Rate-Distortion Optimized Streaming of Packetized Media

Article *in* IEEE Transactions on Multimedia · April 2001

DOI: 10.1109/TMM.2005.864313 · Source: CiteSeer

Philip A. Chou
Google Inc.
226 PUBLICATIONS **13,430** CITATIONS

Zhourong Miao
10 PUBLICATIONS **1,019** CITATIONS

Rate-Distortion Optimized Streaming of Packetized Media

Philip A. Chou and Zhourong Miao
pachou@microsoft.com

February 2001

Technical Report
MSR-TR-2001-35

This paper addresses the problem of streaming packetized media over a lossy packet network, in a rate-distortion optimized way. Out of all the packets that could be transmitted at a given transmission opportunity, we show which packets, if any, to transmit in order to meet a rate constraint while minimizing the end-to-end distortion. Furthermore, if the network supports multiple qualities of service, we show which quality of service to use for each transmitted packet to meet a cost constraint while minimizing the end-to-end distortion. Experimental results show that our system has steady-state gains of 3–7 dB or more over systems that do not have per-packet rate-distortion optimization.

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

1 Introduction

This paper addresses the problem of streaming packetized media over a lossy packet network, in a rate-distortion optimized way. In a streaming media system, a server pre-stores encoded media data and transmits it on demand to a client for playback in real time. The client buffers the data that it receives and begins playback after a short delay of up to several seconds. This delay is fixed and does not depend on the length of the presentation. Once the client begins playback, it is able to continue without interruption until the end of the presentation. It is this continuous playback with fixed delay that distinguishes streaming from download-and-play schemes. Furthermore streaming is distinguished from telephony and conferencing by its ability to store media data encoded off-line, and by its tolerance to a longer playback delay. Streaming, telephony, and conferencing all transmit in real-time; however, for streaming the media data must be encoded without the benefit of knowing the state of the channel during transmission. For this reason, in a streaming media system, the encoding must be flexible and the server must adaptively select and transmit the correct data to the client, as a function of the state of the network as observed by the client or server. In this paper, we show, for arbitrary encodings and packetizations of multiple media, which packets to select for transmission, when to transmit them, and how to transmit them (e.g., with high or low quality of service), such that the rate-distortion performance of the system is optimized. We measure rate as the total cost of bytes transmitted, which is often proportional to the number of bytes transmitted, and we measure distortion as the total end-to-end distortion of the presentation in arbitrary but incrementally additive units.

We set up a general framework for rate-distortion optimized streaming of packetized media, and within it we consider several scenarios. Throughout, we assume that the network loses or corrupts packets at random, and delivers those packets that is does not lose after a random delay. However, the network may or may not have multiple qualities of service (e.g., with different probabilities of loss, corruption, and delay) available at different costs per transmitted byte. Also, the network may or may not allow variations in transmission rate. Finally, the network may or may not provide a back channel, which can be used either for feedback (e.g., acknowledgements) from the receiver in a sender-driven mode, or for control of the sender (e.g., requests for transmission) in a receiver-driven mode. Thus our framework handles a variety of scenarios of current interest: sender-driven or receiver-driven streaming, streaming over best-effort networks such as today's Internet, streaming over multiple overlay networks, streaming over networks with integrated or differentiated services, streaming over combined wireline/wireless networks, and streaming over a network with multiple access to different servers. The same framework has also been shown to handle error control for the case of receiver-driven layered multicast [1, 2, 3, 4].

We present the major ideas in our paper as follows. In Section 2, we show that many encodings and packetizations of multiple media can be abstracted as a single directed acyclic dependency graph, in which each node represents a packetized data unit, and each packet is labeled by a timestamp, a size, and an importance.

In Section 3, we show that any of the aforementioned transmission scenarios can be abstracted as a set of choices for sending single packets, in which each choice π is associated with a cost per byte $\rho(\pi)$ of transmitting the packet and an error probability $\epsilon(\pi)$ of not delivering the packet by its deadline. This leads to the concept of an error-cost function $\epsilon(\rho) = \min_\pi\{\epsilon(\pi) : \rho(\pi) \leq \rho\}$, for which optimal performance (for a single packet) can be achieved by selecting the transmission option π minimizing the Lagrangian $\epsilon(\pi) + \lambda'\rho(\pi)$ for some Lagrange multiplier λ'. For scenarios involving a back channel, we identify the transmission options as policies in a Markov decision process and associate Lagrangians with each policy. We show how to use dynamic programming to find a policy with the mimimal Lagrangian.

In Section 4 we show how to relate the error-cost functions for the packets to the distortion-rate function for the entire multimedia presentation. An optimal distortion-rate performance $D(R)$ for the entire presentation can be achieved by minimizing $D + \lambda R$ for some Lagrange multiplier λ. In turn, $D + \lambda R$ can be minimized by individually minimizing the packet Lagrangians $\epsilon(\pi) + \lambda'\rho(\pi)$ for appropriately chosen Lagrange multipliers λ'. The Lagrange multipliers λ' ultimately depend, cyclically, on the error probabilities $\epsilon(\pi)$. However, we develop an iterative descent algorithm for finding solutions that are locally optimal.

In Sections 5 and 6, we show how to control the window of transmission opportunities for each packet, and how to control the bit rate of the transmission at a packet level, in response to congestion and flow control mechanisms. We also sketch a possible protocol for sender-driven streaming over a best-effort network.

In Section 7 we report experimental results focusing on sender-driven streaming over a best-effort network. Using simulations, we show that our system gains up to 4 dB or more over systems approximating state-of-the-art commercial systems, over networks with 20% packet loss.

To our knowledge, the most closely related contemporaneous work is that by Miao and Ortega [5, 6], which develops a low-complexity heuristic algorithm for sender-driven scheduling of packet transmissions over a best-effort network. Zhou and Li [7] also develop similar heuristics.

The most closely related rigorous work is that by Podolksy, McCanne, and Vetterli [8, 9], which uses a Markov chain analysis to find the optimal policy for transmitting layered media at a fixed rate, including retransmissions, to minimize the end-to-end distortion. To make the analysis tractable, Podolsky et al. assume zero transmission delay and loss-free acknowledgements. Unfortunately they are unable to simulate an optimal system with more than a few source layers and more than a few transmission opportunities per frame, since the space of policies grows exponentially in both of these quantities. One of the main contributions of our paper is showing that this policy space can be factored so that the layers are only loosely coupled, resulting in complexity that grows roughly linearly in the number of layers.

The work of Chande, Jafakhani, and Farvardin [10] was the first that we know of to formulate and solve the problem of optimal transmission over a noisy channel in the presense of feedback, using a Markov decision process framework. The work of Servetto [11] also recognized that optimal transmission over a noisy channel in the presense of feedback is a nonlinear stochastic control problem. Inspired by these works, Chou et al. [1, 2, 3, 4] used an iterative descent algorithm in a Lagrangian framework to find locally optimal transmission policies for hybrid FEC/ARQ, assuming jitter-free delay and loss-free retransmission requests, when the source layers are given by arbitrary directed acyclic graphs. Chou et al. applied their work to receiver-driven layered multicast of audio and video. That work became the starting point for the present paper, when we realized that the same methodology could be used to solve the problem of Podolsky et al. in a practical way.

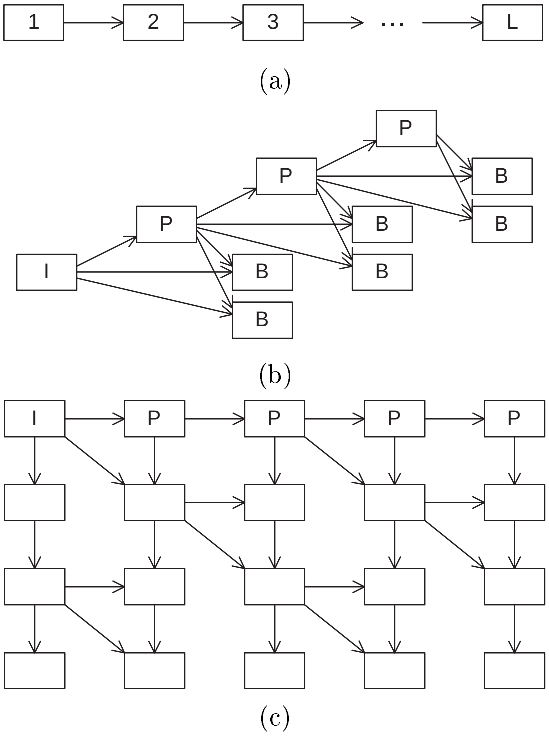
There have been numerous other papers that perform some kind of rate-distortion optimization for transmission of packetized media. Many have focused on the problem of source rate control in the absence of transmission errors [12, 13, 14, 15]. (Works that address the problem of source rate control in the presense of transmission errors, but are not rate-distortion optimized, include [16, 17, 18, 19, 20, 21].) Other works have focused on the problem of error control using forward error correction (FEC). Many of these use the priority encoding transmission (PET) technique of Albanese et al. [22, 23, 24, 25, 26, 27, 28, 29, 30, 31], which can be rate-distortion optimized using the algorithms of [23, 28, 32]. Others of these use a systematic rate-compatible technique [1, 2, 3, 4], which can be rate-distortion optimized using the algorithms of [1, 2, 3, 4, 33, 34, 35]. The present paper is an extension of these latter methods. (Still others use "signal processing FEC" for error control [36, 37, 38, 39, 40], for which rate-distortion optimization is still a research topic [41].) Some papers have investigated the problem of error control using retransmission-based protocols, e.g., [42, 43, 44, 45, 46, 47, 48]. However, with the exception of those works listed in the previous paragraph, to our knowledge, none are rate-distortion optimized. Finally, a few papers suggest the use of multiple qualities of service (e.g., diffserv) to support more cost-effective media transmission at a higher quality [49, 50, 51]. Of these, only [49] attempts to optimize the distortion subject to transmission rate constraints. Our paper substantially furthers the work in this direction.

2 Preliminaries

In this section we define our abstractions of the encoding, packetization, and communication processes, and state the distortion-rate optimization problem that we are trying to solve.

In a streaming media system, the encoded data are packetized into *data units* and are stored in a file on a media server. If the server selects a data unit for transmission, the data unit is put into a packet and sent across the network. If the packet is lost, the data unit may be sent again in another packet. In general we assume a one-to-many correspondence between data units and packets. However, each packet must contain one and only one data unit.

Regardless of how many media objects (audio, video, etc.) there are in a



(a)

(b)

(c)

Figure 1: Directed acyclic dependency graphs. (a) Sequential dependencies typical of embedded codes. (b) Dependencies between IBBPBBPBBP video frames. (c) Typical dependencies for MPEG-4 progressive fine grain scalability mode.

multimedia presentation, and regardless of what algorithms are used for encoding and packetizing those media objects, the result is a set of data units for the presentation whose interdependencies can be expressed by a directed acyclic graph. 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 corresponds to a dependence of data unit l on data unit l'. That is to say, in order for data unit l to be decoded, data unit l' must also be decoded. This induces a partial order between data units, for which we write $l' \prec l$ if l' is an ancestor of l (or equivalently if l is a descendant of l'). Thus, if a set of data units is received by the client, only those data units whose ancestors have all been also received can be decoded.

Typically, the graph of dependencies between *all* of the data units in a presentation is a collection of connected components, where each connected component is itself a directed acyclic graph representing the dependencies between all of the data units of an independently encoded and packetized group of frames (GOF) of one media type. Some such directed acyclic dependency graphs are illustrated in Figure 1. Figure 1a shows a dependence graph typical of an embedded encoding of a group of frames, which is packetized sequentially. Figure 1b shows a dependence graph typical of an encoding by a standard video coder of a group of (IBBPBBPBBP) frames, which is packetized as one data unit per frame. And Figure 1c shows a dependence graph typical of an encoding of a group of frames by MPEG-4's fine grain scalability (FGS) mode [52, 53], which is again packetized as one data unit per frame.

Labeling each data unit l in the dependence graph are three constant quantities: its data unit size B_l in bytes, its importance Δd_l in units of distortion, and its timestamp $t_{DTS,l}$. We now discuss each of these in turn.

The data unit size B_l is the number of source bytes in the data unit.

The importance Δd_l is the amount by which the distortion at the receiver will *decrease* if the data unit is *decoded* (on time) at the receiver. Recall that a data unit can be decoded only if all of the data units on which it depends can also be decoded. For example, in Figure 1a, Δd_l for the third data unit in the sequence is the decrease in distortion if three data units are decoded instead of only two. Similarly, in Figure 1b, Δd_l for a B frame is the decrease in distortion if the B frame is decoded, compared to the distortion if the B frame is not decoded. In this way, the overall distortion can be computed as the initial distortion d_0 (i.e., the distortion if no data units are decoded) less the sum of the decreases Δd_l over all data units l that have been decoded on time. We say the distortion is *incrementally additive* with respect to the partial order given by the dependency graph. An important limitation of this incrementally additive model is that the amount by which the distortion decreases when a data unit is decoded does not depend on whether its sibling or cousin data units are decoded. So for example, in this model, the decrease in distortion when a B frame is decoded does not depend on whether or not any other B frame is decoded. Strictly speaking, this rules out a number of error concealment techniques. Fortunately, the incremental additivity model provides a good approximation to reality even in those cases where it is not exact.

The timestamp $t_{DTS,l}$ is the time by which the data unit must be decoded to be useful (i.e., for the distortion to decrease by Δd_l). This corresponds to the decoder timestamp (DTS) in MPEG terminology, and represents the time at which the decoder extracts the data from its input buffer prior to presentation (which in turn occurs at the presentation timestamp, PTS). Thus, 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 usefully decoded. Packets containing a data unit that arrive after the data unit's delivery deadline are discarded.

Each data unit l is also implicitly labeled by a set of $N = N_l$ transmission opportunities $t_{0,l}, t_{1,l}, \ldots, t_{N-1,l}$ prior to $t_{DTS,l}$ at which the data unit may be put into a packet and transmitted. Often this set of transmission opportunities is a single time $t_{0,l}$ (such as a "send time") 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 $[t_{lag}, t_{lead}]$) prior to the delivery deadline. Determination of this set is addressed in Section 5.

It pays to be careful about the temporal coordinate systems (or clocks) in which time is expressed. In this paper, we deal with three different temporal coordinate systems: the media (or encoder) temporal coordinate system t, the sender (or server) temporal coordinate system s, and the receiver (or client) temporal coordinate system r. Each of these is related to the other by an affine coordinate transformation. For example,

$$r = r_{start} + \delta + (t - t_{DTS0})/\nu = A_{t\to r}(t)$$

is the coordinate transformation that maps media time t into receiver time $r = A_{t\to r}(t)$. Here, r_{start} is the time (on the receiver's clock) that the user effectively pushes the "play" button; δ is the delay between r_{start} and the moment that the first data unit is due to be removed from the client buffer and decoded; t_{DTS0} is the decoder timestamp of the first data unit to be decoded; and ν is the desired playback speed. Likewise,

$$s = s_{start} + (r - r_{start}) = A_{r\to s}(r)$$

is the coordinate transformation that maps receiver time r into sender time $s = A_{r\to s}(r)$. We assume that the sender and receiver clocks run at the same rate, but differ only by a constant offset $s_{start} - r_{start}$, determination of which effectively synchronizes the clocks. We use the notation t_X, s_X, and r_X to denote the time of a single event X in each of the three temporal coordinate systems.

