EVALUATION OF ADAPTIVE FILTERING OF MPEG SYSTEM STREAMS IN IP NETWORKS

Michael Hemy[1], *Peter Steenkiste*[1], *and Thomas Gross*[1,2]

[1]School of Computer Science
Carnegie Mellon
Pittsburgh, PA 15213

[2]Departement Informatik
ETH Zürich
CH 8092 Zürich

ABSTRACT

Congestion and large differences in available link bandwidth create challenges for the design of applications that want to deliver high quality video over the Internet. We investigate the placement of computation agents within the network to provide filtering to adapt MPEG system streams to the available bandwidth without transcoding. This paper presents an evaluation of our implementation in three different operating environments: a networking testbed in a laboratory environment, a home-user scenario (DSL line with 640 Kbit/sec), and a wide area network covering the Atlantic (server in Europe, client in the US). Our architecture leverages previous work and introduces efficient MPEG system filtering to achieve high-quality video over best-effort networks.

1. INTRODUCTION

The infrastructure of the Internet provides an attractive platform for the distribution of all kinds of digital audio and video content. Distribution of continuous media poses a number of challenges. Since the movie is played as it is received, the transfer over the network must proceed at a specific rate to prevent buffer overflow or underflow at the player. If there is competing traffic in the network there is the risk of congestion, and consequently, packets may be dropped or delayed. Finally, if the available network bandwidth is insufficient, and we want to continue to smoothly play the movie, the application must reduce its bandwidth demands intelligently.

The key problem is to ensure that the player receives a continuous feed in the presence of variable congestion while maximizing the transfer of comprehensible information given the available (varying) bandwidth. Bandwidth reduction can be achieved by filtering the video stream before transmitting it on the congested link. Filtering is the process of selectively removing part of the data, while maintaining some of the stream's characteristics. Our solution is based on a filter that can transparently remove frames from an MPEG system stream while maintaining audio & video synchronization. Since the output of the filter is also an MPEG system stream, there is no additional complexity required at the client-end to synchronize multiple separate streams.

If lost data packets are to be retransmitted using a Cyclic UDP scheme [8] such as in the Continuous Media Player [6], or recovered by a Forward Error Correction (FEC) scheme [7], then redundant data needs to be streamed over at least a certain segment of the network. The retransmitted data competes on the already scarce bandwidth resource and introduces a latency on the client at playback. Hence, we choose to not retransmit lost packets.

A responsive adaptation process is key to successful media presentation. Without it, random packet losses in an MPEG stream will be propagated due to the inter-frame dependencies. The adaptation process can be controlled by various types of feedback. Some algorithms (The Vosaic player [3], Li et al.'s player [5], the VoD testbed [2], and OGI's player [1]), monitor packet arrival as well as information obtained after decoding the stream such as frame rate or delayed frames. In contrast to that approach we present an adaptation scheme based only on monitoring network packets. The latter has the advantage of reducing the client's complexity and completely decoupling the adaptation from the format of the continuous media.

2. MPEG SYSTEM SENSITIVE FILTERING

In [4], we present the architecture and implementation of an adaptive MPEG filter that is efficient enough to be executed on a PC, as is shown in Figure 1. It shows the cost of filtering a number of movies at three levels of filtering and compares those costs with the overhead of just demultiplexing the system streams. The costs are expressed as a percentage of CPU load of a Pentium 400 MHz machine while data is being streamed at video rates, and the costs include only the processing overhead but not the network communication. The results show that this particular type of filtering is very efficient and is faster than any transcoding, since the latter would require demultiplexing the system stream as a first step.

2.1. Architecture

The three components of our architecture are: a client, a server, and a filter. Figure 2 depicts a unicast setup where different streams are transmitted to individual clients. Both the filter and the server respond to requests from multiple clients. By placing filters on nodes that connect to network bottlenecks, we can adapt the stream provided by the video-server to match the client's available bandwidth.

2.2. Server-client interaction

Requests for changes in the quality of the transmitted movie must originate at the client side. Such requests could either be entered interactively by the person watching the video, or be triggered by a process analyzing the received stream at the client. Unfortunately, perception-based models are computationally expensive. Since the client is already busy with decoding and rendering a continuous media stream, we need a non-intrusive method to determine when, and how much, that stream should be filtered.

