Rate-Distortion Optimized Receiver-Driven Streaming over Best-Effort Networks

Philip A. Chou[†] and Anshul Sehgal[*]

[†] *Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399*

[*] *Beckman Institute, University of Illinois, Urbana, IL 61801*

pachou@microsoft.com, asehgal@uiuc.edu

Abstract

This paper addresses the problem of streaming packetized media over a lossy packet network, in a rate-distortion optimized way, where the rate-distortion optimization is performed at the receiver. We show how the receiver should compute which packets, if any, to request from the sender, in order to meet a constraint on the transmission rate from the sender to the receiver while minimizing the average end-to-end distortion. Experimental results show that the receiver-driven system has performance within 1 dB of a sender-driven system, while reducing the computational burden at the sender.

1 Introduction

In this paper, we employ a framework for distortion-rate optimized streaming of packetized media, which was introduced in [1, 2], and apply it to the scenario of receiver-driven transmission over a best-effort network. In our receiver-driven scenario, the receiver requests from the sender each unit of data that it wishes to receive, by transmitting a "request packet" to the sender that requests the desired data unit. If the request packet is not lost in the network, the sender replies by transmitting the requested data unit to the receiver. In turn, if the packet containing the data unit is not lost in the network, the receiver buffers the data unit until it is ready to be decoded. However, if the receiver does not receive the requested data unit in a timely fashion, the receiver may request the data unit again. This process is repeated until either the data unit arrives at the receiver on time (i.e., before its decoding deadline) or the receiver gives up.

Every data unit that arrives at the receiver on time reduces the distortion of the presentation by an amount that corresponds to the basic importance of the data unit, provided that all of the data units on which it depends have also arrived at the receiver. In addition, every data unit that is transmitted from the sender incurs a transmission cost. It is of interest to minimize both the distortion and the transmission cost. Towards this end, in our rate-distortion optimization framework, the receiver makes its requests according to a “request policy” that minimizes the expected distortion subject to a constraint on the expected rate of transmission from the sender to the receiver. The receiver determines the optimal request policy by minimizing a Lagrangian, taking into account the data units’ dependence relationships in addition to their different decoding deadlines and basic importances. This minimization results in a form of unequal loss protection, in which *sensitive* data units — data units whose delivery most affects the distortion — are requested or re-requested in preference to less sensitive data units. Consequently the more sensitive data units are often requested far in advance of their decoding times, while the less sensitive data units are requested later if at all. Furthermore the sensitivity of a data unit is not a static quantity, but varies in response to feedback from the receiver. If a data unit is received, then the data units on which it depends increase in sensitivity while the data units that depend on it decrease in sensitivity, and vice versa. Optimization of the request policy is performed by an iterative descent algorithm called the iterative sensitivity adjustment (ISA) algorithm.

In sender-driven streaming [1, 2, 3] the sender decides which packets should be transmitted to the receiver. It does this on the basis of information available to it such as acknowledgements it receives from the receiver for previously received packets (which could be lost or delayed), the rate-distortion characteristics of the presentation, decoding timestamps of the data units, etc. Receiver-driven streaming has certain advantages over sender-driven streaming, stemming from knowledge that the receiver has but the sender does not. In particular, the receiver has perfect and arbitrarily complete knowledge of all the data units that it has ever received from any sender. This knowledge can be especially useful if the receiver wishes to replay any content that it has received in the past (the state of which has been discarded by the sender), or if the receiver wishes to distribute transmission of a single presentation across multiple senders, e.g., for the purposes of reliability or load balancing. However, in receiver-driven streaming, the receiver does not have the knowledge the sender has of the distortion-rate characteristics of the presentation or the decoding timestamps of data units. This paper addresses the problem of receiver-driven streaming in a rate-distortion optimized framework.

2 Preliminaries

In a streaming media system, the encoded data are packetized into *data units* and are stored in a file on a media server. Regardless of how many media objects (audio, video, etc.) there are in a multimedia presentation, and regardless of what algorithms are used for encoding and packetizing those media objects, all of the data units in the presentation have interdependencies, which be expressed by a directed acyclic graph as illustrated in Figure 1. Each node of the graph corresponds to a data unit, and each edge of the graph directed from data unit l' to data unit l implies that data unit l can be decoded only if data unit l' is first decoded.

