RATE-DISTORTION OPTIMIZED VIDEO STREAMING WITH ADAPTIVE PLAYOUT

Mark Kalman, Eckehard Steinbach, and Bernd Girod

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

ABSTRACT

We propose a new scheme for streaming media systems that combines Adaptive Media Playout (AMP) with rate-distortion optimized packet transmission. AMP, the client-controlled, adaptive modification of the media playout rate, allows us to flexibly adjust the playout deadlines of individual packets, and can therefore reduce reconstruction distortion. This added flexibility incurs a subjective cost, however. In this work we introduce functions that assess the subjective cost of a schedule of playout rate modifications, and we show how to optimize the schedule with respect to these costs and distortion. Because the optimal playout rate schedule and the R-D optimal transmission schedule are interdependent, we solve the two problems jointly. In simulations that model a receiver-driven scenario, results for a short media clip show a more than 2 dB improvement in mean PSNR for R-D optimal transmission scheduling combined with a moderate amount of AMP, over R-D optimal transmission scheduling alone.

1. INTRODUCTION

During a streaming media session, packets must successfully cross a lossy channel within a finite time that is largely given by the size of the playout buffer at the client. The purpose of the playout buffer is to absorb end-to-end delay jitter and to allow time for retransmission attempts when packets are lost. Because the playout buffer is finite, however, and because there are constraints on the allowable instantaneous transmission rate, retransmission attempts for lost packets divert transmission opportunities from subsequent packets and reduce the amount of time that subsequent packets have to successfully cross the channel. A streaming media system must make decisions, therefore, that govern how it will allocate transmission resources among packets.

A rate-distortion optimized streaming system allocates time and bandwidth resources among packets in a way that minimizes the expected reconstruction distortion of the media representation. For example, consider a scenario in which uniformly sized frames of media are placed in individual packets, and one packet is transmitted per discrete transmission interval. A rate-distortion optimized streaming system decides which packet to transmit at each opportunity based on the packets' deadlines, their transmission histories, the channel statistics, feedback, the packets' interdependencies, and the reduction in distortion yielded by each packet if it is successfully received and decoded.

We can improve the performance of such a system with Adaptive Media Playout (AMP). AMP allows us to independently scale the playout durations of frames of media and thus gives us some control over arrival deadlines. Playing a frame slowly, for example, extends the arrival deadlines of frames that follow. For video, frame period scaling is accomplished simply by adjusting the duration that each frame is shown. For audio, signal processing done in conjunction with time scaling preserves 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].

While AMP reduces the expected distortion of a media stream by pushing back deadlines, playout rate variations do have a negative subjective impact that we would like to measure. In this paper we introduce functions that assess the subjective cost of a playout schedule. We then show how to find a playout schedule that is optimal with respect to subjective cost and distortion.

Because the optimal playout schedule and the R-D optimal transmission schedule are interdependent, we solve these problems jointly. In [3], the authors present a framework to achieve rate-distortion optimal packet scheduling. In this paper, we incorporate distortion-subjective cost optimized AMP into their framework. While in [3] the authors consider primarily a sender-driven scenario, for our purposes a receiver-driven approach is more appropriate. If the joint optimization is computed at the client, playout schedules, once determined, do not need to be communi-

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

cated across the lossy packet network.

2. R-D OPTIMIZED STREAMING WITH FIXED PLAYOUT RATES

In this section we review the general framework for rate-distortion optimized streaming given by Chou and Miao in [3] but we make modifications to suit our receiver-driven approach, and to ease the incorporation of AMP in Section 3.

Let l be a data unit - a packetized frame or portion of one. In our receiver-driven scenario, the client, at discrete time intervals, sends requests to the server asking that it transmit one or more data units. At any request opportunity the client may only request data units whose deadlines for playout are within a certain time window. This means that each data unit has a limited number of opportunities to be requested from the time it first enters the window until it is due for playout. Let π_l be the request policy for data unit l. The policy governs whether the client requests data unit l or not at each of l's opportunities, given that l has not arrived. The policy consists of a sequence of actions $[a_0, a_1, \ldots, a_{N-1}]$, $a_i \in \{0, 1\}$. $a_i = 1$ indicates that the client makes a request at opportunity i if the data unit has not yet been received.

