarantees for network bandwidth should be able to negotiate these guarantees in advance. In contrast, live video applications usually negotiate for an immediate channel. Further, it should be possible for the application to lower its bandwidth requirement without tearing down the channel. Live video applications generally maintain a fixed bandwidth allocation.

In the following section, a description of the original critical bandwidth algorithm is presented along with an optimal version of the algorithm. A proof of optimality is also presented for the optimal bandwidth algorithm. The evaluation section compares and contrasts the alternative smoothing algorithms using an MPEG encoded movie and several full-length Motion-JPEG encoded movies. Finally, a summary and conclusions about the importance of smoothing to the design of network services is presented.

2.0 Bandwidth Allocation Algorithms

In dealing with compressed video data, smoothing techniques attempt to remove the burstiness with appropriate prefetching and delay. An understanding of how burstiness is introduced into a video stream can pro-

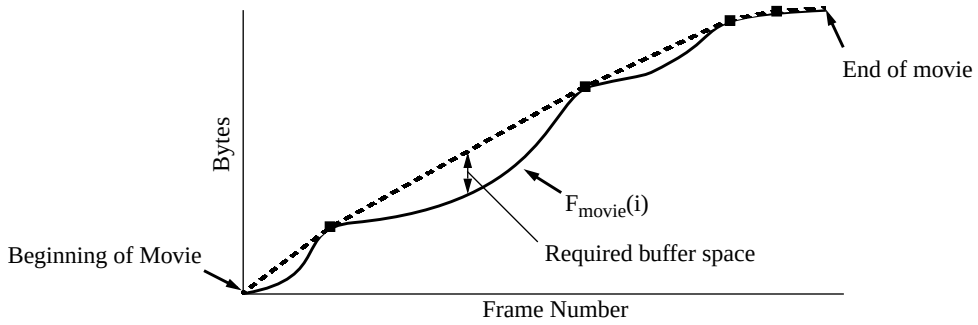


FIGURE 1. The solid line in the graph shows a possible graph for $F_{movie}(i)$. The total delivered bandwidth (up to some frame i), must be greater than or equal to $F_{movie}(i)$. Note: $F_{movie}(i)$ must be monotonically increasing (i.e. no negative frame sizes). The dotted set of lines show the critical bandwidth allocation algorithm's bandwidth plan that requires 5 decreases in bandwidth, while the squares on the dotted lines show the junctures between runs. The minimum buffer size is determined by the maximum vertical distance between the critical bandwidth allocation plan and the function $F_{movie}(i)$.

vide insight into the effectiveness of the various smoothing algorithms. For pre-recorded streams compressed with algorithms such as MPEG[1,5], burstiness occurs at two scales: in small runs of frames as a result of the pattern of frame types (I, P, or B) used in compression and at a larger scale with variations in scene content. As was originally shown, the critical bandwidth allocation algorithm can be an effective algorithm in smoothing variations in scene content[3]. In this section we will review the critical bandwidth algorithm and show that it results in the minimum number of bandwidth increases. In addition, we will motivate, present, and prove the optimal critical bandwidth allocation algorithm.

2.1 Critical Bandwidth Allocation

The critical bandwidth allocation algorithm calculates a plan for the delivery of video data which, into a limited buffer, produces plans with relatively few increases in bandwidth. For clarity, we say that each increase in bandwidth will begin a new *segment*. Thus, the critical bandwidth algorithm will break each video clip into one or more segments with non-increasing bandwidth requirements. Each segment will consist of one or more *runs* with each run having a constant bandwidth requirement.

Given any map of frame sizes for a particular movie, a graph can be drawn that has the following function:

$$F_{movie}(i) = \sum_{j=1}^{i} FrameSize_j$$

This function is the running integral of frame sizes for the movie. By graphing this function (as is shown in Figure 1), any plan that is generated must have the total bandwidth received (TBR) such that the following condition holds for all frames, *i*, within the movie to avoid buffer underflow:

$$F_{movie}(i) \leq TBR$$

The critical bandwidth allocation algorithm allocates bandwidth at the minimum constant bandwidths necessary in order to play back the video without buffer underflow. This corresponds to creating a convex arc from the beginning of the movie to the end of the movie with each run ending at a point on the line

$F_{movie}(i)$. The slope of each line (run) determines the bandwidth allocation that is required for that run and provides a plan for video delivery. The bandwidth allocations for this plan decrease monotonically and are the minimum allocations for any such monotonically decreasing plan (shown in Figure 1 as a sequence of lines with monotonically decreasing slopes). The creation of this plan does not observe any limits in available buffer space but does calculate the minimum possible buffer size necessary to play the video clip with a single monotically decreasing sequence of bandwidth allocations. This required buffer size is determined by the maximum vertical distance between the bandwidth allocation plan and the function $F_{movie}(i)$. The magnitude of this minimum buffer size will vary for the same clip, depending on the encoding scheme used and the long term burstiness that results.

Formally, the critical bandwidth CB_0, in bytes per frame, is defined as

$$CB_0 = \max_{1 \le i \le N} \left(\left\lceil \frac{\sum_{j=1}^{i} frame_j}{j} \right\rceil \right)$$

where N is the number of frames in the video clip and $frame_j$ is the size in bytes of frame number j. Thus, the critical bandwidth is determined by the frame, i, for which the average frame size for i and all prior frames in the video clip is maximized. We call frame i, which forces the critical bandwidth, the *critical point* in the video clip, or CP_0. In the case where the maximum is achieved multiple times, we choose CP_0 to be the last frame at which it is achieved. Suppose we divide the video clip into two runs, one consisting of all frames up to (and including) CP_0 and one following CP_0 to the end of the video. Then, the first run will have the same critical bandwidth as the entire video clip, while the run following CP_0 will have a critical bandwidth less than the critical bandwidth of the entire video clip[3]. For this definition and subsequent definitions involving bandwidth calculations, we assume that reservations are allocated in bytes per frame.

By recursively breaking the video clip at these critical points and finding the critical bandwidth of the remaining clip, we can extend the definition of critical bandwidth to produce a strictly decreasing sequence of critical bandwidths. These critical bandwidths, CB_n, are determined by a sequence of critical points CP_n, where

$$CB_n = \max_{CP_{n-1} < i \le N} \left(\left\lceil \frac{\sum_{CP_{n-1}+1}^{i} frame_j}{j} \right\rceil \right)$$

Finally, this bandwidth allocation plan allows for the commencement of playback to begin after the arrival of the first bandwidth reservation (frame) of data.

2.2 Critical Bandwidth Allocation with Maximum Buffer Constraint

Using the critical bandwidth algorithm results in the calculation of the minimum buffer size necessary to treat the entire video clip as a monotonically decreasing sequence of bandwidth allocations. If this minimum buffer size exceeds the buffer space available, then the client must increase the bandwidth in the middle of the video clip, trading buffer bandwidth for network bandwidth. Using our geometric model, we

