Transactions Papers

Application Layer Error-Correction Coding for Rate-Distortion Optimized Streaming to Wireless Clients

Jacob Chakareski and Philip A. Chou, *Fellow, IEEE*

***Abstract*—This paper addresses the problem of streaming packetized media over a lossy packet network to a wireless client, in a rate-distortion optimized way. We introduce an incremental redundancy error-correction scheme that combats the effects of both packet loss and bit errors in an end-to-end fashion, without support from the underlying network or from an intermediate base station. The scheme is employed within an optimization framework that enables the sender to compute which packets it should send, out of all the packets it could send at a given transmission opportunity, in order to meet an average transmission-rate constraint while minimizing the average end-to-end distortion. Experimental results show that our system is robust and maintains quality of service over a wide range of channel conditions. Up to 8 dB performance gains are registered over systems that are not rate-distortion optimized, at bit-error rates as large as 10^{-2}.**

***Index Terms*—Application layer, audio coding, channel coding, error correction, Internet, Markov processes, multimedia communication, optimal control, protocols, rate distortion, video coding.**

I. Introduction

WE CONSIDER the problem of streaming packetized media over a lossy backbone packet network through a base station to a wireless client. Packets may be lost in the backbone network due to congestion, or they may be corrupted in the wireless link due to fading. The introduction of new hardware and standards for wireless networking [1]–[4] has made streaming of packetized media over the Internet with a wireless last hop a present reality. However, the latest wireless networking standards do not offer adequate support for multimedia communication. The absence of quality-of-service (QoS) support makes it impossible for a multimedia application to effect the link-layer error control for its traffic. Therefore, critically important media packets receive the same error-control treatment as optional media packets. Indeed, all media packets, which are generally error-tolerant but delay-sensitive, receive the same error-control treatment as data packets, which are generally error-intolerant and delay-insensitive. This represents a significant constraint, since the performance of multimedia applications can be significantly improved by some degree of cross-layer awareness.

One mechanism for providing adequate cross-layer awareness is QoS support. Streaming multimedia on demand over networks with support for multiple QoSs, e.g., DiffServ [5], is studied rigorously in [6]. Other papers that study this issue are [7]–[11].

In this paper, we assume a different mechanism for cross-layer awareness in wireless systems, which may be easier to implement. In particular, we assume that the client application may observe bit errors in the packet payload. That is, 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 [12], in which the transport checksum is applied only to the header,[1] 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 application-layer 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 special support for media communication.

We propose a streaming system based on a hybrid forward error correction/automatic repeat request (FEC/ARQ) error-control scheme known as incremental redundancy (IR) transmission. Hybrid FEC/ARQ schemes have been introduced with the purpose of exploiting both the predictable performance of FEC codes and the coding-rate flexibility of ARQ protocols [13]–[16]. Hybrid FEC/ARQ schemes that use rate-compatible FEC codes, where the higher rate codes are embedded in the

Paper approved by L. Rasmussen, the Editor for Iterative Detection, Decoding, and ARQ of the IEEE Communications Society. Manuscript received October 23, 2002; revised March 26, 2003 and December 3, 2003. This paper was presented in part at the Data Compression Conference, Snowbird, UT, April 2002, and in part at the International Conference on Acoustics, Speech, and Signal Processing, Orlando, FL, May 2002.

J. Chakareski is with the Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: cakarz@stanford.edu).

P. A. Chou is with Microsoft Research, Microsoft Corporation, Redmond, WA 98052-6399 USA (e-mail: pachou@microsoft.com).

Digital Object Identifier 10.1109/TCOMM.2004.836436

[1]Header checksum only will become part of the new standard for the broadcast mode in 3GPP in a few months.

lower rate codes, are known as IR schemes, in that the parity symbols are incrementally transmitted to adaptively meet the error-performance requirements of the system. In particular, Hagenauer proposed rate-compatible punctured convolutional (RCPC) codes with such an application in mind [14]. For a comprehensive overview of RCPC/ARQ protocols, the reader is referred to [15] or [16]. The use of turbo codes [17]–[19] in an ARQ protocol was first proposed in [20], in which additional constituent encoders were added to obtain rates lower than that of a conventional turbo encoder $(1/2)$. This is similar to the approach used in our paper. Rates above $1/2$ were considered in [21], where the authors used rate-compatible punctured turbo (RCPT) codes.