2.2.1. Adaptation algorithm

It is the client's responsibility to request modifications in the continuous media stream's bandwidth. To decide whether the filter should

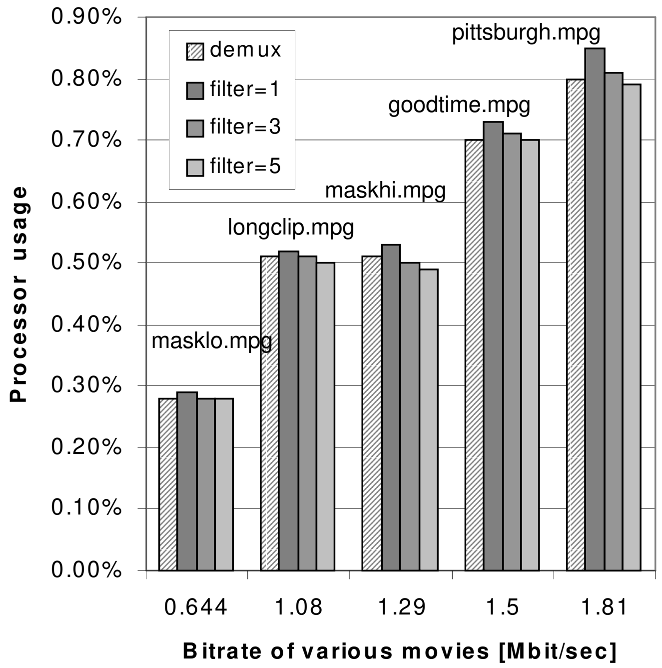

Figure 1: Pentium 400 MHz load for filtering various movies.

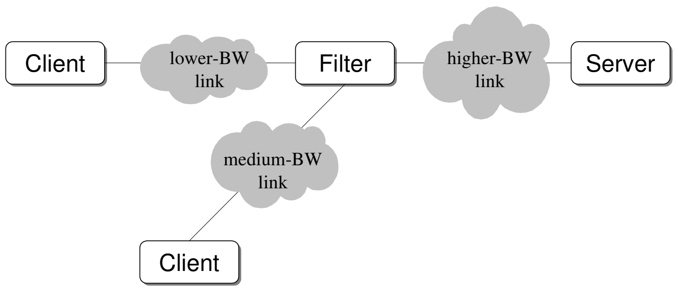

Figure 2: Active-service architecture.

increase or decrease the bandwidth, the receiver continuously measures the current packet loss rate using a sliding window of variable length S packets.

If the packet loss rate reaches a threshold α during a window of length S such that:

$$1 \leq S < S_\alpha$$

the client immediately requests the filter to reduce the bandwidth. The value of α should be such that a packet loss rate of α (or less) still results in acceptable video quality. A second threshold β, such that $\beta < \alpha$ is used to determine when the bandwidth can be increased. if the packet drop rate is less than β, during a period S such that:

$$S \geq 2^n * S_\beta$$

the receiver will ask the filter to increase the bandwidth. Typically $n = 0$, however if an increase in bandwidth is followed by a decrease in bandwidth over a period S such that $S < S_\alpha$, then $n = n{+}1$. Similarly if the opposite condition exists then $n = n{-}1$. For both cases we require: $1 \leq n \leq 5$. The reason for using two thresholds and a multiplier for S_β is to allow the BW recovery to be less aggressive.

After every request to increase or decrease the frame drop rate, the client temporarily suspends measuring the packet loss rate until it is notified by the filter that the drop rate change took place. This period depends on the number of packets 'en-route' and the frame being processed by the filter when it receives the request to change the drop rate.

3. EVALUATION

We present a detailed evaluation of our video streaming application. We first describe our experimental setup. We then compare the video quality with and without filter-based adaptation. Finally we characterize the responsiveness of the filter with dynamic competing loads and the bandwidth sharing of our application with itself and TCP.

3.1. Experimental setup

We present results for three different scenarios. For all the experiments the client was run on a 200 MHz Pentium Pro machine.