Now suppose that we can independently scale the playout duration of each frame of media. Let playout schedule $\boldsymbol{\nu} = \{\nu_1, \nu_2, \ldots, \nu_F\}$ be the set of frame-period scaling factors for an entire sequence of length F frames. If $\nu_1 = 1.25$, for example, the first frame will be shown for $1.25 \times t_F$ seconds, where t_F is the original frame period. In the non-adaptive case the scaling factors are all ones.

Given the playout schedule and the request policy for data unit l, we can compute $\epsilon(\pi_l, \boldsymbol{\nu})$, the probability that l does not reach the client by its deadline, conditioned upon whether or not it has arrived by the current time.

$$\epsilon(\pi_l, \boldsymbol{\nu}) = \begin{cases} \prod_{i:a_i=1} \frac{1-F_{RTT}\left(t_{DTS}-t_i+t_F \sum_{j:j<f_l}(\nu_j-1)\right)}{1-F_{RTT}(t_{now}-t_i)} & \text{if not arrived} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The random variable RTT is the round trip time, the time it takes for a data unit to arrive at the client after a request is made, and F_{RTT} is its cdf. We assume that RTT's are distributed independently. t_{DTS} is the playout deadline for the data unit before playout adaption, t_i is the time at which action i occurs, t_{now} is the current time, and f_l specifies the frame that data unit l belongs to. Given π_l we can also calculate $\rho(\pi_l)$, the expected number of times data unit l will be transmitted under policy π_l.

Let $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_L)$, be the set of policies for all the data units in a group of frames and let $D(\boldsymbol{\pi}, \boldsymbol{\nu}) = \sum_{l=1}^{L} D_l$, be the expected distortion for that group.

$$D_l = \Delta d_l - \Delta D_l \prod_{l' \preceq l} (1 - \epsilon(\pi_{l'}, \boldsymbol{\nu})), \quad (2)$$

where ΔD_l is the expected reduction in distortion if data unit l is decoded on time. In modern codecs, data units are often dependent on the presence of other data units to be decoded. For instance, in a succession of video frames I-P-P-P, each P frame relies on the I frame and preceding P frames. In (2), $l' \prec l$ indicates that l is dependent on l' to be decoded. $\prod_{l' \preceq l}(1 - \epsilon(\pi_{l'}, \boldsymbol{\nu})$ is therefore the probability that l is decodable because it and all the packets it depends on arrive on time. Δd_l refers to the distortion that data unit l removes in the absence of any error concealment. The expression in (2) is actually an approximation that assumes that the expectation ΔD_l is independent of $\prod_{l' \preceq l}(1-\epsilon(\pi_{l'})$ and may be factored separately.

Let $R(\boldsymbol{\pi}, \boldsymbol{\nu}) = \sum_{l=1}^{L} R_l$, be the expected rate incurred by a group of frames under policies $\boldsymbol{\pi}$ and playout schedule $\boldsymbol{\nu}$, where

$$R_l = \frac{B_l \rho_l}{t_{DTS} + t_F \sum_{j:j<f_l}(\nu_j - 1) - t_{lead}}. \quad (3)$$

In (3), t_{lead} is the time that l first enters the transmission window, and B_l is the size of data unit l in bytes.

For some fixed playout schedule $\boldsymbol{\nu}$, a rate-distortion optimal set of policies $\boldsymbol{\pi}$ minimizes the cost function $J(\boldsymbol{\pi}, \boldsymbol{\nu})$ $= D(\boldsymbol{\pi}, \boldsymbol{\nu}) + \lambda R(\boldsymbol{\pi}, \boldsymbol{\nu})$. We find this policy vector iteratively. Starting with some initial policy vector $\boldsymbol{\pi}$, at each iteration the algorithm selects one data unit l, and, holding the policies for the other data units fixed, it adjusts policy π_l to minimize J. Because data units are interdependent, however, each time one policy is adjusted, the optimality of the other policies is no longer assured. The policies for all the data units are therefore iteratively adjusted in a round robin way. Since at each iteration the cost J is reduced, and since the cost is bounded below, the algorithm is assured to converge to at least a locally minimal J.