Associated with each data unit l is a size B_l, a decoding time $t_{DTS,l}$, and an importance Δd_l. The size B_l is the size of the data unit in bytes. The decoding time $t_{DTS,l}$ is the time at which the decoder is scheduled to extract the data unit from its input buffer and decode it. (This corresponds to the decoder timestamp in MPEG terminology.) In the context of the server/client model for streaming, $t_{DTS,l}$ is the delivery deadline by which data unit l must arrive at the client, or be too late to be used. Packets containing a data unit that arrive after the data unit's delivery deadline are discarded. The importance Δd_l is the amount by which the distortion at the receiver will *decrease* if the data unit arrives on time at the receiver and is decoded.

Also associated with each data unit l is a set of $N = N_l$ request opportunities $t_{0,l}, t_{1,l}, \ldots, t_{N-1,l}$ prior to $t_{DTS,l}$ at which the receiver may transmit a request for the data unit. Often this set of request opportunities is a single time $t_{0,l}$ prior to the delivery deadline, but in general we assume it is a finite set of times $t_{0,l}, t_{1,l}, \ldots, t_{N-1,l}$, such as the set of times at $T = 50$ ms intervals within a window prior to the delivery deadline.

We model the network as two independent channels: a backward (or uplink) path from the receiver to the sender and a forward (or downlink) path from the sender to the receiver. Each path is an independent time-invariant packet erasure channel with random delays. This means that if the receiver inserts a request packet into the backward path at time t, then the packet is lost with some probability, say ϵ_B, independent of t. However, if the packet is not lost, then it arrives at the sender at time t', where the backward trip time $BTT = t' - t$ is randomly drawn according to probability density p_B. Each packet is lost or delayed independently of the other packets. (However in practice we allow ϵ_B and p_B to adapt over time to the state of the channel, which may become congested, so that the actual channel has memory lasting perhaps several seconds.) For convenience, we combine the packet loss probability and the packet delay density into a single probability measure, by assigning $BTT = \infty$ in the event that the packet is lost. Thus $P\{BTT > \tau\}$ is the probability that a request packet transmitted by the receiver at time t does not arrive at the sender by time $t + \tau$, whether it is lost or simply delayed. Similarly, we let $P\{FTT > \tau\}$ be the probability that a data packet transmitted by the sender at time t does not arrive at the receiver by time $t + \tau$, whether it is lost or simply delayed. The round trip time $RTT = FTT + BTT$ is by definition the sum of forward and backward trip times. In our experiments we use as densities for p_B and p_F Gamma distributions with parameters (n_B, α_B) and (n_F, α_F) shifted to the right by κ_B and κ_F, respectively.

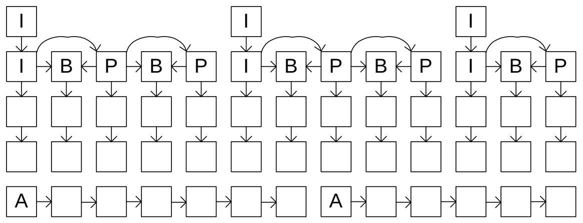


Figure 1: Typical directed acyclic dependency graph for video and audio data units.

3 Iterative Sensitivity Adjustment

We assume that communication of each data unit l can be achieved with a policy π_l selected from a family of policies Π. The family Π is determined by the scenario under consideration. For example, in a sender-driven scenario, Π may be a family of forward transmission schedules according to which the sender transmits the data unit until it receives an acknowledgement for the data unit from the receiver. In this paper, we focus on the receiver-driven scenario in which Π is the family of request policies according to which the receiver requests a data unit until it receives the data unit from the sender, or gives up. However, the rate-distortion optimization framework presented in this section does not depend on the specific scenario.

