Computing Rate-Distortion Optimized Policies for Streaming Media to Wireless Clients

Jacob Chakareski[†,*], Philip A. Chou[*] and Behnaam Aazhang[†]

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

[†] *Center for Multimedia Communication, Rice University, Houston, TX 77055*

cakarz@rice.edu, pachou@microsoft.com, aaz@rice.edu

Abstract

We consider the problem of streaming packetized media over the Internet from a server through a base station to a wireless client, in a rate-distortion optimized way. For error control, we employ an incremental redundancy scheme, in which the server can incrementally transmit parity packets in response to negative acknowledgements fed back from the client. Computing the optimal transmission policy for the server involves estimation of the probability that a single packet will be communicated in error as a function of the expected redundancy (or cost) used to communicate the packet. In this paper, we show how to compute this error-cost function, and thereby optimize the server's transmission policy, in this scenario.

1 Introduction

We consider the problem of streaming packetized media over the Internet from a server to a wireless client. The main difference between this problem and the problem of streaming to a wireline client is that in the former problem, we assume, the client may observe bit errors in the packet payload. In particular, we assume that bit errors may be passed up from the physical and link layers to the network and transport layers, and that the transport layer runs a protocol such as UDP Lite [1], in which the transport checksum is applied only to the header, rather than to both the header and payload. This allows bit errors in the packet payload to be passed up through the protocol stack to the client application. In this way, the application can receive corrupted packets and can provide error control for corrupted packets in a way that is similar to providing error control for lost packets in a streaming media system. Other than exposing bit errors to the application, we assume that neither the network infrastructure nor the client's base station offers enhanced quality-of-service support for streaming media to the wireless client.

For error control, we propose a hybrid FEC/ARQ approach known as incremental redundancy, in which the server can incrementally transmit parity packets in response to negative acknowledgements fed back from the client, until it receives a positive

Proceedings of the DATA COMPRESSION CONFERENCE (DCC'02)
1068-0314/02 $17.00 © 2002 IEEE

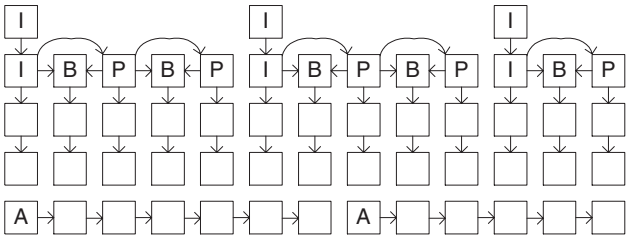

Figure 1: Typical directed acyclic dependency graph for video and audio data units.

acknowledgment. All packets sent in either direction are subject to random loss, delay, and corruption. Therefore the server can never be completely aware of the state of the wireless client. However, the server is aware of the different deadlines, importances, and dependencies of the various media packets to be transmitted. Using this information the server is able to transmit its media and parity packets based on the feedback it receives in a rate-distortion optimized way, that is, minimizing the expected end-to-end distortion subject to a constraint on the expected transmission rate. Such a rate-distortion optimized transmission algorithm, or transmission policy, results in unequal error protection provided to different portions of the media stream.[1]

To compute the rate-distortion optimized transmission policy we use the Iterative Sensitivity Adjustment (ISA) algorithm introduced in [2]. The core step of this algorithm involves estimation of the probability that a single packet will be communicated in error as a function of the expected redundancy, or cost, used to communicate the packet. The lower convex hull of the set of all expected error-cost pairs is called the error-cost function. How to compute this function, for the scenario of sender-driven streaming over a best-effort network to a wireless client, is the focus of this paper.

2 Preliminaries

In a streaming media system, the encoded data are packetized into *data units* and are stored in a file on a media server. All of the data units in the presentation have interdependencies, which can be expressed by a directed acyclic graph as illustrated in Figure 1. Each node of the graph corresponds to a data unit, and each edge of the graph directed from data unit l' to data unit l implies that data unit l can be decoded only if data unit l' is first decoded.