We model the network as an independent time-invariant packet erasure channel with random delays. That means that if the sender inserts a packet into the network at sender time s, then the packet is lost with some probability, say ϵ_F, independent of s. However, if the packet is not lost, then it arrives at the receiver at sender time s', where the forward trip time $FTT = s' - s$ is randomly drawn according to probability density $p_F(\tau|\text{not lost})$. Each packet is lost or delayed independently of the other packets. This independence and time-invariance is reasonable over short time scales. In fact, digital transmission systems are often modeled by hidden Markov models, in which losses are independent given the current state (e.g., "congested" or "not congested") [54]. The current state may change slowly over time. Here, we assume that the loss probability ϵ_F and the delay density $p_F(\tau|\text{not lost})$ can indeed vary slowly over time, depending on the underlying state of the network, which is typically estimated anyway for the purposes of congestion control. (See Section 6.)

For convenience, we combine the packet loss probability and the packet delay density into a single probability measure, by assigning $FTT = \infty$ in the event that the packet is lost. This places mass ϵ_F at infinity, and weights the density $p_F(\tau|\text{not lost})$ by the factor $(1 - \epsilon_F)$, as illustrated in the upper left corner of Figure 2. Thus the cumulative distribution function for the forward trip time is

$$P\{FTT \le \tau\} = \int_0^\tau (1 - \epsilon_F) p_F(t|\text{not lost}) dt,$$

and its complement is

$$P\{FTT > \tau\} = \epsilon_F + \int_\tau^\infty (1 - \epsilon_F) p_F(t|\text{not lost}) dt,$$

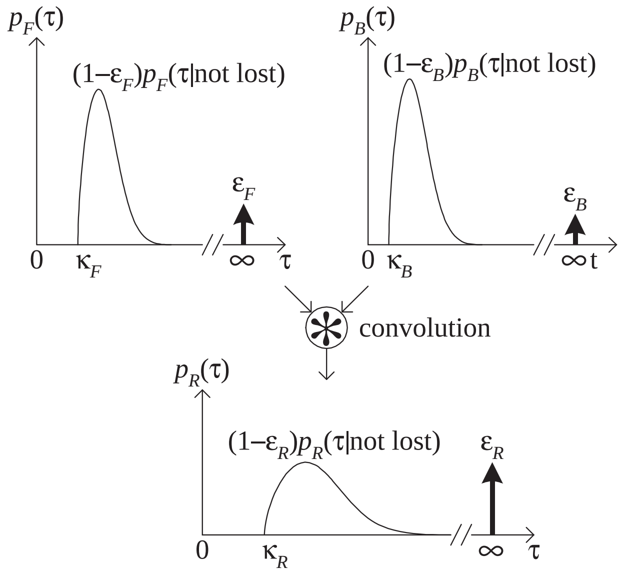


Figure 2: Density for round trip time as a convolution of densities for forward and backward trip times.

which is the probability that a packet sent at time s is not received by time $s+\tau$, whether lost or simply delayed. In principle it is not ever possible to determine by waiting for a packet whether it is lost, or just delayed for a very long time.

We assume that the back channel, if available, can be similarly characterized. If the client sends a packet to the server, then the packet is lost with probability ϵ_B, otherwise it is delayed according to density $p_B(\tau|\text{not lost})$. The probability that the backward trip time is greater than τ is

$$P\{BTT > \tau\} = \epsilon_B + \int_\tau^\infty (1-\epsilon_B)p_B(t|\text{not lost})dt.$$

The round trip time $RTT = FTT + BTT$ is by definition the sum of forward and backward trip times. The probability that the round trip time is greater than τ is therefore the integral of the convolution

$$\begin{aligned} P\{RTT > \tau\} &= \epsilon_F + (1-\epsilon_F)\epsilon_B + (1-\epsilon_F)(1-\epsilon_B)\times \\ &\int_\tau^\infty \int_0^t p_F(t'|\text{not lost})p_B(t-t'|\text{not lost})dt'dt, \end{aligned} \tag{1}$$

which is the integral from τ to infinity of the (unconditional) "density" $p_R(\tau) = p_F(\tau) * p_B(\tau)$ shown in Figure 2. Of course there are a number of alternative ways to compute (1), e.g., as the volume of the joint density $p_F(\tau_F)p_B(\tau_B)$ over the area shown in Figure 3.

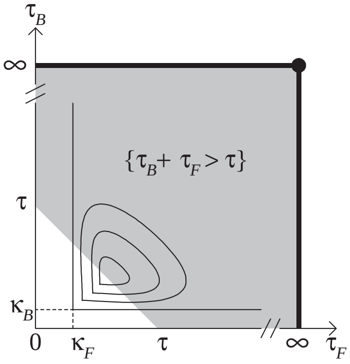


Figure 3: Region for (τ_F, τ_B) over which $\tau_F + \tau_B > \tau$.

We do not assume any particular form for the densities $p_F(\tau|\text{not lost})$, $p_B(\tau|\text{not lost})$, or $p_R(\tau|\text{not lost})$. However, for concreteness in the next section and in Section 7 (Experimental Results) we do model these distributions parametrically. As in [55], we model packet delay as having a shifted Gamma distribution with rightward shift κ and parameters n and α, e.g.,

$$p_F(\tau|\text{not lost}) = \frac{\alpha_F}{\Gamma(n_F)}(\alpha(\tau - \kappa_F))^{n_F - 1} e^{-\alpha_F(\tau - \kappa_F)} \tag{2}$$

for $\tau \geq \kappa_F$. This is the distribution of a random variable that is equal to a constant κ_F plus the sum of n_F independent identically distributed exponential random variables each with parameter α_F [56]. One way to interpret this is that the forward trip time FTT is the result of a packet going through n_F routers, each of which requires a constant processing time κ_F/n_F plus waiting time in a steady state M/M/1 queue [57]. Since an exponential random variable with parameter α has mean $1/\alpha$ and variance $1/\alpha^2$, the forward trip time as modeled by (2) has mean $\mu_F = \kappa_F + n_F/\alpha_F$ and variance $\sigma_F^2 = n_F/\alpha_F^2$. Hence

$$n_F = (\mu_F - \kappa_F)\alpha_F, \tag{3}$$

$$\alpha_F = (\mu_F - \kappa_F)/\sigma_F^2. \tag{4}$$

In this way the parameters n_F and α_F can be determined from estimates of μ_F and σ_F^2, as we consider in Section 7. A similar approach can be taken for backward and round trip times.

We end this section with a discussion of our objective: rate-distortion optimized streaming of packetized media. By *rate* we mean the expected cost R of streaming an entire presentation. Cost may be measured as the number of bytes transmitted. However it can also be measured more generically. As we mentioned in the Introduction, each data unit is sent with some transmission option π, which has cost per source byte $\rho(\pi)$ and hence a data unit cost $B\rho(\pi)$, where B is the size of the data unit in bytes. The cost of streaming the entire presentation is therefore the sum of the data unit costs for all of the data

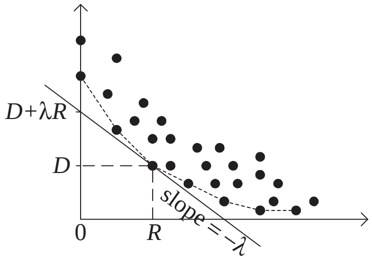


Figure 4: The set of achievable rate-distortion pairs, its lower convex hull (dotted), and an achievable pair (R, D) minimizing the Lagrangian $D + \lambda R$. Each dot is the (R, D) performance of some algorithm (caricature only).

units transmitted. The rate R is the expected value of this total cost, averaged over all possible realizations of the random channel, when a given streaming algorithm is in use.

By *distortion* we mean the expected distortion D incurred for the entire presentation. As we mentioned earlier in this section, whenever a data unit is decoded on time at the receiver, the distortion decreases (from some initial large distortion d_0) by the importance Δd of the data unit. The distortion incurred for the entire presentation is therefore some initial large distortion d_0 less the sum of the importances for all the data units decoded on time. The distortion D is the expected value of this total distortion, averaged over all possible realizations of the random channel, when a given streaming algorithm is in use.

We seek a streaming algorithm that has the minimum possible expected distortion D for its expected rate R. By restricting ourselves to algorithms whose rate-distortion performance (R, D) lies on the lower convex hull of the set of all rate-distortion performances achievable by some algorithm, as illustrated in Figure 4, we can find an optimal algorithm with expected distortion D and expected rate R by minimizing the Lagrangian $D + \lambda R$ for some positive Lagrange multiplier λ. Finding such an algorithm, under various communication scenarios, is the objective of this paper.

3 Transmitting a Single Data Unit

In this section we study the problem of optimally transmitting a single data unit. Knowing whether, when, and how to best transmit each data unit in isolation will lead (in Section 4) to optimal transmission of the entire presentation.

When considering transmission of only a single data unit, the distortion-rate measures can be normalized. Rather than measuring distortion in terms of, say, squared error, expected distortion can be measured as the "error probability" or more precisely the late/loss probability, that is, the probability that the data unit does not arrive at its destination on time. Rather than measuring rate in

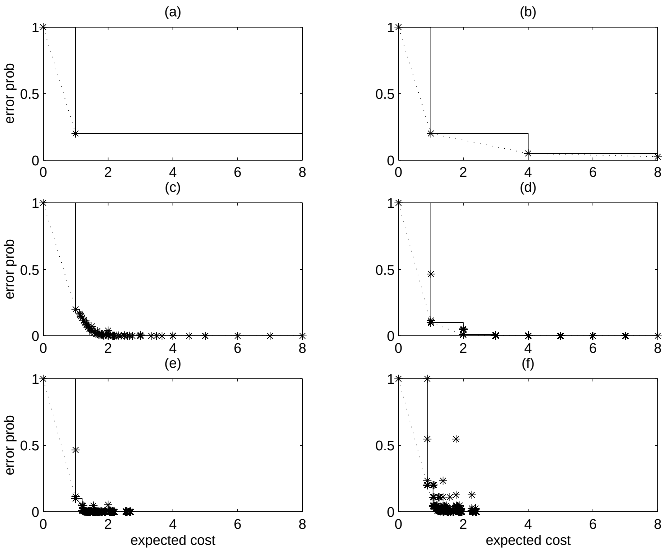


Figure 5: Error-cost functions. (a) Single QoS, no feedback. (b) Multiple QoS, no feedback. (c) FEC, no feedback. (d) Retransmission, no feedback. (e) Sender-driven retransmission, with feedback. (f) Receiver-driven retransmission, with feedback.

terms of, say, bytes per second, rate can be measured as the expected number of times the data unit is transmitted, or more generally, as the expected number of bytes transmitted per source byte, or more generally still, as the expected cost per source byte to transmit the data unit. We refer to these normalized distortion-rate measures as "error-cost" measures.

At each of the transmission opportunities associated with a data unit, the data unit may be transmitted or not. If it is transmitted then depending on the scenario there may be more than one mode in which to transmit the data unit.

The simplest scenario is the following. A data unit with delivery deadline s_{DTS} can either be transmitted at time $s_0 < s_{DTS}$, or not, over a network with a single quality of service (QoS) e.g., a best-effort network. In this scenario, there are two error-cost possibilities. If the data unit is not transmitted, then the error probability is one, while the cost per source byte (expected number of packet transmissions) is zero. On the other hand, if the data unit is transmitted, then the error probability is $\epsilon \equiv P\{FTT > s_{DTS} - s_0\}$, while the cost per source byte is one. The operational error-cost function for this scenario and its convex hull are illustrated in Figure 5a for $\epsilon = 20\%$.

A more interesting scenario is the following. Suppose that there are multiple qualities of service available on the network (or equivalently suppose there are multiple networks) each with its own forward trip time distribution and its own marginal cost per transmitted byte. For example, best-effort service with forward trip time $FTT^{(1)}$ could cost $\rho^{(1)} = 1$ microcent per transmitted byte, a low-loss service with forward trip time $FTT^{(2)}$ could cost $\rho^{(2)} = 4$ microcents per transmitted byte, and a low-delay, low-loss service with forward trip time $FTT^{(3)}$ could cost $\rho^{(3)} = 8$ microcents per transmitted byte. Then a data unit with delivery deadline s_{DTS} transmitted (or not) at time s_0 has one of four distortion-rate possibilities: $\delta^{(0)} = 1$ and $\rho^{(0)} = 0$; $\delta^{(1)} = P\{FTT^{(1)} > s_{DTS} - s_0\}$ and $\rho^{(1)} = 1$; $\delta^{(2)} = P\{FTT^{(2)} > s_{DTS} - s_0\}$ and $\rho^{(2)} = 4$; and $\delta^{(3)} = P\{FTT^{(3)} > s_{DTS} - s_0\}$ and $\rho^{(3)} = 8$. The operational error-cost function for this scenario and its convex hull are illustrated in Figure 5b.

Another scenario is one in which only a single best-effort network is available, but the application emulates different qualities of service over this network using forward error correction schemes of different strengths. For instance, if $k \geq 1$ and $n \geq k$, the application can emulate a higher quality of service for a data unit transmitted at time s_0 by 1) grouping it together with $k-1$ other data units also to be transmitted at time s_0, 2) applying an (n, k) systematic Reed-Solomon code to produce $n-k$ parity units, and 3) transmitting the data packets plus their parity packets at time s_0. The original data unit cannot be recovered at the receiver by time s_{DTS} only if it is late or lost (which happens with probability $\epsilon = P\{FTT > s_{DTS} - s_0\}$) *and* at least $n-k$ of the other $n-1$ packets are also late or lost (which happens with probability $f_{n,k} = \sum_{i=n-k}^{n-1} \binom{n-1}{i} \epsilon^i (1-\epsilon)^{n-1-i}$). Thus the loss/late probability is reduced by a factor $f_{n,k}$ over best-effort, at a cost of n/k transmitted bytes per source byte. The error-cost performances for various values of (n, k) can be plotted (e.g., for $k = 1, \ldots, 8$ and $n = k, \ldots, k+8$) to produce an operational error-cost function, as illustrated in Figure 5c.

In a similar scenario, quality of service can be emulated using retransmissions. Let $s_0, s_1, \ldots, s_{N-1}$ be N discrete transmission opportunities and let s_{DTS} be the delivery deadline. Repeatedly transmitting the data unit at all N opportunities results in a small loss/late probability (equal to $\prod_i P\{FTT > s_{DTS} - s_i\}$) but a large cost (equal to N). On the other hand transmitting the data unit at none of the N opportunities results in a large loss/late probability (equal to 1) but a small cost (equal to 0). Intermediate loss/late probabilities and costs can also be achieved and easily computed for any fixed transmission pattern. For example, suppose $a_0, a_1, \ldots, a_{N-1}$ represents a transmission pattern where $a_i = 1$ if a data packet is transmitted at time s_i and $a_i = 0$ otherwise. Then the loss/late probability is equal to $\prod_{i:a_i=1} P\{FTT > s_{DTS} - s_i\}$ while the cost is equal to $\sum_{i:a_i=1} 1$ transmitted bytes per source byte. The error-cost performances for all 2^N transmission patterns can be plotted to produce an operational error-cost function, as illustrated in Figure 5d.