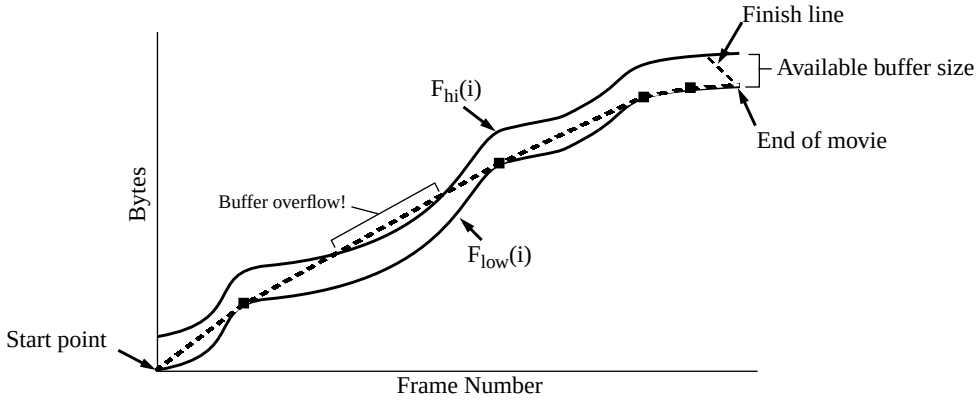


FIGURE 2. This figure shows a possible graph for $F_{hi}(i)$ and $F_{low}(i)$. The total delivered bandwidth (up to some frame i), must lie between $F_{hi}(i)$ and $F_{low}(i)$ or buffer overflow or underflow will occur. The dotted lines represents a critical bandwidth allocation plan that does not consider buffer overflow. Any time the allocation plan goes above $F_{hi}(i)$, buffer overflow will occur.

can graph the following functions, $F_{hi}(i)$ and $F_{low}(i)$, where $F_{low}(i)$ is the same as $F_{movie}(i)$ from the last section.

$$F_{hi}(i) = \left(\sum_{j=1}^{i} FrameSize_j \right) + BufferSize$$

and

$$F_{low}(i) = \sum_{j=1}^{i} FrameSize_j$$

By graphing these vertically equidistant functions (as shown in Figure 2), any plan that is generated must have the total bandwidth received (TBR) such that the following condition holds for all frames, i, within the movie:

$$F_{low}(i) \leq TBR \leq F_{hi}(i)$$

For the critical bandwidth algorithm that uses a maximum buffer constraint, the algorithm allocates at the minimum bandwidths such that the buffer does not underflow or overflow. In the example in Figure 2, the second run which will cause the buffer to overflow is modified to not cross $F_{hi}(i)$, the buffer overflow function, so that the run (line) which increases the bandwidth requirement will have a bandwidth requirement (slope) greater than the second run (see Figure 3). As a result, each run is still allocated at the minimum constant bandwidth necessary to get to the end of each run and increases only occur when prefetching a burst into the buffer will cause it to overflow.

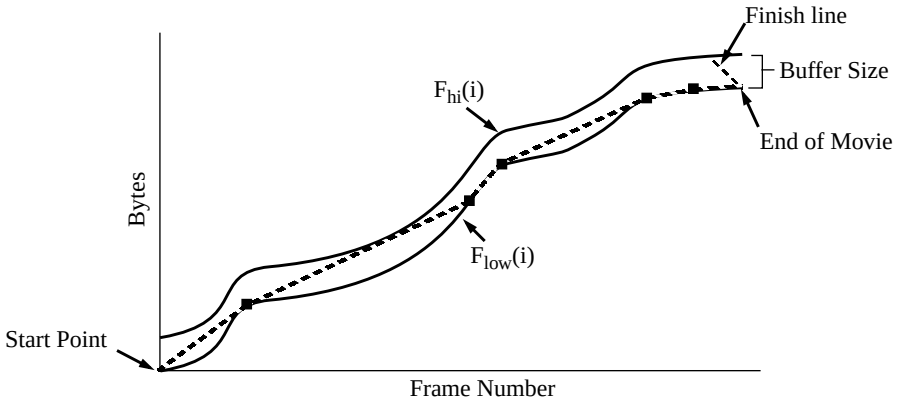


FIGURE 3. The dotted line above shows the modification from Figure 2 that the critical bandwidth algorithm with a maximum buffer constraint makes such that the buffer will not overflow.

In order to algebraically calculate a bandwidth allocation plan that takes advantage of the buffer space available without exceeding it, let

- $FrameAve_i$ be the average frame size from the beginning of the run to the *i*th frame within the run.
- $MaxBW_i$ be the maximum average bandwidth sustainable from the beginning of the run to the *i*th frame that will not overflow the buffer. This can be defined as

$$\min_{1 \le j \le i} \left(FrameAve_j + \left\lceil \frac{BufferSize}{j} \right\rceil \right)$$

Then, a critical bandwidth sequence is defined as a set of sucessive runs, with each run defined as a set of *k* frames such that the following holds for all frames within the run:

$$\max_{1 \le j \le k} FrameAve_j \le MaxBW_k$$

When the above condition becomes false, the buffer space available will be exceeded at some point within the run and a change in bandwidth will be required. The critical bandwidth for the run is determined by

$$CB = \max_{1 \le j \le k} FrameAve_j$$

and the critical point is the frame *j* at which the *FrameAve* was maximized. If the critical point is *k*, then a bandwidth increase will be necessary in the next run. If the critical point is less than *k*, then a decrease in bandwidth will be necessary because the buffer overflowed after a critical point. Typically, a critical point will either be at the end of the run or will be at a frame far enough back to allow the buffer to overflow between the critical point and the end of the run, frame *k*.

Using the method described results in a bandwidth allocation plan for the delivery of the video. At the end of each run, the buffer will be relatively empty. For runs that increase the bandwidth allocation from the last run, the transition can be further smoothed by prefetching data for the next run. In the geometric model, this corresponds to modifying the run that increases the bandwidth requirement to have a smaller slope (see Figure 4). This smoothing corresponds to starting the run with the bandwidth increase earlier in

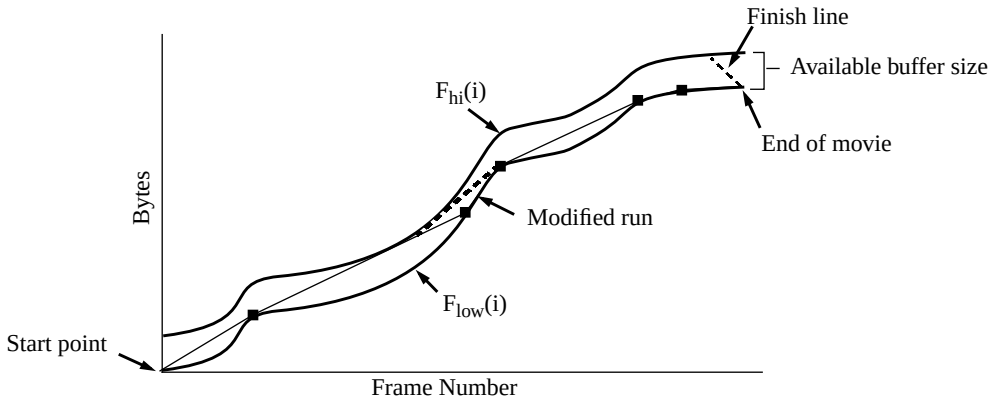


FIGURE 4. The light solid lines in the graph above show the critical bandwidth allocation algorithm with no prefetching, while the dotted line shows the modification that is made to decrease the bandwidth requirement on a bandwidth increase. This decrease in bandwidth corresponds to starting the run earlier (in time) with a smaller bandwidth requirement.