- A dedicated testbed using 230 MHz Pentium Pro systems running FreeBSD as routers. The routers are connected using dedicated 10 MHz Ethernet links that form the bottleneck. The filter is a 400 MHz Pentium II machine and the video server is a DEC 500/266. We set up a reservation of X Mbit/sec (e.g., 2 Mbit/sec) for continuous media streaming and one or more competing sources; the remaining link bandwidth is assigned to a UDP stream that floods the network but is prevented from using the X Mbit/sec pipe by the scheduler.
- A DSL connection between CMU and a suburban home. The path between the server at CMU and client at the home has three segments: the CMU local area network where the server and filter are placed, a T1 line to the telephone office, and a synchronous DSL line to the residence. The last segment is the bottleneck link, its maximum UDP throughput was measured to be 649 Mbit/sec. About 95% of this maximum bandwidth was available for the connection between filter and client.
- A transatlantic connection between at client at CMU and a server at ETH in Zurich, Switzerland. We ran the server and filter on the same Linux box (a Pentium II 400 Mhz) for convenience. Since we use the Internet, we had no control over competing sources. Interestingly, the bandwidth for the transatlantic connection proved to be larger than needed for streaming video and all tries resulted in the whole movie being streamed without losses.

The adaptation parameters were set to

$$S_\alpha = 500; S_\beta = 500; \alpha = 25; \beta = 5$$

In Section 3.5 we report parameter sensitivity by experimenting with different values.

The video stream is received and displayed by a JMF-based client. While it is possible for us to subjectively judge and compare the quality of the received video stream JMF provides no objective quality metrics. For this reason, we developed a second version of the JMF data source to store additional tracing information as well as the MPEG system stream in files for off-line analysis and replay. We also developed a tool (mpegOscope) that analyzes MPEG streams and collects a variety of statistics, e.g., frequency of several types of errors, lost frames due to the propagation of errors, etc. The mpegOscope tool is quite accurate; the number of good frames it reports is typically within 1% of the frames displayed. We verify this number by comparing statistics collected by the Windows Media Player when replaying the received stream.

When comparing different scenarios, we primarily use the following metrics: The first metric is the average rate of correctly received video frames. Two other metrics are the average bandwidth of the transmitted and received MPEG-1 system stream; these metrics capture the network resource utilization. The final metric captures the smoothness of the video stream. This is a fairly subjective metric, but the idea is that we want to distinguish between a video stream where every other frame is missing and a stream where the same number of frames has been dropped, but in a bursty fashion. The metric we use is the spectrum of frame rates over the length of the media stream; a video stream played mostly at one frame rate will be in general more smooth and pleasant than a video stream that has a broad spectrum of rates.

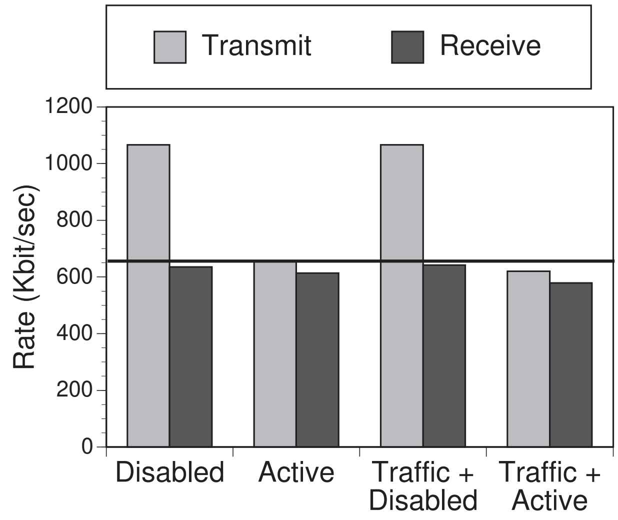

Figure 3: Effect of adaptation on network load.

3.2. Evaluation of video quality

In this section we compare the quality of the received MPEG system stream with and without adaptation under some typical operating conditions.

Figure 3 shows the transmitted and received bandwidth for the video stream (in Kbit/sec) for the DSL scenario using a movie with a bandwidth requirement of 1.07 Mbit/sec. The first two cases compare the throughputs with the filter active and disabled, when the MPEG system is the only traffic in the network. We see that in both cases, roughly the same number of bits are received, but the transmit rate with adaptation is much lower than without adaptation. With adaptation performed by the filter, the output stays close to the maximum bandwidth of the critical link. Without adaptation, the output rate is unconstrained and a large portion of the data is dropped along the way to the client.

Without filtering, the receive rate is equal to the link capacity. When filtering, the adaptation reduces the bandwidth until it reaches a steady state. We note that the average transmit rate is slightly lower than link capacity, since frame filtering provides only for a certain distinct number of filtering levels. While the receive rates are almost similar with and without filtering, Table 1 shows that there is a big difference in the frame rates attained at the client, so the actual movie quality is significantly lower without adaptation.