Suppose there are L data units in the multimedia session. Let π_l be the policy for data unit $l \in \{1, \ldots, L\}$ and let $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_L)$ be the vector of policies for all L data units. Any given policy vector $\boldsymbol{\pi}$ induces an expected distortion $D(\boldsymbol{\pi})$ and an expected transmission rate $R(\boldsymbol{\pi})$ for the multimedia session. We seek the policy vector $\boldsymbol{\pi}$ that minimizes the Lagrangian $D(\boldsymbol{\pi}) + \lambda R(\boldsymbol{\pi})$ for some Lagrange multiplier $\lambda > 0$, and thus achieves a point on the lower convex hull of the set of all achievable distortion-rate pairs.

The expected transmission rate $R(\boldsymbol{\pi})$ is the sum of the expected transmission rates for each data unit $l \in \{1, \ldots, L\}$:

$$R(\boldsymbol{\pi}) = \sum_l B_l \rho(\pi_l), \tag{1}$$

where B_l is the number of bytes in data unit l and $\rho(\pi_l)$ is the *expected cost* per byte, or the expected number of transmitted bytes per source byte under policy π_l. The expected distortion $D(\boldsymbol{\pi})$ is somewhat more complicated to express, but it can be expressed in terms of the *expected error*, or the probability $\epsilon(\pi_l)$ for $l \in \{1, \ldots, L\}$ that data unit l does not arrive at the receiver on time under policy π_l. Specifically,

$$D(\boldsymbol{\pi}) = D_0 - \sum_l \Delta D_l \prod_{l' \preceq l} (1 - \epsilon(\pi_{l'})), \tag{2}$$

where D_0 is the expected reconstruction error for the presentation if no data units are received and ΔD_l is the expected reduction in reconstruction error if data unit l is decoded on time. Derivation of this expression can be found in [1, 2].

With expressions (1) and (2) for the expected transmission rate and expected distortion for any given policy vector now in hand, we are able to find the policy vector $\boldsymbol{\pi}$ that minimizes the expected Lagrangian

$$J(\boldsymbol{\pi}) = D(\boldsymbol{\pi}) + \lambda R(\boldsymbol{\pi}) = D_0 + \sum_l \left[\Delta D_l \left(- \prod_{l' \preceq l} (1 - \epsilon(\pi_{l'})) \right) + \lambda B_l \rho(\pi_l) \right]. \tag{3}$$

However, this minimization is complicated by the fact that the terms involving π_l are not independent. We employ an iterative descent algorithm, called the iterative sensitivity adjustment (ISA) algorithm, in which we minimize the objective function $J(\pi_1, \ldots, \pi_L)$ in (3) one component at a time while keeping the other variables constant, until convergence. Let $\boldsymbol{\pi}^{(0)}$ be any initial policy vector and let $\boldsymbol{\pi}^{(n)} = (\pi_1^{(n)}, \ldots, \pi_L^{(n)})$ be determined for $n = 1, 2, \ldots$, as follows. Select one component $l_n \in \{1, \ldots, L\}$ to optimize at step n, e.g., $l_n = (n \bmod L)$. Then for $l \neq l_n$, let $\pi_l^{(n)} = \pi_l^{(n-1)}$, while for $l = l_n$, let $\pi_l^{(n)} = \arg\min_{\pi_l} J(\pi_1^{(n)}, \ldots, \pi_{l-1}^{(n)}, \pi_l, \pi_{l+1}^{(n)}, \ldots, \pi_L^{(n)})$, or

$$\pi_l^{(n)} = \arg\min_{\pi_l} S_l^{(n)} \epsilon(\pi_l) + \lambda B_l \rho(\pi_l), \tag{4}$$