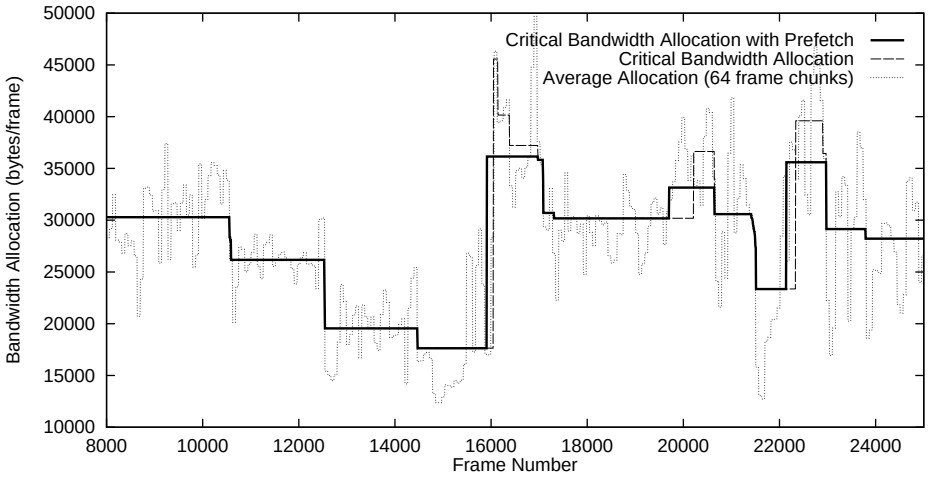


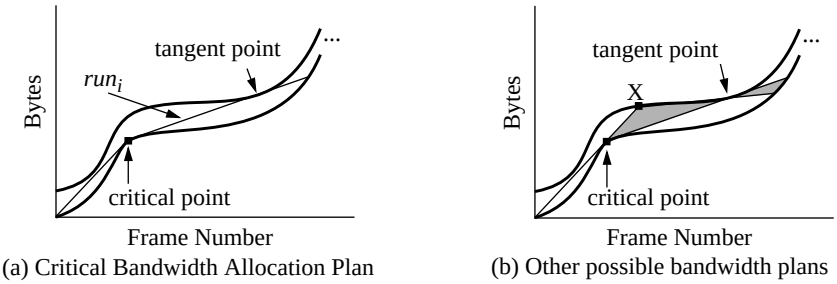
FIGURE 5. Each graph represents the bandwidth requests from a sample of 17,000 frames from the movie *Star Wars*. The medium dotted line is the critical bandwidth allocation algorithms without prefetching and a maximum buffer size of 3 Mbytes. The solid line represents the critical bandwidth algorithm with prefetching and a maximum buffer size of 3 Mbytes. The lightly dotted line shows the average allocation of 64 frame groups.

time but with a smaller bandwidth requirement. The prefetching of data, however, must not violate the buffer limitation and must still obey the critical bandwidth property.

The resulting bandwidth allocation plans for the critical bandwidth algorithms with a maximum buffer constraint are shown in Figure 5. Using either of these algorithms creates a plan that has the minimum number of bandwidth increases necessary for video playback as will be shown by the following theorem.

Theorem 1 : The critical bandwidth allocation algorithm with a fixed maximum buffer constraint results in the minimum number of bandwidth increases required for playback of video without buffer starvation or buffer overflow.

Proof: A key observation about the critical bandwidth algorithm is required for the proof of this theorem. For runs within a segment, the critical bandwidth algorithm allocates each run



(a) Critical Bandwidth Allocation Plan (b) Other possible bandwidth plans

FIGURE 6. These figures show the two condition for which an increase in the bandwidth allocation plan will be necessary for the run, run_{i+1}. In Figure (a), the run chosen by the critical bandwidth allocation plan with a maximum buffer constraint chooses the run such that it starts at the critical point labeled and is drawn through the point labeled "tanget point". The shaded region in Figure (b) shows the area where other plans that intersect both (1) the line from "critical point" to "X" and (2) the tanget point.

such that the critical point for the run is at the end of the run, resulting in an empty buffer at the end of each run.

On a run which increases the bandwidth requirement a situation similar to that in Figure 6 arises. Because all plans must cross the line between the points ''critical point'' and ''X'' in Figure 6b, any other plan that crosses between these points will result in a critical point (end of run), less than or equal to that chosen by the critical bandwidth allocation algorithm. The critical bandwidth allocation algorithm with maximum prefetch required an increase in bandwidth therefore requiring all other plans to increase their bandwidth requirements as well. As a result, because the critical bandwidth allocation plan creates plans that all have critical points at the end of each run (so that all increases are similar to that in Figure 6), the critical bandwidth allocation plan with a maximum buffer constraint has the minimal number of bandwidth increases. //

Using the critical bandwidth algorithm with a fixed size buffer minimizes the number of bandwidth increases required during the playback of a video clip. The algorithm also requires decreasing the bandwidth at critical points throughout each segment, however, this does not minimize the total number of bandwidth changes.

2.3 An Optimal Bandwidth Allocation Algorithm

Using a critical bandwidth based algorithm, it is possible to minimize the total number of bandwidth changes, both increases as well as decreases, given a fixed size buffer. In this section, we will motivate, describe, and prove an optimal strategy for creating a bandwidth allocation plan. For clarity, given a frame for which a run begins, we will use *critical bandwidth* and *critical point* to mean the bandwidth and point to use for the run such that the buffer does not exceed its limits for the run.

2.3.1 Motivation behind the Optimal Bandwidth Allocation Algorithm

Several observations about the critical bandwidth algorithm lead us to believe that it can be used as a basis for minimizing the total number of bandwidth changes necessary to play a video back without buffer underflow or overflow. Our hypothesis was that applying a greedy approach to each run in the critical bandwidth algorithm such that each run selected was as long as possible would result in the minimum

number of bandwidth changes. Suppose we have a run that is determined by critical point, *CP*, and critical bandwidth, *CB*, then several observations arise:

- Allocating bandwidth any greater than *CB* during the run will result in unused data being stored in the buffer during the run, placing a larger burden on the buffer.
- Allocating bandwidth any less than *CB* will result in buffer starvation unless an increase in bandwidth greater than *CB* occurs before *CP* to make up for "lost" bandwidth. Therefore, using *CB* should result in the minimal load placed on the buffer.
- Adding prefetch to the beginning of a run will push the critical point, *CP*, further out until a point where too much prefetch will result in a necessary increase (due to buffer overflow), which, in turn, reduces *CP* (and thereby shortening the run).

Using these observations, regulating the amount of prefetch for each run and allocating each run at its critical bandwidth will be key in minimizing the total number of bandwidth changes. In addition, the amount of prefetch for each run should be calculated such that each run is as long as possible. As mentioned above, the critical points can be moved forward in time by using prefetching, however, simply prefetching as much as possible before a run will not result in the critical points being moved the furthest into the future. Adding additional prefetch will push the critical point further out up to a certain point, say CP_{opt}. After this point any additional prefetch will result in buffer overflow before CP_{opt}, resulting in an increase in bandwidth required before CP_{opt}.

Depending on whether a run is an increase or decrease in bandwidth from the last run determines how the prefetching can be handled. In the case of a decrease in bandwidth, suppose we have a run, run_0, described by critical point CP_0 and critical bandwidth CB_0, then the prefetch for the next run, run_1, described by CP_1 and CB_1 can be accomplished in one of two ways. First, prefetch can be accomplished during run_0 by setting the bandwidth for the run to CB_0' (as in Figure 7a). The second method involves extending CB_0 into run_1 to accomplish the prefetch. We are guaranteed, that the maximum amount of prefetch is possible because CB_1 is less than CB_0. These two cases are shown in Figure 7. Using the second

