Smoothing and buffering for delivery of prerecorded compressed video

Wu-chi Feng and Stuart Sechrest

Software Systems Research Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, Michigan 48105

ABSTRACT

The transportation of compressed video data generally requires the network to adapt to large fluctuations in bandwidth requirements if the quality of video is to remain constant. Techniques that use averaging for smoothing video data, such as those found in video conferences, allows for some smoothing at the expense of delay. With video-on-demand systems on the horizon, smoothing techniques for prerecorded video data are necessary for the efficient use of network resources. Simply extending algorithms that smooth via averaging for use in video playback cannot smooth the burstiness in bandwidth requirements without using large amounts of buffering. In this paper, we introduce the notion of critical bandwidth allocation which allows for the most effective use of buffering while allowing for long durations between bandwidth increase requests. A comparison between critical bandwidth allocation algorithms and other smoothing algorithms is presented.

Keywords: video delivery, bandwidth allocation, MPEG, video-on-demand, buffering, bandwidth management strategies, video servers

1 INTRODUCTION

The efficient transportation of compressed video remains a challenge for the research community. Compression introduces burstiness into video streams which makes the optimal scheduling of resources more difficult. For low-latency live video, one must generally choose between compromising quality of service or network utilization for its delivery. Prerecorded video, however, allows a third option, the smoothing of bandwidth requirements with buffering on the receiving end. Sufficient smoothing can increase channel utilization and simplify delivery for networks or storage devices. Many services, including video-on-demand services, can take advantage of smoothing provided that the receivers have modest buffering capabilities.

Two approaches have been taken for the delivery of bursty data in an attempt to control overallocation of resources. Statistical multiplexing[8,1] averages bandwidths across channels, assuming that bursts are unlikely to occur simultaneously in independent streams. This approach cannot provide firm quality-of-service guarantees because of its statistical nature. Alternatively, bandwidths can be made more uniform within a stream by grouping frames and sending them at the average bandwidth for the group with some amount of buffering at the receiver side[4] or using a leaky bucket approach.[7] For some applications, the best approach may be to eliminate burstiness by counteractively adjusting picture quality,[5,6] but the loss of quality may be too large for others.

For streams compressed with algorithms such as MPEG,[3,2] 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. In this paper, we will discuss smoothing algorithms appropriate for prerecorded video data that can be used to control burstiness at this larger scale. We introduce the notion of critical bandwidth allocation, which allows for the most effective use of buffer space while allowing for long durations between bandwidth increase requests. Our analysis will include consideration of tradeoffs between smoothing, buffering, and delay. In the following section, a description of the various bandwidth smoothing algorithms is presented. Section 3 compares and contrasts the various algorithms. Finally, some conclusions and future applications are presented.

2 BANDWIDTH ALLOCATION ALGORITHMS

2.1 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. This is illustrated in Figure 1. By using this method, clients can guarantee that the bandwidth needed will be constant throughout the chunk. However, 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 chunk unless it has buffered at least one chunk ahead. Buffering one chunk ahead implies that the delay between channel setup and playback must be at least one chunk in length. As a result, 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.

2.2 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 1 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, while 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 while drops are sharper.

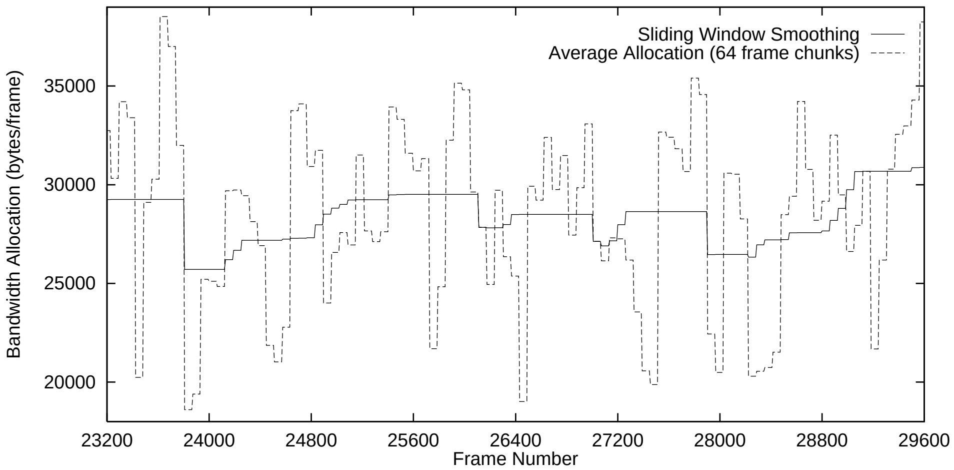

Figure 1: Each line in the graph represents a sample of bandwidth requests from a 6400 frame sample from the movie *Starwars*. The video has been MPEG-encoded using I-frames only. The burstiness is therefore entirely due to differences in scene content. The dotted line shows the average allocation algorithm for 64 frame chunks. The solid line represents the sliding window smoothing applied to 64 frame chunks with a window size of 10 chunks.