This scenario becomes more realistic when combined with feedback. Suppose the receiver sends an acknowledgement packet back to the sender the instant that it receives a data packet, and that the sender truncates its transmission pattern upon receipt of the acknowledgement packet. Then although the loss/late probability remains the same, the expected number of data packet transmissions is reduced to $\sum_{i:a_i=1}(\prod_{j<i:a_j=1} P\{RTT > s_i - s_j\})$.[1] This scenario, which we refer to as sender-driven transmission over a single-QoS network using retransmissions with feedback, is the principal scenario considered in this paper. The operational error-cost function for this scenario is illustrated in Figure 5e.

A receiver-driven version of the above scenario is the following. The receiver initiates transmission by sending a request packet to the sender; the sender responds by sending a data packet to the receiver. Let $r_0, r_1, \ldots, r_{N-1}$ be N discrete request opportunities at which the receiver can transmit a request packet, and let r_{DTS} be the deadline for delivery of the data unit to the receiver. Suppose $a_0, a_1, \ldots, a_{N-1}$ represents a request pattern where $a_i = 1$ if a request packet is transmitted at time r_i and $a_i = 0$ otherwise. Suppose the sender transmits the data packet to the receiver the instant that it receives a request, and that the receiver truncates its request pattern upon receipt of the data unit. Then it is not too hard to show that the loss/late probability is equal to $\prod_{i:a_i=1} P\{RTT > r_{DTS} - r_i\}$ (which is larger than in the sender-driven case) while the expected data packet transmission rate is equal to $\sum_{i:a_i=1}(\prod_{j<i:a_j=1} P\{RTT > r_i - r_j\}) P\{BTT < \infty\}$ (which is smaller than in the sender-driven case). The operational error-cost function is illustrated in Figure 5f.

Hybrids of any of the above scenarios are also possible. For example, receiver-driven transmission over a multiple-QoS network using retransmissions with feedback can be handled by letting $a_i = Q_i \in \{1, 2, 3\}$ indicate a request by the receiver at time r_i for a data unit to be transmitted with quality of service Q_i (and $a_i = 0$ otherwise). In this case, the loss/late probability is equal to $\prod_{i:a_i \neq 0} P\{RTT^{(Q_i)} > r_{DTS} - r_i\}$, and the cost is equal to $\sum_{i:a_i \neq 0} \rho^{(Q_i)}(\prod_{j<i:a_j \neq 0} P\{RTT^{(Q_j)} > r_i - r_j\}) P\{BTT < \infty\}$, where $RTT^{(Q)} = BTT + FTT^{(Q)}$ is the round trip time over the single backward channel and the Q'th forward channel. This last "intserv/diffserv" scenario is mathematically equivalent to a number of other scenarios of interest, such as the "overlay" scenario in which the receiver is connected to the sender by multiple physical networks, each offering a different quality of service, and the "multiple access" scenario in which the receiver is connected to multiple senders over different network paths, each offering a different quality of service.

In the remainder of this section, we study in detail, using a Markov decision process framework, the scenario of sender-driven transmission over a single-QoS network using retransmissions with feedback. At the end of this section, we show how the Markov decision process framework allows further generalizations to other scenarios, such as a wireless scenario in which the receiver can respond to a corrupted packet with a negative acknowledgement (NAK), as well as respond to a cleanly received packet with a positive acknowledgement (ACK).

A Markov decision process with finite horizon N is an N-step stochastic process through a state space in which an action can be taken at each state in

[1] To see this, consider that the quantity in parentheses is the expected value of the indicator function of the event that a data packet is transmitted at time s_i, which in turn is the probability that none of the previously transmitted packets are acknowledged by time s_i.

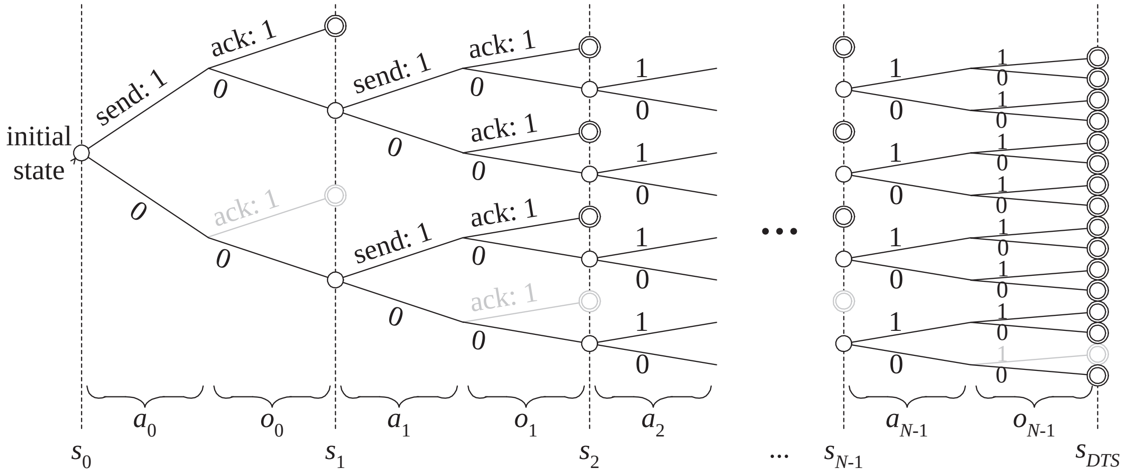


Figure 6: Trellis for a Markov decision process. Final states are indicated with double circles.

a corresponding trellis of length N to influence the outgoing transition probabilities and thereby maximize an expected reward or minimize an expected cost along the transitions. The assignment of actions to trellis states is called a *policy* (denoted by π) and the optimal policy, in our context, is the one that minimizes the expected cost $\epsilon_\pi + \lambda'\rho_\pi$ of traversing the trellis in N steps starting from a known initial state.

Figure 6 shows the trellis for the Markov decision process associated with the problem of sender-driven transmission over a single-QoS network using retransmissions with feedback. The process begins in the initial state at time s_0. In this state, the sender can choose either to send the data unit, taking action $a_0 = 1$, or not to send the data unit, taking action $a_0 = 0$. If the sender chooses to send the data unit, then just prior to time s_1, the sender can observe either that some packet containing the data unit has been acknowledged, in which case $o_0 = 1$, or that no packet has been acknowledged, in which case $o_0 = 0$. If a packet containing the data unit has been acknowledged by time s_1, then the process enters a final state at time s_1. Otherwise the process enters a non-final state at time s_1, and the sender can once again choose either to send the data unit, or not, repeating the process up to a total of N times.

Each state in the trellis (circles in Figure 6) captures the action-observation history leading up to that state from the initial state. That is, a state q_i at time s_i represents a sequence of i action-observation pairs, $(a_0, o_0) \circ (a_1, o_1) \circ \ldots \circ (a_{i-1}, o_{i-1})$.

The action taken at a state determines the transition probabilities to the next state. If the next state $q_{i+1} = q_i \circ (a_i, o_i)$ is the current state q_i followed by the action-observation pair (a_i, o_i), then (in this scenario)

$$P(q_{i+1}|q_i, a_i) = \begin{cases} \prod_{j \le i : a_j = 1} P\{RTT > s_{i+1} - s_j | RTT > s_i - s_j\} \\ \qquad \text{if } o_i = 0 \text{ (not ACK'd)}, \\ 1 - \prod_{j \le i : a_j = 1} P\{RTT > s_{i+1} - s_j | RTT > s_i - s_j\} \\ \qquad \text{if } o_i = 1 \text{ (ACK'd)}. \end{cases}$$

That is, the probability that no acknowledgement arrives by time s_{i+1} given that no acknowledgement arrived by time s_i is the product of the probabilities for each data packet sent at time s_j that no acknowledgement arrives by time s_{i+1} (i.e., $RTT > s_{i+1} - s_j$) given that no acknowledgement arrived by time s_i (i.e., $RTT > s_i - s_j$). Note that these conditional probabilities can be easily calculated in terms of known unconditional probabilities, since by Bayes' rule and the fact that $RTT > \tau + \delta$ implies $RTT > \tau$ (for $\delta \ge 0$), $P\{RTT > \tau + \delta | RTT > \tau\} = P\{RTT > \tau + \delta\} / P\{RTT > \tau\}$.

Now, any policy $\pi : q \mapsto a$ assigning actions to states induces a Markov chain with transition probabilities

$$P_\pi(q_{i+1}|q_i) \equiv P(q_{i+1}|q_i, \pi(q_i)).$$

Let $\mathcal{Q}_\pi$ be the set of all complete paths through this Markov chain, and let $\boldsymbol{q} = (q_0, q_1, \ldots, q_F) \in \mathcal{Q}_\pi$. That is, let $\boldsymbol{q}$ satisfy $q_{i+1} = q_i \circ (a_i, o_i)$ where $a_i = \pi(q_i)$ and $o_i = 0$ for $i < F - 1$. Then $\boldsymbol{q}$ has probability

$$P_\pi(\boldsymbol{q}) = \prod_{i=0}^{F-1} P_\pi(q_{i+1}|q_i),$$

transmission cost

$$\rho_\pi(\boldsymbol{q}) = \sum_{i=0}^{F-1} a_i,$$

and error (loss/late probability)

$$\epsilon_\pi(\boldsymbol{q}) = \begin{cases} 0 \\ \qquad \text{if } o_{F-1} = 1 \text{ (ACK'd)}, \\ \prod_{j : a_j = 1} P\{FTT > s_{DTS} - s_j | RTT > s_{DTS} - s_j\} \\ \qquad \text{otherwise}. \end{cases}$$

The latter expression follows from the facts that if the path leads to an acknowledgement, then the probability that the data packet is lost or late is zero, while if the path leads to no acknowledgement by time s_{DTS}, then the probability that the data packet is lost or late is the product of the probabilities that each data packet transmitted is lost or late ($FTT > s_{DTS} - s_j$) given that no acknowledgement is received for that packet ($RTT > s_{DTS} - s_j$). Note that again these conditional probabilities can be easily calculated in terms of known unconditional probabilities, since by Bayes' rule and the fact that $FTT > \tau$ implies $RTT > \tau$, $P\{FTT > \tau | RTT > \tau\} = P\{FTT > \tau\}/P\{RTT > \tau\}$.

Armed with definitions of probability, transmission cost, and error for each path, one can now express the expected cost and error for the Markov chain induced by policy π:

$$\rho_\pi \equiv E_\pi \rho_\pi(\boldsymbol{q}) = \sum_{\boldsymbol{q} \in \mathcal{Q}_\pi} P_\pi(\boldsymbol{q}) \rho_\pi(\boldsymbol{q})$$

$$\epsilon_\pi \equiv E_\pi \epsilon_\pi(\boldsymbol{q}) = \sum_{\boldsymbol{q} \in \mathcal{Q}_\pi} P_\pi(\boldsymbol{q}) \epsilon_\pi(\boldsymbol{q}).$$

In principle it is possible to enumerate all possible policies π, plot the error-cost performances $\{(\rho_\pi, \epsilon_\pi)\}$ in the error-cost plane, and produce an operational error-cost function for this scenario as we did in Figure 5e. Unfortunately, in general it may not be feasible to enumerate all possible policies. However if one is only interested in finding a point on the convex hull of the operational error-cost function, it is relatively simple matter to find the policy minimizing the expected Lagrangian

$$J_\pi \equiv \epsilon_\pi + \lambda' \rho_\pi = \sum_{\boldsymbol{q} \in \mathcal{Q}_\pi} P_\pi(\boldsymbol{q}) J_\pi(\boldsymbol{q}), \tag{5}$$

where $J_\pi(\boldsymbol{q}) \equiv \epsilon_\pi(\boldsymbol{q}) + \lambda' \rho_\pi(\boldsymbol{q})$. This can be accomplished with dynamic programming by extending the domain of definition of $J_\pi(\boldsymbol{q})$ not only to complete paths through the trellis (under π) but also to partial paths. Let

$$J_\pi(q_i) = \begin{cases} \epsilon_\pi(\boldsymbol{q}) + \lambda' \rho_\pi(\boldsymbol{q}) \\ \qquad \text{if } q_i \text{ is final in } \boldsymbol{q} \ (i = F), \\ \sum_{q_{i+1}} P(q_{i+1} | q_i, \pi(q_i)) J_\pi(q_{i+1}) \\ \qquad \text{otherwise} \end{cases}$$

be the expected Lagrangian of all paths through q_i (under π). Then let

$$J^*(q_i) = \begin{cases} \epsilon_\pi(\boldsymbol{q}) + \lambda' \rho_\pi(\boldsymbol{q}) \\ \qquad \text{if } q_i \text{ is final in } \boldsymbol{q} \ (i = F), \\ \min_a \sum_{q_{i+1}} P(q_{i+1} | q_i, a) J^*(q_{i+1}) \\ \qquad \text{otherwise.} \end{cases} \tag{6}$$

By induction, $J^*(q_i) \leq J_\pi(q_i)$ for all q_i and all π, with equality if $\pi = \pi^*$, where

$$\pi^*(q_i) = \arg\min_a \sum_{q_{i+1}} P(q_{i+1} | q_i, a) J^*(q_{i+1}) \tag{7}$$

for all non-final states q_i. Thus the optimal policy (minimizing (5)) can be computed efficiently using (6) and (7). In this paper, whatever algorithm is

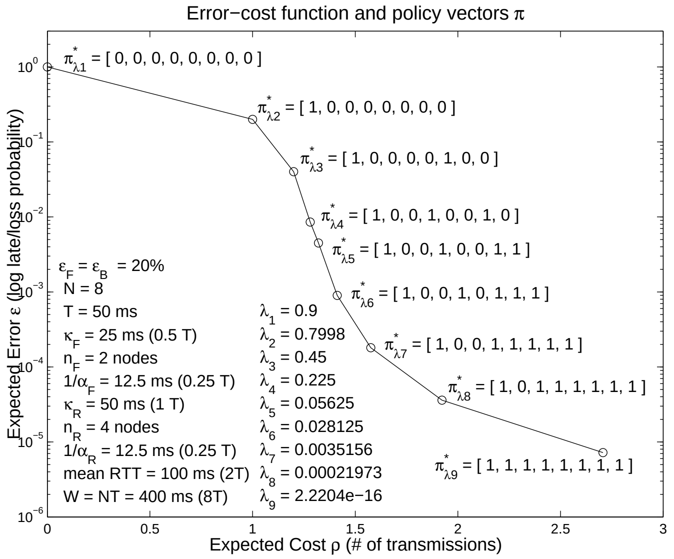


Figure 7: Optimal policies and their error-cost performances. The optimal policies for different values of λ' are shown as sequences of actions $[a_0, a_1, \ldots, a_{N-1}]$.

used to compute the optimal choice π minimizing $\epsilon_\pi + \lambda' \rho_\pi$ will be called the transmit-one (X1) algorithm.