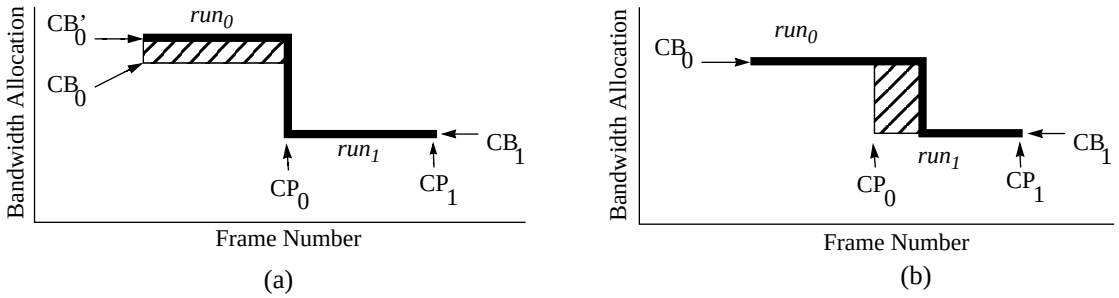
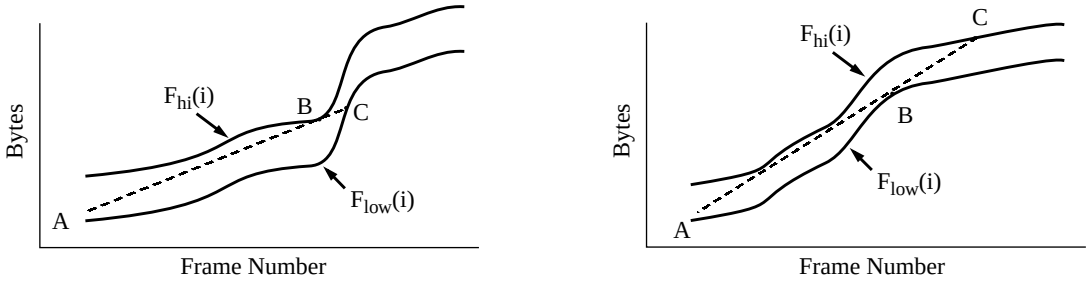


FIGURE 7. These figures show the possibilities for prefetching data for a run, run_1, that decreases the bandwidth allocation from the last run. Figure (a) allocates bandwidth in run_0 higher than its critical bandwidth to accomplish prefetch. Figure (b) shows how prefetch can be accomplished by extending the bandwidth allocation for run_0 into run_1. The bold lines indicate the actual allocation with prefetch.

method of prefetch is preferable because the maximum amount of prefetch is always obtainable, while the buffer capacity may limit the amount of prefetch available to the first method. Furthermore, if run_0 was an area of increased bandwidth, the increase in bandwidth will not be the minimum possible increase if prefetching with the first method is used.



(a) Bandwidth Increase Condition (b) Bandwidth Decrease Condition

FIGURE 9. These figures show the two conditions in which a change in bandwidth is made using the optimal bandwidth allocation algorithm. In maximizing the furthest point reached by C in the figures, the lines must touch both $F_{hi}(i)$ and $F_{low}(i)$. A search on the line between B and C are used to find the furthest point for the next run.

For runs that will end before an increase in the bandwidth allocation plan, only one method of prefetch is possible. Suppose we have run_0 which requires an increase in bandwidth in the next run. Allocating at any bandwidth higher than the critical bandwidth, CB_0, for run_0 will result in an overflowed buffer. Therefore, we increase the bandwidth before CP_0 to start prefetching for run_1. To accomplish this, a search is performed between the beginning and end of the run to find a starting point such that the maximum amount of prefetch is possible for the next run and that the buffer will not overflow in run_0 and the critical point for the next run is as far out as possible. This prefetching is graphically shown in Figure 8.

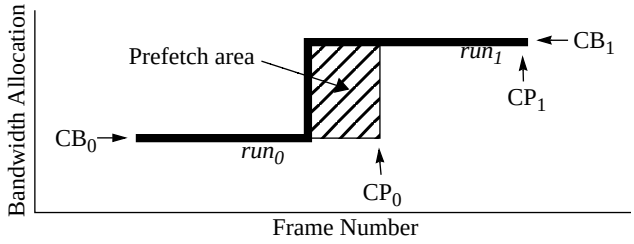


FIGURE 8. This figure shows the only possibility for prefetch for a run, run_1, that increases the bandwidth allocation from the last run, run_0, and keeps the number of bandwidth changes constant. The bold line indicates the actual allocation with prefetch.

2.3.2 The Optimal Bandwidth Allocation Algorithm

The optimal bandwidth allocation algorithm allocates runs such that the critical point for each run is as far out as possible. For our geometric model, allocating runs such that the critical points are as far out as possible corresponds to drawing the longest lines possible that do not cross $F_{hi}(i)$ or $F_{low}(i)$. As a result, at the end of a particular line (run), there are two possibilities for the next run, either increase or decrease the bandwidth requirement. These cases are shown in Figure 9. A search along the line that touches $F_{hi}(i)$ and $F_{low}(i)$ is performed to find the intersection point of a line segment that reaches the farthest point out in time. This new line segment will maximize the critical point for the next run, while providing a transition from the last run to the current run. A sample construction is shown in Figure 10.

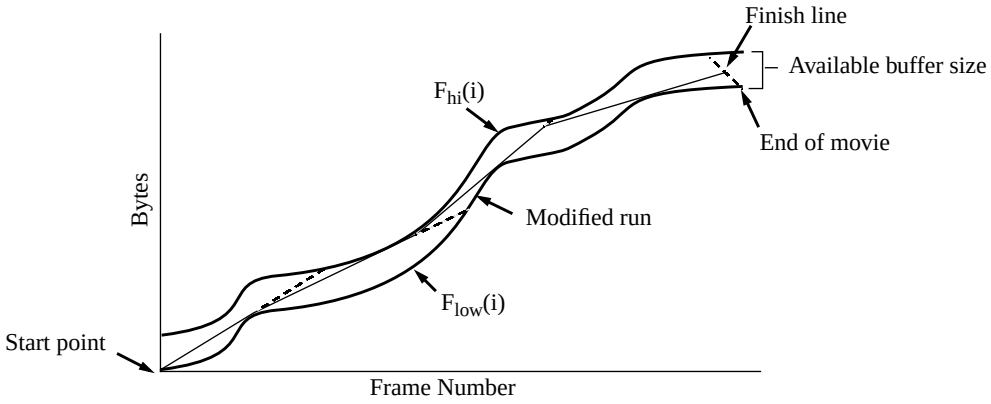


FIGURE 10. This figure shows a sample construction of the optimal bandwidth allocation algorithm. Note, this plan includes only 4 runs while the plan in Figure 4 required 6 runs. The heavy solid lines show $F_{hi}(i)$ and $F_{low}(i)$, while the light solid lines show the slopes (bandwidths) selected by the optimal critical bandwidth allocation algorithm. The dotted lines show the lines along which the searches were performed to maximize the critical points of the following runs.

Formally, the construction of the optimal bandwidth allocation plan consists of three types of allocations: the beginning run, a run that decreases the bandwidth allocation, and a run that increases the bandwidth allocation.

The beginning run does not have any prefetch in order to minimize the latency between channel set-up and the beginning of playback. Therefore, the first run, is set to the critical bandwidth and critical point for the run starting at the beginning of the movie.

In the calculation of a run that decreases the bandwidth allocation, a search (as described in Figure 9b) is performed to determine how far the bandwidth should be held past the end of the last run into the current run. The search finds a frame, *j*, such that using the same bandwidth allocation from the end of the last run results in the critical point in the current run to be as far out as possible. The bandwidth for the run is then set to the critical bandwidth of the last run up to, and including, frame *j*, while the bandwidth from frame *j+1* to the critical point is set to the critical bandwidth of the current run.

In the calculation for a run that increases the bandwidth allocation, a search (as described in Figure 9a) is performed in the end of the last run to find a frame, *k*, to start prefetching for the current run such that the current run will be as long as possible. The current run is then started on frame *k* and has its bandwidth allocation set to the critical bandwidth starting from frame *k* with its critical point determining the end of the run.