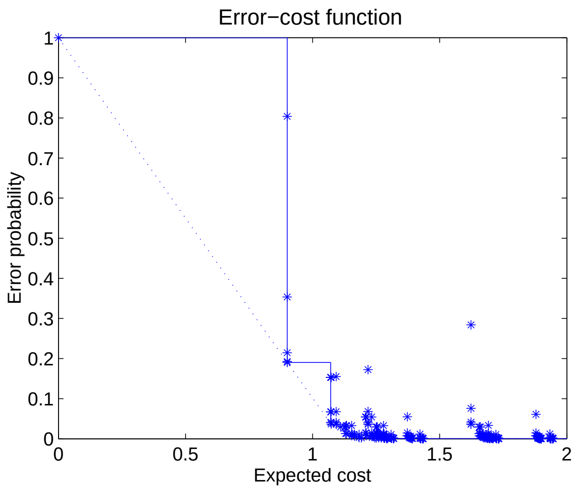
where (4) follows from (3) with $S_l^{(n)} = \sum_{l' \succeq l} \Delta D_{l'} \prod_{l'' \preceq l' : l'' \neq l} (1 - \epsilon(\pi_{l''}^{(n)}))$. The factor S_l can be regarded as the *sensitivity* to losing data unit l, i.e., the amount by which the expected distortion will increase if data unit l cannot be recovered at the receiver, given the current policies for the other data units.

The minimization (4) is now simple, since each data unit l can be considered in isolation. Indeed the optimal policy $\pi_l \in \Pi$ for data unit l minimizes the "per data unit" Lagrangian $\epsilon(\pi_l) + \lambda' \rho(\pi_l)$, where $\lambda' = \lambda B_l / S_l^{(n)}$. Thus to minimize (4) for any l and λ', it suffices to know the lower convex hull of the function $\epsilon(\rho) = \min_{\pi \in \Pi} \{\epsilon(\pi) : \rho(\pi) \leq \rho\}$, which we call the *error-cost* function. The error-cost function can be considered as a normalized distortion-rate function pertaining to the transmission of a single dimensionless data unit, and depends only on the transmission scenario and the channel characteristics. The next section investigates the error-cost function for the scenario of receiver-driven transmission over a best-effort network.

4 The Error-Cost Function

First consider the simplified scenario where the receiver transmits only a single request for a data unit, say at time $t_0 < t_{DTS}$, where t_{DTS} is the data unit's delivery deadline. If the request packet is not lost on its way to the sender along the backward path, then the sender transmits the data unit in a data packet along the forward path, incurring a cost of 1 transmitted bytes per source byte. Thus the expected cost is equal to the probability that the request packet is not lost, or $P\{BTT < \infty\}$, using our convention that $BTT = \infty$ if the packet is lost. Further, the expected error is equal to the probability that either the request packet is lost on its way to the sender along the backward path, or the data packet is lost on its way back to the receiver along the forward path, or the round trip is longer than $t_{DTS} - t_0$. The probability that of any of these events occur is $P\{RTT > t_{DTS} - t_0\}$, again using our convention.

Now consider the more complex scenario where the receiver may transmit multiple requests for a data unit. Let $t_0, t_1, \ldots, t_{N-1}$ be N discrete opportunities at which the receiver may transmit a request for a data unit and let $t_{DTS} > t_{N-1}$ be the data unit's delivery deadline. Repeatedly transmitting the data unit at all N opportunities results in a small expected error (equal to $\prod_i P\{RTT > t_{DTS} - t_i\}$) but a large expected cost (equal to $N \cdot P\{BTT < \infty\}$). On the other hand, transmitting the data unit at none of the N opportunities results in a large expected error (equal to 1) but a small expected cost (equal to 0). Intermediate expected errors and costs can also be achieved and are easily computed for any fixed request pattern. For example,



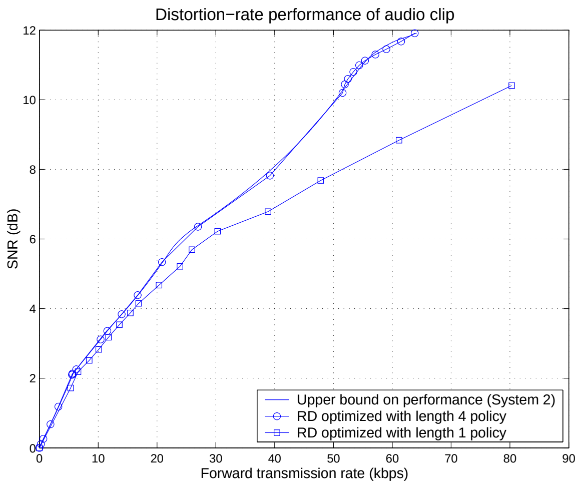


Figure 2: (a) Error-cost function for receiver-driven streaming. (b) Performance of system for policy length = 1 vs. policy length = 4.