The IR transmission scheme in our paper operates end-to-end, and hence, does not need QoS support from the underlying network infrastructure nor from the intermediate base station. Instead, the scheme inserts unequal amounts of redundancy at the application layer, exploiting the ability of the client to observe corrupted payloads. In the IR scheme, the server can incrementally increase the amount of transmitted redundancy in response to negative acknowledgment (NAK) fed back from the client. Hence, the IR scheme is adaptive to the error characteristics of the network. Furthermore, the scheme uses the code-combining principle [22], [23] to ensure that none of the received redundancy information is discarded, which makes it appealing from a bandwidth-efficiency viewpoint.

Optimal use of the IR transmission scheme is determined by an optimization procedure known as the iterative sensitivity adjustment (ISA) algorithm [24], [25]. Since all packets sent in either direction are subject to stochastic loss, delay, and corruption, 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 and the optimization procedure, the server is able to transmit its media and redundancy 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,[2] results in unequal error protection provided to different portions of the media stream. The core step of the optimization procedure involves trading off the expected redundancy (the cost used to communicate the packet) for the probability that a single media packet will be communicated in error.

In summary, the major contributions of this paper are the introduction of an IR scheme for application-layer error control in multimedia streaming to wireless clients, and an extension of our previous rate-distortion optimization framework to accommodate this new scheme. Parts of this work have already been presented in [26] and [27].

We present the major ideas in our paper as follows. In Section II, we define our abstractions of the encoding, packetization, and communication processes. In Section III, we present in detail our IR transmission scheme. In Section IV, we show how the entire multimedia presentation can be transmitted in a rate-distortion optimized way, using as a building block an algorithm for rate-distortion optimized transmission of a single media packet. This algorithm is the subject of Section V. In Section VI, we report our experimental results. Using simulations, the performances of the algorithm from Section V and of our system as a whole are closely examined. We show up to 8-dB gains over state-of-the-art systems, at bit-error rates (BERs) as large as 10^{-2}.

II. 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. Associated with each data unit l is a size B_l, a decoding time $t_{\mathrm{DTS},l}$, and an importance Δd_l. The size B_l is the size of the data unit in bytes. $t_{\mathrm{DTS},l}$ is the *delivery deadline* by which data unit l must arrive at the client, or be too late to be usefully decoded. Packets containing data units that arrive after the data 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 a composite 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.[3] However, if the packet is not lost, then it arrives at the client device at time t', where the forward trip time $\mathrm{FTT} = t' - t$ 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.[4] 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 IP/UDP/RTP header has been corrupted. If N_h is the size of the header in bits, then $\epsilon_H = 1 - (1 - \mathrm{BER}_F)^{N_h}$ 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 $P\{\mathrm{FTT}' > \tau\} = \epsilon'_F + (1 - \epsilon'_F) \int_\tau^\infty p_F(t)\, dt$ 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

[2]In this paper, a *transmission policy* is an algorithm for transmitting data, following the terminology of Markov decision processes, in which a *policy* is a mapping of each state (in a finite-state machine) to an action, which together with the state and the next observation, determines the next state.

[3]Our assumption that the packet losses are independent is for the purposes of tractability of our analysis, which requires independence of the packet transmissions related to any given data unit. This is justified, provided that the retransmissions are infrequent (e.g., the retransmission intervals are larger than the one-way delay), as noted for example by Bolot [28].

