ADAPTIVE PLAYOUT FOR REAL-TIME MEDIA STREAMING

Mark Kalman, Eckehard Steinbach†, and Bernd Girod

Information Systems Laboratory
Department of Electrical Engineering
Stanford University
{mkalman, bgirod}@stanford.edu

ABSTRACT

When media is streamed over best-effort networks, a buffer at the client protects against playout interruptions due to variations in the data arrival rate. While the amount of protection offered grows with the size of the client's buffer, so does the latency that is introduced. In this paper we show how adaptive media playout (AMP) - the variation of the playout speed of media frames depending on the condition of the channel - allows smaller buffer sizes and correspondingly smaller delays for a given level of protection against buffer underflow. We demonstrate this with the results of Markov chain analyses and with simulations. In addition, we consider AMP as a form of receiver-driven rate scalability, allowing clients access to streams encoded at higher source rates than their connections would ordinarily allow.

1. INTRODUCTION

Streaming systems rely on buffering at the client to protect against the random packet losses and delays that characterize a best-effort network. Buffering reduces a system's sensitivity to short-term fluctuations in the data arrival rate by absorbing variations in end-to-end delay and allowing margin for retransmission attempts when packets are lost.

Buffering has drawbacks, however. While the amount of protection a buffer offers grows with its size, so does the latency that it introduces. Latency is most noticeable to the user as pre-roll delay, the time it takes for the buffer to fill with data and for playout to begin after the user makes a request. However, in streams of live events or in two-way communication, latency is noticeable throughout the session. Furthermore, buffering is generally only useful as long as the mean data arrival rate at the client remains at or above the source rate. If the rate offered by the channel is below, or falls below that of the source, a buffer will soon underflow. What the system needs, in this case, is a way to reduce the source rate.

Adaptive Media Playout (AMP) is a client-controlled means to do just this. AMP allows the client to flexibly adjust its data consumption rate by varying the speed at which frames of media are played out. For video, the client simply adjusts the duration that each frame is shown. For audio, the client performs signal processing in conjunction with time scaling to preserve the pitch of the signal. Informal subjective tests have shown that slowing the playout rate of video and audio up to 25% is often un-noticeable, and that time-scale modification is preferable subjectively to halting playout or errors due to missing data [1],[2].

In this paper we examine two applications of AMP. First, we show that AMP, a form of rate-scalability, can allow clients to access streams which are encoded at a higher source rate than their connection would ordinarily allow. Second, we show how AMP can be used to improve the inherent tradeoff between buffer underflow probability and latency.

2. RECEIVER-DRIVEN RATE SCALABILITY WITH AMP

In the absence of a widely accepted and truly efficient, fine-grained rate-scalable codec, media content providers attempt to accommodate a wide range of viewer connection speeds by offering multiple versions of stored programs, each encoded at a different rate [3]. A user requests the stream that is closest to, but does not exceed, the available connection speed. When a user's connection speed is nearly, but not quite, sufficient for a particular encoding, the stream that must be selected is at a rate and quality that does not take full advantage of the connection that is available.

Given the ability to scale the playout rate of media data, however, the user can select a higher quality stream and reduce its effective source rate to better match the connection speed. Thus, AMP provides the receiver with a mechanism for fine-grained scalability in which picture quality can be improved at the cost of a moderate increase in the playout duration of the streamed media content. Fig. 1 illustrates this flexibility. In the video streaming example shown, the server offers different versions of the same program with a 20% difference in average rate from stream to stream. The light gray area corresponds to the operational domain if the receiver does not offer playout rate scalability. If, for instance, the mean connection rate is 90 kbps, the receiver has to select the pre-encoded stream with a mean source rate of 80 kbps. In this example, this results in a reproduction fidelity of 32 dB PSNR. The dark gray area in Fig. 1 shows the additional operational domain that AMP provides. With an AMP-equipped receiver, the 100 kbps media stream can be streamed over the 90 kbps connection with a playout speed scaling factor of $90\text{kbps}/100\text{kbps} = 0.9$. The reconstruction quality is now 34 dB PSNR, an improvement of 2dB.