Associated with each data unit l is a size B_l, a decoding time $t_{DTS,l}$, and an importance Δd_l. The size B_l is the size of the data unit in bytes. The decoding time $t_{DTS,l}$ is the time at which the decoder is scheduled to extract the data unit from its input buffer and decode it. (This is the decoder timestamp, in MPEG terminology.) Thus $t_{DTS,l}$ is the delivery deadline by which data unit l must arrive at the client, or be too late to be used. Packets containing data units that arrive after the data

[1] In this paper, a *transmission policy* is a decision procedure, or algorithm for transmitting data. In general, a policy is a state machine that can move to a new state and take an action as a function of its current state and its current input. Thus a transmission policy could be a transmission algorithm for hybrid FEC/ARQ, ARQ, or FEC, depending on the scenario at hand.

units' delivery deadlines are discarded. The importance Δd_l is the amount by which the distortion at the client will *decrease* if the data unit arrives on time at the client and is decoded.

The communication channel, which in effect is an aggregate of the backbone packet network and the wireless link, is modeled as an independent time-invariant packet erasure channel with random delays at the packet level and as a binary symmetric channel (BSC) at the bit level. This means that if the server inserts a packet into the network at time t, then the packet is lost with some probability, say ϵ_F, independent of t. However, if the packet is not lost, then it arrives at the client device at time t', where the forward trip time $FTT = t' - t$ is randomly drawn according to probability density p_F. Furthermore, the individual bits of the received packet are independently and symmetrically corrupted with a probability BER_F. Therefore even though the packet may arrive at the client device, it may still be discarded by the client platform before reaching the client application if the packet's RTP/UDP/IP header has been corrupted. If N_h is the size of the header in bits, then $\epsilon_H = 1 - (1 - BER_F)^{N_h}$ is the probability of header corruption, and a modified probability of packet loss, $\epsilon'_F = \epsilon_F + (1 - \epsilon_F)\epsilon_H$, is the probability that the packet is either lost in the network or else is discarded by the client platform due to a corrupted header. We let $P\{FTT' > \tau\} = \epsilon'_F + (1 - \epsilon'_F) \int_\tau^\infty p_F(t)dt$ denote the probability that a packet transmitted by the server at time t does not arrive at the client application by time $t + \tau$, whether it is lost in the network, discarded by the client platform, or simply delayed by more than τ. Even if the packet arrives at the client application, its payload may still be corrupted with probability $\epsilon_P = 1 - (1 - BER_F)^{N_p}$, where N_p is the size of the payload in bits.

The client application immediately transmits either a positive or negative acknowledgement packet over a backward channel whenever it receives a data packet on the forward channel. The backward channel is similarly characterized by the probability of packet loss ϵ_B, delay density p_B, and bit error rate BER_B. These induce a modified probability ϵ'_B of packet loss on the backward channel, and a distribution $P\{BTT' > \tau\}$ on the modified backward trip time. In turn, these induce a modified probability $\epsilon'_R = 1 - (1 - \epsilon'_F)(1 - \epsilon'_B)$ of losing a packet either on the forward or backward channel, and a modified round trip time distribution $P\{RTT' > \tau\} = \epsilon'_R + (1 - \epsilon'_R) \int_\tau^\infty p_R(t)dt$, where $p_R = p_F * p_B$ is the convolution of p_F and p_B. Note that $P\{RTT' > \tau\}$ is the probability that the server does not receive an acknowledgement packet (positive or negative) by time $t + \tau$ for a data packet tranmitted by the server at time t.

3 Incremental Redundancy Transmission

In our incremental redundancy system, for each data unit that is sufficiently important to transmit, the server augments the data unit with a 2-byte CRC and transmits the data unit and its CRC in a packet, herein called a *systematic* packet, to the client. The server may transmit systematic packets periodically if necessary, until it receives a positive or negative acknowledgement from the client or until the server gives up

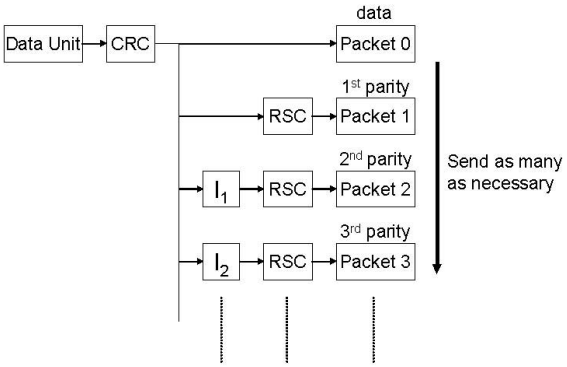

Figure 2: The Incremental Redundancy transmission scheme.