As an example, Figure 7 shows the error-cost performances of different optimal policies $\pi^*_{\lambda'}$ computed by the X1 algorithm described above for different values of λ'. The expected error is shown on a logarithmic scale. Each optimal policy is shown as the sequence of actions $[a_0, a_1, \ldots, a_{N-1}]$ taken along the longest path in the Markov chain defined by the policy, that is, the sequence of actions taken by the sender at each transmission opportunity until it receives an acknowledgement. The all-zeros policy (which never transmits) is shown at the upper left with expected error equal to 1 and expected cost equal to 0. The all-ones policy (which always transmits) is shown at the lower right with expected error equal to 7×10^{-6} and expected cost equal to 2.7. Intermediate policies are shown in between. As λ' decreases, the optimal policy decreases in error but increases in cost. In this example, there are $N = 8$ transmission opportunities every $T = 50$ ms. The mean forward trip time is $\kappa_F + n_F/\alpha_F = T$ and the mean round trip time is $\kappa_R + n_R/\alpha_R = 2T$, using the parametric models discussed in Section 2.

To conclude this section, we discuss how this Markov decision process framework illustrated in Figure 6 allows further generalizations to other scenarios besides sender-driven transmission over a single-QoS network. Indeed the multiple-QoS scenario can be treated by adding additional action branches out of each state, e.g., 0 = don't send, 1 = send with QoS$^{(1)}$, 2 = send with QoS$^{(2)}$, and 3 = send with QoS$^{(3)}$. The receiver-driven scenario can be treated by relabeling "send" with "request" and relabeling "ACK" with "send". And a sender-driven wireless scenario, in which the receiver can respond to a corrupted received packet with a negative acknowledgement (NAK) as well as respond to a cleanly received packet with a positive acknowledgement (ACK), can be treated by adding additional observation branches after each action, e.g., 0 = no ACK, 1 = ACK, and 2 = NAK. Negative acknowledgements can improve performance by informing the sender that it can retransmit immediately without waiting for a timeout. A receiver-driven wireless scenario can be treated in the framework as well. Refer to [1, 2, 3, 4] for how the framework is used for hybrid FEC/Pseudo-ARQ in the receiver-driven multicast scenario.

4 Transmitting a Group of Data Units

In this section we study how a whole group of interdependent data units can be transmitted in a distortion-rate optimized way, using as a building block the scenario-appropriate method for transmitting a single data unit.

Suppose we wish to transmit a group of L data units whose dependencies are specified by an arbitrary directed acyclic graph. These L data units could be all the data units in a session, all the data units in a group of frames, or only those data units whose delivery deadlines lie in a limited time window.

Let π_l be the transmission policy for data unit $l \in \{1, \ldots, L\}$ and let $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_L)$ be the vector of transmission policies for all L data units in the group. Any given policy vector $\boldsymbol{\pi}$ induces an expected distortion and an expected transmission cost for the group. The expected transmission cost is the sum of the expected transmission costs for each data unit in the group. In turn, the expected transmission cost for each data unit $l \in \{1, \ldots, L\}$ is the expected cost per byte of transmitting the data unit, $\rho(\pi_l)$, times its size in bytes, B_l. That is, we have for the expected transmission cost

$$R(\boldsymbol{\pi}) = \sum_l B_l \rho(\pi_l). \tag{8}$$

The expected distortion for the group is somewhat more complicated to express. Let I_l be the indicator random variable that is 1 if data unit l arrives at the receiver on time, and is 0 otherwise. Then $\prod_{l' \preceq l} I_{l'}$ is 1 if data unit l is *decodable* by the receiver on time, and is 0 otherwise. If data unit l is decodable by the receiver on time, then the reconstruction error is reduced by the quantity Δd_l; otherwise the reconstruction error is not reduced. Hence the total reduction in reconstruction error for the group is $\sum_l \Delta d_l \prod_{l' \preceq l} I_{l'}$. Subtracting this quantity from the reconstruction error for the group if no data units are received, and taking expectations, we have for the expected distortion

$$D(\boldsymbol{\pi}) = D_0 - \sum_l \Delta D_l \prod_{l' \preceq l} (1 - \epsilon(\pi_{l'})), \tag{9}$$

where D_0 is the expected reconstruction error for the group if no data units are received, ΔD_l is the expected reduction in reconstruction error if data unit l is decoded on time, and $\epsilon(\pi_l)$ is the probability that data unit l does not arrive at the receiver on time (as computed in the previous section). Here we have used the assumption that the data packet transmission processes are independent, and are independent of the source process, in order to factor the expectation in (9).

With expressions (8) and (9) for the expected transmission cost and expected distortion for any given policy vector now in hand, we are able to optimize the policy vector to minimize the expected distortion subject to a constraint on the expected transmission cost. By restricting ourselves to solutions on the lower convex hull of the set of rate-distortion pairs $\{(R(\boldsymbol{\pi}), D(\boldsymbol{\pi}))\}$, we can solve the problem by finding the policy vector $\boldsymbol{\pi}$ that minimizes the expected Lagrangian

$$\begin{aligned} J(\boldsymbol{\pi}) &= D(\boldsymbol{\pi}) + \lambda R(\boldsymbol{\pi}) \\ &= D_0 + \sum_l \left[\Delta D_l \left(-\prod_{l' \preceq l} (1 - \epsilon(\pi_{l'})) \right) + \lambda B_l \rho(\pi_l) \right] .. \end{aligned} \tag{10}$$

The solution to this problem is completely characterized by the dependency graph, the set of distortion increments ΔD_l, and packet sizes B_l (which are determined by the source, source code, and packetization) and the error-cost functions $\epsilon(\pi)$ and $\rho(\pi)$ (which are determined by the transmission scenario and channel characteristics). This simplifies the problem of determining the quantities needed for the optimization. However, the minimization itself is complicated by the fact that the expression for the expected distortion cannot be split into a sum of terms each depending on only a single π_l, as is usually the case in rate allocation problems. Hence, we solve the problem using an iterative descent algorithm.

Our iterative approach is based on the method of alternating variables for multivariate minimization [58]. The objective function $J(\pi_1, \ldots, \pi_L)$ in (10) is minimized one variable at a time, keeping the other variables constant, until convergence. To be precise, 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. This can be done round-robin style, 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

$$\begin{aligned} \pi_l^{(n)} &= \arg\min_{\pi_l} J(\pi_1^{(n)}, \ldots, \pi_{l-1}^{(n)}, \pi_l, \pi_{l+1}^{(n)}, \ldots, \pi_L^{(n)}) \\ &= \arg\min_{\pi_l} S_l^{(n)} \epsilon(\pi_l) + \lambda B_l \rho(\pi_l), \end{aligned} \tag{11}$$

where (11) follows from (10) with

$$S_l^{(n)} = \sum_{l' \succeq l} \Delta D_{l'} \prod_{\substack{l'' \preceq l' \\ l'' \neq l}} (1 - \epsilon(\pi_{l''}^{(n)})). \tag{12}$$

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

```
Given $\lambda$, $\{(B_l, \Delta D_l)\}_{l=1}^{L}$,
    0. Initialize:
         $\epsilon_1 = \cdots = \epsilon_L = \min_\pi \epsilon(\pi)$, $\rho_1 = \cdots = \rho_L = \max_\pi \rho(\pi)$,
         $D = D_0 - \sum_l \Delta D_l \prod_{l' \preceq l}(1 - \epsilon_{l'})$, $R = \sum_l B_l \rho_l$,
         $J^{(0)} = D + \lambda R$, and
         $n = 1$.
    1. $l = l_n = (n \bmod L)$
    2. $S_l = \sum_{l' \succeq l} \Delta D_{l'} \prod_{l'' \preceq l' : l'' \neq l}(1 - \epsilon_{l''})$
    3. $\lambda'_l = \lambda B_l / S_l$
    4. $\pi^*_l = \arg\min_\pi \epsilon(\pi) + \lambda'_l \rho(\pi)$ [Algorithm X1]
    5. $\epsilon_l = \epsilon(\pi^*_l), \quad \rho_l = \rho(\pi^*_l)$
    6. $D = D_0 - \sum_l \Delta D_l \prod_{l' \preceq l}(1 - \epsilon_{l'}), \quad R = \sum_l B_l \rho_l$
    7. $J^{(n)} = D + \lambda R$
    8. If $J^{(n)} = J^{(n-1)}$ stop; else $n = n + 1$ and go to Step 1.
Return $\pi^*_1, \ldots, \pi^*_L$.
```

Figure 8: The sensitivity adaptation (SA) algorithm.

recovered at the receiver, given the current transmission policies for the other data units. Another interpretation of S_l is as the partial derivative of (9) with respect to $\epsilon_l = \epsilon(\pi_l)$, evaluated at $\boldsymbol{\pi}^{(n)}$. See [1, 2, 3, 4].

Now, the solution to (11) is simple. This is the problem of transmitting a single data unit, which can be solved with the X1 algorithm as described in the previous section: find the transmission policy π_l minimizing $\epsilon(\pi_l) + \lambda' \rho(\pi_l)$, where $\lambda' = \lambda B_l / S_l$. This can be accomplished by minimizing (5) using (6) and (7) as described in the previous section, or by using the equivalent procedure (e.g., exhaustive search) for the appropriate scenario. In this way, the policy vector $\boldsymbol{\pi}^{(n)}$ is determined and the process is repeated until $J(\boldsymbol{\pi}^{(n)})$ converges. Convergence is guaranteed because $J(\boldsymbol{\pi}^{(n)})$ is non-increasing and bounded below. The overall algorithm, which we call the sensitivity adaptation (SA) algorithm, is summarized in Figure 8.

In summary, using the SA algorithm in conjunction with the X1 algorithm we are able to find transmission policies $\pi^*_1, \ldots, \pi^*_L$ for each of the L data units in a group such that after following the policies, the expected distortion $D(\boldsymbol{\pi}^*)$ and the expected transmission cost $R(\boldsymbol{\pi}^*)$ are minimal (or at least locally minimal) for each other, since they lie on the convex hull of all operational rate-distortion pairs $(R(\boldsymbol{\pi}), D(\boldsymbol{\pi}))$, for the given set of transmission opportunities.

Actually, when the transmission scenario involves feedback, the above statement is true only approximately. The reason is that the development of the SA algorithm assumes that the transmission processes are independent of each other in order to factor the expectation across the product in (9). While independence between transmission processes is a good model for transmission without feedback, it is not a good model for transmission with feedback, since feedback about one data unit being recovered ideally affects not only subsequent transmissions of that data unit, but also subsequent transmissions of other data units.

Nevertheless, in transmission scenarios involving feedback, we are still able to provide stepwise-optimal rate-distortion performance by re-running the SA algorithm at every transmission opportunity. This allows all of the optimized transmission policies $\pi_1^*, \ldots, \pi_L^*$ to be updated at every transmission opportunity to take into account the most recent information from feedback on any of the data packets. To be more specific, for simplicity assume that each data unit l has the same set of transmission opportunities $s_0, s_1, \ldots, s_{N-1}$. Let $q_{l,0}, q_{l,1}, \ldots, q_{l,N-1}$ be the state of the transmission process for data unit l at each of these transmission opportunities. Run the SA algorithm at each transmission opportunity s_i, but in Step 4 of Figure 8 constrain policy $\pi_l^* = \pi_{l,i}^*$ to pass through state $q_{l,i}$. This can be done efficiently using (6) and (7) as usual, but stopping the computation at state $q_{l,i}$ as if it were the initial state. The resulting quantities $\epsilon(\pi_{l,i}^*)$ and $\rho(\pi_{l,i}^*)$ are then *conditional* expectations of the error and cost, respectively, of following policy $\pi_{l,i}^*$ from state $q_{l,i}$ given the history of actions and observations $(a_0, o_0) \circ (a_1, o_1) \circ \cdots \circ (a_{i-1}, o_{i-1})$ for data unit l leading up to $q_{l,i}$. Thus for example if data unit l is acknowledged by time s_i, then $\epsilon(\pi_{l,i}^*) = 0$, causing the sensitivities of all other data units (computed in Step 2) to increase, and causing the Lagrange multipliers of all other data units (computed in Step 3) to decrease. Upon convergence, each data unit l has independently optimized its transmission policy $\pi_{l,i}^*$ going forward from state $q_{l,i}$ at time s_i, given all information up to time s_i. Each data unit transmission process then takes one step forward, i.e., takes action $a = \pi_{l,i}^*(q_{l,i})$, and the procedure is repeated at the next transmission opportunity s_{i+1}.

This stepwise-optimal procedure can be likened to a procedure common among human agents who are assigned separate tasks towards a common goal, where achievement of the goal depends to varying degrees on achievement of the individual tasks. Suppose the agents call each other at the end of each day to report their state of progress. Since it is infeasible for an agent to follow from the outset an optimal strategy involving contingency plans for every possible daily state of every other agent, each agent instead optimizes his own long-term strategy assuming an expected level of success for each other agent. The agent is able to modify his long-term strategy daily, as the expected level of success of the other agents changes according to their status reports.

Although this stepwise-optimal procedure does not necessarily produce the optimal performance when feedback is available, it should produce near-optimal performance. Moreover, it is tractable, because it factors the full state space into a product of state spaces, one for each data unit (or agent). Although these state spaces are coupled, the coupling is loose. Hence it is possible to separately solve for the optimal policy for each data unit, and then run the SA algorithm to couple these solutions. In contrast, the truly optimal solution involves a state space that grows exponentially in the number of data units as well as the number of transmission opportunities. Podolsky, McCanne, and Vetterli studied such a solution in [8, 9], and even with a simplified channel model concluded that the problem is intractable when there are more than two data units and more than two transmission opportunities. Our factorization of the problem into loosely coupled problems of transmitting only a single data unit is one of the main contributions of our work.

5 Window and Rate Control

In the previous two sections, we showed how a system can achieve locally optimal distortion-rate performance in non-feedback scenarios, and stepwise-optimal distortion-rate performance in feedback scenarios. However, because we measure distortion-rate performance in an *average* sense, it is possible for such a distortion-rate optimized system to transmit most of the data units in each group in a single burst, resulting in a large instantaneous rate despite a low average rate. When the group of data units is large, e.g., the entire session, this is untenable. Two complementary solutions to this problem are window control and rate control.

In window control, the data units are given different windows of transmission opportunities, based on their delivery deadlines. To be specific, at any given transmission time s, only those data units l whose delivery deadlines $t_{DTS,l}$ fall within the window $[t_{lag}(s), t_{lead}(s)]$ are given the opportunity to transmit. The window boundaries $t_{lag}(s)$ and $t_{lead}(s)$ advance monotonically with s.

Figure 9 graphs typical values of $t_{lag}(s)$ and $t_{lead}(s)$ as a function of the transmission time s, $s \geq s_{start}$. For any given transmission time s, the vertical interval $\{t_{DTS} : t_{lag}(s) \leq t_{DTS} \leq t_{lead}(s)\}$ is the set of delivery deadlines whose data units are eligible for transmission at time s. Conversely, for any given delivery deadline t_{DTS}, the horizontal interval $\{s : t_{lag}(s) \leq t_{DTS} \leq t_{lead}(s)\}$ is the set of times at which data units with delivery deadline t_{DTS} are eligible for transmission. It is during this horizontal interval that data units with delivery deadline t_{DTS} can be transmitted.