In order to create the optimal bandwidth allocation plan, we first find the critical bandwidth and critical point for the run beginning on the first frame of the video. Next, using the conditions from Figure 9, we determine whether the next run will increase or decrease the bandwidth allocation. Accordingly, we then apply the appropriate calculation for a run that decreases the bandwidth allocation or for a run that increases the bandwidth allocation. For each subsequent run, we simply apply the same algorithm to determine which of the calculations to use (whether for increasing or decreasing the bandwidth). This results in a plan that has the minimum total required bandwidth changes necessary to play back the video without

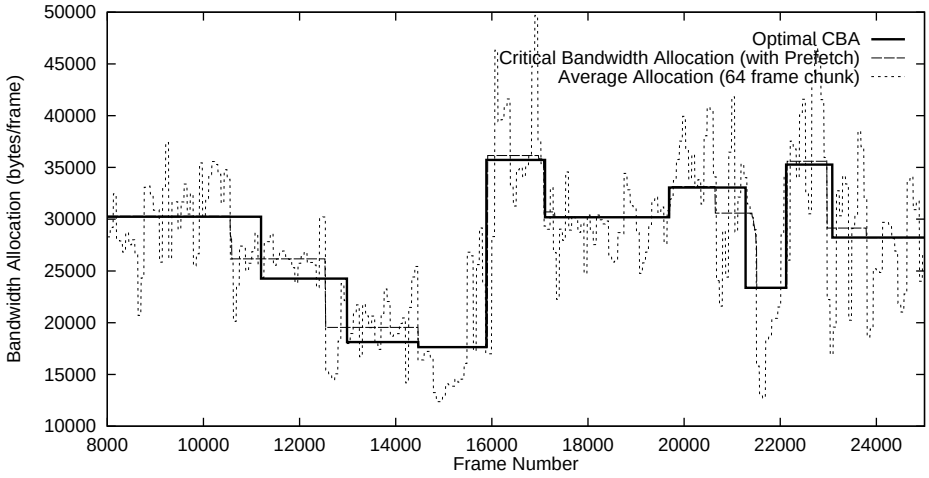


FIGURE 11. Each graph represents the bandwidth requests from a sample of 17,000 frames from the movie *Star Wars* The medium dotted line is the critical bandwidth allocation algorithms with prefetching and a maximum buffer size of 3 Mbytes. The solid line represents the optimized critical bandwidth algorithm with prefetching and a maximum buffer size of 3 Mbytes. The lightly dotted line shows the average allocation of 64 frame groups.

buffer overflow or underflow, while utilizing all of the bandwidth that has been reserved. A sample allocation plan is shown in Figure 11.

2.3.3 A Proof of Optimality

To show that the optimal bandwidth allocation algorithm results in the minimum number of bandwidth changes, we will show that the bandwidths chosen by any other plan will result in at least as many required bandwidth changes.

Theorem 2 : For video playback allocation plans using a fixed size buffer, for which (1) the bytes deliverable are equal to the aggregate size of the video clip and (2) where prefetching at the start of the movie are disallowed, the optimal critical bandwidth algorithm results in the minimum number of bandwidth changes.

Proof: We will show by contradiction that the optimal bandwidth allocation algorithm results in the minimum number of bandwidth changes for such plans. As described previously, the optimal critical bandwidth algorithm chooses each run such that the critical points are as far out as possible.

Suppose the optimal bandwidth allocation algorithm creates a bandwidth plan, $plan_{opt}$, that has X bandwidth changes in it. Further, suppose that this plan is not optimal in the number of bandwidth changes. Therefore, another plan, $plan_{better}$, must exist that has *fewer* than X bandwidth changes in it. As a result, there must exist at least one run in $plan_{better}$ that spans greater than one run from $plan_{opt}$. As will be shown, this cannot happen.

The first run in $plan_{opt}$, whether it requires an increase or decrease in bandwidth in the next run, will result in a plan that has a critical point greater than or equal to the first run in $plan_{better}$. If an increase in bandwidth is required in the next run (Figure 9a), $plan_{opt}$ picks the bandwidth such that any more bandwidth would result in buffer overflow. Any bandwidth higher results in buffer overflow before the critical point of the first run in $plan_{opt}$. Any less bandwidth results in a critical point that is before the critical point of the first run in $plan_{opt}$. If a decrease in bandwidth is required in the next run, then by definition, $plan_{opt}$ has chosen the

minimal bandwidth necessary without overflow resulting in the furthest critical point possible. Thus, $plan_{better}$ cannot have a critical point that is further out than $plan_{opt}$ for the first run.

For each run after the first run, $plan_{opt}$ starts by examining the end of the last run and finds a starting frame that will maximize the critical point of the current run. This search is always performed between points B and C as shown in Figure 9. Because this search is on a line that connects F_{hi} and F_{low}, $plan_{better}$ cannot pick a next run that is longer than the one chosen by $plan_{opt}$, otherwise, $plan_{opt}$ would have found it in its search. We continue this process for all runs within the $plan_{opt}$. Because every ith run in $plan_{better}$ cannot have a critical point further than the ith run in $plan_{opt}$, $plan_{better}$ must have at least as many runs as $plan_{opt}$. Therefore, $plan_{opt}$ results in the fewest number of bandwidth changes.//

As shown by Theorem 2, the optimal bandwidth allocation algorithm results in the minimum number of bandwidth changes required to play back a video clip without buffer underflow or overflow. It is possible, however, that there will be several plans which require the same minimum number of bandwidth changes. For runs which end in a bandwidth increase, the bandwidth used by the optimal bandwidth allocation algorithm is the highest possible bandwidth without buffer overflow. Additionally, the runs which require a bandwidth decrease in the next run are allocated at the minimum possible bandwidth without buffer starvation. Therefore, the optimal critical bandwidth algorithm not only finds the minimum number of bandwidth changes necessary, it also creates a bandwidth plan with the minimum total increases (in magnitude) of bandwidth.

3.0 Evaluation of Algorithms

From the point of view of network management, load estimation and admission control are crucial to providing guarantees of service. These can be greatly simplified if all channels exhibit constant behavior. In the absence of an entirely constant bandwidth allocation, the amount each channel strays from this constant allocation will determine the network's performance. Two measures that influence this performance are the frequency of requests for increased bandwidth and the size of these increases. The frequency and size of decreases can be interesting as well if the network management makes some provision for lowering a bandwidth reservation.

To compare and contrast the differences between the critical bandwidth allocation based approaches, we digitized several full-length movies to use as test data. Our experiments confirmed our theoretical hypothesis and showed that for a small amount of client-side buffering we can indeed reduce the number of bandwidth changes necessary to a small number. To our surprise, we found that having 20 MBytes of buffer space on the client-side could reduce the total number of changes for any of the clips tested to less than 10. In the rest of this section, we will describe our experimental set-up and the video clips that were digitized. In addition, a more in-depth look at the performance of the critical bandwidth allocation based algorithms will be presented.

3.1 Experimental Set-up

To test the effectiveness of the various smoothing algorithms, we needed to digitize a few full-length movies. In previous work, we found that using the critical bandwidth allocation approach on smaller clips created plans that required no increases in bandwidth and required small amounts of buffering to achieve this. At the beginning of our work on video bandwidth smoothing, our test data consisted of a single *Star*

Wars MPEG-compressed movie. For this paper, we used a PC based video capture testbed to digitize the movies *Jurassic Park* and *E.T. - the Extra Terrestrial*. To test various bit rates for the same movie, we digitized *E.T.* at three different constant picture qualities.

Our PC testbed consisted of a Pioneer Laser Disc player, a MiroVideo DC1tv capture board, and a Pentium 90 processor with 32 MB of memory. The MiroVideo Capture board is a Motion-JPEG compression board. Because the critical bandwidth algorithms are most sensitive to the changes in scene content, we felt the additional (order of magnitude) cost for a real-time MPEG encoder was not worth the cost. Furthermore, because the basic routine for encoding I-frames within an MPEG video are based on the JPEG compression standard, the frame sizes for our experimental video data are roughly equivalent to all I-frame encoded MPEG video movies. The MiroVideo board digitized the movies at 640x480 and then subsampled them to 320x240 with guaranteed VHS picture quality.