2.3 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 define the *critical bandwidth* as the minimum constant bandwidth necessary to play a segment of video through without buffer starvation. We define the *critical point* in the segment as the frame at which any smaller bandwidth allocation will result in buffer starvation. This is the frame for which the average frame size for all prior frames in the segment is maximized.

With these definitions, two particular properties of critical bandwidth allocation for a video segment can be extracted. Suppose the segment has critical bandwidth B_{crit} and critical point p. Then, the critical bandwidth for the subsegment from the critical point p to the end of the sequence must be *less than or equal* to B_{crit} and will be determined by a new critical point within the subsegment. Thus, the critical bandwidth of the remaining portion of a segment decreases monotonically as one moves through the segment. In addition, playback of the video can commence immediately upon channel establishment. If the user chooses to have minimal latency, an initial bandwidth spike (generally the size of the first frame) will be required. In the next section, we will discuss how this initial bandwidth spike can be lowered to allow for an initially lower bandwidth requirement in exchange for a little delay between channel establishment and playback.

Given sufficient buffer space, an entire video can be treated as a single segment. After an initially high bandwidth allocation, the bandwidth used would then decrease monotonically throughout the video. Any bursts of data would simply be prefetched sufficiently far in advance. In a system where all videos are treated as a single segment, admission control becomes trivial; if there is enough bandwidth to satisfy the initial request, then admit the new channel. Because each channel will then monotonically return bandwidth back to the network, we are guaranteed that no one will need additional resources during the playback of their videos. In practice, we have found that for a 13 minute clip of I, P, B encoded frames from *Honey, I Blew Up the Kid*, a buffer size of 3.4 MBs would suffice for critical bandwidth allocation. For a 45 minute clip of I encoded *Starwars* video, a buffer

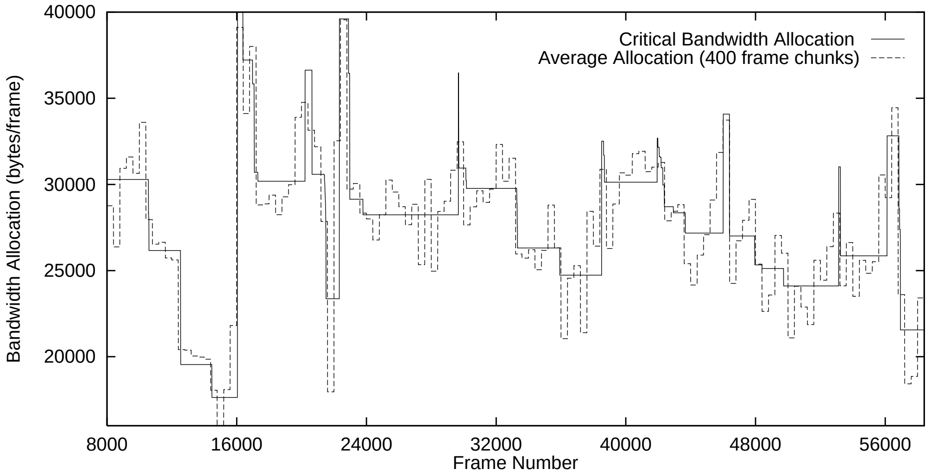

Figure 2: Each graph represents the bandwidth requests for a sample of 50,000 frames from the movie *Starwars* The dotted line shows the average allocation of 400 frame chunks. The solid line represents the critical bandwidth algorithm with a maximum buffer of 3 MBytes. The segment has been divided into 9 subsegments varying in size from 2120 to 8832 frames (71 and 294 seconds, respectively).

size of 38 MBs was necessary to treat the video clip as a single segment.

If the buffer size is to be constrained, it will probably be necessary to increase the bandwidth at several points in the video, segmenting it into a set of runs of monotonically decreasing bandwidth allocations. The plan for data delivery when using the critical bandwidth allocation algorithm thus increases the bandwidth allocation at exactly those points that will prevent the buffer from overflowing and decreases the bandwidth at exactly those points where the buffer is empty. Such a plan is illustrated in Figure 2. As can be seen, the critical bandwidth allocation algorithm results in fairly long times between bandwidth changes even with a relatively small buffer size.