3. OPTIMIZED STREAMING WITH AMP

Consider a media stream of length F frames. The deadline and thus the expected distortion of the final frame, F, is dependent on the playout speeds of frames 1 through $F - 1$. Therefore, to determine the optimal playout rate of, for instance, frame 1, an algorithm would also need to know the playout rates expected for frames 2 through $F - 1$. The algorithm would furthermore need to calculate the request policies expected for all the remaining frames. In fact, each time the system needed to decide the playout rate of the next frame, or decide which data units to request at a discrete request opportunity, the entire playout and the entire request schedules would need to be recomputed. This is not feasible, and not really necessary. Just as, at any time during the streaming session, we only consider requesting data units whose deadlines fall within a certain time window, at a given time during the session we will only consider the portion of the playout schedule corresponding to frames whose deadlines are within a certain time window.

Let $\tilde{\boldsymbol{\nu}} = \{\tilde{\nu_1}, \tilde{\nu_2}, \ldots, \tilde{\nu}_M\}$ be the window of frames whose playout scaling factors are under consideration, where $\tilde{\nu_1}$ is the scaling factor of the frame following the one that is currently playing.

Let $G_1(\tilde{\boldsymbol{\nu}})$ and $G_2(\tilde{\boldsymbol{\nu}})$, $G_j : \Re^M \to \Re$, be functions that assign a subjective cost to $\tilde{\boldsymbol{\nu}}$ for manipulating the playout rates of a block of frames. We propose the following subjective cost functions:

$$G_1(\tilde{\boldsymbol{\nu}}) = \sum_{i=1}^{M} (\tilde{\nu}_i - 1)^2 \tag{4}$$

$$G_2(\tilde{\boldsymbol{\nu}}) = \sum_{i=1}^{M} (\tilde{\nu}_i - \tilde{\nu}_{i-1})^2 \tag{5}$$

G_1 places a cost on the deviation of the frame-period scaling factors from 1, and G_2 puts a price on variations in the playout rate from one frame to the next ($\tilde{\nu}_0$ refers the frame-period scaling factor of the currently-playing frame).

We perform jointly optimal packet and playout scheduling by minimizing

$$J(\boldsymbol{\pi}, \tilde{\boldsymbol{\nu}}) = D(\boldsymbol{\pi}, \tilde{\boldsymbol{\nu}}) + \lambda R(\boldsymbol{\pi}, \tilde{\boldsymbol{\nu}}) + \lambda_1 G_1(\tilde{\boldsymbol{\nu}}) + \lambda_2 G_2(\tilde{\boldsymbol{\nu}}). \tag{6}$$

To minimize, we begin with initial values for vectors $\boldsymbol{\pi}$ and $\tilde{\boldsymbol{\nu}}$. Similar to the procedure in [3], we iteratively optimize each π_l while holding the other policies and the playout schedule fixed. Then, at the end of an optimization pass, we hold the policies $\boldsymbol{\pi}$ fixed and minimize (6) with respect to $\tilde{\boldsymbol{\nu}}$. We repeat until convergence, which is guaranteed again because at each stage the cost J is reduced, and it is bounded below. The algorithm runs at each transmission opportunity and each time a new frame is to begin playout. Upon convergence, the actions in $\boldsymbol{\pi}$ that correspond to the current time specify what requests are sent, and $\tilde{\boldsymbol{\nu}}_{\mathbf{1}}$ species the playout rate of the next frame.

For fixed $\boldsymbol{\pi}$, we find the best $\tilde{\boldsymbol{\nu}}$ using the method of iterated linearized least squares. Let

$$\begin{aligned}\mathbf{b} = \big(&\sqrt{D_1}, \sqrt{D_2}, \ldots, \sqrt{D_L}, \sqrt{\lambda R_1}, \sqrt{\lambda R_2}, \ldots, \sqrt{\lambda R_L},\\ &\sqrt{\lambda_1 G_{1,1}}, \sqrt{\lambda_1 G_{1,2}}, \ldots, \sqrt{\lambda_1 G_{1,M}},\\ &\sqrt{\lambda_2 G_{2,1}}, \sqrt{\lambda_2 G_{2,2}}, \ldots, \sqrt{\lambda_2 G_{2,M}}\big),\end{aligned} \tag{7}$$