[4]Our assumption that the bit errors are independent and identically distributed (i.i.d.) is principally for the purposes of simulation and is not a critical part of our analysis; a bursty bit-error channel that gives the same header and payload error rates should produce roughly the same results.

may still be corrupted with probability $\epsilon_P = 1-(1-\mathrm{BER}_F)^{N_p}$, where N_p is the size of the payload in bits.

In our IR scheme, the client application immediately transmits either a positive or negative acknowledgment (ACK/NAK) 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 BER_B. These induce a modified probability ϵ'_B of packet loss on the backward channel, and a distribution $P\{\mathrm{BTT}' > \tau\}$ 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 $P\{\mathrm{RTT}' > \tau\} = \epsilon'_R + (1-\epsilon'_R)\int_\tau^\infty p_R(t)\,dt$, where $p_R = p_F * p_B$ is the convolution of p_F and p_B. Note that $P\{\mathrm{RTT}' > \tau\}$ is the probability that the server does not receive an acknowledgment packet (positive or negative) by time $t+\tau$ for a data packet transmitted by the server at time t. Finally, it should be pointed out that the BER of the BSC channel is, in fact, the residual BER[5] after error-correction decoding has been performed at the physical and link layers at the client or at the base station.

III. IR Transmission

A server implementing our IR scheme enables a client to exploit corrupted payloads by sending to the client redundancy packets (henceforth denoted *parity* packets), in addition to the regular data units (henceforth denoted *systematic* packets). A parity packet contains the parity bits of a codeword obtained when a systematic error-correction code is applied to a systematic packet. Parity packets are sent only if necessary. Fig. 1 illustrates the packet-generation process.

First, the server augments the data unit with a binary cyclic redundancy check (CRC) code having a generator polynomial $g(x)$. This enables the client to verify the integrity of the data unit. It is assumed that bit errors cannot go unnoticed by the CRC code. The CRC-encoded data unit represents a systematic packet, denoted Packet 0 in Fig. 1, and is created and transmitted at the first transmission decision for the data unit.

The server may transmit systematic packets periodically if necessary, until it receives an ACK/NAK from the client or until the server gives up transmitting the data unit. Once the server receives a NAK, it may begin to transmit parity packets periodically if necessary, until it receives a positive acknowledgment (ACK) or until it gives up. A NAK indicates that the client received the packet but was unable to decode the data unit. An ACK indicates that the client received the packet and successfully decoded the data unit. The server generates the parity packets, denoted Packets k ($k \geq 1$) in the figure, by applying a rate-$1/2$ recursive systematic convolutional (RSC) code with a generator matrix $G(x)$ to different pseudorandom interleavings I_{k-1} of the data unit and its CRC, as shown. Note that there is no interleaving involved when Packet 1 is created.

The client attempts to decode each packet that it receives, using the CRC for systematic packets, or using the list Viterbi algorithm [29] for the first parity packet,[6] or using the turbo decoding algorithm [19] for subsequent parity packets. The client immediately transmits an ACK packet to the server upon successful decoding, or a NAK packet upon decoding failure. The header of the acknowledgment packet identifies the data packet being positively or negatively acknowledged; the payload of the acknowledgment packet is empty.

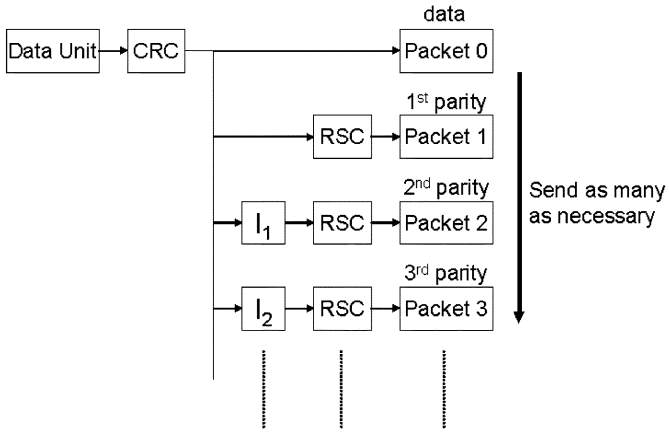

Fig. 1. IR transmission scheme.