To show that the frame rate achieved with adaptation (filter level 3) is optimal we perform two additional experiments: one with limiting the maximum filter level to 2 and one with statically configuring the filter level to 3 without any adaptation. The results are also presented in Table 1.

DSL Experiment	Average fps
Filtering disabled	0.5
Filtering active	6
Filtering limited to level 2	1.2
Filtering fixed at level 3	7.5
Traffic + Filtering disabled	0.5
Traffic + Filtering active	5

Table 1: Filter performance evaluation

A static level 3 filtering is optimal in that it provides the highest fps for the current network conditions. We come close to this result when allowing adaptation to occur, only losing a few frames initially (until bandwidth reduction is achieved), and when probing for available bandwidth.

Figure 3 and Table 1 also show results of experiments we ran with competing traffic. For these experiments we used another machine sharing the DSL line to browse the web and also issue file transfers (using FTP). The results represent averages of various runs. With the filter, the video stream was able to adapt, using less bandwidth than without the traffic. When the filter was disabled the server continued to stream the continuous media at full rate, effectively flooding the DSL link and preventing any other traffic. While filtering allowed the competing streams to proceed, most of the bandwidth was used by the video stream. The movie clip we used for these experiments was too short to notice long-term adaptation in presence of competing traffic. In later experiments (Sections 3.4 and 3.5) we evaluate bandwidth sharing.

Figure 4 shows the frame rate distributions for cases one and two and compares them with the distribution for local replay. For each frame rate we show the percentage of all frames that are delivered at this rate to the client. We see that with adaptation, the frame rates are centered around 5 fps and 10 fps, whereas without adaptation, the peak is around 0 fps (i.e., no playable frames arrive at the client).

The MPEG movie used in these experiments had the following structure: $IBBPBBPBBPBB$. Removing all the B frames (filtering level 2), still results in equally distributed frames $I_{BB}P_{BB}P_{BB}P_{BB}$, with a rate of 10 fps, i.e. 1/3 of the original. The filter level needed for correct adaptation in this experiment was 3, requiring to remove the last P frame to maintain inter-frame dependencies. The resulting stream $I_{BB}P_{BB}P_{BBPBB}$ has a frame rate of 7.5 fps, but the interval between the frames is no longer constant. The uneven spacing of frames causes peaks in the frame rates distribution shown in Figure 4.

Figure 5 shows the transmit and receive throughputs for a 1.5 Mbit/sec movie with a bottleneck link bandwidth of 1.1 Mbit/sec. The adaptation behavior is similar to that on the DSL line. Figure 6 shows the distribution of good, lost/dropped, and damaged frames with and without adaptation. Clearly, adaptation pays off. Finally, Figure 7 shows how many audio frames are correctly received with and without adaptation: again, adaptation pays off.

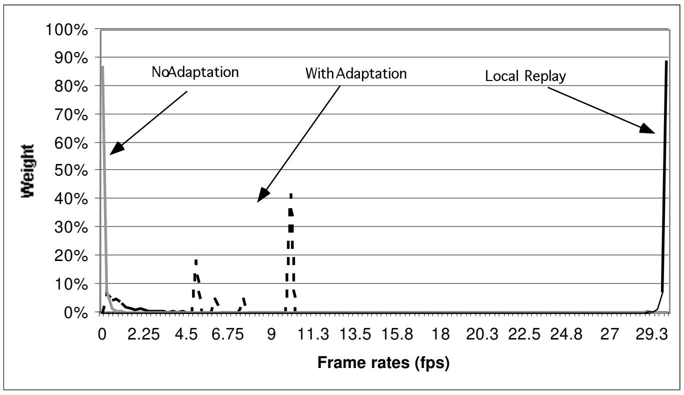

Figure 4: Distribution of video frame rates for different experiments.

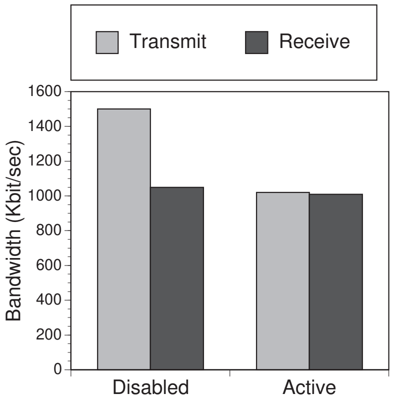

Figure 5: Frame rates in testbed experiment.

