Computer Communications (Special Issue on System Support for Multimedia Computing)
Vol. 18, No. 10, October 1995, pp. 709-717

Critical Bandwidth Allocation for the Delivery of Compressed Video

Wu-chi Feng and Stuart Sechrest
Software Systems Research Lab
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, Michigan

Abstract

The transportation of compressed video data without loss of picture quality requires the network to support large fluctuations in bandwidth requirements. These fluctuations can be smoothed, but straightforward approaches to smoothing can still suffer from excessive buffering requirements, poor buffer utilization, and an excessive number of bandwidth changes. This paper introduces critical bandwidth allocation, which reduces the number of bandwidth changes to a very small number, and achieves the maximum effectiveness from client-side buffers. A comparison between critical bandwidth allocation algorithms and other smoothing algorithms is presented, the sensitivity of the algorithm to jitter is examined, and implications for the design of network services are discussed.

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

Introduction

The transportation of video for multimedia applications relies on both high speed networking and data compression. To an extent these reliances are contradictory because efficient network management is made more difficult by the burstiness that effective data compression induces. For live video applications, the problem of delivering video data is constrained by the fact that decisions must be made on-line and delay between sender and receiver must be minimized. As a result, live video applications may have to settle for weaker guarantees of service or some degradation in quality of service. Work on problems raised by the requirements of live video includes work on statistical multiplexing[2,10], smoothing in exchange for delay[5], jitter control[11,13], and adjusting the quality of service to fit the resources available[7,8]. With stored video, however, the approaches to delivery of the video data are less constrained because the full video stream is known *a priori*. As a result, prefetching can be used as an effective tool to control the burstiness of the bandwidth required of the underlying network without loss of picture quality.

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 volatile. 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 this paper, we introduce the notion of *critical bandwidth allocation*, an algorithm that provides plans for smoothing the network bandwidth by utilizing a buffer of a fixed size. Smoothing algorithms that look at only a small window within the stream will often fail to prefetch sufficient data to overcome large demand bursts. The critical bandwidth allocation algorithm is able to make full use of the prefetch buffer because it looks at the entire clip to be smoothed, as is possible for stored video applications. The algorithm ensures that the available buffer space will be filled in advance of demand, which allows the network bandwidth requirement to be kept as low as possible. Increases in bandwidth, which will require negotiation with a network manager if bandwidth guarantees are offered, are made only when the buffer size limit prevents further prefetching. Thus, the algorithm guarantees that the plan will not cause the available buffer to underflow or to overflow while producing the fewest possible increases in requested bandwidth.

The critical 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, but may be relatively large. 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. Finally, while live video requires low-latency jitter-free delivery, these are of much lesser importance when using buffering for network bandwidth smoothing.

In the following section, approaches to bandwidth smoothing, including critical bandwidth allocation, are described. The evaluation section compares and contrasts the alternative smoothing algorithms using an MPEG encoded movie of sufficient length to study the behavior of the algorithms over time. In addition, the sensitivity of the critical bandwidth algorithm to network jitter is discussed. Finally, a summary and conclusions about the importance of smoothing to the design of network services is presented.

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 provide insight into the effectiveness of the various smoothing algorithms. For pre-recorded streams compressed with algorithms such as MPEG[1,6], 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. Simple averaging algorithms are good at smoothing out local pattern burstiness, but do little to control the variability at the larger scale.

Average Bandwidth Allocation

A very simple method for easing bandwidth fluctuations is to group some number of frames together into a ``chunk'' and send them at the average bandwidth requirement for the group. Figure 1 shows simplified code for this and subsequent algorithms. By using this method, clients can guarantee that the bandwidth needed will be constant throughout the chunk. The amount of smoothing, however, is directly related to the chunk size. Using this constant bandwidth for the entire chunk implies that the end user is not guaranteed that local buffer starvation will not occur during the display of the chunk unless it has buffered at least one chunk ahead. Using bandwidth smoothing with large chunk sizes becomes impractical because of this buffering requirement. In addition, because this algorithm merely groups a fixed number of frames together, no smoothing occurs across chunks. This can be seen in Figure 2.

Average Allocation

```
num_chunks = num_frames/chunk_size;
for (i=0 ; i<num_chunks ; i++)
   {
      ave = average frame size of chunk;
      set allocation for the chunk = ave
   }
```

Sliding Window Smoothing

```
num_chunks = num_frames/chunk_size;
calculate the average allocation graph;
for (i=0 ; i < num_chunks - window_size ; i++)
   {
      ave = average for chunks within the window;
      for (j=i ; j < i + window_size ; j++)
         {
            if (allocation of chunk j > ave )
               smooth allocation of chunk j by prefetching it
                    earlier in the window;
         }
   }
```

Critical Bandwidth Allocation

```
i=0;
while (i < last_frame)
   {
      maxave = maximum ave. bytes/frame from i to last_frame;
      maxpt = point where maxave occurred;

      // check for buffer overflow
      if (buffer from i to maxpt will be exceeded at maxave)
         {
            calculate maxave and maxpt such that the new
                    maxpt will be the furthest point from i
                    without buffer starvation or overflow.
         }

      set bandwidth allocation from i to maxpt = maxave;
      i = maxpt+1;
   }
```

FIGURE 1. This pseudo-code presents a high-level description of the bandwidth smoothing algorithms described. Details of implementation and optimization are omitted.