The function $t_{lead}(s)$ determines when a data unit becomes elegible for transmission, while the function $t_{lag}(s)$ determines when a data unit becomes inelegible for transmission. A good choice for when a data unit becomes inelegible for transmission is its delivery deadline. Hence, a good choice for the function $t_{lag}(s)$ is the temporal coordinate transformation

$$t_{lag}(s) = t_{DTS0} + \nu[s - (s_{start} + \delta)] = A_{s \to t}(s).$$

This is because at time s all data units with delivery deadlines t_{DTS} prior to $t = A_{s \to t}(s)$ have expired. Put another way, all data units with delivery deadline

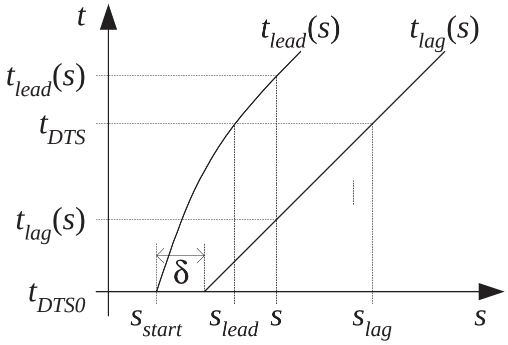


Figure 9: Window control. The horizontal interval between s_{lead} and s_{lag} is the window of transmission opportunity for data units with delivery deadline t_{DTS}. The vertical interval between $t_{lag}(s)$ and $t_{lead}(s)$ is the set of delivery deadlines t_{DTS} whose data units are elegible for transmission at time s.

t_{DTS} will expire when s reaches s_{DTS}, where $A_{s\to t}(s_{DTS}) = t_{DTS}$.

A good choice for the function $t_{lead}(s)$ is less evident, since there are a number of considerations. For one, the horizontal gap between the functions t_{lead} and t_{lag} determines the instantaneous buffer size, in seconds, at the receiver. A large buffer size implies a large storage requirement at the receiver, but it also implies a large window of opportunity during which data units can be retransmitted so that they can be reliably received before their delivery deadlines. A large buffer size also usually means a large preroll (start-up) delay δ, unless the instantaneous buffer size is allowed to begin small (at size δ seconds), and grow over time. For practical reasons most implementations limit the receiver buffer size to some level. This implies that $t_{lead}(s)$ eventually parallels $t_{lag}(s)$ at a common slope ν (the playback speed). However, the limiting buffer size can vary greatly between implementations. Finally, it is important not to allow too many data units to become elegible for transmission all at the same time (e.g., at the start). Without rate control, this would cause a large spike in instaneous rate, while with rate control, this would lead to other anomalous behavior (discussed later in this section). For the experimental results reported in Section 7, we choose $t_{lead}(s)$ so that buffer delay begins at δ and grows linearly at rate ν up to a constant delay $\delta_{\max}$. However, we have also used

$$t_{lead}(s) = t_{DTS0} + \nu(s - s_{start}) + (b/a)\ln(a(s - s_{start}) + 1)$$

for appropriately chosen a and b. This choice permits the receiver buffer size to go to infinity, implying absolute robustness asymptotically.

Now let us consider the transmission dynamics. Beginning at time s_{start}, at each transmission opportunity s (say, every $T = 50$ ms), the SA algorithm runs on the group of data units elegible for transmission at time s. We call this group of data units, $\{l : t_{lag}(s) \le t_{DTS,l} \le t_{lead}(s)\}$, the *transmission buffer* at time

s. For each data unit l in the transmission buffer, the SA algorithm iteratively computes the sensitivity S_l (according to (12)), and produces the optimal policy π_l (minimizing $S_l\epsilon(\pi_l) + \lambda B_l \rho(\pi_l)$ according to (11)). Upon convergence, each data unit l in the transmission buffer is sent (or not) according to the first action in policy π_l. Thus at each transmission opportunity, some of the data units in the transmission buffer are sent, and others are not. This process is repeated at each transmission opportunity.

In this way, each data unit builds up a transmission history over the (horizontal) interval of time in which the data unit remains in the transmission buffer, that is, the interval $[s_{lead}, s_{lag}]$ where $t_{lead}(s_{lead}) = t_{DTS,l}$ and $t_{lag}(s_{lag}) = t_{DTS,l}$. The transmission history for a data unit is the list of actions taken at the previous transmission opportunities during the interval, and the accompanying observations.

When the SA algorithm runs at each time s, it takes into account for each data unit in the transmission buffer its transmission history up to time s i.e., the current state of the data unit's transmission process. Thus, if data unit l has been acknowledged, then given its transmission history the conditional expected error $\epsilon(\pi_l)$ is zero regardless of the policy π_l followed moving forward. Hence, minimizing $S_l\epsilon(\pi_l) + \lambda B_l \rho(\pi_l)$ produces an optimal policy π_l that has the least possible expected cost $\rho(\pi_l)$, which is a policy that never transmits again. Similarly, if a data unit l has been very recently transmitted, then given its transmission history the conditional expected error $\epsilon(\pi_l)$ is small regardless of the policy π_l followed moving forward. So again, minimizing $S_l\epsilon(\pi_l) + \lambda B_l \rho(\pi_l)$ produces an optimal policy π_l that is unlikely to transmit right away, unless $\lambda' = \lambda B_l / S_l$ is exceptionally low.

It can be seen that when the SA algorithm runs on the transmission buffer at time s, very few of the data units in the buffer are actually selected for transmission at time s. It is clear from the error-cost function and the expression $S_l\epsilon(\pi_l) + \lambda B_l \rho(\pi_l)$ that a data unit has a greater chance of being selected for transmission at time s under the following conditions: if it has not yet been transmitted, if it has been transmitted only in the distant past (e.g., more than one RTT ago) and has not received an acknowledgement, if λ is low, if its size B_l is low, or if its sensitivity S_l is high. In turn, it is clear from (12) that a data unit's sensitivity S_l tends to be high if its distortion increment Δd_l is high, or if the distortion increment $\Delta d_{l'}$ of any of its descendants $l' \succ l$ is high, and if for any such high distortion increment $\Delta d_{l'}$, the probability is high that all ancestors l'' of l' arrive on time. Thus, most of the data units that are selected for transmission are important data units (with high Δd_l) that have recently entered the transmission buffer. A few data units, whose prior packet transmissions have not been acknowledged within a round trip time or two, are also selected for (re)transmission. And occasionally, a less important data unit will be selected for transmission for the first time as instantaneous bandwidth becomes available (see rate control later in this section) or as its sensitivity increases when its ancestors' data packet transmissions are acknowledged. In this way, the various factors that influence whether a data unit should or should not be transmitted at any given instant are perfectly blended together in one coherent computation.

Rate control can be used in conjunction with window control to smooth the instantaneous transmission rate still further. The rate control mechanism we propose is similar to the rate control mechanisms found in standard video encoders, but there are differences. In standard video encoders, the rate control mechanism typically adjusts a quantization stepsize Q [59] or possibly a Lagrange multiplier λ [60, 61] to affect the instantaneous bit rate out of the encoder into an encoder buffer. If the encoder buffer is close to empty, then Q or λ is decreased to keep the buffer from underflowing, while if the encoder buffer is close to full, then Q or λ is increased to keep the buffer from overflowing. Bits drain out of the buffer at a constant rate and can feed a constant bit rate (CBR) channel. If the channel is a variable bit rate (VBR) channel with a leaky bucket constraint, then the encoder buffer can be a virtual buffer matched to the VBR channel. The rate-controlled bit stream out of the encoder can then directly feed the VBR channel, while conforming to the channel's leaky bucket constraint [13].

In this paper we propose a similar mechanism for controlling the instantaneous rate of data packet transmissions out of the SA algorithm. Data units selected for transmission by the SA algorithm have their packets go into a virtual buffer. If the virtual buffer is close to empty, then λ is decreased, while if the virtual buffer is close to full, then λ is increased. If desired, the SA algorithm can be repeatedly re-run with new values of λ until a precise instantaneous rate is achieved at each transmission opportunity s. As a special case, λ can be adjusted at each transmission opportunity s to achieve a constant instantaneous rate out of the SA algorithm. This is usually done with an iterative binary search.

Of particular interest is the special case in which the rate control algorithm adjusts λ so that exactly one data unit is selected for transmission at each transmission opportunity. That is, λ is increased until there remains only one data unit l in the transmission buffer for which the optimal policy π_l (minimizing $S_l\epsilon(\pi_l)+\lambda B_l\rho(\pi_l)$) has "send" as its first action. This one data unit is relatively simple to find, without iteration, by creating a list of Lagrange multipliers $\Lambda = \{\lambda_l\}$, where for each data unit l, λ_l is the threshold for λ above which the data unit is not transmitted, and below which the data unit is transmitted, at the current transmission opportunity. Once this list is created, the data unit l to transmit is the data unit with the largest λ_l on the list.

The problem now is to compute λ_l for each data unit. Let B_l be the size of data unit l, let S_l be the sensitivity of data unit l (assuming there are no further transmissions of any other data units), and let λ'_l be the threshold for λ' above which the policy π_l minimizing $\epsilon(\pi_l)+\lambda'\rho(\pi_l)$ has "don't send" as its first action, and below which it has "send" as its first action. Then $\lambda_l = \lambda'_l S_l/B_l$.

It turns out that λ'_l is simple to approximate. Let $\pi_{l,0}$ be the policy for data unit l with no further transmissions, and let $\pi_{l,1}$ be the policy for data unit l with exactly one further transmission, which is at the current transmission opportunity. Then λ'_l is approximately equal to the slope of the line between the points $(\rho(\pi_{l,0}), \epsilon(\pi_{l,0}))$ and $(\rho(\pi_{l,1}), \epsilon(\pi_{l,1}))$, that is, $\lambda_l' \approx \hat{\lambda}_l'$, where

$$\hat{\lambda}_l' = \frac{\epsilon(\pi_{l,0}) - \epsilon(\pi_{l,1})}{\rho(\pi_{l,1}) - \rho(\pi_{l,0})} = \epsilon(\pi_{l,0}) - \epsilon(\pi_{l,1}). \tag{13}$$

In turn, $\epsilon(\pi_{l,0})$ and $\epsilon(\pi_{l,1})$ are easy to calculate as the probability that none of the previously transmitted packets for data unit l arrive at the receiver by the delivery deadline. (In the case of $\pi_{l,1}$ this includes the packet sent at the current time s). That is,

$$\begin{aligned} \epsilon(\pi_{l,0}) &= \prod_{j<i:a_j=1} P\{FTT > s_{DTS} - s_j | RTT > s_i - s_j\}, \\ \epsilon(\pi_{l,1}) &= \epsilon(\pi_{l,0}) \cdot P\{FTT > s_{DTS} - s_i\} \end{aligned}$$

where $s_0, \ldots, s_{i-1}$ are the previous transmission opportunities, $s_i = s$ is the current transmission opportunity, and s_{DTS} is the delivery deadline. Thus

$$\begin{aligned} \hat{\lambda}_l' = (1 - P\{FTT > s_{DTS} - s_i\}) \times \\ \prod_{j<i:a_j=1} P\{FTT > s_{DTS} - s_j | RTT > s_i - s_j\}. \end{aligned} \tag{14}$$

As we showed in Section 3, the conditional probabilities $P\{FTT > s_{DTS} - s_j | RTT > s_i - s_j\}$ can be expressed as $P\{FTT > s_{DTS} - s_j\} / P\{RTT > s_i - s_j\}$. Thus if we can further approximate $P\{FTT > \tau\}$ as 1 for $\tau < \mu_F$ and as ϵ_F for $\tau \geq \mu_F$ (and likewise for $P\{RTT > \tau\}$), then

$$\hat{\lambda}_l' \approx \begin{cases} 0, \text{ if } s_{DTS} - s_i < \mu_F, \text{ and} \\ (1 - \epsilon_F) \displaystyle\prod_{\substack{j<i:a_j=1, \\ s_i - s_j \geq \mu_R}} \frac{\epsilon_F}{\epsilon_R} \prod_{\substack{j<i:a_j=1, \\ s_i - s_j < \mu_R}} \epsilon_F \\ \text{otherwise.} \end{cases} \tag{15}$$

Hence, if the current time s_i is within a forward trip time μ_F of the delivery deadline s_{DTS}, then $\lambda_l' \approx 0$ (and the data unit will not be sent). Otherwise, λ_l' is approximately 1, multiplied by ϵ_F for every packet containing this data unit already transmitted within the last round trip time μ_R (typically at most one such data unit), multiplied still further by ϵ_F / ϵ_R (typically about 1/2) for every packet containing this data unit already transmitted longer ago than one round trip time μ_R. Thus data units within the transmission window are out of the running for transmission if 1) they are within a forward trip time of their deadlines, 2) they have already been transmitted and have been acknowledged, or 3) they have already been transmitted within a round trip time of the current time. The other data units are ranked by their sensitivities per unit cost, S_l / B_l, discounted by a factor of 2 for each unacknowledged transmission. The sensitivities can also be approximated in a similar way. Hence approximate rate-distortion optimal streaming can be achieved at very low computational complexity. Indeed using these approximations we have prototyped a rate-distortion optimized streaming media server running in Java that requires only about one percent of the CPU on a 700 MHz Pentium III, for a 40 Kbps audio stream. The accuracy of these approximations is evaluated in Section 7.

When it is desired to send more than one data unit at each transmission opportunity, a possibility is to simply choose the top candidates from the list Λ. This will be only approximately correct, however, since in this case the sensitivities S_l should ideally be recomputed for each possible subset of selected data units.

The rate control algorithm and the window control algorithm interact. Let $R(s)$ be the instantanous rate of transmission at time s in bits per second, and let $t'_{lead}(s)$ be the derivative of the function $t_{lead}(s)$ defining the leading edge of the transmission window. It turns out that the effective encoding bit rate for the content transmitted at time s is $(1-\epsilon_F)R(s)/t'_{lead}(s)$. The effective encoding bit rate is the expected number of valid bits received for each second of content, and hence determines the instantaneous playback quality. Thus if $R(s)$ gets larger, the effective encoding bit rate goes proportionally up, while if $t_{lead}(s)$ gets steeper, the effective encoding bit rate goes proportionally down. This is because in the interval $[s, s+\Delta s]$, approximately $R(s)\cdot\Delta s$ bits are transmitted, approximately $(1-\epsilon_F)R(s)\cdot\Delta s$ of these are eventually received, and the window advances by approximately $t'_{lead}(s)\cdot\Delta s$ seconds. Since most data units are transmitted along the leading edge of the window (if they are transmitted at all), the number of bits eventually received per second of content is $(1-\epsilon_F)R(s)\Delta s/t'_{lead}(s)\Delta s$, and the result follows. The impact of window control on the effective encoding bit rate is a further consideration in its design.

6 Flow and Congestion Control

Flow control refers to the ability (by the receiver, usually) to limit the sender's transmission rate so that the receiver's resources are not overwhelmed by too much data. Congestion control refers to the ability (by the network, the sender, or the receiver) to limit the sender's transmission rate so that the network's resources are not overwhelmed by too much data, from this communication or from any other communication. Both flow control and congestion control are necessary features in most communication systems that operate over best-effort networks.