3.3. Responsiveness

In the next set of experiments we characterize the responsiveness of the filter-based adaptation. We use the testbed, but instead of using a steady competing stream of UDP packets, we use a bursty stream. The burst length is variable, and the interburst gap is 10 seconds. The net effect is that during bursts, the 1.5 Mbit/sec video stream has 1.1 Mbit/sec of bandwidth available, and between bursts it has 10 Mbit/sec available.

Figure 8 shows the percentage of good frames received for different burst lengths. For burst lengths of 2 seconds or less, the adaptation does not kick in, since there are not sufficient packet losses. For burst lengths of 2.5 seconds and up, there is adaptation. We see that the percentage of correctly received frames is still high.

3.4. Bandwidth sharing

In the next set of experiments we start to characterize how the adaptive video application shares bandwidth with other network users. We use our testbed and we set up a pipe with a bandwidth of 1.2 Mbit/sec.

In the first experiment, we have an FTP stream compete with the streaming of a 1.5 Mbit/sec movie for bandwidth on the 1.2 Mbit/sec pipe. FTP was started first and it was able to use the available bandwidth (1.2 Mbit/sec). When the video streaming starts, the FTP slows down considerably as a result of packet loss and timeouts, as one would expect. After a while, FTP grabs back some

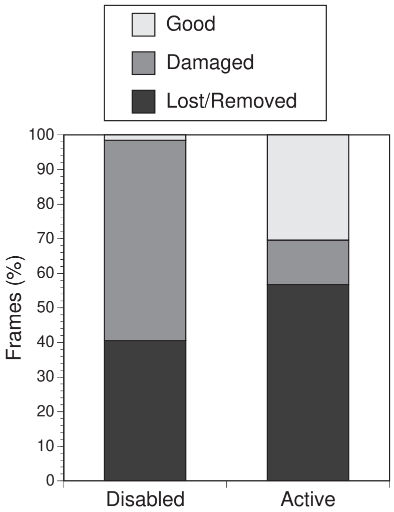

Figure 6: Effect of packet losses on video frames.

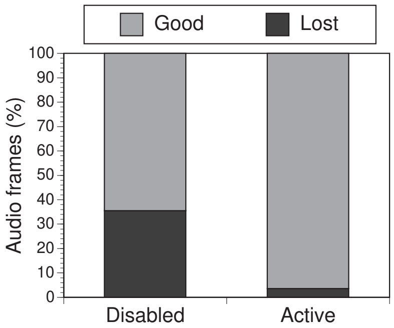

Figure 7: Effect of packet losses on audio frames.

of the bandwidth, and it peaks at about 600 Kbit/sec. This competing stream reduced the effective frame rate to 7.58 fps, compared to about 10 fps without competing TCP (FTP) stream.

In the second experiment we have two video streams competing for the 1.2 Mbit/sec of available bottleneck bandwidth. Under steady state conditions, the first stream gets about one third of the bandwidth and the second stream two thirds.

These results shows that the video streaming leaves bandwidth to other users. While it appears that the video streaming is more aggressive than TCP, more experiments are needed to characterize this property more precisely.

3.5. Parameter sensitivity

All experiments previously mentioned were taken with a specific set of adaptation parameters. Our current experience has shown that α and S_β are the more sensitive parameters. In most cases α, the number of packets lost to trigger a decrease in the stream request, is reached before a period S_α. On the other hand since β is typically small, it has less of an impact than S_β on bandwidth recapturing, since it enforces a delay equivalent to a window of at least that size before making a request to increase the stream. It is clear that reducing α will cause a faster adaptation. The question is 'will it result in more good frames?'.

To answer that question we devise an experiment with constant competing load. We chose a disturbing signal such that its char-

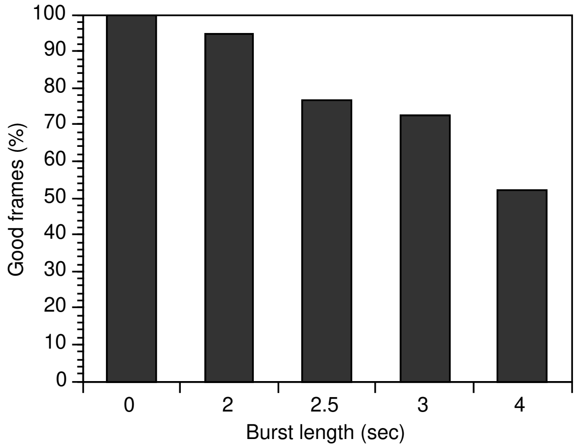

Figure 8: Inference of bursty traffic.

acteristics in terms of BW load would require an adaptation of 2 levels of filtering. In other words, removing all the B frames from this movie should result in a video stream that would fit in a congested network where the load was constant. Column 2 of Table 2 shows the number of good frames received for various values of α for one movie (goodtime.mpg). The first row in the table provides a reference.