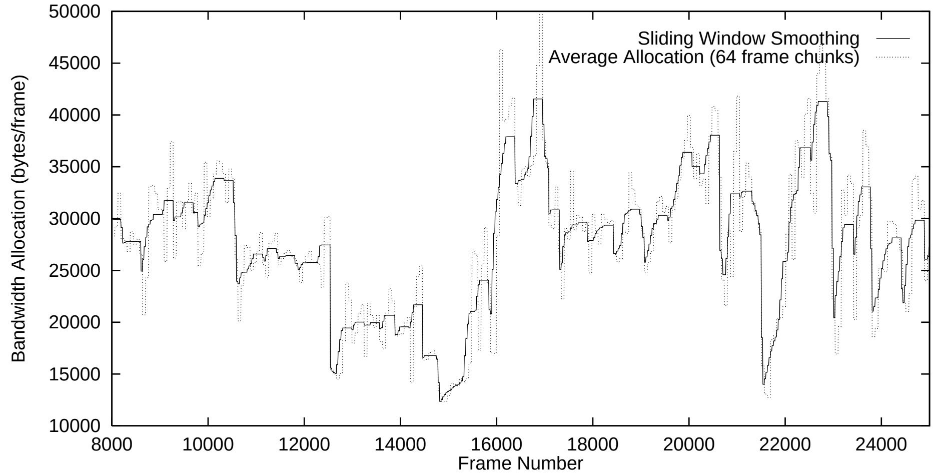


FIGURE 2. Each line in the graph represents a sample of bandwidth requests from a 17000 frame (approx. 9.5 minute) sample from the movie *Star Wars*. The video has been MPEG-encoded using I-frames only. The burstiness is therefore entirely due to differences in scene content. Each algorithm was run assuming a maximum buffer of 3 MBytes. The dotted line shows the average allocation algorithm for 64 frame chunks. The solid line represents the sliding window smoothing applied to 32 frame chunks with a window size of 6 chunks.

Sliding Window Smoothing

Rather than simply averaging within fixed chunks, a sliding window can be used to smooth within a larger region. A moving window of *n* chunks is smoothed by shifting bandwidth from the last chunk to earlier chunks that stand below the average for the window as a whole. This has the effect of causing the data to be prefetched in advance of bursts of large frames. However, redistribution is limited by the size of the window, so peaks and valleys will still occur.

Figure 2 shows the effect of sliding window smoothing. The amount of smoothing is determined by the window size. A window size of 1 yields no smoothing at all and the graphs are equivalent to the average bandwidth allocation algorithm. With sufficiently large windows considerable smoothing is possible. This smoothing is accomplished through prefetching and will require buffer space in the receiver, but far less than would be required to achieve equivalent smoothing with the average allocation algorithm.

Sliding window smoothing, however, has some drawbacks for planning bandwidth allocation. While the number of adjustments in bandwidth is less than required for the average allocation algorithm, frequent adjustments are still necessary. Furthermore, rises in bandwidth are generally accomplished in several steps, each requiring negotiation with the network manager, while drops are sharper.

Critical Bandwidth Allocation

A client using either of the previous algorithms must ensure that its buffer will not be exhausted by buffering one chunk ahead of the chunk being displayed. To remove this restriction, we introduce the notion of critical bandwidth allocation. We describe an algorithm to calculate a plan for the delivery of video data. Here, we say that each video clip consists of one or more *segments* with non-increasing bandwidth requirements, and each segment consists of one or more *runs* requiring constant bandwidth. In contrast to the previous algorithms, the critical bandwidth allocation algorithm produces plans with very few increases in bandwidth and, hence, few segments.

We define the *critical bandwidth* as the minimum constant bandwidth necessary to play a video clip through without buffer starvation. This critical bandwidth, CB_0, is defined as

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

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.

If we divide a video clip in two following CP_0, the run ending at CP_0 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. The initial run will have the same critical bandwidth as the entire clip since it is for this initial run that the average frame size is maximized. To see that the run following CP_0 must have a critical bandwidth less than CP_0, assume that this run has a critical bandwidth CB determined by a critical point CP. By the definition of critical bandwidth, CB_0 must be greater than the average frame size from frame 1 to frame CP. This requires the following inequality to be true

$$CB_0 > \frac{(CP_0 \cdot CB_0) + (CP - CP_0)CB}{CP}$$

By rearrangement, this requires

$$CB_0 > CB_0 + \frac{(CP - CP_0)(CB - CB_0)}{CP}$$

to hold. Since CP is greater than CP_0, CB must be less than CB_0.

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

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

(We ignore rounding in the definition of CB_n. To calculate critical bandwidths precisely, the calculation of CB_n must be adjusted to account for the rounding up of the critical bandwidth in CB_{n-1}.)