transmitting the data unit. Once the server receives a negative acknowledgement (NAK), it may begin to transmit parity packets periodically if necessary, until it receives a positive acknowledgement (ACK) or until it gives up. An ACK indicates that the client has successfully recovered the data unit. The server generates the parity packets for a data unit by applying a rate-1/2 recursive systematic convolutional (RSC) code to different pseudo-random interleavings of the data unit and its CRC, as shown in Figure 2. Hence each parity packet is unique. The client attempts to decode each packet that it receives, using the CRC for systematic packets, or using the List Viterbi algorithm [5] for the first parity packet, or using the Turbo decoding algorithm [6] for subsequent parity packets. The client immediately transmits an ACK to the server upon successful decoding, or a NAK upon decoding failure. For more details on the incremental redundancy scheme, we refer the reader to [3, 4].

The server may transmit a packet for a data unit at any of a finite set of transmission opportunities $t_0, t_1, \ldots, t_{N-1}$. Whether or not the server transmits a packet at any given transmission opportunity is determined by the server's transmission policy for the data unit, which takes into account the feedback received from the client, the data unit's dependencies on other data units, the data unit's deadline, importance, and size, and the server's overall transmission rate budget. In the next section we show how these transmission policies can be rate-distortion optimized.

4 R-D Optimization using the ISA Algorithm

We assume that the transmission policy π_l for each data unit l is selected from a family of policies Π. The family Π is determined by the scenario under consideration. For example, in a pure forward error correction scenario, Π may correspond to a family of forward error correction codes with parameters (n, k). In this paper, we focus on the incremental redundancy scenario, in which Π corresponds to a family of transmission policies defined precisely in the next section. In this section it is sufficient to let Π remain abstract.

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

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

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

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

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

where D_0 is the expected reconstruction error for the presentation if no data units are received and ΔD_l is the expected reduction in reconstruction error if data unit l is decoded on time.

Finding a policy vector $\boldsymbol{\pi}$ that minimizes the expected Lagrangian $J(\boldsymbol{\pi}) = D(\boldsymbol{\pi}) + \lambda R(\boldsymbol{\pi})$, for $\lambda > 0$, is difficult since the terms involving the individual policies π_l in $J(\boldsymbol{\pi})$ are not independent. Therefore, we employ an iterative descent algorithm, called the Iterative Sensitivity Adjustment (ISA) algorithm, in which we minimize the objective function $J(\pi_1, \ldots, \pi_L)$ one component at a time while keeping the other variables constant, until convergence [2]. It can be shown that the optimal individual policies at iteration n, for $n = 1, 2, \ldots$, are given by

$$\pi_l^{(n)} = \arg\min_{\pi_l} S_l^{(n)} \epsilon(\pi_l) + \lambda B_l \rho(\pi_l), \tag{3}$$

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

The minimization (3) is now simple, since each data unit l can be considered in isolation. Indeed the optimal transmission policy $\pi_l \in \Pi$ for data unit l minimizes the "per data unit" Lagrangian $\epsilon(\pi_l) + \lambda' \rho(\pi_l)$, where $\lambda' = \lambda B_l / S_l^{(n)}$. Thus to minimize (3) for any l and λ', it suffices to know the lower convex hull of the function $\epsilon(\rho) = \min_{\pi \in \Pi} \{\epsilon(\pi) : \rho(\pi) \leq \rho\}$, which we call the expected *error-cost* function. In the next section we show how to compute the expected error-cost function for the family of transmission policies corresponding to the incremental redundancy scheme.