In the Internet, the transmission control protocol (TCP) employs both flow control and congestion control by limiting the size of the window of data that the sender is trying to transmit at any given time. Without going into the particulars of TCP, it suffices to say that *identical* flow and congestion control mechanisms can be used in our framework for streaming packetized media. The basic idea is to simulate TCP. When the TCP simulator schedules a segment for transmission, then the scheduled time becomes a transmission opportunity, at which time any data unit may be transmitted (assuming fixed-size data units the same size as a segment), e.g., the data unit l for which λ_l is largest, as described in the previous section. In this way, packetized media can be streamed for real-time playback such that its transmission pattern is indistinguishable from that of TCP. While it is true that TCP's transmission pattern is not smooth (alternating between additive rate increases and multiplicative rate decreases), our packet-oriented framework does not require a smooth transmission pattern if the transmission window $[t_{lead}(s), t_{lag}(s)]$ becomes sufficiently large. The advantages of this approach are total compatibility with TCP, effective flow and congestion control, and a proven, stable protocol.

An alternative approach is to use "TCP-friendly" equation-based congestion control [62, 63, 64, 65, 66, 67, 68, 69]. In this approach, the congestion control mechanism makes a local estimate of TCP's long-term average transmission rate using an equation for the number of seconds between data units such as [68]

$$T = \mu_R\sqrt{2\epsilon_R/3} + 3(\mu_R + 4\sigma_R)\epsilon_R(1 + 32\epsilon_R^2)\sqrt{3\epsilon_R/8}, \tag{16}$$

where ϵ_R, μ_R, and σ_R^2 are short-term estimates of the packet loss probability, the mean round trip time, and the round trip time variance, respectively. The estimated transmission rate is reduced if necessary to effect flow control, and is then passed to the rate control mechanism (such as described in the previous section) to maintain the designated transmission rate. The advantages of this approach are its TCP-friendliness and its simplicity.

There are certainly other forms of flow and congestion control, such as those found in commercial streaming media products, which we will not describe here. However, any congestion control mechanism will include some kind of channel estimation, e.g., estimation of channel parameters such as ϵ_R, μ_R, and σ_R^2. This is because in the absense of explicit feedback from the network, the sender and/or receiver must infer the state of the network by observing data units as they enter and leave the network.

We do not advocate particular mechanisms for flow or congestion control. However, it is worth noting that our framework is compatible with a variety of such mechanisms.

We close this section with a possible protocol for server-driven streaming over a best-effort network, as sketched in Figure 10. The client initially contacts the server using a protocol such as the real time streaming protocol (RTSP) [70], and communicates information such as the filename, the presentation time t_{PTS0} at which to begin playback, the playback speed ν, and the playback (buffering) delay δ. The server then uses an index in the file to find the first data unit l to send, and notes its decoder timestamp $t_{DTS0} \equiv t_{DTS,l}$. When the server is ready to send, it records the time $s_{start}^{\min}$, sends to the client a synchronization packet containing a unique identifier as well as the first decoder timestamp t_{DTS0}, sets a timer, and awaits a reply. If the server does not receive a reply before the timer expires, then it repeats the process. Eventually, after a forward trip time, the client receives the synchronization packet, records t_{DTS0} and the current time r_{start}, sends an acknowledgement packet containing both the unique identifier and r_{start}, sets a playback timer for δ seconds, and awaits data packets from the server. If it receives a new synchronization packet in the meantime, then it repeats the process. Eventually, after a backward trip

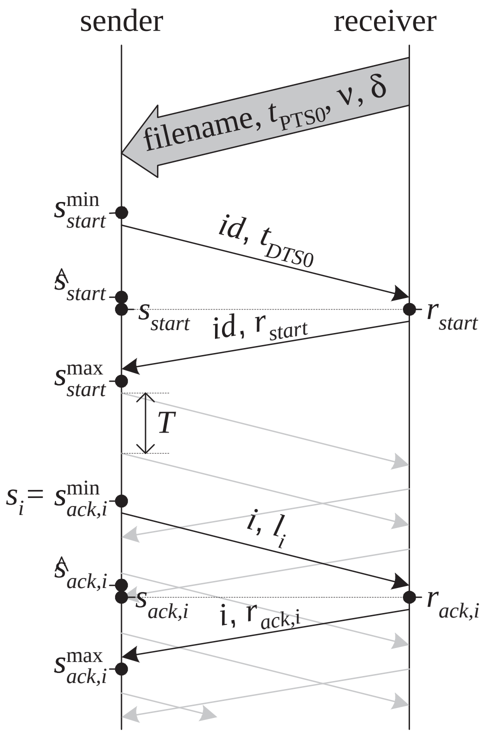


Figure 10: A real-time transmission protocol.

time, the server receives the acknowledgement data unit, records r_{start} and the current time $s_{start}^{\max}$, cancels its timeout timer, estimates s_{start} (corresponding to r_{start}) as the midpoint

$$\hat{s}_{start} = (s_{start}^{\min} + s_{start}^{\max})/2,$$

and begins sending data packets to the client as fast as the congestion control mechanism will allow, e.g., sending the data unit with the highest $\lambda_l' S_l / B_l$ every T seconds, where T is computed by (16). The server uses a unique sequence number i for each data packet sent, and records for each sequence number the time $s_i = s_{ack,i}^{\min}$ at which the packet is sent as well as an identifier l_i for the data unit in the packet. If and when the client receives a packet with sequence number i, it notes the current time $r_{ack,i}$, and returns an acknowledgement packet containing the sequence number i and the time $r_{ack,i}$ (and possibly additional acknowledgements of previous data units). The client also buffers the received data unit, eliminates duplicate data units if necessary, and sorts the data units in the buffer in order of their decoder timestamps $t_{DTS,l}$. When the client's playback timer expires at time $r_{start} + \delta$, the client begins to remove from the buffer and decode any data units l with timestamps $t_{DTS,l}$ equal to

$$t = t_{DTS0} + \nu(r - (r_{start} + \delta)) = A_{r\to t}(r), \tag{17}$$

where r is the current time at the client. Data units that arrive at time r with timestamps later than (17) are discarded. If and when the server receives an acknowledgement packet containing sequence number i and timestamp $r_{ack,i}$, it notes the current time $s^{\max}_{ack,i}$ and estimates $s_{ack,i} = s_{start} + (r_{ack,i} - r_{start}) = A_{r\to s}(r_{ack,i})$ as $\hat{s}_{ack,i} = \hat{s}_{start} + (r_{ack,i} - r_{start})$. These quantities can be used to compute the round trip time $RTT_i = s^{\max}_{ack,i} - s^{\min}_{ack,i}$ and estimate the forward and backward trip times $\widehat{FTT}_i = \hat{s}_{ack,i} - s^{\min}_{ack,i}$ and $\widehat{BTT}_i = s^{\max}_{ack,i} - \hat{s}_{ack,i}$, where $\widehat{FTT}_i + \widehat{BTT}_i = RTT_i$.

The server can use the samples $\{RTT_i\}$, $\{\widehat{FTT}_i\}$, and $\{\widehat{BTT}_i\}$ to compute short-term estimates of the means, variances, and distributions of these quantities. For example,

$$\begin{aligned} \Delta &\leftarrow RTT_i - \mu_R, \\ \mu_R &\leftarrow \mu_R + \eta_1 \Delta, \\ \sigma_R^2 &\leftarrow \sigma_R^2 + \eta_2(\Delta^2 - \sigma_R^2), \end{aligned}$$

where η_1 and η_2 are constants such as 1/8 and 1/4, respectively, as in TCP [71]. Then, using the relations (3) and (4) in Section 2, the parameters of a translated Gamma distribution for RTT can be computed as

$$\begin{aligned} \kappa_R &\leftarrow \min\{\kappa_R, RTT_i\}, \\ \alpha_R &\leftarrow (\mu_R - \kappa_R)/\sigma_R^2, \\ n_R &\leftarrow (\mu_R - \kappa_R)\alpha_R. \end{aligned}$$

The same can be done for $\widehat{FTT}$ and $\widehat{BTT}$.

Clearly, the forward and backward trip time estimates $\widehat{FTT}_i$ and $\widehat{BTT}_i$, their means $\mu_{\hat{F}}$ and $\mu_{\hat{B}}$, and the translations of their distributions $p_{\hat{F}}(\tau|\text{not lost})$ and $p_{\hat{B}}(\tau|\text{not lost})$ are biased upward and respectively downward by the same constant $\hat{s}_i - s_i = \hat{s}_{start} - s_{start} \equiv \Delta$. However, this bias is exactly cancelled out in the relevant calculations in Section 3, because for example

$$\begin{aligned} &P\{FTT > s_{DTS} - s_i\} \\ &\quad = P\{FTT + \Delta > s_{DTS} + \Delta - s_i\} \\ &\quad = P\{\widehat{FTT} > \hat{s}_{DTS} - s_i\}. \end{aligned}$$

Thus precise synchronization between the server and client is not necessary; $\widehat{FTT}$ and $\hat{s}_{DTS}$ can be used instead of FTT and s_{DTS}.

7 Experimental Results

In this section, we report our experimental results only for the scenario of sender-driven transmission over a single-QoS network with retransmissions. First we

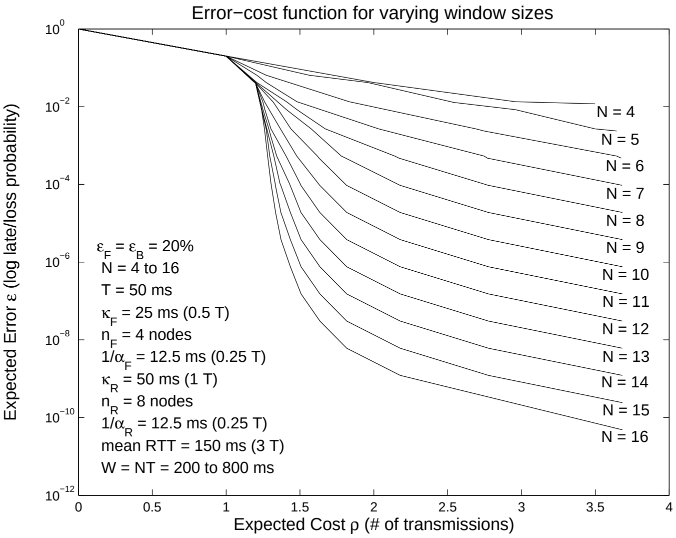


Figure 11: Error-cost functions for varying window sizes.

examine in detail error-cost optimized transmission of a single data unit, and later examine rate-distortion optimized streaming of an entire audio presentation. Throughout the section we assume that each data unit has transmission opportunities every T seconds, at sender times $s_0, s_1, \ldots, s_{N-1}$ with delivery deadline $s_{DTS} = s_N$ where $s_n = s_0 + nT$ for $n = 1, \ldots, N$. Thus $W = NT$ is the size of the window of transmission opportunity for each data unit.

The error-cost function for transmission of a single data unit was shown in Figures 5e and 7 of Section 3 for the parameters shown in Figure 7. Figure 7 shows the error-cost function on a log-linear scale, with each vertex of its convex hull labeled by the sequence of actions $[a_0, a_1, \ldots, a_7]$ for the optimal policy π^*_λ corresponding to the Lagrange multiplier λ for that vertex. By following policy $\pi^*_{\lambda_4}$, which transmits up to three times at intervals of $3T$ within a window of duration $8T$, is is possible to reduce the late/loss probability to less than one percent at an expected cost of only about 1.25 transmitted packets per data unit. The transmission interval $3T$ is equal to a slightly aggressive timeout interval: the mean RTT ($\kappa_R + n_R/\alpha_R = 2T$) plus two times the standard deviation ($2\sqrt{n_R}/\alpha_R = T$).

It is natural to ask whether extending the window size larger than $8T$, which is four times the mean RTT, can allow arbitrary further reductions in the late/loss probability, or whether the reductions saturate. Figure 11 answers this question, by plotting the error-cost function for the window size $W = NT$ ranging from $4T$ to $16T$, with T fixed at 50 ms and N ranging from 4 to 16. (In this set of experiments we increased the mean RTT to $3T$ and the standard deviation to about $0.7T$ in order to improve the dynamic range.) The figure shows that at any given expected cost ρ greater than $1/(1 - \epsilon_F) = 1.25$ transmitted

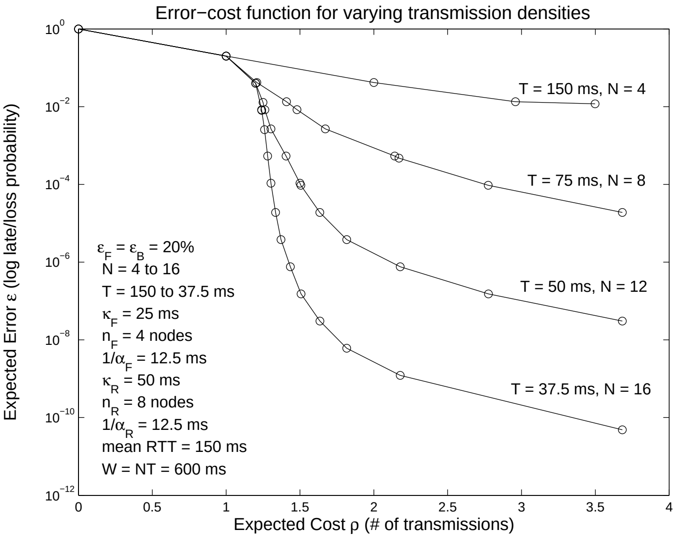


Figure 12: Error-cost functions for varying transmission densities.

packets per data unit, the late/loss probability can be reduced arbitrarily by increasing the window size to allow a larger number of retransmissions. However, for any expected cost ρ less than $1/(1-\epsilon_F) = 1.25$, the late/loss probability saturates at a lower bound equal to $1-\rho(1-\epsilon_F)$. This is because the information-theoretic capacity of an erasure channel with erasure probability ϵ_F is $C = (1-\epsilon_F)$ source units per transmission unit [72][§8.1], which cannot be increased by feedback [72][§8.12]. Any attempt to transmit at a rate $R = 1/\rho$ greater than C source units per transmission unit (i.e., at a rate ρ less than $1/C$ packet transmissions per data unit) inherently leads to a positive erasure probability, which must be at least $(R-C)/R = (1/\rho - (1-\epsilon_F))\rho = 1-\rho(1-\epsilon_F)$. At capacity ($\rho = 1.25$), it appears that the late/loss probability saturates at $N = 8$, or at a window size $8T$ equal to about 2.67 times the mean RTT.

It is also natural to ask, for a *fixed* window size W, whether the late/loss probability can be improved by increasing the density of transmission opportunities. Figure 12 answers this question, by plotting the error-cost function for a fixed window size $W = NT = 600$ ms, for $N = 4, 8, 12, 16$ and $T = 150, 75, 50, 37.5$ ms, respectively. The mean RTT remains fixed at 150 ms. The figure shows that at any given expected cost ρ greater than $1/(1-\epsilon_F) = 1.25$ transmitted packets per data unit, the late/loss probability can be reduced arbitrarily by increasing the transmission density (assuming, as we do here, that each transmission is independent.) However, for any expected cost ρ less than $1/(1-\epsilon_F) = 1.25$, the late/loss probability again saturates at the lower bound $1-\rho(1-\epsilon_F)$. At capacity, it appears that the late/loss probability saturates at $N = 8$, or at a transmission density equal to about two transmissions per mean RTT.