Delivering video data at the critical bandwidth prior to each critical point and decreasing step-wise at the critical point to the next critical bandwidth 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. This plan does not observe any limits in available buffer space. In practice, the buffer size used depends on the long-term burstiness of the video, which in turn, depends on the encoding scheme used. For example, we have found that for a 13 minute clip of I, P, B encoded frames from the movie *Honey, I Blew Up the Kid*, a buffer size of 3.4 MBs would suffice for the critical bandwidth allocation algorithm. For a 45 minute clip of *Star Wars* encoded using only I-frames, and consequently having sustained peaks and valleys in frame sizes, required a buffer size of 38 MBs.

The buffer requirements of the plan generated by the critical bandwidths may exceed the buffer space available. However, the buffer requirement can be reduced by breaking the video clip into shorter segments, each with a monotonically decreasing bandwidth allocation plan. Each shorter segment can begin with a relatively high network bandwidth allocation, lessening the reliance on the buffer to smooth out burstiness. To calculate a bandwidth allocation plan that takes full 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 ith frame within the run.
- $MaxBW_i$ be the maximum average bandwidth sustainable from the beginning of the run to the ith frame that will not overflow the buffer. This can be defined as

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

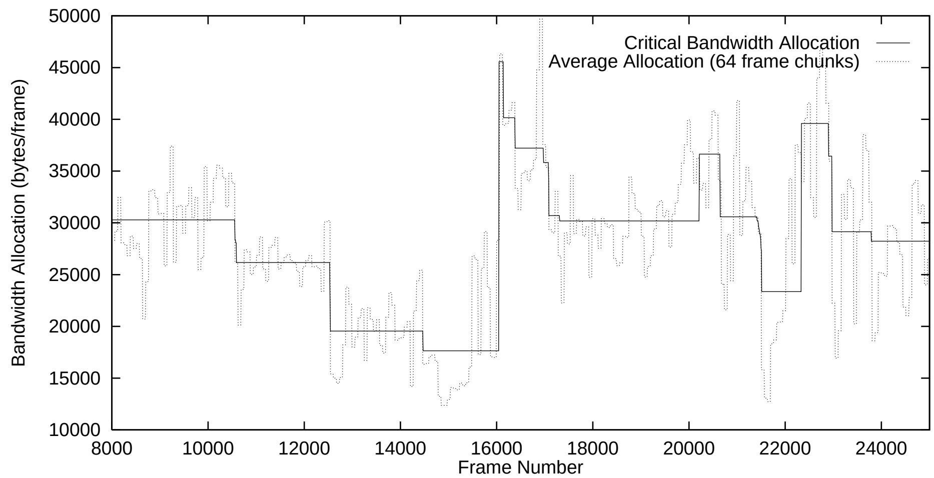


FIGURE 3. Each graph represents the bandwidth requests for a sample of 17,000 frames (approx. 9.5 minutes) from the movie *Star Wars*. The dotted line shows the average allocation of 64 frame chunks. The solid line represents the critical bandwidth algorithm with a maximum buffer of 3 MBytes. The video clip has been divided into 4 segments varying in size from 2120 to 9056 frames (71 and 302 seconds, respectively). Negotiations for increased bandwidth are, thus, quite infrequent.

Then, in calculating the critical bandwidth sequence with a constrained buffer, we check two conditions at each frame, i, in a run. First, if

$$FrameAve_i > MaxBW_i$$

then the bandwidth required to reach frame i will overflow the buffer before frame i is reached. We must therefore make frame i the first frame of a new segment where the bandwidth can be increased. We set the bandwidth for the run ending at frame i-1 to $MaxBW_i$. Second, if

$$\max_{1 \le j \le i} FrameAve_j > MaxBW_i$$

then the run will exceed the available buffer space at frame i. In this case, the frame j for which the average frame size is maximized is a critical point. The critical bandwidth for the run ending at frame i will be $FrameAve_j$, and a new run with a lower bandwidth allocation can be started with frame j+1. Similarly, if frame i is the last frame of the video, then the frame j is again a critical point and the same procedure is followed.

A data delivery plan produced by the critical bandwidth allocation algorithm increases the bandwidth allocation at exactly the points where additional prefetching is not an option since the buffer reaches capacity during the previous run. It decreases the bandwidth at those points when the burst requiring prefetching has passed and the prefetched data has been displayed. These decreases are also accompanied by an empty buffer. A plan using a 3 MByte buffer is