Let $\epsilon_P^{(n)}$ denote the probability of the event $E^{(n)}$ that it takes more than n parity packets, *arriving at the client* after a systematic packet, for successful decoding. Clearly, the events $E^{(0)} \supseteq E^{(1)} \supseteq E^{(2)} \supseteq \cdots$ are nested. Thus, $\epsilon_P = \epsilon_P^{(0)} \geq \epsilon_P^{(1)} \geq \epsilon_P^{(2)} \geq \cdots$, and the conditional probability that the nth parity packet that arrives at the client will fail to decode, given that the previous parity packets have failed to decode, is $\epsilon_P^{(n)}/\epsilon_P^{(n-1)}$. In Section VI, we evaluate these quantities empirically, and use them in our optimization procedure.

The server may transmit a systematic or parity 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 a given transmission opportunity is determined by the server's transmission policy for the data unit, which takes into account any 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 chosen to minimize the average end-to-end distortion, subject to a constraint on the average transmission rate.

IV. $R-D$ Optimization Using the ISA Algorithm

Suppose there are L data units in the multimedia presentation. 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. Π corresponds to a family of transmission policies defined precisely in the next section. Any given policy vector $\boldsymbol{\pi}$ induces an expected distortion $D(\boldsymbol{\pi})$ and an expected number of transmitted bytes $R(\boldsymbol{\pi})$ for the multimedia presentation. We seek the policy vector $\boldsymbol{\pi}$ that minimizes

[5]These are the bits that are still in error after decoding.

[6]It would also be possible to decode the first parity packet using the *a posteriori* probability (APP) module used for the turbo decoding, which maximizes the APP of each symbol. However, the Viterbi algorithm tends to provide a lower block-error rate. List decoding decreases the block-error rate still further.

$D(\boldsymbol{\pi})$ subject to a constraint on $R(\boldsymbol{\pi})$. This can be achieved by minimizing the Lagrangian $J(\boldsymbol{\pi}) = D(\boldsymbol{\pi}) + \lambda R(\boldsymbol{\pi})$ for some Lagrange multiplier $\lambda > 0$, thus achieving a point on the lower convex hull of the set of all achievable distortion-rate pairs.

We now compute expressions for $R(\boldsymbol{\pi})$ and $D(\boldsymbol{\pi})$. The expected number of transmitted bytes $R(\boldsymbol{\pi})$ is the sum of the expected number of bytes transmitted 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 number of transmitted bytes per source byte (under policy π_l), called the *expected cost*. The expected distortion $D(\boldsymbol{\pi})$ 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 $\Delta d_l \prod_{l' \preceq l}(1 - \epsilon(\pi_{l'}))$ is the expected reduction in reconstruction error associated with data unit l. Here, $l' \preceq l$ means that l depends directly or indirectly on l'.

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. Specifically, the expression for the expected distortion cannot be split into a sum of terms, each depending on a single π_l. Therefore, we employ an iterative descent algorithm, called the ISA algorithm, in which we minimize the objective function $J(\pi_1, \ldots, \pi_L)$ one variable at a time, while keeping the other variables constant, until convergence [24]. 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 $S_l^{(n)} = \sum_{l' \succeq l} \Delta d_{l'} \prod_{l'' \preceq l', l'' \neq l}(1 - \epsilon(\pi_{l''}^{(n)}))$ is 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.[7] Another interpretation of $S_l^{(n)}$ is as the partial derivative of (2) with respect to $\epsilon_l = \epsilon(\pi_l)$, evaluated at $\boldsymbol{\pi}^{(n)}$.

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. The error-cost function can be considered as a normalized distortion-rate function pertaining to the transmission of a single dimensionless data unit, and it depends only on the transmission scenario and the channel characteristics. In the next section, we show how to compute the expected error-cost function for the family of transmission policies corresponding to the IR scheme.