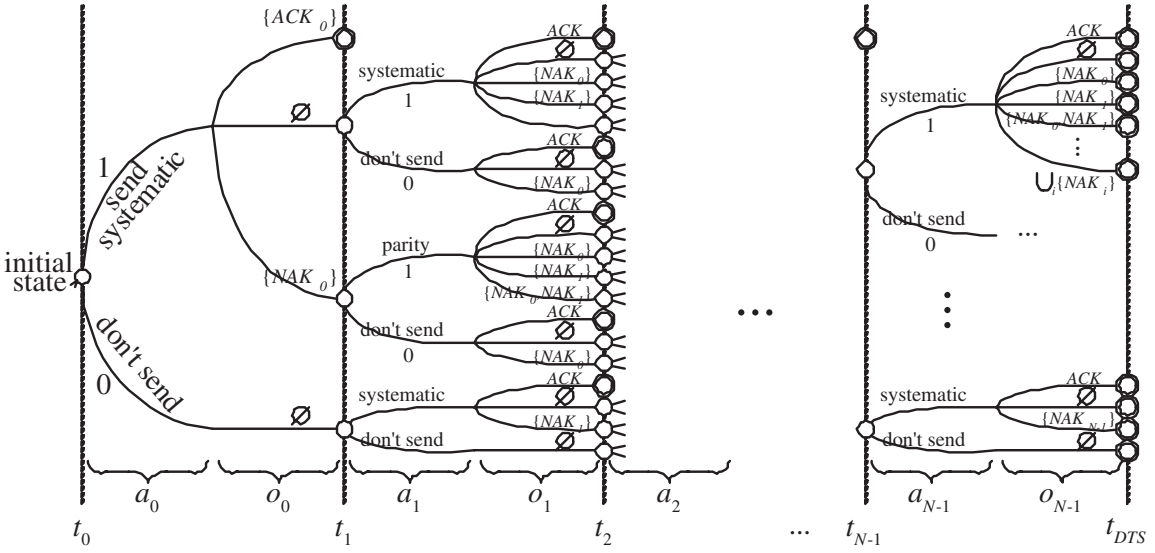

Figure 3: State space for a Markov decision process.

5 Computing the Expected Error-Cost Function

For transmitting a single data unit, we assume that there are N discrete transmission opportunities $t_0, t_1, \ldots, t_{N-1}$ prior to the data unit's delivery deadline t_{DTS} at which the server is allowed to transmit either a systematic packet or a parity packet for the data unit. The server need not transmit a packet at every transmission opportunity. As described in Section 3, if the server transmits a packet at a transmission opportunity, it must be a systematic packet if a NAK has not yet been received; otherwise, it must be a parity packet. The server does not transmit any packets after an ACK is received.

At each transmission opportunity t_i, $i = 0, 1, \ldots, N-1$, the server takes an action a_i, where $a_i = 1$ if the server sends a packet and $a_i = 0$ otherwise. Then, at the next transmission opportunity t_{i+1}, the server makes an observation o_i, where o_i is the set of acknowledgements received by the server in the interval $(t_i, t_{i+1}]$. For example, $o_i = \{NAK_{j_1}, ACK_{j_2}\}$ means that during the interval $(t_i, t_{i+1}]$, a NAK arrived for the packet sent at time t_{j_1} and an ACK arrived for the packet sent at time t_{j_2}. The history, or the sequence of action-observation pairs $(a_0, o_0) \circ (a_1, o_1) \circ \cdots \circ (a_i, o_i)$ leading up to time t_{i+1}, determines the state q_{i+1} at time t_{i+1}, as illustrated in Figure 3. If the final observation o_i includes an ACK, then q_{i+1} is a final state. In addition, any state at time $t_N = t_{DTS}$ is a final state. Final states in Figure 3 are indicated by double circles.

The action a_i taken at a non-final state q_i determines the transition probabilities $P(q_{i+1}|q_i, a_i)$ to the next state q_{i+1}. For example, in Figure 3, if the action taken at the initial state q_0 is $a_0 = 1$ (send), then the transition probabilities to the four states at time t_1 are respectively $P\{RTT' \leq t_1 - t_0\}(1 - \epsilon_P)$, $P\{RTT' > t_1 - t_0\}$, $P\{RTT' \leq t_1 - t_0\}\epsilon_P$, and 0. (Recall that $P\{RTT' > t_1 - t_0\}$ is the probability that no response is received by the server by time t_1 for a packet transmitted at time

t_0, and that ϵ_P is the probability that the payload will be corrupted on the forward channel.) On the other hand, if the action taken is $a_0 = 0$ (don't send), then the transition probabilities are respectively 0, 0, 0, and 1.