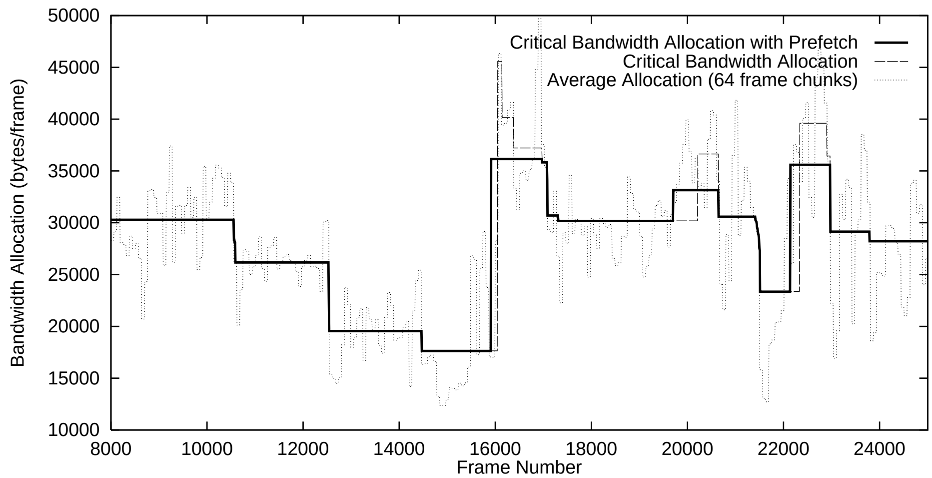


FIGURE 4. Each graph shows the bandwidth requests from a sample of 17,000 frames from the movie *Star Wars*. The solid line represents the critical bandwidth algorithm with prefetching and a maximum buffer size of 3 Mbytes. The remaining two lines represent the critical bandwidth allocation algorithm without prefetching and the average allocation algorithm as in Figure 3.

illustrated in Figure 3. As can be seen, the critical bandwidth allocation algorithm results in fairly long times between bandwidth changes even with a relatively small buffer size.

Critical Bandwidth Allocation with Prefetching

A problem with the critical bandwidth allocation algorithm is that prefetching is not done across segment boundaries. Consequently, at the end of each segment, the buffer will be drained and a bandwidth spike may be needed at the beginning of each new segment.

It is generally possible to modify the bandwidth increase junctures between segments to begin prefetching data from the latter segment without causing the buffer to overflow. Given a bandwidth delivery plan created with the critical bandwidth algorithm, an adjusted plan with prefetching can be created. To calculate the critical bandwidth allocation with prefetching, each bandwidth increase in the plan is examined. The size of the bandwidth increase needed is reduced by substituting a smaller increase towards the end of the previous segment, at the earliest point that will not cause the buffer to overflow. The modified algorithm must still satisfy the property that within a segment the total bandwidth must decrease monotonically. This adjustment is illustrated in Figure 4.

Evaluation of the Critical Bandwidth Allocation Algorithm

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. Another concern is the sensitivity of the allocation plans to fluctuations in network performance. In particular, we examined the likelihood of buffer overflows and underflows due to jitter in the arrival rate of packets from the network.

Costs of Bandwidth Changes

To test the effectiveness of the various smoothing algorithms, each algorithm was run on an MPEG-compressed 45-minute video segment from the movie *Star Wars*. The statistics for this I-frame encoded stream were obtained from Bellcore.

Figures 5 and 6 show the results of smoothing using a 15 MByte buffer. These graphs show that the critical bandwidth approach produces smooth allocations with infrequent adjustments that take full advantage of the available buffer. The entire 45 minutes could be played with only 3 negotiations for increased bandwidth.

To avoid overtaxing network resources, bandwidth allocations have to be tracked and approved by some network manager. Increases 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.

Increases in Bandwidth

In Figure 6, the total number of bandwidth increases is shown as a function of buffer size. Sliding window algorithms must make many small bandwidth increase requests. The relatively small chunk sizes that they use can require bandwidth increases every 1 to 2 seconds. The average allocation algorithm uses larger chunks for an equivalent buffer size and thus makes fewer increases in bandwidth than the sliding window algorithm. The critical bandwidth algorithm makes a very low number of increase requests.

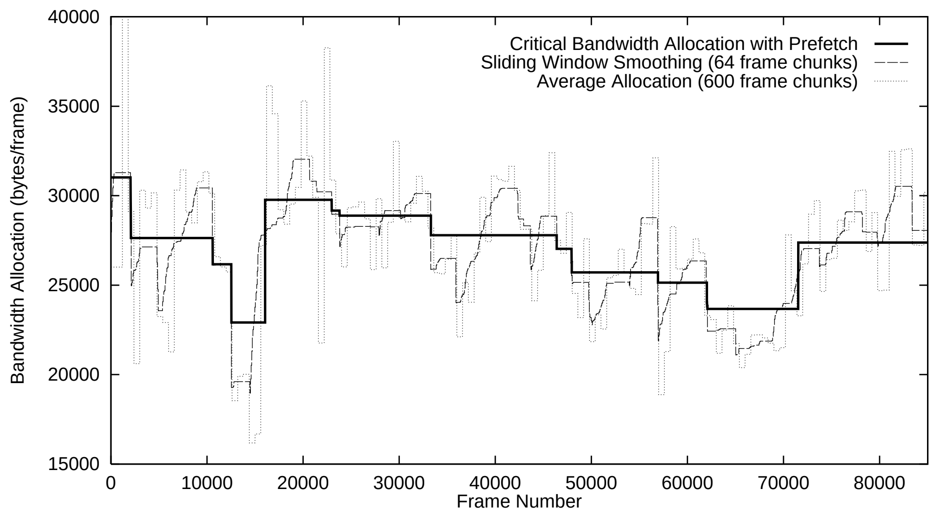


FIGURE 5. This graph shows results of smoothing using the three algorithms with 15 MBytes of buffering available. The data above was obtained by using a 45-minute segment from the movie *Star Wars* (85000 frames). For the average allocation algorithm, the chunk size was increased to take full advantage of the buffer. Because the sliding window smoothing algorithm has two possible parameters, chunk size and window size, we fixed the chunk size at 64 frames and expanded the window to take advantage of the buffer.