2.4 Critical Bandwidth Allocation with Prefetching

A problem with the critical bandwidth allocation algorithm described above 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 will generally be needed at the beginning of each new segment.

It is generally possible to modify the 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. The modified plan must still satisfy the property that within a segment the total bandwidth must decrease monotonically and buffers must not overflow. This adjustment is illustrated in Figure 3.

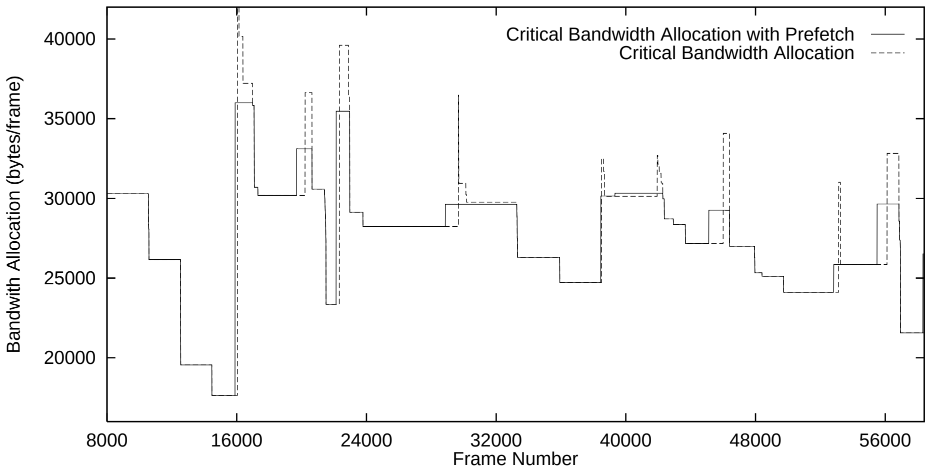

Figure 3: Each graph represents the bandwidth requests from a sample of 50,000 frames from the movie *Starwars* The dotted line is the critical bandwidth allocation algorithms without prefetching and a maximum buffer size of 3 Mbytes. The solid line represents the critical bandwidth algorithm with prefetching and a maximum buffer size of 3 Mbytes.

3 EVALUATION OF ALGORITHMS

Network admission control and load estimation, used to provide service guarantees, are 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 test the effectiveness of the various smoothing algorithms, each algorithm was run on an MPEG-compressed 45-minute video segment from the movie *Starwars*. The statistics for this I-frame encoded stream was obtained from Bellcore. We also encoded a 13-minute segment from *Honey, I Blew Up the Kid*, however, because this segment required less than 4 MBs of disk space for the critical bandwidth allocation, this clip was unsuitable for comparison purposes. We expect to study the effects of coding on critical bandwidth in the future.

Figures 4 and 5 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.

3.1 Cost of Increasing Bandwidth

If we never overallocate network resources, each adjustment in bandwidth will require an exchange with 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 or establishing an alternative route. In Figure 6, the total number of bandwidth increases is shown for each algorithm as a function of buffer size. Sliding window algorithms suffer from having to 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.

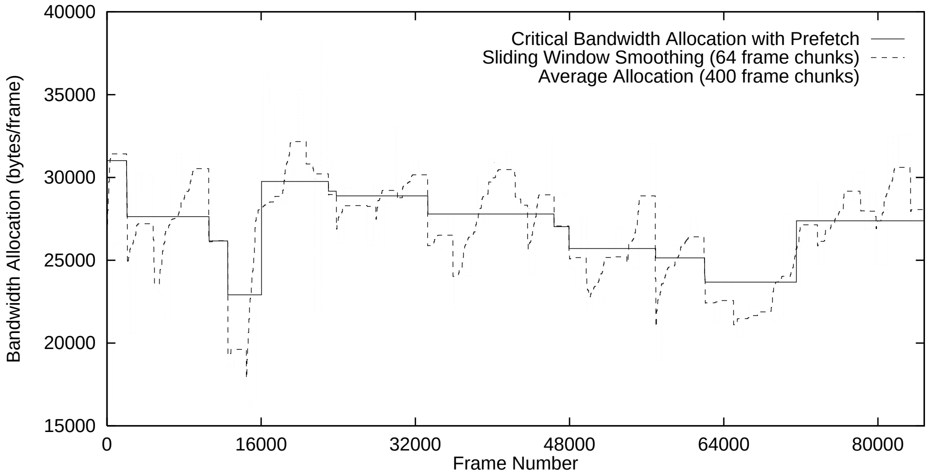

Figure 4: This graph shows results of smoothing using the three algorithms with 15 MBytes of buffering available. The data above was obtained by using a 45-minute segment from the movie *Starwars* (85000 frames). For the average allocation algorithm, the chunk size was increased to take full advantage 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. The critical bandwidth algorithm increased the bandwidth requirements when a 15 MByte buffer would be filled.