Note that the adjustment of playout rates is not without precedent in broadcast media. Motion pictures which are shot at 24 frames per second are shown on European television at 25 fps, the frame rate dictated by PAL, which constitutes a speed-up of 4.17%.

*This work has been supported, in part, by a gift from Intel Corporation, and, in part, by the Stanford Networking Research Center.

†Now with the Institute of Communication Networks, Media Technology Group, Technical University of Munich, steinb@lkn.ei.tum.de

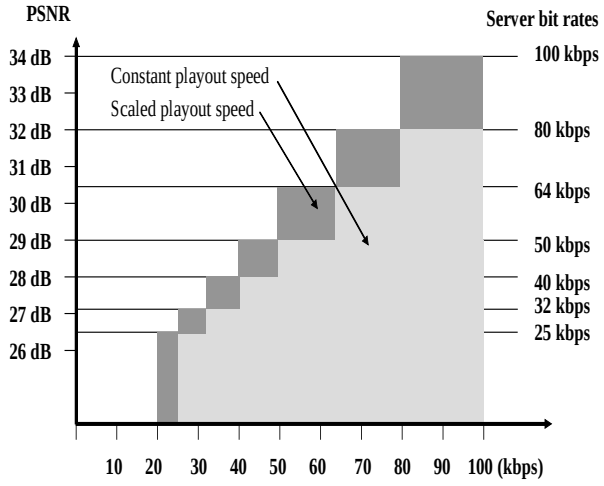

Fig. 1. Allowable server bit rates and the corresponding PSNRs as a function of the available channel rate, with and without AMP.

3. LATENCY REDUCTION WITH AMP

In Section 1 we noted the tradeoff between latency and protection against buffer underflow: both increase with buffer size. In this section we explain how AMP can improve this tradeoff in three distinguishable modes of use. For a more detailed description of these modes please see [4].

The first mode, AMP-Initial, aims to reduce pre-roll delay. In this mode, the client begins playout of a stream before the playout buffer has been filled to its target level, N_{start}. Playout begins at a reduced rate, however, with frame-periods stretched by a scaling factor $s > 1$. Thus, the mean arrival rate of data out paces the mean consumption rate, and the buffer slowly fills. When the buffer backlog reaches its target level, playout resumes at normal speed.

In the second mode, AMP-Robust, whenever the playout buffer backlog dips below a threshold level N_{adapt}, the playout rate is reduced. When the backlog regains the threshold level, playout resumes normally. With AMP-Robust, if the mean data arrival rate at the client remains at the source rate but fluctuates about the mean, slow playout periods increase the mean backlog at the client leading to more robust protection against underflow. This is desirable for stored programs, where latency is not noticeable after the pre-roll buffering period.

For streams of live events, or for two-way communication, however, latency is noticeable and thus the playout buffer backlog cannot be allowed to increase without bound. AMP-Mean, the third mode that we distinguish, reduces playout speed when the playout buffer backlog falls below the target level by stretching frame-periods by scaling factor s, but it also plays faster than normal when the backlog grows larger than some upper threshold. During faster playout frame periods are scaled by a factor $f < 1$. As we will see in Section 5, analysis and simulation results show that by playing slowly *and* quickly, the mean buffer backlog (and thus latency) can be held at lower mean level than would be allowable otherwise, for a given underflow probability.

4. SIMULATION EXPERIMENTS AND MARKOV CHAIN ANALYSIS

We quantify the improvements in buffer underflow probability versus latency realizable with AMP using Markov chain analysis and simulation experiments. This section is a brief review of our analysis and simulation procedures. They are described in more detail in [4].

4.1. Channel Model

In both our Markov chain analyses and simulation experiments, we model the channel with a two-state, Markov-modulated Poisson process. The channel transitions randomly between a good and a bad state. When a packet appears at the server end of the channel, it must wait an exponentially distributed random time to be transferred to the client. During the transfer, however, the packet may be lost with some probability. The mean waiting times and loss probabilities are higher when the channel is in the bad state compared to the good state. The channel is characterized by parameters

$$C = (t_{prop}, \lambda_G, \lambda_B, T_G, T_B, p_G, p_B) \quad (1)$$

were G signifies the good state, and B the bad. λ_G and λ_B are the mean arrival rates, T_G and T_B are the mean channel state durations, and p_G and p_B are the packet loss probabilities. t_{prop} specifies the one-way propagation time between server and client that is incurred in addition to the waiting time. The channel remains in the good and bad states for random holding times that are distributed according to $exp(1/T_G)$ and $exp(1/T_B)$, respectively.