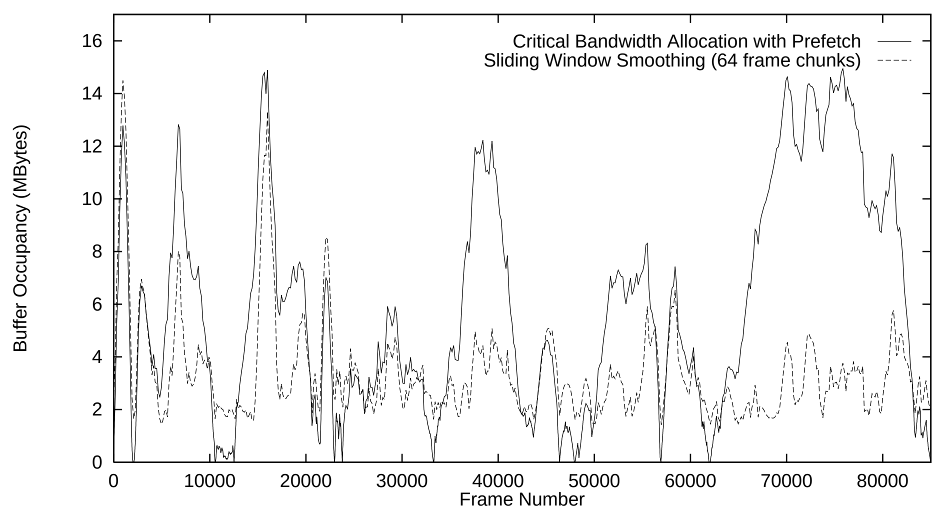


FIGURE 6. This graph shows the utilization of a 15 Mbyte buffer for the sliding window smoothing and the critical bandwidth algorithms. The size of the sliding window is a constant, set in such as way as to make use of the full buffer in the worst case. However, the algorithm is only able to use the buffer in the first part of the segment. A larger window in the second half of the segment would have taken better advantage of the buffer. The critical bandwidth algorithm adjusts its bandwidth to take full advantage of the buffer whenever necessary.

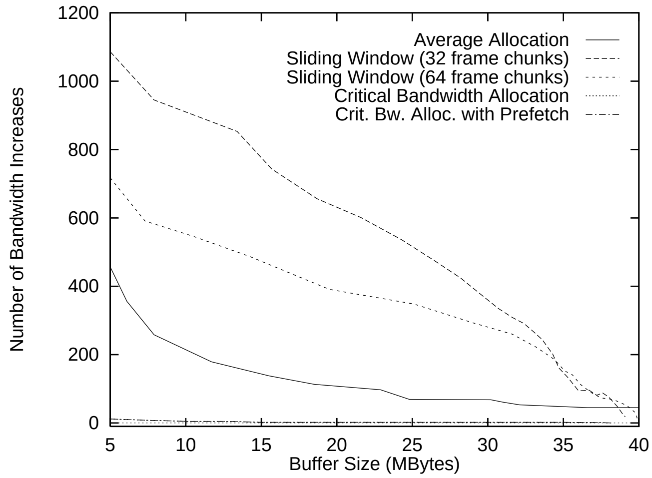


FIGURE 7. The total number of bandwidth increases represent the number (not magnitude) of bandwidth increase requests for the 45-minute *Star Wars* segment. Each algorithm was applied to the movie segment with different buffer capacities. The average allocation algorithm varied the chunk size in order to make use of the extra buffer space. The sliding window algorithms chunk sizes were fixed at 32 and 64 frame chunks and then the window size was varied to obtain the amount of smoothing given some buffer space. The critical bandwidth algorithms used the buffer size to determine the bandwidth requests.

The size of a requested increase can be important, since the larger the increase, the less likely the network manager will have sufficient bandwidth available. Critical bandwidth allocation makes adjustments that are larger on average than does sliding window smoothing. However, this is often because sliding window smoothing will break a large change in bandwidth into a number of requests. It is, therefore, useful to examine the total size of increases requested from the different algorithms.

As Figure 8 shows, the average allocation algorithm makes the worst use of buffer space. The primary limitation of this algorithm lies in its inability to effectively prefetch bytes. The only way for the algorithm to obtain a lower increase cost was to increase the size of the chunks. By using the sliding window smoother, the size of the chunk can remain relatively small, alleviating the amount of buffer space used to store the next chunk. By using this extra buffer space to prefetch bursts of large frames, the sliding algorithms are more effective in smoothing bandwidth curves than the average allocation algorithm. The critical bandwidth algorithm without prefetching resulted in a total cost less than the sliding window algorithm for buffer sizes under 27 MBytes and slightly higher above 27 MBytes. Without prefetching, the critical bandwidth algorithm can have fairly large spikes at the beginning of each segment resulting in a somewhat unstable cost. With prefetching, however, the total cost is lower.