Using our testbed, we digitized 4 movies, *Jurassic Park* once, and *E.T.* at three frame qualities, 75, 90, and 100. The different picture qualities determine how coarse the quantization matrices are during encoding. For our sample *E.T.* video, picture qualities of 75, 90, and 100 resulted in bits per pixel measurements of 0.66, 0.94, and 1.64 bits per pixel, respectively. According to an introductory JPEG paper, 0.66 bits per pixel corresponds to ‘‘good to very good’’ quality, 0.94 bits per pixel corresponds to ‘‘excellent’’ picture quality, and 1.64 bits per pixel corresponds to a quality that is ‘‘usually indistinguishable from the original’’[10]. The statistics for these digitized movies along with the statistics for *Jurassic Park* and *Star Wars* are summarized in Table 1. It is interesting to note that the *Star Wars* frame sizes were much larger on

Video Clip Name	Coding	Quality	Length	Ave. Bit Rate	Max Frame Size (bytes)	Min Frame Size (bytes)
Star Wars	MPEG	N/A	1 hour 35 min	6.6 Mbps	78459	321
ET 75	MJPEG	75	1 hour 50 min	1.5 Mbps	14269	717
ET 90	MJPEG	90	1 hour 50 min	2.2 Mbps	19961	885
ET100	MJPEG	100	1 hour 50 min	3.8 Mbps	30553	6827
Jurassic Park	MJPEG	90	2 hours 5 min	2.7 Mbps	23883	1267

TABLE 1: This table shows the statistics of the Motion-JPEG video clips that were digitized with the MiroVideo capture board and used in the evaluation of the critical bandwidth algorithms.

average than the *E.T.* and *Jurassic Park* films, however, they resulted in fairly similar results in terms of bandwidth changes as will be shown shortly.

3.2 Bandwidth changes

To avoid overtaxing network resources, bandwidth allocations have to be tracked and approved by some admission control manager. The total number of changes in bandwidth are particularly important because they may be denied and some adjustment will have to be made, such as a change in quality of service, the establishment of an alternative route, or perhaps the allocation of additional prefetch buffer space. Bandwidth decreases should be simpler to handle from the network point of view because no additional resources from the network will be required to satisfy the request.

The total number of changes in bandwidth can be important, since each change may require an interaction with the network manager. As Figure 12 shows, the optimal bandwidth allocation algorithm resulted

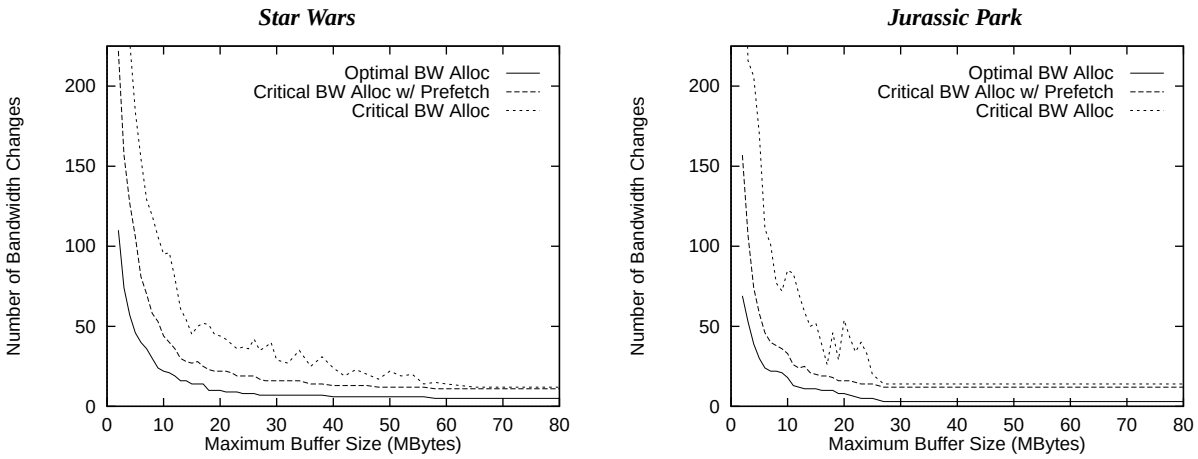


FIGURE 12. The graphs above show the total number of required bandwidth allocation change requests for the *Star Wars* and *Jurassic Park* videos. Each algorithm was run on the entire video clip for varying buffer sizes.

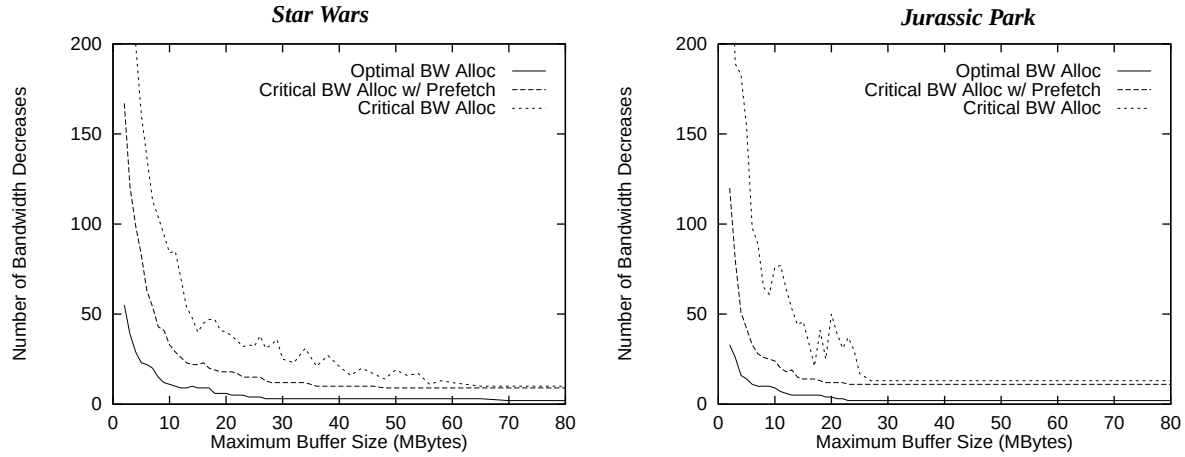


FIGURE 13. The graphs above show the total number of bandwidth allocation decrease requests for the *Star Wars* and *Jurassic Park* videos. Each algorithm was applied to the movie segment with different buffer capacities.

in the fewest number of bandwidth changes for a given buffer size. As an example, using the *Star Wars* video and a 10 MByte buffer resulted in 22 bandwidth changes over the 95 minute movie, while the critical bandwidth algorithm with and without prefetching required 44 and 95 changes in bandwidth, respectively. On average the optimal bandwidth allocation algorithm required a bandwidth change approximately every 4 minutes. After the initial start of the movie, the *Star Wars* movie using the optimal critical bandwidth algorithm had a minimum run length of approximately 2 minutes and a maximum run length of approximately 8 minutes and 20 seconds. The distinction between increases and decreases in the bandwidth allocation plan can be useful because the requests for decreases in bandwidth can generally be satisfied, while increase may require further negotiations with the network. In addition, it highlights the main differences between the various algorithms.