[7]Here, $\sum_{l' \succeq l}$ denotes a summation over the data units l' that depend directly or indirectly on l. Likewise, $\prod_{l'' \preceq l', l'' \neq l}$ denotes a product over the data units $l'' \neq l$ on which l' depends, directly or indirectly.

V. 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 III, 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.[8]

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 acknowledgments received by the server in the interval $(t_i, t_{i+1}]$. For example, $o_i = \{\mathrm{NAK}_{j_1}, \mathrm{ACK}_{j_2}\}$ 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 Fig. 2. Therefore, a state uniquely represents this sequence of action-observation pairs. If the final observation o_i includes an ACK, then q_{i+1} is a final state. In addition, any state at time $t_N = t_{\mathrm{DTS}}$ is a final state. Final states in Fig. 2 are indicated by double circles.

The action a_i taken at a nonfinal state q_i determines the transition probability $P(q_{i+1}|q_i, a_i)$ to the next state q_{i+1}. For example, in Fig. 2, 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, $P\{\mathrm{RTT}' \leq t_1 - t_0\}(1 - \epsilon_P), P\{\mathrm{RTT}' > t_1 - t_0\}, P\{\mathrm{RTT}' \leq t_1 - t_0\}\epsilon_P$, and 0. (Recall that $P\{\mathrm{RTT}' > t_1 - t_0\}$ 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 nonfinal 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.

[8]As pointed out by an anonymous referee, other protocols may make sense as well. For example, in the low BER regime, it may make sense for the server to begin sending parity packets even before it has received a NAK (or an ACK), since even if the client does not first receive a systematic packet, it can successfully decode a single parity packet if the parity packet is error free (which can be verified by checking the CRC after a trial decoding). However, we have chosen a scheme that ensures delivery of at least one systematic packet to the client. The two schemes have almost identical performance in the low BER regime, while our scheme has superior performance in the high BER regime, because the client will never be forced to decode corrupted parity packets only, which is not necessarily true for the alternative scheme outlined above. Nonetheless, one may wish to consider different protocols, depending on the expected channel parameters.

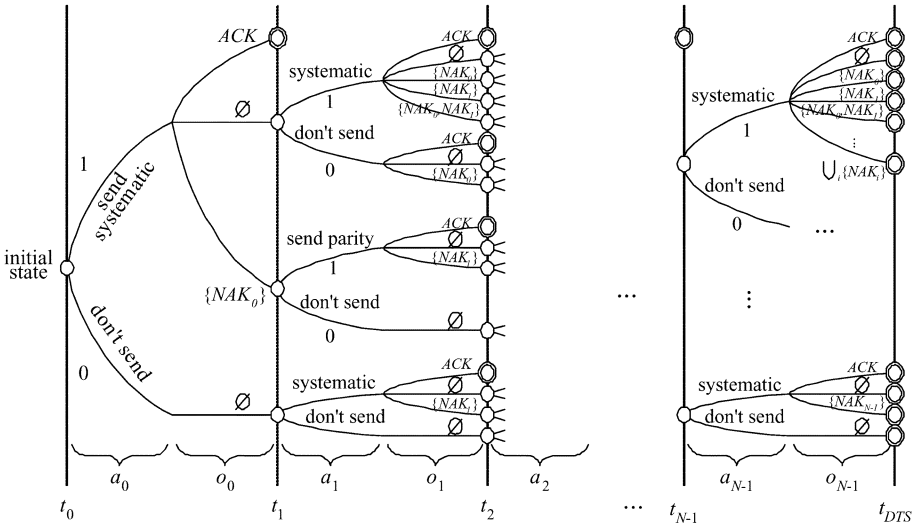

Fig. 2. State space for a Markov decision process.