4.2. Streaming System Model

In our simulation experiments we incorporate the channel model described above into the system shown in Fig. 2. Our system model consists of a source, a server, a channel, and a client. It operates as follows.

The source generates frames and passes them to the server. If the source is a live program, throughout the streaming session it passes a new frame to the server every t_F seconds. If the source is a stored program, all of the frames are transferred to the server at the start of the session. Once frames are transferred to the server, they are each placed in a packet and added to the transmission queue (TX Queue). A copy of each packet is saved in the packet store for later retransmission if necessary. The server also contains a queue of packets for which retransmission requests have been received (RTX Queue). The packets in this queue are sorted according to playout deadline.

The channel services these queues with priority given to packets in the RTX Queue since these packets are nearer to their playout deadlines. The random inter-service times are distributed as described in Section 4.1. When packets cross the channel they are placed in the client's playout queue.

After playout begins, the playout queue is serviced deterministically at a rate $\mu(n)$, which is constant during non-adaptive playout, and varies with n, the number of packets in the queue, during adaptive playout. When a packet arrives at the client with a non-contiguous sequence number, the client assumes that any missing packets have been lost. The client places retransmission requests for the missing packets into the retransmission request queue (RTX Req.). The service times for this queue are the same

as in the forward direction. We assume that the retransmission requests are never lost with the rationale that since the requests are much smaller than frames of media, they can be adequately protected. After a fixed propagation delay, the retransmission requests arrive at the server where the appropriate packet is fetched from the packet store and placed in the retransmission queue.

By simulating this system, we can find mean pre-roll times, mean latencies and buffer underflow probabilities given a set of channel parameters and an adaptive playout policy. Below we describe how we find the same results using Markov chain analysis.

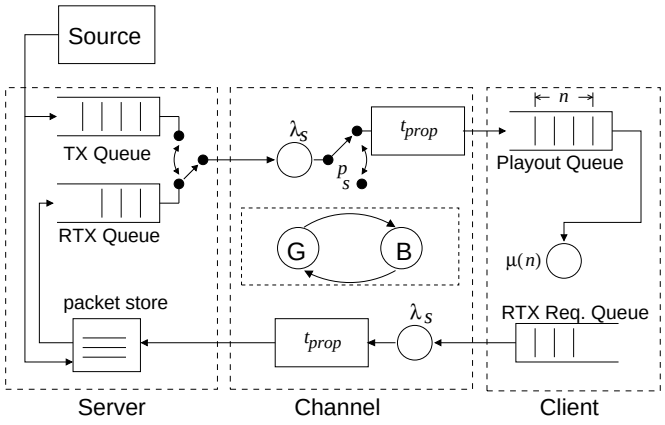

Fig. 2. Streaming video system model

4.3. Markov Chain Analysis

To perform Markov chain analysis, we define a finite state space describing all the possible states of the system and then find the probabilities of transitioning between each pair of states from one discrete time step to the next. The steady state probability distribution over the states of the system can then tell us the quantities we seek, namely the mean backlog in the server and playout queues (latency) and the probability of the playout queue backlog being zero when a packet is needed for playout (underflow). Similarly, we can find values for mean pre-roll delay. For more detail please see [4].

To perform Markov chain analysis we must simplify the system shown in Fig. 2, however [5]. The number of states needed to characterize the system in Fig. 2 would be too large to allow a tractable analysis.

To sidestep this problem we observe that for reasonable packet loss rates (typically $\leq 20\%$), after a few retransmission attempts the probability that a packet is received is nearly 1. With a playout buffer that is significantly longer than a round-trip time, our system will behave very similar to an erasure channel with an unlimited number of retransmissions allowable for each packet. We can thus model packet loss as a reduction in throughput [6]. Fig. 3 illustrates the simplified system that we assume in our analysis. Let parameters of the simplified channel model be:

$$\hat{C} = (\hat{\lambda}_G, \hat{\lambda}_B, T_G, T_B) \quad (2)$$

where $\hat{\lambda}_G = (1 - p_G) \cdot \lambda_G$, and $\hat{\lambda}_B = (1 - p_B) \cdot \lambda_B$. Also note that we have dropped t_{prop} as it does not affect our analysis to assume $t_{prop} = 0$.