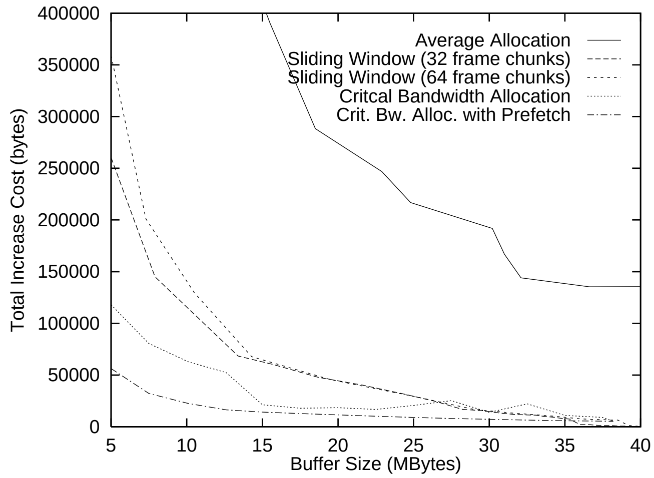


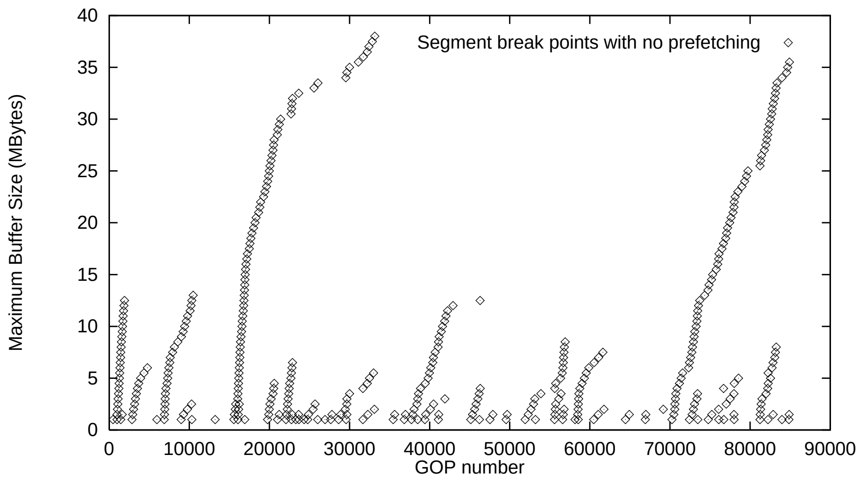
FIGURE 8. The total increase cost of each algorithm is the summation of all bandwidth *increase* requests for the 45-minute *Star Wars* segment. While both sliding window smoothing and critical bandwidth allocation lead to a small cost for large buffers, sliding window allocation spreads its increases over a much greater number of requests.

Because the critical bandwidth algorithm segments the movie clip only when necessary, increasing the available buffer space can delay a given break point or may eliminate the need for it. This causes the number of break points to monotonically decrease as more buffer space is allowed. Figure 9 shows the segment break points for the *Star Wars* clip for different buffer sizes. It is notable that a very small amount of buffering will eliminate the need for almost all of the increases in network bandwidth.

The break points represent the times when an increase in network bandwidth is needed and thus represent a required interaction with the network manager. When this interaction would actually take place will depend on the nature of the underlying service. If the bandwidth requested for the next segment is not available, it may be possible to allocate additional buffer space and adjust the plan to prefetch a greater amount of the data, decreasing the bandwidth request, but advancing it in time. This possibility is only open if the underlying service allows bandwidth allocations to be negotiated in advance.

Decreases in Bandwidth

A decrease in required bandwidth may require an interaction with the network manager, but will not be disallowed due to contention for resources. Some approaches to access control assume that bandwidths will be constant, but smoothing shows that within lengthy segments it may be monotonically decreasing. Thus, there may be benefits in developing admission con-