suppose $a_0, a_1, \ldots, a_{N-1}$ represents a request pattern where $a_i = 1$ if a data packet is requested at time t_i and $a_i = 0$ otherwise. Then the expected error is equal to $\prod_{i:a_i=1} P\{RTT > t_{DTS} - t_i\}$ while the expected cost is equal to $\sum_{i:a_i=1} P\{BTT < \infty\}$. The latter expression assumes that the receiver does not alter its transmission behavior upon receipt of the data unit.

Finally, consider the most realistic scenario where the receiver may transmit multiple requests for a data unit, and terminates its request pattern upon receipt of the data unit. Then although the expected error remains the same, the expected cost is reduced to $\sum_{i:a_i=1}(\prod_{j<i:a_j=1} P\{RTT > t_i - t_j\})P\{BTT < \infty\}$. To see this, note that the factor $(\prod_{j<i:a_j=1} P\{RTT > t_i - t_j\})$ is the probability that none of the previously transmitted request packets is acknowledged with a data packet by time t_i, which in turn is the probability that a request packet is transmitted at time t_i.

In the latter scenario, the policy π_l for data unit l is identified with the request pattern $a_0, a_1, \ldots, a_{N-1}$ used to request the data unit. Thus if there are N request opportunities for a data unit, then there are 2^N possible policies π_l in the family Π of policies for a data unit, each with its own expected error $\epsilon(\pi_l)$ and expected cost $\rho(\pi_l)$. In this context, N is called the policy length.

Figure 2a shows error-cost performances of all 2^N policies in the last receiver-driven streaming scenario. Here, $N = 8$ and the interval between request opportunities is $T = 50$ ms, so that $t_{i+1} = t_i + T$ for $i = 0, 1, \ldots, N-1$, and $t_{DTS} = t_N$. The channel parameters are $\epsilon_B = \epsilon_F = 10\%$, $n_B = n_F = 2$, $\alpha_B = \alpha_F = 1/(25$ ms$)$, and $\kappa_B = \kappa_F = 50$ ms.

5 Experimental Results

We investigate the overall distortion-rate performance for streaming one minute of packetized audio content using the proposed approach and its variants. The audio content, the first minute of Sarah McLachlan's *Building a Mystery*, is compressed using a scalable version of the Windows Media Audio codec. The codec produces a group of twelve 500-byte data units every 750 ms for a maximum data rate of 64 Kbps. All twelve data units in the gth group receive the same decoding timestamp, equal to $g \times 750$ ms. The channel parameter values given in the previous section are used for simulations. These values yield a mean RTT of 200 ms and a one-way loss rate of 10%, representative of a typical Internet channel. Our simulations use a pseudo-random number generator initialized with the same seed for each system method tested.

Two systems are considered:

System 1: receiver driven rate-distortion optimized streaming. In this system, the rate-distortion optimized streaming algorithm described in the previous sections is applied to streaming of packetized media. The Lagrange multiplier λ is fixed for the entire session. At each request opportunity, the receiver requests data units based on the channel characteristics, the distortion-rate characteristics of the stream, and the decoding timestamps of the data units. The results are averaged over multiple runs to smooth out the effects of particular channel realizations. The performance of various variants of this system are also evaluated.

System 2: An "ideal" rate-distortion optimal system working at channel capacity. This ideal system provides an upper bound on the performance of any rate-distortion optimization based system. The performance of such a system is computed using the rate-distortion characteristics of the stream and the characteristics of the channel. If the sender transmits each data unit an infinite time before its deadline, the channel acts as a binary erasure channel with a drop probability ϵ_F. The capacity of this channel is $C = 1 - \epsilon_F$. Thus, to successfully transmit each source byte, the sender has to transmit at least $1/C$ bytes over the channel. Therefore, transmission of a data unit with a distortion decrement ΔD_l and size B_l requires B_l/C bytes. Knowing ϵ_F, ΔD_l, B_l and the dependence structure for each data unit, the performance of this system can be computed using an optimal pruning algorithm [4, 5].