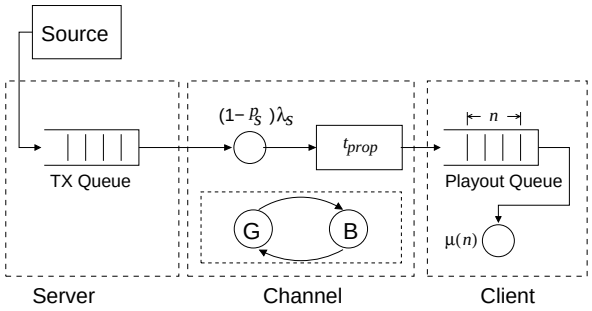

Fig. 3. The simplified streaming video system model. To allow a tractable Markov chain analysis, we translate the packet loss rate into a reduction in packet arrival rate.

5. RESULTS

Markov chain analysis and simulation results show the extent to which AMP can reduce latency for a given playout buffer underflow probability. In Fig. 4, we see a plot of Mean Time Before Buffer Underflow (MTBBU, a function of the underflow probability) versus the mean latency for three slow-down factors s paired with three speed-up factors f. This is an example of AMP-Mean, where the stream is a 'live' program and it is therefore desirable to minimize latency - the mean time between live actions and their appearance at the receiver - for a given MTBBU. We see that for a fixed MTBBU, AMP-Mean reduces latency by 25-30% for these channel parameters.

Fig. 5 answers the question: given a stored program of a given length, what pre-roll delay is required so that the underflow probability is less than 0.01? It plots mean pre-roll time versus program length, such that the program plays to completion without underflow 99% of the time. The results are for a system that combines AMP-Initial with AMP-Robust. For a given program length, the client buffers N_{start} frames before playout begins. The frame periods during playout are stretched by a scaling factor s, however, whenever the backlog in the buffer falls below $N_{adapt} = 100$. Since the normal frame rate of the simulated system is 10 fps, throughout the playout of the program, whenever less than 10 seconds of data are available in the playout buffer, the playout speed is reduced. For these channel parameters, $t_{prop} = 10$ ms, $\lambda_G = 1.2824$, $\lambda_B = .4706$, $T_G = 20$ s, $T_B = 2$ s, $p_G = 0.15$, and $p_B = .15$, we see that the necessary pre-roll delays with AMP can be a fraction of would otherwise be required.

6. CONCLUSION

In this work we have explored adaptive media playout (AMP) as a means to scale the source rate of a media stream at the receiver. We have shown two distinct ways to make use of this flexibility. First, AMP can allow users to access streams that are encoded at a source rate that is greater than their connections would ordinarily allow. Second, AMP can improve the tradeoff between buffering-induced latency and underflow probability. We have outlined the simulation models and Markov analysis that we have used to quantify improvements in this tradeoff, and our results show that AMP can lead to pre-roll times that are a fraction of what they would be otherwise, and to reductions of 25-30% in mean latency for live

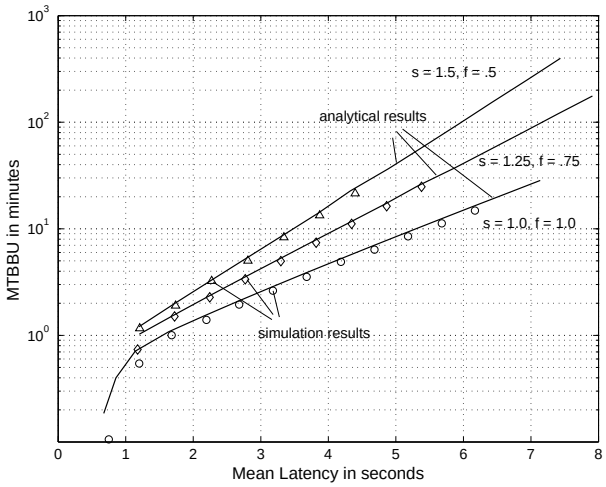

Fig. 4. Mean Time Between Buffer Underflow (MTBBU) vs. Mean Latency for a channel C characterized by $t_{prop} = 10$ ms, $\lambda_G = 1.583$, $\lambda_B = 0$, $T_G = 28.5$ s, $T_B = 1.5$ s, $p_G = .157$, and $p_B = 1$.