and note that $||\mathbf{b}||^2 = J(\boldsymbol{\pi}, \tilde{\boldsymbol{\nu}})$. We minimize $||\mathbf{b}||^{\mathbf{2}}$ with respect to $\tilde{\boldsymbol{\nu}}$ iteratively. For each iteration n, beginning at $n = 0$ with initial scaling factors $\tilde{\boldsymbol{\nu}}^{(0)}$, we create the matrix $A^{(n)} \in \Re^{(2L+2M)\times M}$, where

$$[A^{(n)}]_{i,j} = \left.\frac{\partial b_i}{\partial \nu_j}\right|_{\tilde{\boldsymbol{\nu}} = \tilde{\boldsymbol{\nu}}^{(\boldsymbol{n})}}, \tag{8}$$

and the vector $\mathbf{b}^{(\mathbf{n})}$ where

$$\mathbf{b}^{(\mathbf{n})} = \mathbf{b}|_{\tilde{\boldsymbol{\nu}} = \tilde{\boldsymbol{\nu}}^{(\boldsymbol{n})}}. \tag{9}$$

We then find $\Delta\tilde{\boldsymbol{\nu}}^{(\boldsymbol{n})} \in \Re^M$:

$$\Delta\tilde{\boldsymbol{\nu}}^{(\boldsymbol{n})} = \arg\min_{\Delta\tilde{\boldsymbol{\nu}}} \left\| A^{(n)} \Delta\tilde{\boldsymbol{\nu}} + \mathbf{b}^{(n)} \right\|, \tag{10}$$

which is a least-squares problem. We update:

$$\tilde{\boldsymbol{\nu}}^{(n+1)} = \tilde{\boldsymbol{\nu}}^{(n)} + \Delta\tilde{\boldsymbol{\nu}}^{(n)}, \tag{11}$$

and the process repeats until convergence.

In practice we add regularization to matrix A to ensure that it is well conditioned and that the $\Delta\tilde{\boldsymbol{\nu}}$ are not too large during each iteration. The allotted space prohibits us from providing expressions for the entries of A, but closed form expressions for these are readily found in terms of f_{RTT} and F_{RTT} - the pdf and cdf of the round trip time.

In actual implementation, we would allow the user to control the weights λ_1 and λ_2. The user would select a playout adaption versus distortion setting according to taste.

4. SIMULATION RESULTS

In our implementation, we constrain the transmission rate by allowing the client to make one transmission request per request opportunity. We assume that requests, though delayed randomly, reach the server without error, reasoning that the small packets may be adequately protected. The media packets traveling from server to client face both delay and losses, however.

Every 47.5 ms, our algorithm makes the R-D optimal request based on the non-iterative method for single transmissions given in [3]. It also finds the expected actions for 15 future slots. Thus it populates the policy vector $\boldsymbol{\pi}$ and can use the method in Section 3 to find the playout schedule $\tilde{\boldsymbol{\nu}}$ with $M = 4$. It avoids the complexity of iterating between solving for $\boldsymbol{\pi}$ and $\tilde{\boldsymbol{\nu}}$, and finds each only once. At each request interval, however, it bases its request and new $\boldsymbol{\pi}$ on the $\tilde{\boldsymbol{\nu}}$ found during the previous slot.

The results shown in Figures 1 and 2 are for *Foreman* encoded with H.26L at 10 fps, with no B frames and with I frames every .5 seconds. Thus, a group of frames is arranged I-P-P-P-P. Since the I frame is roughly four times larger than P frames, it is broken into four interdependent packets. Therefore, packets are roughly equally-sized and, because requests are not lost, the transmission rate from the server is relatively constant. We model FTT, the time it takes a request to reach the server, as a random variable with a right shifted Γ distribution as in [3], with parameters $n_F = 3$, $\alpha_F = 33.33$, and $\kappa_F = 10$ ms, yielding a mean FTT of 100 ms [4]. The round-trip time, RTT, is distributed according to $(1 - p_{loss})(f_{FTT} * f_{FTT}) + p_{loss}\delta(t - \infty)$, where the loss probability $p_{loss} = .2$ [3]. Playout begins after 15 transmission intervals yielding a pre-roll delay of $.7125$ seconds. The client uses rudimentary error concealment. When a frame is not decodable by its playout deadline, the client plays the most recent frame that is decodable. In our algorithm we approximate the value of ΔD_l with $\Delta \hat{D}_l = 700$ for all l, based on empirical observation.