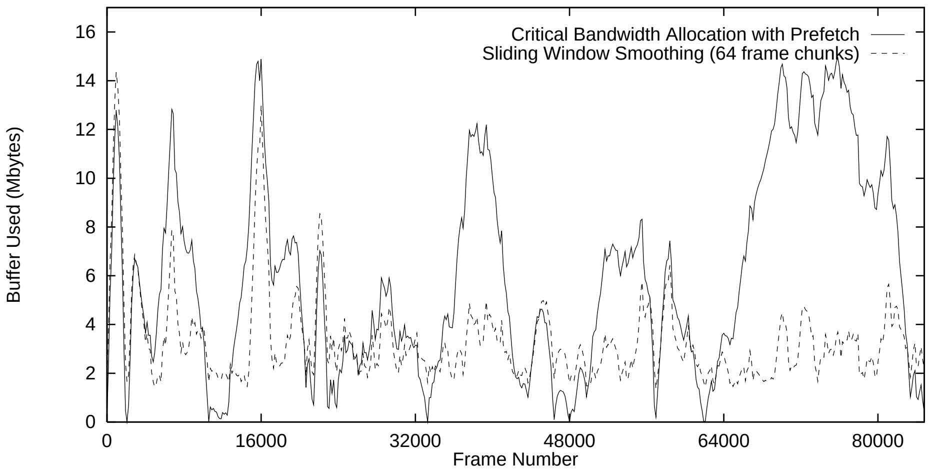

Figure 5: This graph shows the buffer utilization for the sliding window smoothing and the critical bandwidth algorithms. The size of the sliding window was set uniformly in such as way as to make use of the full buffer in the worst case. However, the algorithm is only able to use the buffer in the first 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.

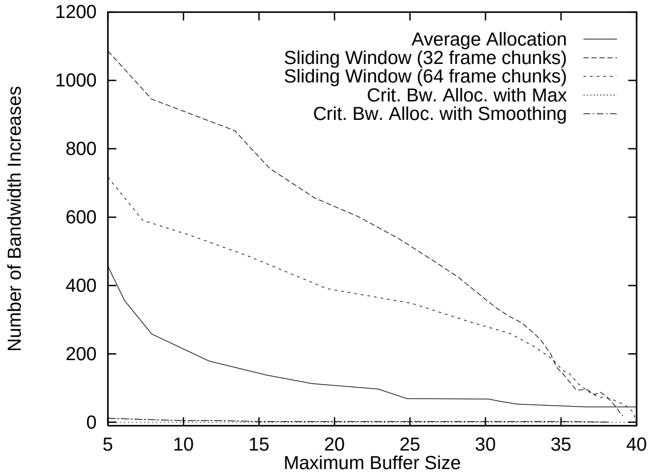

Figure 6: The total number of bandwidth increases represent the number (not magnitude) of bandwidth increase requests for the 45-minute *Starwars* segment. Each algorithm was applied to the movie segment with different maximum buffer capacities. The average allocation algorithm varied the chunk size 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 just used the maximum 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.

In Figure 7, 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 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.

3.2 Cost of Decreasing Bandwidth

A decrease in required bandwidth may require an interaction with the network manager but will not be disallowed due to contention for resources. Smoothing shows that within lengthy segments, bandwidths may be monotonically decreasing rather than constant. Thus, there may be benefits in developing admission control policies that allow reserved bandwidth to be released without terminating a channel. The critical bandwidth algorithm identifies blocks of bandwidths that can be released (see Figure 8).

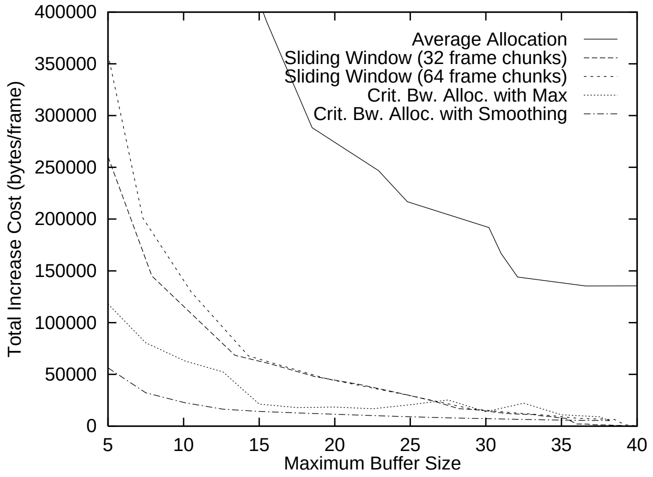

Figure 7: The total increase cost of each algorithm is the summation of all bandwidth *increase* requests for the 45-minute *Starwars* segment.