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 packets transmitted under policy π 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 Fig. 2, then a systematic packet was sent at every transmission opportunity $t_0, t_1, \ldots, t_{N-1}$ and no acknowledgment was received. In that case, $\epsilon_\pi(q_F) = \prod_{i=0}^{N-1} P\{\mathrm{FTT}'' > t_{\mathrm{DTS}} - t_i | \mathrm{RTT}' > t_{\mathrm{DTS}} - t_i\}$, where $P\{\mathrm{FTT}'' > \tau\} = P\{\mathrm{FTT}' > \tau\} + P\{\mathrm{FTT}' \leq \tau\}\epsilon_P$ is the probability that a packet transmitted at time t either does not arrive at the client application by time $t + \tau$, or else its payload is corrupted. Therefore, Π is a collection of all possible paths through the state space described in Fig. 2.

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 unfeasible, 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 online, incremental approach with two steps of look-ahead. More than two steps of look-ahead unfortunately becomes mathematically intractable. Thus, 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 $t_{i+k} \geq t_{\mathrm{DTS}}$, 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 $t_{i+k} < t_{\mathrm{DTS}}$. Note that three cases of transmission history can be differentiated: 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.

As shown in the Appendix, the expressions for the error-cost performances of the four policies $[0,0], [1,0], [0,1]$, and $[1,1]$ for transmission history cases 1) and 2) can be represented together as

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

where

$$P_{\mathrm{ACK}}([a_i]) = 1 - \prod_{j \le i: a_j = 1} (\epsilon_P + (1-\epsilon_P) \times P\{\mathrm{RTT}' > t_{i+k} - t_j \,|\, \mathrm{RTT}' > t_i - t_j\}).$$

The conditional probabilities in these expressions can be computed simply by converting them to unconditional probabilities using Bayes' rule, e.g., $P\{\mathrm{FTT}' > \tau + \delta | \mathrm{RTT}' > \tau\} = P\{\mathrm{FTT}' > \tau + \delta\}/P\{\mathrm{RTT}' > \tau\}$.

Unfortunately, expressions for the error-cost performances of the four policies for the transmission history case 3) are intractably complex for general delay densities. However, when the forward and backward delays are constant (i.e., $\mathrm{FTT} = \tau_F$ and $\mathrm{BTT} = \tau_B$ for constants τ_F and τ_B), then exact expressions for the error-cost performances are possible

$$\epsilon([a_i, a_{i+k}]) = P_{\mathrm{sp}} \sum_{n_1=0}^{N_1} p(n_1) \sum_{n_2=0}^{N_2} p(n_2) \times \sum_{n_3=0}^{N_{\mathrm{new}}+a_i+a_{i+k}} p_{a_i+a_{i+k}}(n_3) \frac{\epsilon_P^{(N_0+n_1+n_2+n_3)}}{\epsilon_P^{(N_0+n_1)}}$$

$$\rho([a_i, a_{i+k}]) = a_i + a_{i+k}(1 - P_{\mathrm{ACK}}([a_i]))$$

where $P_{\mathrm{sp}} = (P\{\mathrm{FTT}'' > \tau_F | \mathrm{RTT}' > \tau_R\})^{N_{s1}}$, $N_{s1}, N_1, N_2, N_{\mathrm{new}}$, and N_0 are numbers that depend on the transmission history, $p(n_1), p(n_2)$, and $p_{a_i+a_{i+k}}(n_3)$ are binomial distributions, and $P_{\mathrm{ACK}}([a_i])$ for this case is defined in the Appendix. We evaluate $\epsilon_P^{(n)}$ empirically in the first part of the next section.

VI. Experimental Results

In this section, we report our experimental results. First we examine the performance of the IR scheme. Then, we analyze the error-cost optimized transmission of a single data unit for the three scenarios of transmission history, using the equations from the previous section. Finally, we evaluate the end-to-end distortion-rate performance of our system for streaming of an entire audio presentation.