Formally a policy π is a mapping $q \mapsto a$ from non-final states to actions. Thus any policy π induces a Markov chain with transition probabilities $P_\pi(q_{i+1}|q_i) \equiv P(q_{i+1}|q_i, \pi(q_i))$, and consequently also induces a probability distribution on final states. Let q_F be a final state with history $(a_0, o_0) \circ (a_1, o_1) \circ \cdots \circ (a_{F-1}, o_{F-1})$, and let $q_{i+1} = q_i \circ (a_i, o_i)$, $i = 1, \ldots, F-1$, be the sequence of states leading up to q_F. Then q_F has probability $P_\pi(q_F) = \prod_{i=0}^{F-1} P_\pi(q_{i+1}|q_i)$, transmission cost $\rho_\pi(q_F) = \sum_{i=0}^{F-1} a_i$, and error $\epsilon_\pi(q_F) = 0$ if o_{F-1} contains an ACK and otherwise $\epsilon_\pi(q_F)$ is equal to the probability that none of the transmitted packets results in successful decoding by time t_{DTS}, given q_F. For example, if q_F is the second state from the top at time t_{DTS} in Figure 3, then a systematic packet was sent at every transmission opportunity $t_0, t_1, \ldots, t_{N-1}$ and no acknowledgements were received. In that case, $\epsilon_\pi(q_F) = \prod_{i=0}^{N-1} P\{FTT'' > t_{DTS} - t_i\}$, where $P\{FTT'' > \tau\} = P\{FTT' > \tau\} + P\{FTT' \leq \tau\}\epsilon_P$ is the probability that either a packet transmitted at time t does not arrive at the client application by time $t + \tau$ or else its payload is corrupted.

Armed with definitions of probability, transmission cost, and error for each final state, we can now express the expected cost and error for the Markov chain induced by policy π: $\rho(\pi) = E_\pi \rho_\pi(q_F) = \sum_{q_F} P_\pi(q_F)\rho_\pi(q_F)$, $\epsilon(\pi) = E_\pi \epsilon_\pi(q_F) = \sum_{q_F} P_\pi(q_F)\epsilon_\pi(q_F)$. We wish to find the policy π that minimizes $\epsilon(\pi) + \lambda'\rho(\pi)$, as discussed in the previous section. In principle it is possible to find this by enumerating all possible policies π, plotting the error-cost performances $\{\rho(\pi), \epsilon(\pi))\}$ in the error-cost plane, and producing an operational error-cost function for our scenario. Unfortunately, this is infeasible since the number of states is on the order of $(2^N)^N$, and hence the number of policies is on the order of $2^{(2^N)^N}$.

Rather than evaluating all possible policies π, we evaluate only a subset of them, taking an on-line, incremental approach with two steps of look-ahead, as follows. At the current transmission opportunity t_i, the server evaluates only four policies, which are all consistent with the history $(a_0, o_0) \circ (a_1, o_1) \circ \cdots \circ (a_{i-1}, o_{i-1})$ leading up to state q_i at time t_i. These four policies are denoted $[a_i, a_{i+k}] =$ [0,0], [1,0], [0,1], and [1,1], where a_i is the action to be taken at the current time t_i and, unless a final state is reached in the interim, a_{i+k} is the action to be taken at a future time t_{i+k}. Of these four policies, the policy π^* is sought that minimizes $\epsilon(\pi) + \lambda'\rho(\pi)$, where $\rho(\pi)$ and $\epsilon(\pi)$ are calculated conditioned on q_i. Then a_i is set to the first action $\pi^*(q_i)$ of π^*, and the procedure is repeated at each successive transmission opportunity until a final state is reached. We furthermore allow $\lambda' = \lambda B_l / S_l$ to change at each step to reflect the updated sensitivity S_l, which we adjust at every transmission opportunity using the ISA algorithm of the previous section. It should be noted that if $t_{i+k} \geq t_{DTS}$, then it does not make sense to consider a second transmission at time t_{i+k}. In that case we evaluate only two policies, denoted $[a_i] =$ [0] and [1], which are respectively equal to $[a_i, a_{i+k}] = [0, 0]$ and $[1, 0]$, for $t_{i+k} < t_{DTS}$. If there are no previous acknowledgements, i.e., $o_0 = o_1 = \cdots = o_{i-1} = \emptyset$, then the error-cost

performances of the four policies [0,0], [1,0], [0,1], and [1,1] can be expressed