First, we present experimental results that evaluate the effect of simplifications to and approximations of the proposed approach on its performance. Then, we demonstrate the replay functionality mentioned in Section 1 and compare the performance of receiver-driven streaming to sender-driven streaming.

Effect of short policy length: Our first experiment evaluates the impact of policy length on the performance of the algorithm. Policy length refers to the number of request opportunities, N, that the algorithm looks ahead while computing the policy $\pi_l = \{a_{0,l}, a_{1,l}, \ldots, a_{N,l}\}$ for data unit l. Policy length may be less than the total number of request opportunities available for requesting the data unit before its delivery deadline. This is because at each request opportunity, we permit the algorithm to recompute an optimal length-N policy, given all currently available information for all data units, before taking the first action in the policy, $a_{0,l}$. This is repeated at each request opportunity until the data unit's delivery deadline. Having short policy lengths is beneficial in a practical system since the computational complexity of the algorithm grows exponentially with policy length. For this experiment, the time interval between successive request opportunities is set to $T = 200$ ms and the

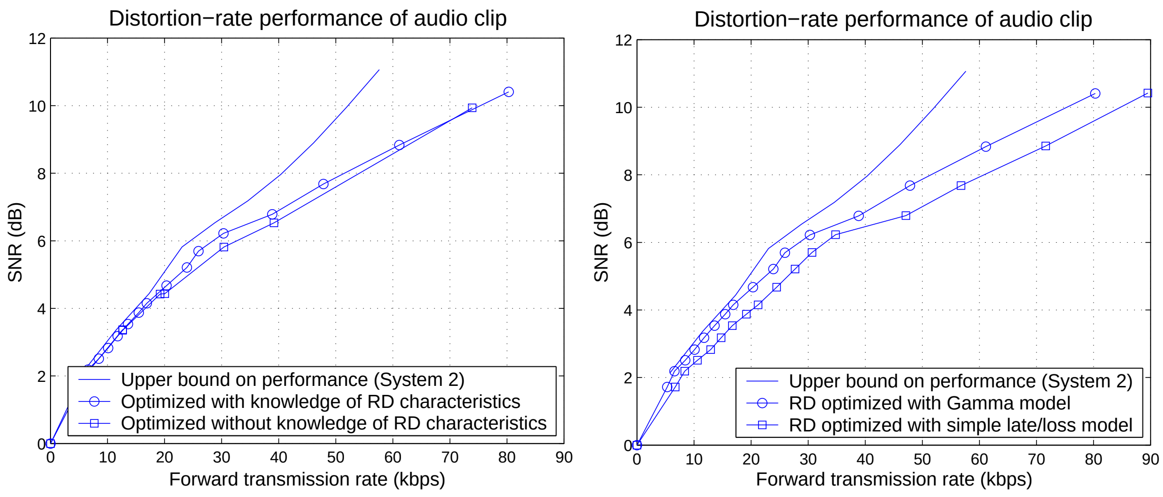
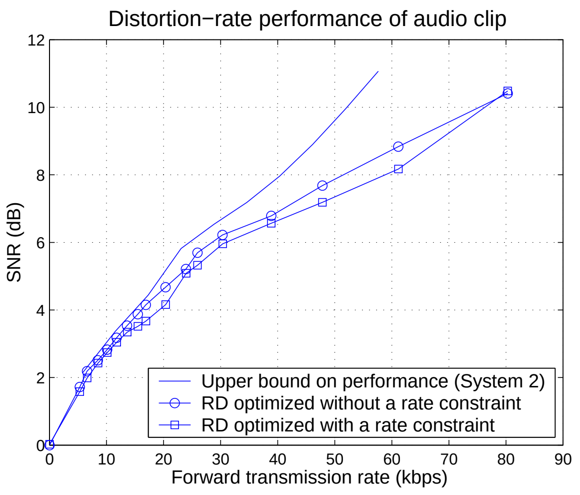


Figure 3: (a) Estimated vs. a priori knowledge of source RD characteristics. (b) Simple late/loss probability model vs. Gamma distribution model.

performances of policies of length four and length one are compared. It is noted that the mean RTT of the channel is 200 ms. Thus, for each data unit, a policy of length four looks ahead four RTTs while making the current decision. Figure 2b compares the performances of System 1 for length four and length one policies. We infer from the figure that at low rates, using a policy of length one results in only a minor degradation in performance (a fraction of a dB) compared to a policy of length four. At high rates, however, the difference in the performance becomes larger. For ensuing experiments, the value of T was set back to 50 ms.