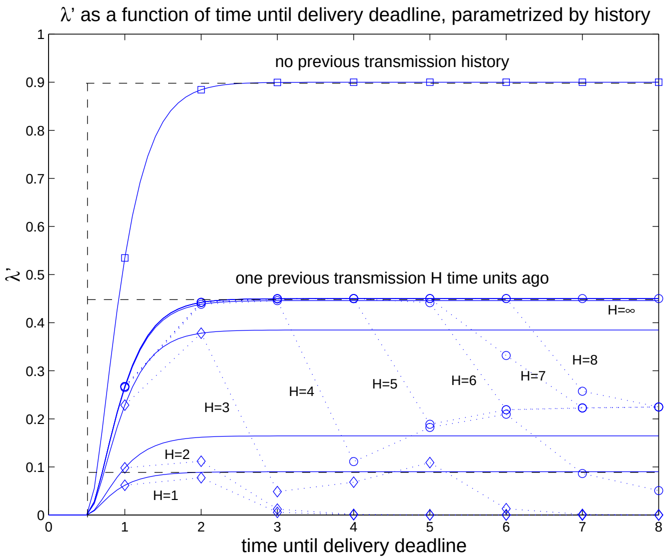


Figure 13: Value of λ' below (above) which optimal policy has "send" ("don't send") as its first action.

The above results show that for $\rho < 1/(1-\epsilon_F)$, a sufficient value for N is probably in the range 8–12. However, the results also also indicate that for $\rho > 1/(1-\epsilon_F)$, increasing N can dramatically improve performance. Unfortunately, the computational complexity of determining an optimal strategy grows exponentially in N, even using dynamic programming (6)–(7). Fortunately, it is possible to compute an approximate stepwise-optimal policy with computational complexity essentially linear in N using the approximations (14) or (15) for λ', the threshold below and above which the optimal policy has "send" and "don't send" (respectively) as its first action. Our next set of results justify these approximations.

Figure 13 shows λ' as a function of time until the delivery deadline, in units of $T = 50$ ms for the channel parameters $\kappa_F, n_F, \alpha_F, \kappa_R, n_R, \alpha_R$ shown in Figure 7. When there have been no previous transmissions, λ' is given by the square markers, the approximation $\hat{\lambda}'$ of (14) is given by the solid line through the square markers, and the approximation of (15) is given by the dashed line through the square markers. It is evident that the approximation $\hat{\lambda}'$ of (14) is essentially exact, and the approximation of (15) is essentially exact when the delivery deadline is further away than $2T$ (the mean RTT).

When there has been one previous transmission H time units ago, for $H = 4, 5, 6, 7, 8, \infty$, λ' is given by the circle markers, the approximation $\hat{\lambda}'$ of (14) is given by the solid line through the circle markers, and the approximation of (15) is given by the dashed line through the circle markers. It is evident that the approximation $\hat{\lambda}'$ of (14) is essentially exact when the time H since the last

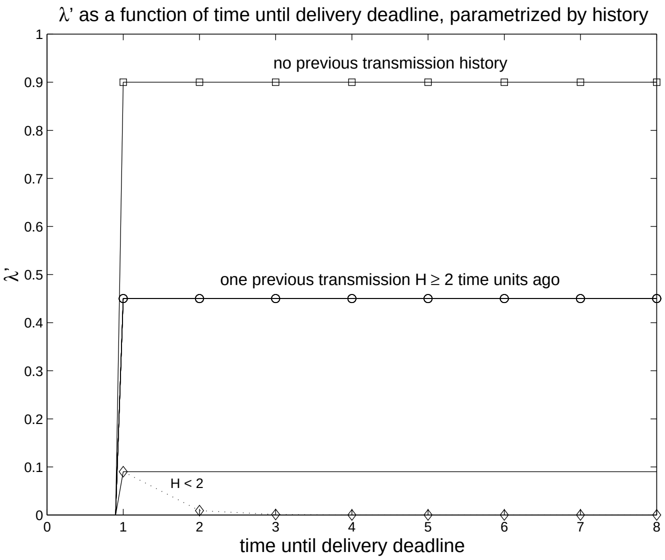


Figure 14: Value of λ' when the mean RTT is one time unit and its variance is zero.

transmission is large. As H gets smaller the approximation is exact only when the time until the delivery deadline is smaller than H, i.e., when the delivery deadline is closer (in the future) than the last transmission (in the past). Otherwise, when the delivery deadline is further away than H, λ' falls off as the value of immediately retransmitting the data unit falls off. When the last retransmission is very recent, i.e., $H = 1, 2, 3$, λ' is given by the diamond markers, the approximation $\hat{\lambda}'$ of (14) is given by the solid lines that are initially through the diamond markers, and the approximation of (15) is given by the lowest dashed line. It is evident that the approximation $\hat{\lambda}'$ of (14) is still good when the delivery deadline is closer (in the future) than the last transmission (in the past). However, for delivery deadlines further than H away, the approximation of (15) is better.

The approximations (14) and (15) become even better as the variance of the round trip time becomes lower. Figure 14 shows λ' and its approximations as in Figure 13, where the mean RTT remains equal to one time unit but the variance has been reduced to zero (i.e., $\kappa_R + n_R/\alpha_R = 1T$ while $n_R/\alpha_R^2 \to 0$). It is clear that the approximations are excellent, especially as ϵ_F (the height of the lowest line) becomes small.

When there are multiple previous transmissions, λ' and its approximations simply scale down together, by a factor of ϵ_F/ϵ_R, which is a factor of about 1/2 in these experiments, for each previous transmission. This can be seen in Figure 15.

Finally, we consider the overall distortion-rate performance of various meth-

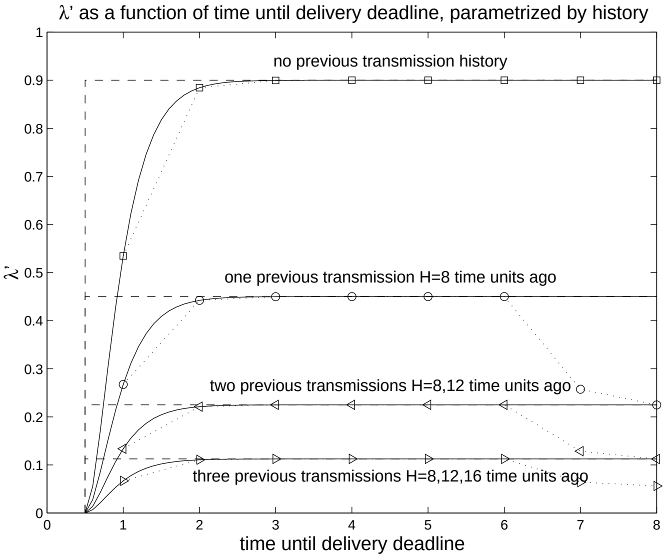


Figure 15: Value of λ' when there are multiple previous transmissions.

ods of streaming one minute of packetized audio content. 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 performs perceptual weighting on lapped orthogonal transform coefficients, followed by bitplane coding to produce an embedded bit string for each group of frames (GOF) of duration about 0.75 s. The bit string for each GOF is partitioned into segments of length 500 bytes, and packetized into data units. Twelve 500-byte data units are kept for each GOF, for a maximum bit rate of $12 \times 500 \times 8/0.75 = 64$ Kbps. The twelve data units per GOF are sequentially dependent, as shown in Figure 1a. Each data unit l is labeled by the decrease Δd_l in the perceptually weighted squared error if the data unit is decoded on time and all of its predecessors in the same GOF are decoded on time. All twelve data units in the Mth GOF receive the same decoding timestamp, equal to $0.75M$.

We compare several streaming systems. All of the systems use the same playback delay $\delta = 420$ ms and the same transmission buffer size, which ramps from 420 ms to 840 ms during the preroll period (420 ms) according to the functions $t_{lag}(s)$ and $t_{lead}(s)$ as specified in Section 5. The systems also use the same channel parameters, which are shown in Figure 16. Transmitted packets are dropped at random, and those not dropped receive a random delay, using a pseudo-random number generator. The pseudo-random number generator is initialized to the same seed for each of the systems compared. For each system at each transmission rate, performance is averaged over multiple runs to smooth out the effects of particular channel realizations. Figure 16 shows, for each of the systems compared, the signal-to-noise ratio in dB of the end-to-end perceptual

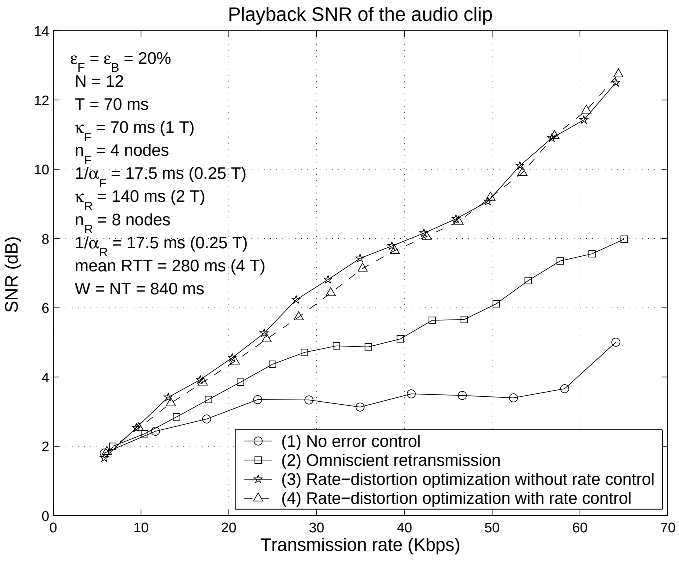


Figure 16: Layered audio transmission

distortion as a function of the transmission rate (in Kbps) averaged over the one minute audio clip. We now describe the compared systems in detail.

System 1: No error control. In this system, there is no error control. Data units are transmitted at most once, in GOF order. A data unit is transmitted only if all of its predecessors in the same GOF are also transmitted. The number of data units transmitted in each GOF is proportional to the transmission rate. As shown in Figure 16, the performance of this system saturates as the transmission rate increases. This is because in the absense of error control, base layer packets are being lost 20% of the time, limiting overall performance, regardless of the transmission rate. This shows the need for some sort of error control.

System 2: Omniscient retransmission. In this system, error control is provided by retransmissions, which may occupy up to 20% of the channel bandwidth (equal to the packet loss probability). Data units for which the server receives negative acknowledgements (NAKs) from the client are queued, and are retransmitted from the queue on a space-available basis. However, data units that are still in the retransmission queue past their delivery deadlines are removed from the queue and are not retransmitted. The remaining 80% or more of the channel bandwidth is used for first-time transmission of data units in the same manner as in System 1. *Omniscient* refers to the manner in which the client sends NAKs. Commonly, a client will send a NAK whenever a packet sequence number is detected missing for more than some timeout interval, but more sophisticated strategies are possible and are implemented in commercial streaming media systems. Here, we provide an upper bound on the performance of any such system by simulating an omniscient, though unrealizable, strategy in which the client sends precisely one NAK for each packet that is lost, at precisely the moment that the packet would have arrived at the client if it had not been lost. As Figure 16 shows, such a system with omniscient retransmission can perform up to three or more dB better than a system without error control.

System 3: Rate-distortion optimization without rate control. In this system, rate-distortion optimization as described in the previous sections is applied without rate control to the scheduling of packet transmissions at the sender. Unlike System 2, no NAKs are available; only ACKs are sent back to the server upon receipt of a packet by the client. The server uses its history of previous transmissions as well as its history of acknowledgments to determine which packets to transmit (or retransmit) at each transmission opportunity. The Lagrange multiplier λ is fixed for the entire presentation. However, the number of data units selected at each transmission opportunity may vary, resulting in a variable transmission rate during the presentation. Nevertheless, the *average* transmission rate during the presentation is well behaved and monotonically increases as λ decreases. As Figure 16 shows, rate-distortion optimization without rate control outperforms the system with omniscient retransmissions by up to four or more dB, and outperforms the system without any error control by up to an additional three or more dB, for a total gain up to seven or more dB.

System 4: Rate-distortion optimization with rate control. This system is the same as System 3 with the addition of rate control. In this case we fix a bandwidth limit for each group of frames (instead of fixing λ as in System 3). In order to get as close as possible to the bandwidth limit at each group of frames we iterate the rate-distortion optimized selection procedure by adjusting λ until the limit is reached but not exceeded. As Figure 16 shows, there is very little penalty (a fraction of a dB) for using a rate control mechanism to impose a fixed rate constraint on the transmission. Thus, it is clear that the rate-distortion optimized systems obtain superior performance by using the available bandwidth in the most cost-effective way.

As a final comment, it is worth noting that the number of runs required to smooth out the effects of particular channel realizations differs significantly for the different systems. Both Systems 1 and 2 require over 300 runs to obtain statistically "stable" results (in which averaging over additional realizations does not change the result much, or converges). This is because Systems 1 and 2 do not distinguish between packets of differing importance, and hence different channel realizations lead to a large dynamic range in playback quality. On the other hand, System 3 requires fewer than 30 runs, and System 4 requires fewer than 5 runs; sometimes one or two runs are sufficient. This clearly indicates that the rate-distortion optimized systems provide a relatively stable playback quality when operated over a lossy network.

8 Conclusion

Perhaps the main lesson of this paper is that rate-distortion optimized streaming, or more generically, cost-importance optimized streaming, can be achieved in a variety of scenarios by concentrating on cost-error optimized transmission of one packet at a time, and gluing the resulting transmission policies together. In this regard, this paper has made a number of important specific contributions, including the following:

- Abstraction of the encoding and packetization of multiple media in terms of a single directed acyclic dependency graph, in which each node represents a data unit labeled by its delivery deadline, size, and importance, and the concept of an incrementally additive distortion measure with respect to this dependency graph.
- Abstraction of a variety of different transmission scenarios, including sender-driven or receiver-driven transmission over best-effort networks, multiple overlay networks, integrated or differentiated services networks, combined wireline/wireless networks, and multiple access networks, in terms of a finite alphabet of choices π for sending a single data unit and measures of error probability $\epsilon(\pi)$ and cost $\rho(\pi)$ for each choice.
- The concept of the error-cost function $\epsilon(\rho) = \min_\pi\{\epsilon(\pi) : \rho(\pi) \leq \rho\}$ of a data unit and its relationship to the average distortion D and average rate R of an entire multimedia presentation.
- The generic algorithm X1 for sending a single data unit, $\pi^* = \arg\min_\pi \epsilon(\pi) + \lambda'\rho(\pi)$, which achieves optimal error-cost performance by selecting the transmission option π that minimizes a Lagrangian.
- The specific algorithm X1 for scenarios involving feedback, in which transmission options are identified as policies in a Markov decision process, Lagrangians of the error-cost function are associated with each policy, and a policy minimizing the Lagrangian is found using dynamic programming.
- The concept of sensitivity of the distortion to not receiving a particular data unit on time, and its computation in terms of error probabilities of other data units.
- The algorithm SA for jointly choosing the set of transmission options for all data units minimizing the Lagrangian $D + \lambda R$, using a locally optimal iterative descent algorithm. Also, stepwise application of the SA algorithm, in feedback scenarios, for stepwise locally optimal transmission in the distortion-rate sense.
- Formalization of the problem of window control, and characterization of lagging and leading window boundaries.
- Development of a fast rate control algorithm for use with the SA algorithm, and a sample protocol.