In Figure 1, the first plot is the mean PSNR for the *Foreman* sequence as a function of the mean frame period scaling factor as controlled by the parameters λ_1 and λ_2 in (7). The second shows mean PSNR versus the average maximum frame period scaling factor. We see a 2.25 dB improvement in mean PSNR for mean playout rates that are slowed down by less than 20% on average, if the user is willing to tolerate a peak slowdown of 30 - 35%.

The first trace in Figure 2 shows the frame-by-frame PSNR of the *Foreman* sequence, with no adaption, but with receiver-driven R-D optimal streaming as described above. The second trace shows, for the same random seed, the frame-by-frame PSNR when AMP is used. The third trace shows the frame-by-frame playout scaling factors chosen by our algorithm.

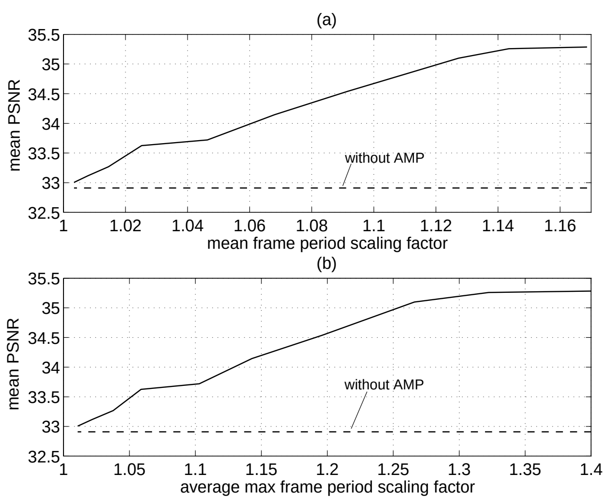

Fig. 1. (a) Mean PSNR vs. mean playout speed, (b) and the average maximum playout speed.

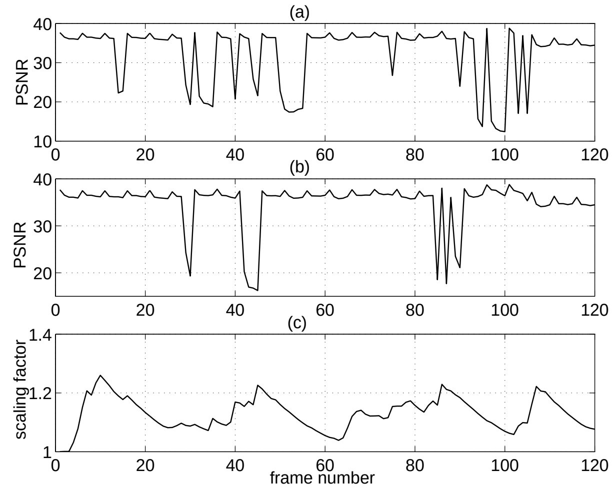

Fig. 2. PSNR vs. frame number for (a) one simulation outcome without AMP, (b) with AMP , and (c) playout speed vs. frame number with AMP

5. CONCLUSION

In this paper we have presented a method to combine subjectively optimized AMP with R-D optimized packet scheduling. After introducing functions that assess the subjective cost of a schedule of playout rates, we have shown how to find a playout schedule that is optimal with respect to distortion and subjective cost, jointly with the R-D optimal transmission schedule. In simulations of receiver-driven video streaming, we have shown that optimized AMP can improve the mean PSNR of a short clip by more than 2 dB for moderate amounts of adaption.

6. 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] P. A. Chou and Z. Miao, “Rate-distortion optimized streaming of packetized media,” IEEE Transactions on Multimedia, February 2001. Submitted. `http://research.microsoft.com/~pachou`

[4] A. Leon Garcia, *Probability and Random Processes for Electrical Engineering*, Addison Wesley, 1994.