Effect of estimating the rate-distortion characteristics of the source: At each request opportunity, the receiver requests data units based on an optimization procedure that requires knowledge of the distortion decrement ΔD_l and size B_l of each data unit l. In general, this rate-distortion information is not available at the receiver. However, the receiver can predict these quantities using previously received data units. For example, if the source is coded in layers, then the distortion decrements for the data units in all the layers in the current group can be predicted using exponentially weighted moving averages such as $\overline{\Delta D_l} \leftarrow \alpha\overline{\Delta D_l} + (1-\alpha)\Delta D_l$, where α is a weighting factor (set to 0.8 in our experiments). The averages can be initialized during session setup to the distortion decrements for the first group of data units. This procedure can also be used to estimate B_l, though in our experiments the data units have a fixed size (500 bytes) so prediction is not required. Figure 3a compares the performance of System 1 with this estimation procedure to its performance when the receiver has perfect knowledge of the rate-distortion characteristics. The graph shows that estimation leads to only minor degradations in the performance of the system, demonstrating the robustness of the algorithm to small perturbations in its knowledge of the rate-distortion characteristics of the stream.

Effect of a simple late/loss model: In our next experiment, we approximate the late/loss probability model based on the Gamma distribution with a simple late/loss probability model [1, 2]. As shown in [1, 2], this leads to simplifications in the



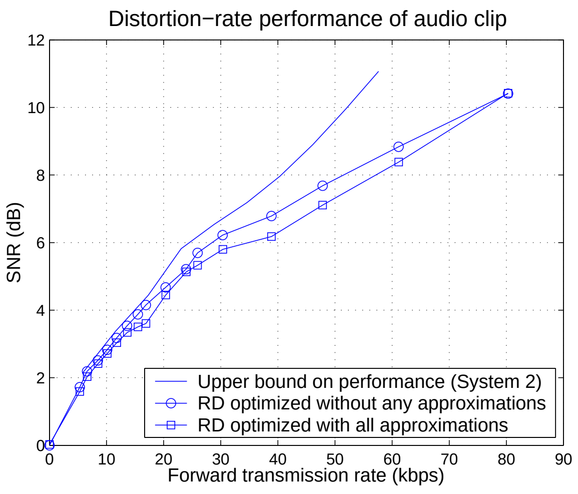


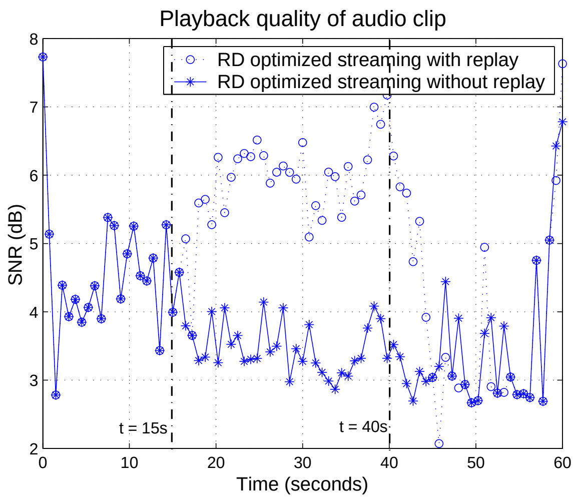
Figure 4: (a) Streaming with and without a rate constraint. (b) Streaming with and without proposed approximations.

optimization algorithm. The simple late/loss model approximates the Gamma distribution with $P\{RTT > \tau\} = 1$ for $\tau < \mu_R$ and as ϵ_R for $\tau \geq \mu_R$ (likewise for $P\{FTT > \tau\}$). Though packets are still dropped according to the Gamma distribution model, the algorithm makes its decisions based on the simple late/loss probability model. Figure 3b compares the performance of System 1 with the simple late/loss model to its performance with the Gamma model. As shown in Figure 3b, performance degrades by only a fraction of a dB with this approximation.