$$\begin{aligned}
\rho([a_i, a_{i+k}]) &= a_i + a_{i+k} \prod_{j \le i: a_j = 1} (\epsilon_P + (1-\epsilon_P) P\{RTT' > t_{i+k} - t_j | RTT' > t_i - t_j\}) \\
\epsilon([a_i, 0]) &= \prod_{j \le i: a_j = 1} P\{FTT'' > t_{DTS} - t_j | RTT' > t_i - t_j\} \\
\epsilon([a_i, 1]) &= \epsilon([a_i, 0]) \left(\epsilon_P^{(1)} + (1-\epsilon_P^{(1)}) P\{FTT' > t_{DTS} - t_{i+k}\}\right) \\
&\quad + P\{FTT' \le t_{DTS} - t_{i+k}\} \left(\epsilon_P - \epsilon_P^{(1)}\right) \\
&\quad \times \prod_{j \le i: a_i = 1} (\epsilon_P P\{RTT' > t_{i+k} - t_j | RTT' > t_i - t_j\} \\
&\quad + (1-\epsilon_P) P\{FTT' > t_{DTS} - t_j | RTT' > t_i - t_j\})
\end{aligned}$$

where $\epsilon_P^{(1)}$ is the probability that a parity packet arriving after a systematic packet cannot be successfully decoded. Derivation of these expressions, as well as derivations for the error-cost performance of the four policies in the case where there are previous NAKs in the transmission history, can be found in [3].

6 Experimental Results

In this section we show the results of error-cost performances computed for different bit error rates and different transmission histories. In our experiments we use $T = 100$ ms as the nominal time between transmission opportunities $t_0, t_1, \ldots, t_i$. We set $t_{i+k} = t_i + 2T$ with $t_{DTS} \ge t_{i+k} + T$. The forward and backward communication channels are symmetric and are given by $\epsilon_F = \epsilon_B = 10\%$ and $p_F(t) = p_B(t) = \delta(t - \tau_F)$, where $\tau_F = T$. That is, unless there is loss, $FTT = T$ and $RTT = 2T$. We examine bit error rates $BER \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$ for the wireless link. We assume a payload size $N_p = 4016$ bits for data packets on the forward channel, a payload size $N_p = 0$ for acknowledgement packets on the backward channel, and a packet header size $N_h = 320$ bits for both data and acknowledgement packets. We consider three history cases: (1) no previous transmissions, i.e., $a_0 = a_1 = \cdots = a_{i-1} = 0$, (2) previous transmissions but no previous NAKs, i.e., $o_0 = o_1 = \cdots = o_{i-1} = \emptyset$, and (3) previous transmissions as well as previous NAKs.

Figure 4 shows error-cost functions in the case of no previous transmissions, for various bit error rates. It can be seen that for policy [0,0], the expected cost is 0 and the expected error is 1, since nothing is transmitted. The expected error-cost for policies [0,1] and [1,0] is identical, since there are no previous transmissions and transmissions at either t_i or t_{i+k} have the same chance of arriving at the client by t_{DTS}. The expected error is smallest, and conversely the expected cost is largest, for policy [1,1]. The error-cost performance gets worse as the BER increases, until it is nearly useless to send a single systematic packet, since its payload will be corrupted with probability near one.

Figure 5 shows error-cost functions in the cases of previous transmissions with and without previous acknowledgements, for various bit error rates. In particular, Fig-

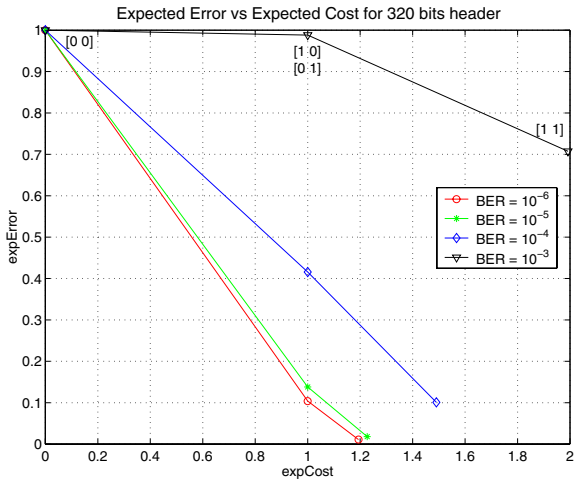

Figure 4: Error-Cost function without previous transmissions.

ures 5a through 5c show error-cost functions for a history of one single transmission, at $t_i - T$ for Figure 5a and at $t_i - 2T$ for Figures 5b and 5c. Histories in the latter two figures are distinguished by their acknowledgments. In Figure 5b the systematic packet transmitted at time $t_i - 2T$ has not been acknowledged by time t_i, while in Figure 5c the packet has been (negatively) acknowledged. Figures 5d through 5f show error-cost functions for a history of two transmissions: a NAKed systematic packet at $t_i - 3T$ and a parity packet at $t_i - T$ for Figure 5d, or a NAKed systematic packet at $t_i - 4T$ and a parity packet at $t_i - 2T$ for Figures 5e and 5f. Histories in the latter two figures are again distinguished by their acknowledgements for the last transmission. In Figure 5e the parity packet transmitted at time $t_i - 2T$ has not been acknowledged by time t_i, while in Figure 5f the packet has been (negatively) acknowledged.