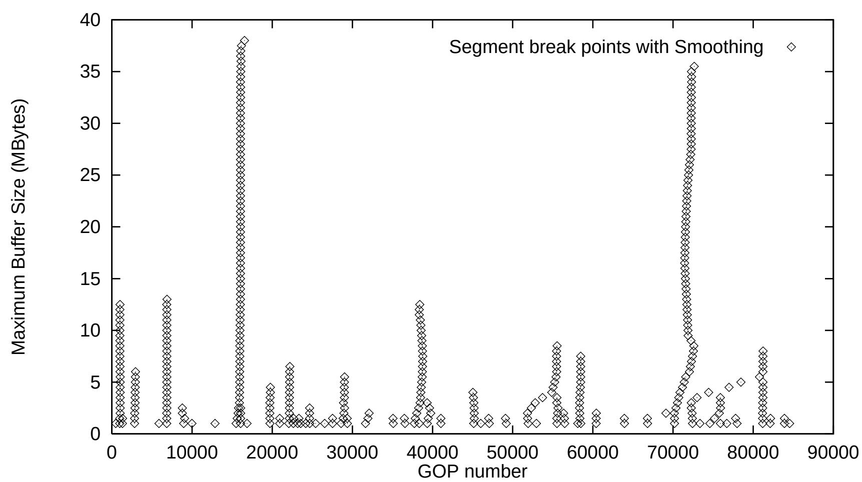


FIGURE 9. The graphs above show the segmentation points of the critical bandwidth allocation approach for various buffer sizes. From this graph, we can see for a given buffer size, the number of bandwidth increases (and consequently the number of interactions with the network admission control) necessary for the display of the clip. In the top graph, the break points for the critical bandwidth algorithm without prefetching is shown. In the bottom graph, the critical bandwidth algorithm with prefetch is shown. For moderately sized buffers, the number of segments reduces rapidly.

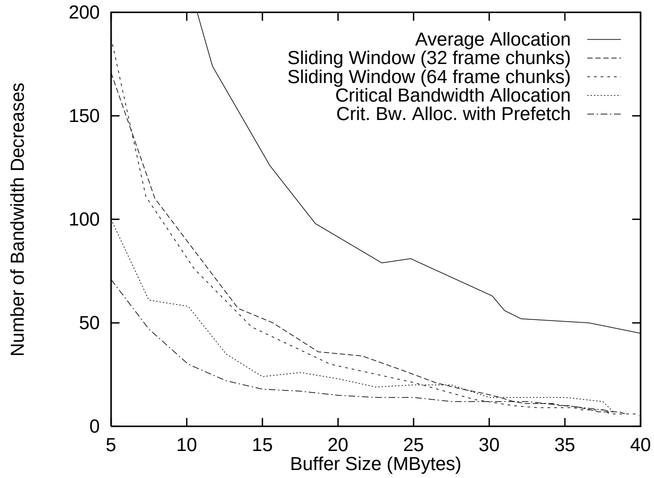


FIGURE 10. The total number of bandwidth decreases represent the number (not magnitude) of bandwidth decrease requests for the 45-minute *Star Wars* segment. The critical bandwidth algorithm makes more decrease requests than increase requests. The sliding window smoothing algorithm on the other hand, make many fewer requests to decrease bandwidth than to increase, since increases are frequently spread across several requests

trol policies that allow reserved bandwidth to be released without terminating a channel. The critical bandwidth algorithm does a good job of identifying blocks of bandwidth that can be released in a low number of requests (see Figure 10).

Sensitivity of the Critical Bandwidth Algorithm to Network Jitter

Most of the time, the client's buffer will be partially filled, holding several seconds of video data. At these times, the display of data will be immune from jitter in the delivery system. When the buffer is nearly full or nearly empty, however, early or late arriving packets could cause the buffer to overflow or underflow. Because the critical bandwidth algorithm results in buffers that are nearly empty at every critical point in the movie, and buffer that are nearly full preceding each segmentation point, the effects of jitter on the delivery of packets must be accounted for.

Discovering the exact margins of safety required when using a particular delivery system would require details of that system. It is possible, however, to find rough bounds through simple analysis. We assume that each packet delivered will have a fixed delay for delivery from server to client, plus a stochastic jitter. We assume that the jitter is normally distributed, and that the delay of any packet past its display time will result in the loss of that frame. Note that this is a conservative assumption, since error recovery techniques such as those proposed by Ghanbari[4] can often salvage the frame.

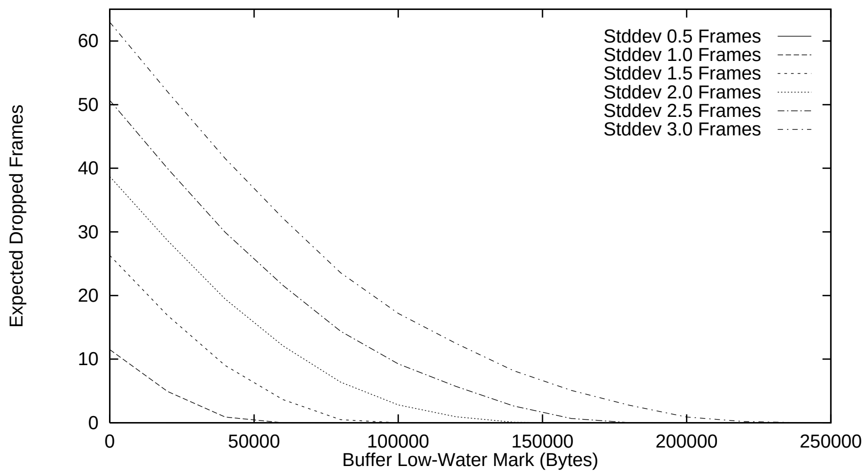


FIGURE 11. This graph shows the effects of adding additional buffer space to avoid buffer underflow due to network jitter. Underflow is possible when the buffer data approaches zero. By delaying the display of the video, we can effectively create a minimum reserve in the buffer. The graph shows the expected number of dropped frames as a function of this low-water mark for normally distributed jitter with standard deviation ranging from 0.5 to 3.0 frame intervals (15 to 100 milliseconds). Small amounts of buffering can eliminate the problem of dropped frames even with substantial jitter.