A. Performance of the IR Transmission Scheme

We assess the performance of the IR scheme by evaluating the probability of decoding failure (PoDF), introduced earlier as $\epsilon_P^{(n)}$. As explained, this is the probability of the event $E^{(n)}$ that it takes more than n parity packets, arriving at the client after a systematic packet, for successful decoding. The PoDF were obtained through extensive simulations. Each simulation run consisted of a transmission of a set of 1000 data units, each of size 500 bytes. The data units were transmitted through a BSC, with a different BER for every simulation run. The set of BER values that was used is $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$.

We used a CRC code with generator polynomial $g(x) = x^{16} + x^{15} + x^2 + 1$ [30]. The RSC code was a rate-$1/2$ binary code with generator polynomial matrix $G(x) = [1 \;\; (x^2 + x + 1)/(x^2 + 1)]$ [31]. For every data unit, there could be at most 10 (1 systematic+9 parity) packets sent by the server and received by the client at the other side of the communication channel. This is an overestimate of the maximum number of packets needed for successful decoding, as the results will show later on. The list size of the Viterbi decoder was also set to 10, as was the maximum number of iterations for the turbo decoding algorithm. The first parity packet was obtained by terminating the trellis of the RSC code with two tail bits. The trellises for the successive parity packets were not terminated.

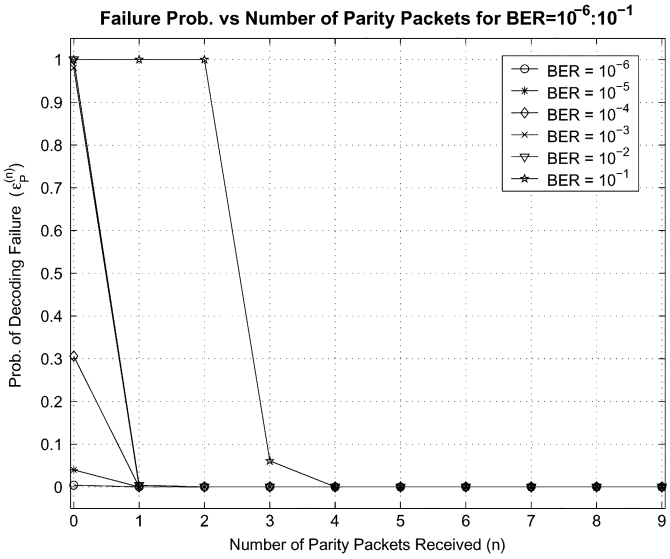

Fig. 3. PoDF versus number of parity packets received.

For every data unit in a simulation run, given a channel realization with a particular BER value, we recorded the number of parity packets needed for successful decoding. The probabilities of decoding failure $\epsilon_P^{(n)}$ for a given BER value were calculated using histograms of these numbers over all data units.

It can be seen from Fig. 3 that for BERs $\le 10^{-2}$, at most two parity packets are needed for decoding. On the whole, including $\mathrm{BER} = 10^{-1}$, not more than four parity packets suffice to observe success in decoding in every instance. In [32], we also evaluate the PoDF as a function of the number of paths for the list Viterbi decoder, and the number of iterations for the turbo decoding algorithm. Those results are not included here, due to space considerations.

B. Error-Cost Performances

In this subsection, we examine the expected error-cost function for transmission of a single data unit, computed for different BERs and different transmission histories, using the probabilities of decoding failure evaluated in the previous section. 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 $t_{\mathrm{DTS}} \ge t_{i+k} + T$. The latter inequality ensures that a packet transmitted at time t_{i+k} has a nonzero probability of arriving at the client on time; the exact value of t_{DTS} is otherwise immaterial. 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, $\mathrm{FTT} = T$ and $\mathrm{RTT} = 2T$. We examine residual bit-error rates $\mathrm{BER} \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$ 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 acknowledgment packets on the backward channel, and a packet header size $N_h = 320$ bits for both data and acknowledgment packets.