We believe that these contributions significantly advance the state of the art in streaming media systems, and lay the groundwork for future work on streaming media over diffserv, wireless, and multiple access networks, as well as via caching proxies, which are just now being deployed.

References

[1] P. A. Chou, A. E. Mohr, A. Wang, and S. Mehrotra. FEC and pseudo-ARQ for receiver-driven layered multicast. In *Communication Theory Workshop*, Aptos, CA, May 1999. IEEE.

[2] P. A. Chou, A. E. Mohr, A. Wang, and S. Mehrotra. FEC and pseudo-ARQ for receiver-driven layered multicast of audio and video. Technical Report MSR-TR-99-86, Microsoft Research, Redmond, WA, November 1999.

[3] P. A. Chou, A. E. Mohr, A. Wang, and S. Mehrotra. FEC and pseudo-ARQ for receiver-driven layered multicast of audio and video. In *Proc. Data Compression Conference*, Snowbird, UT, March 2000. IEEE Computer Society.

[4] P. A. Chou, A. E. Mohr, A. Wang, and S. Mehrotra. Error control for receiver-driven layered multicast of audio and video. *IEEE Trans. Multimedia*, 2001. To appear.

[5] Z. Miao and A. Ortega. Optimal scheduling for streaming of scalable media. In *Proc. Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, November 2000.

[6] Z. Miao. *Algorithms for Streaming, Caching and Storage of Digital Media.* PhD thesis, University of Southern California, Los Angeles, CA, August 2001. Expected.

[7] J. Zhou and J. Li. Scalable audio streaming over the Internet with network-aware rate-distortion optimization. In *Proc. Int'l Conf. Image Processing*, Thessaloniki, Greece, October 2001. IEEE. Submitted.

[8] M. Podolsky, S. McCanne, and M. Vetterli. Soft ARQ for layered streaming media. Technical Report UCB/CSD-98-1024, University of California, Computer Science Division, Berkeley, CA, November 1998.

[9] M. Podolsky, S. McCanne, and M. Vetterli. Soft ARQ for layered streaming media. *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology, Special Issue on Multimedia Signal Processing*, November 2001. To appear.

[10] V. Chande, H. Jafarkhani, and N. Farvardin. Joint source-channel coding of images for channels with feedback. In *Proc. Information Theory Workshop*, San Diego, CA, February 1998. IEEE.

[11] S. D. Servetto. *Compression and Reliable Transmission of Digital Image and Video Signals.* PhD thesis, University of Illinois, Urbana-Champaign, 1999.

[12] A. Ortega, K. Ramchandran, and M. Vetterli. Optimal trellis-based buffered compression and fast approximation. *IEEE Trans. Image Processing*, 3:26–40, January 1994.

[13] C.-Y. Hsu, A. Ortega, and A. Reibman. Joint selection of source and channel rate for VBR video transmission under ATM policing constraints. *IEEE Journal on Selected Areas in Communications*, 15(5):1016–1028, August 1997.

[14] J.-J. Chen and D. W. Lin. Optimal bit allocation for coding of video signals over ATM networks. *IEEE J. Selected Areas in Communications*, 15(6):1002–1015, August 1997.

[15] D. A. Turner and K. W. Ross. Optimal streaming of layer-encoded multimedia presentations. In *Proc. Int'l Conf. Multimedia and Exhibition*, New York, NY, July 2000. IEEE.

[16] I. Busse, B. Deffner, and H. Schulzrinne. Dynamic QoS control of multimedia applications based on RTP. In *First International Workshop on High Speed Networks and Open Distributed Platforms*, St. Petersburg, Russia, June 1995.

[17] H. Song, J. Kim, and C.-C. J. Kuo. Real-time encoding frame rate control for H.263+ video over the Internet. *Signal Processing: Image Communication*, 15(1–2):127–148, September 1999.

[18] K. Chawla, Z. Jiang, X. Qiu, and A. Reibman. Transmission of streaming video over an EGPRS wireless network. In *Proc. Int'l Conf. Multimedia and Exhibition*, New York, NY, July 2000. IEEE.

[19] A. Reibman, Y. Wang, X. Qiu, Z. Jiang, and K. Chawla. Transmission of multiple description and layered video over an EGPRS wireless network. In *Proc. Int'l Conf. Image Processing*, volume 2, pages 136–139, Vancouver, BC, October 2000. IEEE.

[20] T. Tian, A. H. Li, J. Wen, and J. D. Villasenor. Prority dropping in network transmission of scalable video. In *Proc. Int'l Conf. Image Processing*, volume 3, pages 400–403, Vancouver, Canada, October 2000. IEEE.

[21] P.-C. Hu, Z.-L. Zhang, and M. Kaveh. Channel condition ARQ rate control for real-time wireless video under buffer constraints. In *Proc. Int'l Conf. Image Processing*, volume 2, pages 124–127, Vancouver, BC, October 2000. IEEE.

[22] A. Albanese, J. Blömer, J. Edmonds, M. Luby, and M. Sudan. Priority encoding transmission. *IEEE Trans. Information Theory*, 42:1737–1744, November 1996.

[23] G. Davis and J. Danskin. Joint source and channel coding for image transmission over lossy packet networks. In *Conf. Wavelet Applications to Digital Image Processing*, Denver, CO, August 1996. SPIE.

[24] J. M. Boyce. Packet loss resilient transmission of MPEG video over the Internet. *Signal Processing: Image Communication*, 15(1–2):7–24, September 1999.

[25] U. Horn, K. Stuhlmüller, M. Link, and B. Girod. Robust Internet video transmission based on scalable coding and unequal error protection. *Signal Processing: Image Communication*, 15(1–2):77–94, September 1999.

[26] A. E. Mohr, E. A. Riskin, and R. E. Ladner. Graceful degradation over packet erasure channels through forward error correction. In *Proc. Data Compression Conference*, Snowbird, UT, March 1999. IEEE Computer Society.

[27] A. E. Mohr, E. A. Riskin, and R. E. Ladner. Unequal loss protection: Graceful degradation of image quality over packet erasure channels through forward error correction. *IEEE J. Selected Areas in Communications*, 18(6):819–829, June 2000.

[28] R. Puri and K. Ramchandran. Multiple description source coding through forward error correction codes. In *Proc. Asilomar Conference on Signals, Systems, and Computers*, Asilomar, CA, October 1999. IEEE.

[29] W. Zhu, Q. Zhang, and Y.-Q. Zhang. Network-adaptive rate control with unequal loss protection for scalable video over Internet. In *Proc. Int'l Symp. Circuits and Systems*, Sydney, May 2001. IEEE.

[30] Q. Zhang, G. Wang, W. Zhu, and Y.-Q. Zhang. Robust scalable video streaming over Internet with network-adaptive congestion control and unequal loss protection. In *Proc. Packet Video Workshop*, Kyongju, Korea, April 2001. EURASIP/IEEE.

[31] P. A. Chou and K. Ramchandran. Clustering source/channel rate allocations for receiver-driven multicast with error control under a limited number of streams. In *Proc. Int'l Conf. Multimedia and Exhibition*, New York, NY, July 2000. IEEE.

[32] A. E. Mohr, R. E. Ladner, and E. A. Riskin. Approximately optimal assignment for unequal loss protection. In *Proc. Int'l Conf. Image Processing*, Vancouver, BC, September 2000. IEEE.

[33] J. Lu, A. Nosratinia, and B. Aazhang. Progressive source-channel coding of images over bursty error channels. In *Proc. Int'l Conf. Image Processing*, Chicago, IL, October 1998. IEEE.

[34] M.J. Ruf and J.W. Modestino. Operational rate-distortion performance for joint source and channel coding of images. *IEEE Trans. Image Processing*, 8(3):305–320, March 1999.

[35] V. Chande and N. Farvardin. Progressive transmission of images over memoryless noisy channels. *IEEE J. Selected Areas in Communications*, 18(6):850–860, June 2000.

[36] J.-C. Bolot and A. Vega-Garcia. The case for FEC-based error control for packet audio in the Internet. *ACM Multimedia Systems Journal*, 1998. To appear.

[37] M. Podolsky, C. Romer, and S. McCanne. Simulation of fec-based error control for packet audio on the internet. In *Proc. Infocom*, San Francsico, CA, March 1998. IEEE.

[38] J. Bolot, S. Fosse-Parisis, and D. Towsley. Adaptive FEC-based error control for interactive audio on the Internet. In *Proc. Infocom*, New York, NY, March 1999. IEEE.

[39] W. Jiang and A. Ortega. Multiple description coding via polyphase transform and selective quantization. In *Proc. Visual Communications and Image Processing*, San Jose, CA, January 1999. SPIE.

[40] A. C. Miguel, A. E. Mohr, and E. A. Riskin. SPIHT for generalized multiple description coding. In *Proc. Int'l Conf. Image Processing*, Kobe, Japan, October 1999.

[41] S. Mehrotra. *Multiple Description Coding Using Overcomplete Linear Expansions*. PhD thesis, Stanford University, Stanford, CA, June 2000.

[42] C. Papadopoulos and G. M. Parulkar. Retransmission-based error control for continuous media applications. In *Proc. Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Zushi, Japan, July 1996.

[43] G. Carle and E. W. Biersack. Survey of error recovery techniques for IP-based audio-visual multicast applications. *IEEE Network Magazine*, 11(6):24–36, November 1997.

[44] X. Li, S. Paul, P. Pancha, and M. H. Ammar. Layered video multicast with retransmissions (LVMR). In *Proc. Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, St. Louis, MO, May 1997.

[45] H. Radha, Y. Chen, K. Parthasarathy, and R. Cohen. Scalable Internet video using MPEG-4. *Signal Processing: Image Communication*, 15(1–2):95–126, September 1999.

[46] D. Wu, Y. T. Hou, W. Zhu, H.-J. Lee, T. Chiang, Y.-Q. Zhang, and H. J. Chao. On end-to-end architecture for transporting MPEG-4 video over the Internet. *IEEE Trans. Circuits and Systems for Video Technology*, 10(6):923–941, September 2000.

[47] D. Wu, Y. T. Hou, and Y.-Q. Zhang. Transporting real-time video over the Internet" challenges and approaches. *Proceedings of the IEEE*, 88(12):1855–1875, December 2000.

[48] F. Yang, Q. Zhang, W. Zhu, and Y.-Q. Zhang. An efficient transport scheme for multimedia over wireless Internet. In *Proc. Int'l Conf. Third Generation Wireless and Beyond*, San Francisco, CA, May 2001. IEEE.

[49] S. D. Servetto, K. Ramchandran, K. Nahrstedt, and A. Ortega. Optimal segmentation of a VBR source for its parallel transmission over multiple ATM connections. In *Proc. Int'l Conf. Image Processing*, Santa Barbara, CA, October 1997. IEEE.

[50] V. Padmanabhan. Using differentiated services mechanisms to improve network protocol and application performance. In *RTAS Workshop on QoS Support for Real-Time Internet Applications*, Vancouver, Canada, July 1999. IEEE.

[51] J. Shin, J. Kim, and C.-C. J. Kuo. Relative priority based QoS interaction between video applications and differentiated service networks. In *Proc. Int'l Conf. Image Processing*, Vancouver, Canada, October 2000. IEEE.

[52] W. Li and Y. Chen. Experiment result on fine granularity scalability. contribution M4473, ISO/IEC JTC1/SC29/WG11, Seoul, March 1999.

[53] S. Li, F. Wu, and Y.-Q. Zhang. Experimental results with progressive fine granularity scalable (PFGS) coding. Technical Report m5742, ISO/IEC JTC1/SC29/WG11, Noordwijkerhout, NL, March 2000.

[54] W. Turin. *Digital Transmission Systems: Performance Analysis and Modeling*. McGraw-Hill, 1999.

[55] A. Mukherjee. On the dynamics and significance of low frequency components of internet load. *Internetworking: Res. Experience*, 5:163–205, December 1994.

[56] A.M. Mood, F.A. Graybill, and D.C. Boes. *Introduction to the Theory of Statistics*. McGraw-Hill, 3rd edition, 1974.

[57] S.M. Ross. *Stochastic Processes*. Wiley, 1974.

[58] R. Fletcher. *Practical Methods of Optimization*. Wiley, 2nd edition edition, 1987.

[59] ITU-T SG16/Q15 (T. Gardos, ed.). Video codec test model number 10 (TMN-10). *ITU-T SG16/Q15 document Q15-D-65*, April 1998.

[60] J. Choi and D. Park. A stable feedback control of the buffer state using the controlled lagrange multiplier method. *IEEE Trans. Image Processing*, 3(5):546–588, September 1994.

[61] G. J. Sullivan and T. Wiegand. Rate-distortion optimization for video compression. *IEEE Signal Processing Magazine*, 15(6):74–90, November 1998.

[62] J. Madhavi and S. Floyd. TCP-friendly unicast rate-based flow control. *Technical note sent to the end2end-interest mailing list*, January 1997. http://www.psc.edu/networking/papers/tcp_friendly.html.

[63] M. Mathis, J. Semke, S. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communication Review*, 27(3):67–82, July 1997.

[64] T. Turletti, S.F. Parisis, and J.-C. Bolot. Experiments with a layered transmission scheme over the Internet. Technical Report 3296, INRIA, Sophia Antipolis, France, November 1997.

[65] D. Sisalem and H. Schulzrinne. The loss-delay adaptation algorithm: a TCP-friendly adaptation scheme. In *Proc. Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, UK, July 1998.

[66] W.-T. Tan and A. Zakhor. Internet video using error resilient scalable compression and cooperative transport protocol. In *Proc. Int'l Conf. Image Processing*, volume 3, pages 458–462, Chicago, IL, October 1998.

[67] R. Rejaie, M. Handley, and D. Estrin. RAP: an end-to-end based congestion control mechanism for realtime streams in the Internet. In *Proc. INFOCOM*, New York, NY, March 1999. IEEE.

[68] S. Floyd, M. Handley, and J. Padhye. Equation-based congestion control for unicast applications. Technical Report TR-00-03, International Computer Science Institute, Berkeley, CA, March 2000.

[69] Q. Zhang, Y.-Q. Zhang, and W. Zhu. Resource allocation for audio and video streaming over the Internet. In *Proc. Int'l Symp. Circuits and Systems*, volume IV, pages 21–24, Geneva, Switzerland, May 2000. IEEE.

[70] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol (RTSP). Technical Report RFC-2236, IETF, April 1998.

[71] D.E. Comer. *Internetworking with TCP/IP*, volume 1. Prentice-Hall, 3 edition, 1995.

[72] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1991.