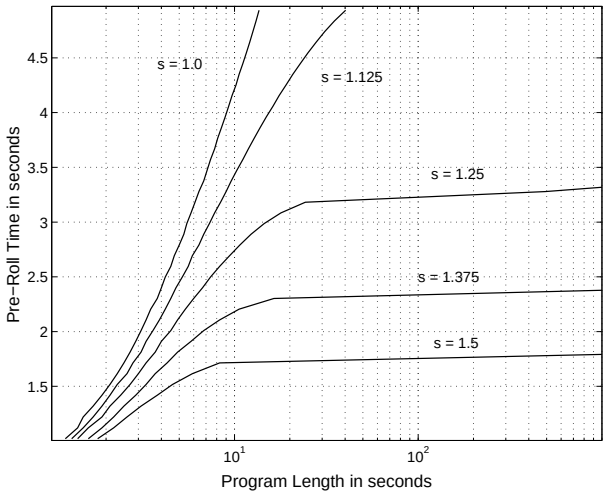

Fig. 5. A plot of the mean required pre-roll time as a function of program length so that $Pr\{\text{underflow}\} \leq 0.01$ for a combination of AMP-Initial and AMP-Robust.

streams.

7. REFERENCES

[1] E. G. Steinbach, N. Färber, and B. Girod, "Adaptive Playout for Low Latency Video Streaming," *Proc. International Conference on Image Processing (ICIP-01)*, Thessaloniki, Greece, Oct. 2001.

[2] Y. J. Liang, N. Färber, and B. Girod, "Adaptive Playout Scheduling Using Time-scale Modification in Packet Voice Communication," *Proc. ICASSP '01*, Salt Lake City, May 2001.

[3] A. Lippman, "Video coding for multiple target audiences," *Proc. Visual Communications and Image Processing '99*, vol. 3653, pp. 780-782, Jan. 1999.

[4] M. Kalman, E. Steinbach, and B. Girod, "Adaptive Media Playout for Low Delay Video Streaming over Error-Prone Channels," *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Wireless Video*, submitted August 2001. `http://www.stanford.edu/~mkalman`

[5] M. Podolsky, S. McCanne, and M. Vetterli, "Soft ARQ for layered streaming media," *Tech. Rep. UCB/CSD-98-1024, University of California, Computer Science Division, Berkeley, CA*, Nov. 1998.

[6] S. Lin, D.J. Costello Jr., "Automatic repeat request error control schemes", *IEEE Commun. Mag.*, vol. 22, no. 12, Dec. 1984.

[7] G.J. Conklin, G.S. Greenbaum, K.O. Lillevold, A.F. Lippman, and Y.A. Reznik, "Video coding for streaming media delivery on the Internet," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 269 - 281,vol. 11, no. 3, March 2001.

[8] Y.J. Liang, E.G. Steinbach, and B. Girod, "Real-time Voice Communication over the Internet Using Packet Path Diversity," *Proc. ACM Multimedia 2001*, Ottawa, Canada, Sept./Oct. 2001.

[9] M.C. Yuang, S.T. Liang, and Y.G. Chen, "Dynamic Video Playout Smoothing Method for Multimedia Applications," *Multimedia Tools and Applications*, vol. 6, pp. 47-59, 1998.

[10] A. Stenger, K. Ben Younes, R. Reng, and B. Girod, "A new error concealment technique for audio transmission with packet loss," *Proc. EUSIPCO '96*, Trieste, Italy, Sept. 1996.

[11] W. Verhelst and M. Roelands, "An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech," *Proc. ICASSP '93*, pp. 554-557, April 1993.