7 Conclusions

A methodology has been presented for computing the expected error-cost function for streaming packetized media to a wireless client. This is an essential part of a larger R-D optimization framework. The computation of the error-cost function is done on a trellis for a Markov decision process with finite horizon N, associated with the transmission scenario under consideration. By exploiting the fact that over short segments of the trellis an analytical solution is still tractable, we are able to compute an approximation to the expected error-cost function and thus overcome the difficulties imposed by the immense complexity of the trellis. This allows us to obtain approximately rate-distortion optimized policies for streaming media over a lossy packet network to a wireless client.

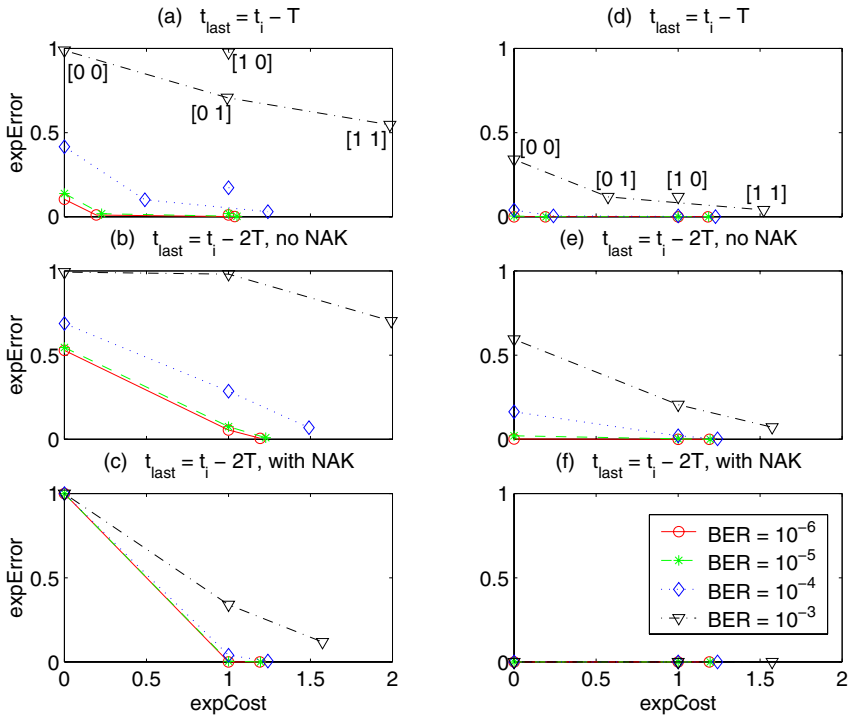

Figure 5: Error-Cost function for transmission scenarios 2 and 3, and 320 bits header.

References

[1] L.A. Larzon, M. Degermark, and S. Pink, *UDP Lite for Real Time Multimedia Applications*, Technical Report HPL-IRI-1999-001, HP Labs, Bristol,United Kingdom, April 1999.

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

[3] P .A. Chou and J. Chakareski, *Application L ayer Error Correction Coding for Improved Performance of Wireless Str eaming Media*, Technical Report MSR-TR-2002-xx, Microsoft Research, Redmond, WA, F ebruary2002.

[4] J. Chakareski and P .A. Chou, *Application L ayer Error Correction Coding for Rate-Distortion Optimized Str eamingto Wireless Clients*, submitted to IEEE Intern. Conf. on Acoustics, Speech, and Signal Proc. (ICASSP) 2002, Orlando, FL, USA, May 2002.

[5] N. Seshadri and C. Sundberg, *List Viterbi de co dingalgorithms with applications*, IEEE T ransactions onCommunications, pp. 313-323, February 1994.

[6] C. Heegard and S.B. Wicker, *T urbo Coding*, Kluwer Academic Publishers, January 1999.