Given an allocation plan from the critical bandwidth algorithm, we can find the probability of individual frames being dropped. Using these probabilities of dropping frames, end users can choose to reduce the probability of lost frames by setting a low-water mark to protect the buffer from underflow. This low-water mark will appear as a delay at the beginning of a clip, before any frames can be shown.

Frame losses due to underflow will tend to appear in clusters at critical points in a clip. The expected number of frames lost will depend upon the low-water mark set. Figure 11 shows graphs of the expected number of dropped frames for the critical points in the *Star Wars* clip as a function of the low-water mark for jitter with standard deviation ranging up to 100 msec. Changes in buffer size will result in more or fewer critical points but will not change the behavior as the buffer approaches empty. The danger of lost frames can be eliminated with a modest commitment of resources.

Creating a high-water mark somewhat below the full buffer size will, similarly, eliminate the danger of buffer overflow. A very conservative estimate of the extra buffer requirement can be arrived at by estimating the expected number of early arriving packets and using Markov's inequality to find a buffer requirement that will bound the probability of overflow. Using this approach, we found that for jitter with a standard deviation of 100 msec, setting the high-water mark to 120 Kbytes below the buffer size was sufficient to reduce the probability of lost frames to 5%. This is an extremely conservative estimate; establishing a tighter bound, however, is probably only worthwhile when the actual jitter characteristics of the delivery system are known.

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. The critical bandwidth algorithm makes better use of the available buffer bandwidth and minimizes the need to increase the bandwidth required of the network.

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. Streams, such as movies, for stored video applications, on the other hand, 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 algorithm and explains its better performance.

The amount of buffering needed in any system will depend upon the size of the data stream and the effectiveness of encoding. The MPEG clip of *Star Wars* used for our analysis here does not make the most effective use of compression. Nonetheless, it is 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.

The use of the critical bandwidth allocation has implications for the design of underlying network services that differ from those drawn from live video applications. Many current proposals for network services frame the problem of video delivery as one of providing a single channel, commencing immediately, with a fixed, hard- or statistically-guaranteed bandwidth, the expected case for applications such as teleconferencing[3,12,9]. The critical bandwidth algorithm breaks a clip into a sequence of segments with differing bandwidth requirements. These segments must be played back so that each segues smoothly into its successor. Because the location of the segments are known before the video is started, it may be useful to negotiate in advance the bandwidth guarantees from the underlying network. This differs from live video where a bandwidth guarantee generally remains constant for an entire exchange. By providing reservations in advance, the network services would allow critical bandwidth algorithm to plan ahead and respond appropriately to denials of bandwidth.

Video on demand services are expected to eventually support full VCR functionality. The critical bandwidth approach bases its plan on a known rate of consumption. Therefore, VCR functions, besides play, violate the intended plan that was created. These variations must be dealt with in one of two ways: either the buffer will have to accommodate the change or a change in the network plan will be required. Short pauses and seeks can be dealt with by establishing larger reserves in the buffer to prevent overflow or underflow. Beyond some point, however, it will always be necessary to renegotiate the network bandwidth because of the altered consumption rate.

By prefetching effectively, critical bandwidth allocation allows long durations of non-increasing bandwidth requirements. 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.

Finally, the jitter of packets across networks that posed problems for live video does not cause as much a problem with critical bandwidth allocation. Even with fairly large jitter, small amounts of additional buffer space can be used to prevent the jitter from lowering the quality of service seen by the end application.

Acknowledgements

This research was supported in part by the National Science Foundation under grant IRI-9212087

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 D. Ferrari, D. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 8, No 3, pp 368-379, March 1990.

4 M. Ghanbari, V. Seferidis, "Cell-Loss Concealment in ATM Video Codecs," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 3, No 3, pp 238-247, June 1993.

5 S. Lam, S. Chow, D. Yau, "An Algorithm for Lossless Smoothing of MPEG Video," In *ACM SIGCOMM Conference Proceedings*, 1994.

6 D.J. LeGall, "A Video Compression Standard for Multimedia Applications," *Communications of the ACM*, Vol. 34, No 4, pp 46-58, April 1991.

7 P. Pancha, M. El Zarki, "Prioritized Transmission of Variable Bit Rate MPEG Video," In *IEEE Globecom 1992 Proceedings*, 1992, pp. 1135-1139.

8 P. Pancha, M. El Zarki, "Bandwidth Allocation Schemes for Variable Bit Rate MPEG Sources in ATM Networks," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 3, No 3, pp 190-198, June 1993.

9 A. Parekh, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks," Ph.D. Thesis, The Massachusetts Institute of Technology, 1992.

10 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.

11 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.

12 G. Woodruff, R. Kositpaiboon, "Multimedia Traffic Management Principles for Guaranteed ATM Network Performance," *IEEE Journal on Selected Areas in Communications*, Vol. 8, No 3, pp 437-446, March 1990.

13 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.