As shown in Figure 13, the total number of bandwidth decreases for both *Star Wars* and *Jurassic Park* had graphs very similar to their respective total number of bandwidth change graphs. Thus, a large percentage of the bandwidth changes are due to decreases in bandwidth, which from a network point of view

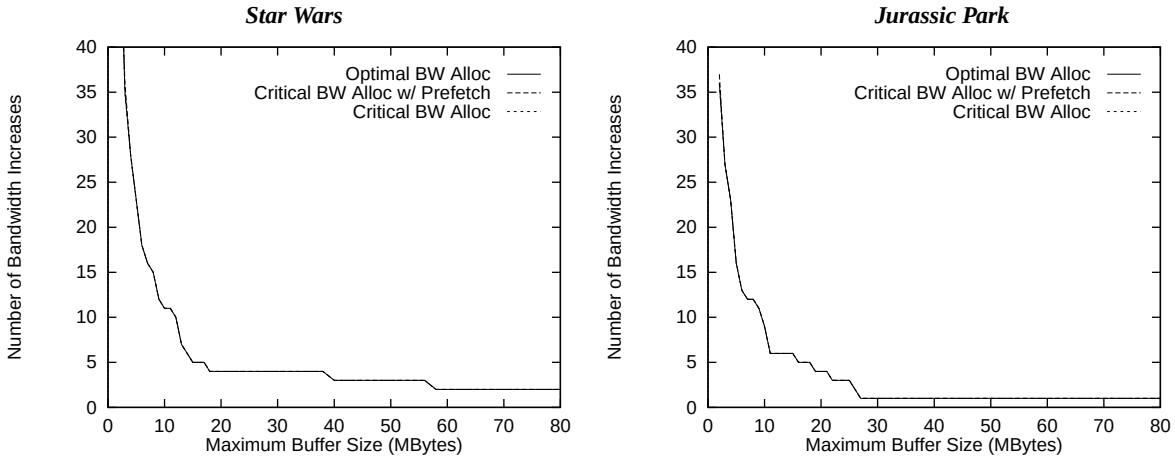


FIGURE 14. The graph above shows the total number of bandwidth allocation increase requests for the *Star Wars* and *Jurassic Park* videos. Each algorithm was applied to the respective movies with different buffer capacities, all resulting in the same number of bandwidth increases as shown above.

should be easier to satisfy. The original critical bandwidth algorithm resulted in an unstable number of bandwidth changes for varying buffer sizes. These are mainly due to the sharp increases in bandwidth at the beginning of segments without prefetching. With a high bandwidth requirement at the beginning of each segment, the critical bandwidth algorithm required more decreases to avoid buffer overflow (See Figure 5). By smoothing these segment boundaries, a lower beginning bandwidth requirement could be used thus reducing the number of required decreases in bandwidth. In comparing the optimal critical bandwidth algorithm with the critical bandwidth algorithm with prefetching, we see that the main difference between these algorithms is in the number of bandwidth decreases (as shown by the same relative differences in total bandwidth changes and total number of decreases). The critical bandwidth allocation algorithm allocates each run at the minimum bandwidth requirement to avoid underflow, while the optimal bandwidth starts each run at the minimum bandwidth requirement but holds the bandwidth past the critical point of the run to prefetch data for the next run.

For bandwidth increases, using the three different algorithms resulted in the same number of increases across all buffer size constraints for each movie, therefore, verifying Theorem 1. As shown in Figure 14, the number of increases required for *Star Wars* and *Jurassic Park* drop below 5 for buffers greater than 15 MBytes.

3.3 Bandwidth Increase Magnitude

Requests for increases in bandwidth allocation will typically require interaction with the network manager for more resources. The frequency and magnitude of these increases will determine the network's ability to adapt to changing network load. Comparison of increases in bandwidth magnitudes are only relevant when the total number of increase requests is the same. As an example, if one stream requests 1000 bytes/frame more bandwidth and another asks for 100 bytes/frame on ten separate occasions, no comparison can be made. Because the critical bandwidth based algorithms have the same number of increases for a given movie, looking at the total magnitude in requested increases in bandwidth allocation may give an advantage to one of the algorithms. As shown in Figure 15, the critical bandwidth algorithm with prefetching and the optimal bandwidth allocation algorithm produce roughly the same total magnitude of band-

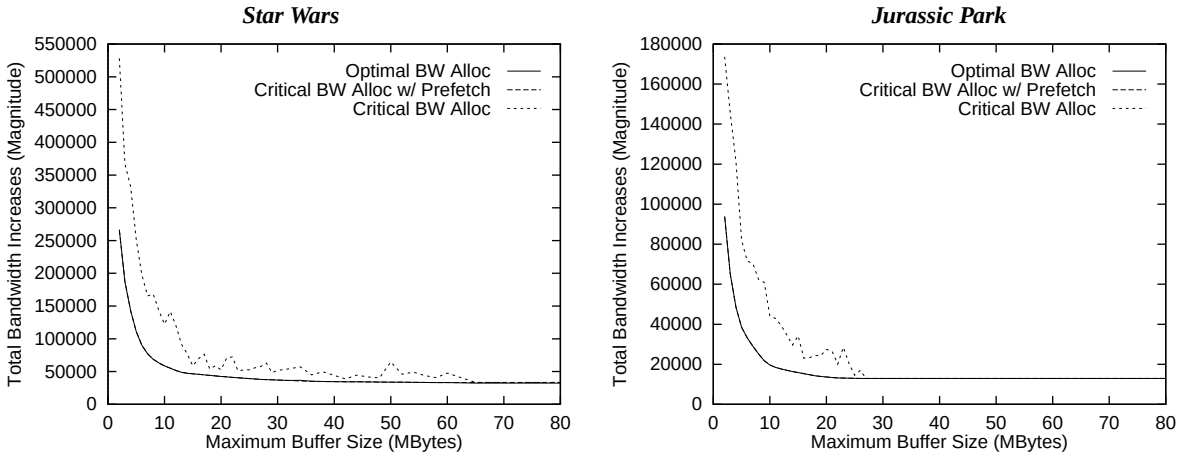


FIGURE 15. The graph above shows the total sum (magnitude) of all the bandwidth allocation increase requests for the *Star Wars* and *Jurassic Park* videos. Each algorithm was applied to both movies with different buffer capacities. The optimal bandwidth allocation algorithm and the critical bandwidth allocation algorithm with prefetch resulted in nearly the same total.

width increases for a given movie, while the critical bandwidth algorithm with no prefetching makes larger increase requests. The difference in the overall magnitude in increases between the *Star Wars* and *Jurassic Park* video is mainly due to larger differences in the small and large frame sizes (and not the actual bit-rates).

3.4 A Comparison of the Various Algorithms

The original intent of the critical bandwidth algorithm was to find the minimal amount of constant bandwidth necessary to play back a video without an initial delay. This results in fairly large spikes at the beginning of each run, but results in fairly long periods of non-increasing bandwidth requirements. Prefetching can be used between the segments to smooth out the bandwidth increase requests. It is interesting to note that, if using the critical bandwidth algorithm with prefetching, the increase requests for bandwidth actually arrive before they are absolutely necessary (except for the initial allocation). Thus, in times of heavy network load with no advanced bandwidth reservations, the end user can delay its need for bandwidth in exchange for an increased bandwidth request. See Figure 16.

The optimal critical bandwidth algorithm makes the fewest possible changes in bandwidth for the duration of video. The trade-off for this optimal allocation is the need to perform a search at the end of each run calculated in the movie. The critical bandwidth algorithm with prefetching requires a search only on runs which will have an increase in the bandwidth allocation in the next run. Finally, the critical bandwidth algorithm with no prefetching requires no searching, but results in larger bandwidth increase requests.