Effect of rate constraint: Next, we evaluate the performance of System 1 with a request rate constraint. As shown in [1, 2], at each available request opportunity, the algorithm can adjust the Lagrange multiplier λ until it selects exactly one data unit to request. To make a comparison between the performance of the rate constrained and the rate unconstrained case, first the ISA algorithm is run with a fixed λ and the total number of requests throughout the session (N) is obtained. To simulate the rate-controlled case, these N requests are uniformly spaced over the duration of the session to obtain an average interval between successive requests. The rate-controlled algorithm [1, 2] is run with this inter-request interval. Figure 4a shows the performance of System 1 in these two cases. As seen from the plots, the presence of a request rate constraint results in only a small penalty in terms of SNR.

Figure 4b evaluates the cumulative effect of using a length one policy, estimation of ΔD_l, a simple late/loss probability model, and rate-control. Comparison is made with a system that does not make these approximations. Figure 4b shows that these approximations lead to minor degradations in the quality of the received stream.

Next, we evaluate the performance of System 1 with replay functionality. For this experiment, the stream was played until $t = 40s$, at which point a request was made to replay the stream from $t = 15s$. Figure 5a plots the per-GOF SNR versus time for the first request and the "replay request" for the stream. During the replay period, the receiver requests only data units that it does not already have, and instead may use the available bandwidth to retrieve enhancement data. This is an idea due to Jim



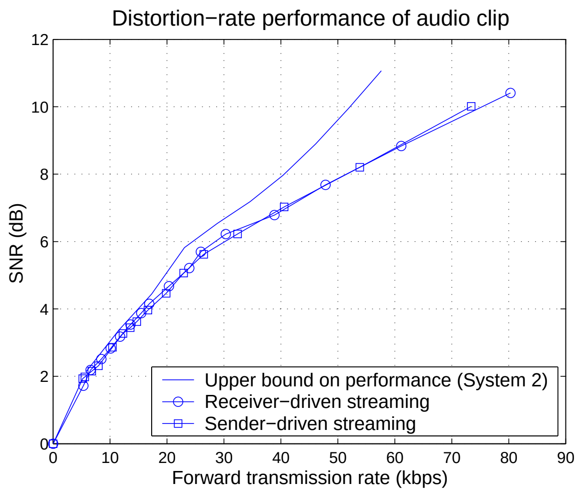


Figure 5: (a) RD optimized streaming with replay. (b) Receiver- vs. sender-driven streaming.

Gemmel [6]. As the figure shows, during the replay period ($t = 15s$ to $t = 40s$), the play back quality for the replay request is strictly better than for the first request.

Finally, Figure 5b compares receiver-driven and sender-driven streaming. The same forward and backward channels are used in both scenarios. As can be seen from the figure, the two systems have almost identical performance. This validates the efficacy of receiver-driven streaming in scenarios where it is advantageous to have distributed computation of transmission policies.

References

[1] P. A. Chou and Z. Miao. Rate-distortion optimized streaming of packetized media. Technical Report MSR-TR-2001-35, Microsoft Research, Redmond, WA, February 2001.

[2] P. A. Chou and Z. Miao. Rate-distortion optimized streaming of packetized media. *IEEE Trans. Multimedia*, 2001. Submitted.

[3] P. A. Chou and Z. Miao. Rate-distortion optimized sender-driven streaming over best-effort networks. In *Proc. Workshop on Multimedia Signal Processing*. IEEE, Cannes, France, October 2001.

[4] Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Trans. Acoustics Speech and Signal Processing*, 36:1445–1453, September 1988.

[5] P. A. Chou, T. Lookabaugh, and R. M. Gray. Optimal pruning with applications to tree structured source coding and modeling. *IEEE Trans. Information Theory*, 35(2):299–315, March 1989.

[6] Jim Gemmel. Layered video idea. Personal communication, March 2000.