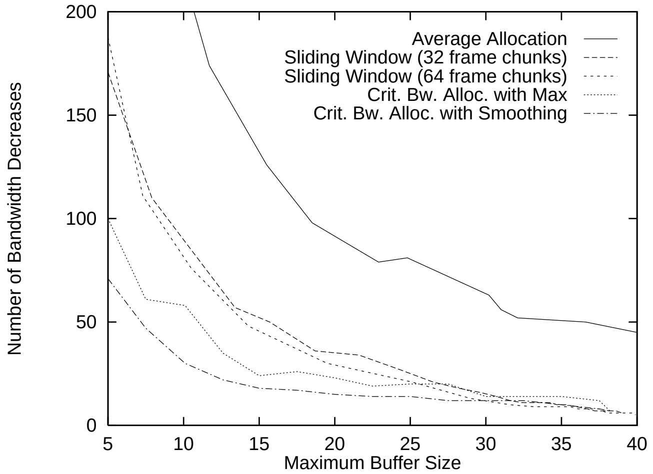

Figure 8: The total number of bandwidth decreases represent the number (not magnitude) of bandwidth decrease requests for the 45-minute *Starwars* segment. The critical bandwidth algorithm makes more decrease requests than increase requests. The sliding window smoothing algorithm on the other hand, make many fewer requests to decrease bandwidth than to increase, since increases are frequently spread across several requests.

3.3 Summary of Results

In summary, we found that any form of averaging to smooth bandwidth requests has to trade of buffer space for number of bandwidth increases. Unlike the averaging algorithms, the critical bandwidth algorithms were effective in using the available buffer space while keeping the number of bandwidth increase requests to a minimum. In order to minimize the fluctuations seen by the network, algorithms must use the global data and not just attempt to smooth via averaging.

4 CONCLUSION

In this paper, we have introduced the notion of critical bandwidth allocation for the delivery of prerecorded compressed video data. Because critical bandwidth allocation can smooth across large regions of video without a very large buffer, it is able to minimize the interaction with the network and effectively utilize receiver's buffers. Although, we emphasized just the bandwidth allocation smoothing in this paper, the critical bandwidth algorithm can be used to help networks in admission control as well as help clients adapt to network load.

Creating an appropriate admission control policy has proven to be a difficult problem due to the unpredictability of bursty video data. This stems from the fact that at any point the network cannot tell whether a channel will increase or decrease its allocation. Many investigators have ignored the possibility of buffering, sometimes because they wish to address low-latency applications. Relatively small amounts of buffering, however, can be used to greatly smooth the bandwidth requirements for MPEG-encoded video. In particular, using critical bandwidth allocation, bandwidths can be made non-increasing for lengthy video segments with little buffering cost.

5 REFERENCES

[1] D.M. Cohen and D.P. Heyman. A simulation study of video teleconferencing traffic in ATM networks. In *IEEE INFOCOM 1993*, pages 894–901, 1993.

[2] D. Le Gall. MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, April 1991.

[3] ISO. Coding of moving pictures and associated audio. Technical Report ISO11172: ISO/MPEG 90/176, ISO, Dec. 1990.

[4] Simon S. Lam, Simon Chow, and David Yau. An algorithm for lossless smoothing of MPEG video. In *ACM SIGCOMM 1994*, 1994.

[5] P. Pancha and M. El Zarki. Prioritized transmission of variable bit rate MPEG video. In *IEEE GLOBECOM 1992*, pages 1135–9, 1992.

[6] P. Pancha and M. El Zarki. Bandwidth-allocation schemes for variable bit rate MPEG sources in ATM networks. *IEEE Transactsion on Circuits and Systems for Video Technology*, 3(3):190–98, June 1993.

[7] A.R. Reibman and A. Berger. On VBR video teleconferencing over ATM networks. In *IEEE GLOBECOM 1992*, pages 314–319, 1992.

[8] D. Reininger, D. Raychaudhuri, B. Melamed, B. Sengupta, and J. Hill. Statistical multiplexing of VBR MPEG compressed video on ATM networks. In *IEEE INFOCOM 1993*, pages 919–26, 1993.