3.5 A Trip to the Movies

The optimal critical bandwidth algorithm results in the smallest number of bandwidth changes required for the playback of a video clip. As Figure 17 shows, for moderately sized buffers the number of bandwidth changes can be reduced to a small number for the playback of each video. Over the range of different bit rates for these videos, each displayed approximately the same number of required bandwidth changes. Thus, the sizes of the individual frames within a movie are not the limiting factor in the number

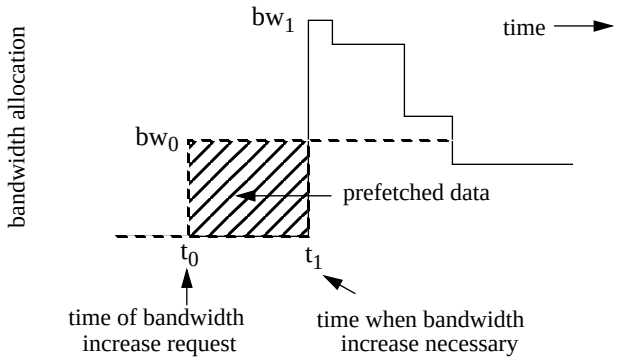


FIGURE 16. This figures shows how prefetching can be traded for network bandwidth when using the critical bandwidth allocation algorithm with prefetching or the optimal bandwidth allocation algorithm. At the time of the increase request, an increase to bw_0 is necessary, however, the network can still meet the required bandwidth necessary by giving more bandwidth (up to bw_1) until t_1. Note, the additional bandwidth required is not a linear interpolation between t_0 and t_1.

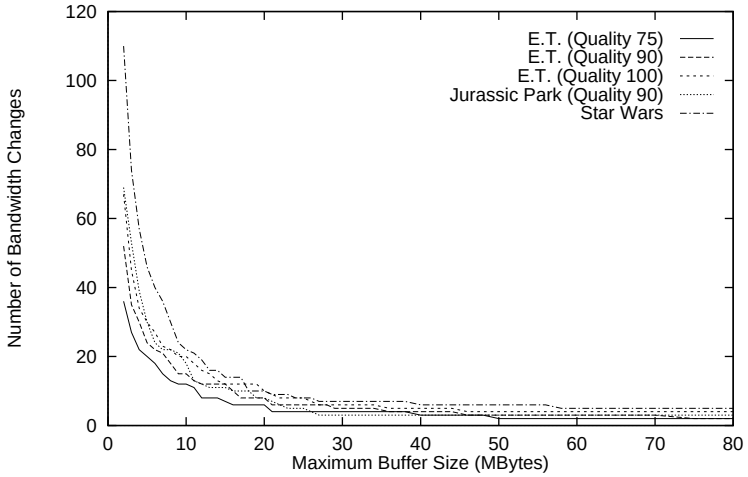


FIGURE 17. This graph shows the total number of bandwidth changes required given some fixed amount of buffer space to play back the various videos without buffer overflow or underflow. As shown above, using a nominal amount of buffer greater than 10MBytes allowed for very few bandwidth changes.

of bandwidth changes require. Instead, it is the difference in sustained peaks and valleys that matters in the total number of bandwidth changes. Using a 20 MByte buffer resulted in all movies requiring less than 10 bandwidth changes, while using a 60 MByte buffer resulted in all movies requiring less than 5 bandwidth changes.

4.0 Summary and Conclusions

The smoothing of video data will play an important role in the design of video playback systems. Smoothing of compressed video data through prefetching allows the data delivery service to substitute buffer bandwidth for network bandwidth. In this paper, we have proven that the critical bandwidth allocation algorithm with a maximum buffer constraint results in the minimum number of bandwidth increases necessary for the playback of stored video streams. We have also introduced the notion of the optimal bandwidth

allocation algorithm which is based on the critical bandwidth algorithm. As we have shown, this optimal bandwidth allocation algorithm makes the best use of the available buffer bandwidth and minimizes the total number of bandwidth changes required for playback.

Many live video applications have fairly consistent bandwidth requirements across time, although compression will lead to pattern burstiness at a small scale. This sort of video stream can be smoothed by an algorithm with a narrow window. On the other hand, streams such as stored movies are inhomogeneous at the level of scenes within the movie. To effectively smooth such streams, it is necessary to examine longer segments within the stream. This is the approach taken by the critical bandwidth and the optimal bandwidth allocation algorithms. Both algorithms aggressively prefetch data to smooth network bandwidth requirements and differ on the approach used in returning bandwidth back to the network. The critical bandwidth algorithm calculates a bandwidth plan such that the bandwidth is returned as soon as the critical point that forced the bandwidth is passed. The optimal bandwidth allocation algorithm, on the other hand, continues to hold the bandwidth until the optimal amount of prefetch for the next run is obtained.

The choice between using the optimal bandwidth allocation algorithm and the critical bandwidth algorithm with a maximum buffer constraint depends on the network cost model. In some cases, it may be more beneficial for the network to have the clients allocate at their minimum constant bandwidth requirements, making the critical bandwidth algorithm more useful. In other cases, it may be more beneficial for the network to have as few interactions with the video application as possible, in which case, the optimal bandwidth allocation algorithm may be the appropriate choice.

The amount of buffering needed in any system will depend upon the size of the data stream and the effectiveness of encoding. The video clips used in our analysis do not take advantage of inter-frame dependencies, however, these interframe dependencies will tend to reduce the number of bandwidth changes required as long as the pattern of frame types is repeating. Nonetheless, our results are indicative of the resources required for smooth video delivery. These requirements are not large in terms of today's workstations and should be achievable for tomorrow's televisions.

By prefetching effectively, critical bandwidth based allocation algorithms minimize the total number of bandwidth increases necessary while the optimal critical bandwidth algorithm minimizes the total number of bandwidth changes made as well. Because these segments persist for considerable amounts of time, lowering their bandwidth requirement periodically, the ability to return bandwidth back to the network without re-establishing the channel can be beneficial.

5.0 References

[1] CCITT Recommendation MPEG-1, ''Coded Representation of Picture, Audio, and Multimedia/ Hypermedia Information,'' ISO/IEC 11172, Geneve Switzerland, 1993.

[2] D.M. Cohen, D.P. Heyman, ''A Simulation Study of Video Teleconferencing Traffic in ATM Networks. In *IEEE INFOCOM 1993*, pp 894-901, 1993.

[3] W. Feng, S. Sechrest, ''Critical Bandwidth Allocation for the Delivery of Compressed Prerecorded Video,'' *Computer Communications*, Oct. 1995.

[4] S. Lam, S. Chow, D. Yau, ''An Algorithm for Lossless Smoothing of MPEG Video,'' In *ACM SIGCOMM Conference Proceedings*, 1994.

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

[6] P. Pancha, M. El Zarki, ‘‘Prioritized Transmission of Variable Bit Rate MPEG Video,’’ In *IEEE Globecom 1992 Proceedings*, 1992, pp. 1135-1139.

[7] D. Reininger, D. Raychaudhuri, B. Melamed, B. Sengupta, J.Hill, ‘‘Statistical Multiplexing of VBR MPEG Compressed Video on ATM networks,’’ In *IEEE INFOCOM 1993*, pp 919-926, 1993.

[8] L. Rowe, K. Patel, B. Smith, K. Liu, ‘‘MPEG Video in Software: Representation, Transmission, and Playback,’’ In Proceedings *SPIE High Speed Networking and Multimedia Computing 1994*, San Jose, CA, Feb. 1994.

[9] D. Stone, K. Jeffay, ‘‘Queue Monitoring: A Delay Jitter Management Policy,’’ In *Proceedings of the 4th International Workshop on Network and OS Support for Digital Audio and Video*, 1993.

[10] G.K Wallace,‘‘The JPEG Still Picture Compression Standard’’,*Communications of the ACM*, Vol. 34, No. 4, (Apr. 1991), pp. 30-44.

[11] H. Zhang, D. Ferrari, D.C. Verma, ‘‘Delay Jitter Control for Real-Time Communications in a Packet Switching Network,’’ *Computer Communications*, Vol. 15, No 6, pp 367-373, Jul./Aug. 1992.