α	good frames	max filter level
(no traffic)	5988	0
40	2580	2
25	2591	2
15	2605	3
10	2555	3
5	2524	3
2	2476	4

Table 2: Adaptation evaluation with constant competing traffic.

The difference in the number of good frames is not significant for this scenario. However it is interesting to observe that reducing α to 2 causes an average frame loss of an additional 2% of the full movie. To understand the reason behind this, we have added the third column in the table: the maximum level of filtering reached. When we reduce α, the system becomes more sensitive to network losses. A few losses can cause the filter to reduce the stream even further than needed on average for the current conditions. The bandwidth will eventually be recaptured but it will take more time to get to the correct level of filtering due to being at a higher filtering level and having a minimum period S_β requirement to reduce each level.

For this experiment the optimal α is: $15 \leq \alpha \leq 25$. However a general characterization of α can only be done with additional experiments with different number of hops between the client and server, as well as with additional movies.

4. CONCLUDING REMARKS

The MPEG format presents a number of challenges to a system that attempts to deliver MPEG streams over a best effort network. However, given the amount of MPEG-1 movies available, supporting their delivery in their original format (without conversion or transcoding) is an attractive solution.

Our system has given us an opportunity to evaluate an adaptive filter in scenarios with vastly different characteristics. The MPEG system sensitive filter provided an efficient solution that can be hosted on current mid-range PCs. Although we have an operational that performed as designed, a number of parameters deserve further study.

As new mid-range communication services (like DSL) become available to residential customers, an adaptive filter as presented here provides a cost-effective solution to allow homes to receive standard continuous media. The adaptation algorithm presented here is deliberately slower to recover bandwidth after congestion than it backs off in the presence of congestion, to ensure ”TCP-friendly” behavior. Experiments with sharing have confirmed this design decision: cooperation with other network users is an essential property to scale up to Internet-wide deployment.

5. REFERENCES

[1] S. Cen, C. Pu, R. Staehli, C. Cowan, and J. Walpole. A Distributed Real-Time MPEG Video Audio Player. In *Proceedings of NOSSDAV'95*, pages 18–21, Durham, New Hampshire, April 1995.

[2] S.-F. Chang, D. Eleftheriadis, D. Anastassiou, S. Jacobs, H. Kalva, and J. Zamora. Columbia's VOD and Multimedia Research Testbed With Heterogeneous Network Support. *Journal on Multimedia Tools and Applications*, 5(2):171–184, Sept 1997.

[3] Z. Chen, S.-M. Tan, R.H. Campbell, and Y. Li. Real Time Video and Audio in the World Wide Web. In *Proceedings of Fourth International World Wide Web Conference*, Boston, Massachusetts, Dec 1995.

[4] M. Hemy, U. Hengartner, P. Steenkiste, and T. Gross. Mpeg system streams in best-effort networks. In *Proc. Packet Video'99*, page (Session X), New York, April 1999. Columbia University. CD ROM and online proceedings at www.research.att.com/ mrc/pv99.

[5] X. Li, S. Paul, P. Pancha, and M. Ammar. Layered Video Multicast with Retransmission (LVMR): Evaluation of Error Recovery Schemes. In *Proceedings of NOSSDAV'97*, St. Louis, Missouri, May 1997.

[6] K. Mayer-Patel and L.A. Rowe. Design and Performance of the Berkeley Continuous Media Toolkit. In *SPIE Proceedings Vol. 3020*, pages 194–2006, San Jose, California, Feb 1997.

[7] J. Nonnenmacher, E. Biersack, and D. Towsley. Parity-Based Loss Recovery for Reliable Multicast Transmission. In *Proceedings of ACM SIGCOMM '97*, pages 298–300, Cannes, France, September 1997.

[8] B.C. Smith. *Implementation Techniques for Continuous Media Systems and Applications*. PhD thesis, University of California at Berkeley, 1994.