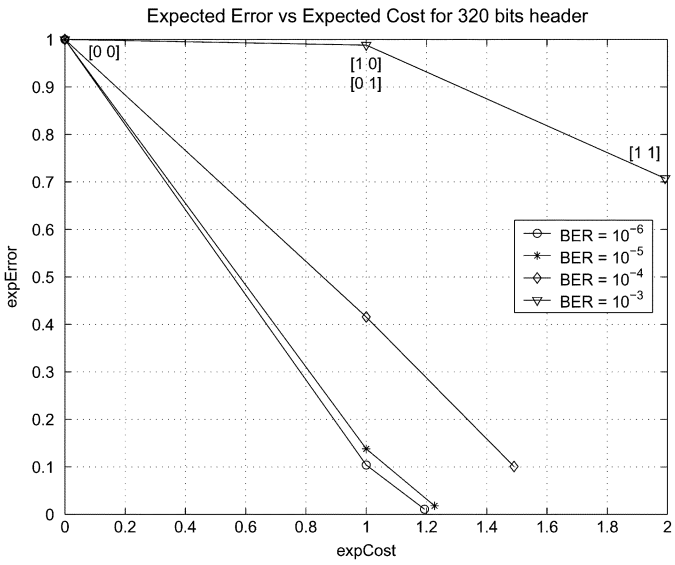

Fig. 4. Error-cost function without previous transmissions.

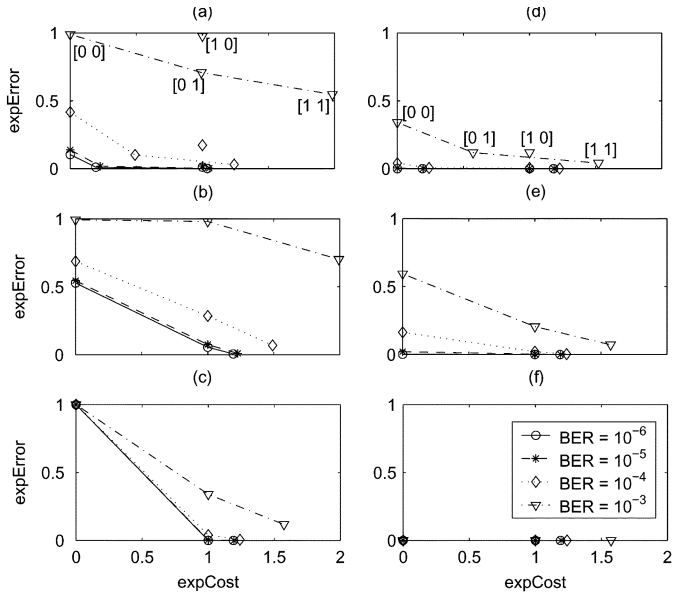

Fig. 5. Error-cost function for transmission history cases 2) and 3).

Fig. 4 shows error-cost functions in the case of no previous transmissions, for various BERs. 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 performances for policies $[0,1]$ and $[1,0]$ are identical, since there are no previous transmissions, and transmission at either t_i or t_{i+k} has 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.

Fig. 5 shows error-cost functions in the cases of previous transmissions with and without previous acknowledgment, for various BERs. In particular, Fig. 5(a)–(c) show error-cost functions for a history of one single transmission of a systematic packet, at time $t_i - T$ for Fig. 5(a), and at time $t_i - 2T$ for Fig. 5(b) and (c). Histories in the latter two figures are distinguished by their acknowledgment. In Fig. 5(b), the previous transmission is not acknowledged, while in Fig. 5(c), the previous transmission is negatively acknowledged, by time t_i. Fig. 5(a) shows, in contrast to Fig. 4, that the expected error for policy $[0,0]$ is generally not 1 (unless the BER is sufficiently high), because of the earlier transmission of a systematic packet at time $t_i - T$, which is likely to be received without corruption. However, Fig. 5(b) and (c) show that as time progresses and the earlier transmission falls further into the past at time $t_i - 2T$, if there is no ACK for the systematic packet, then the expected error rises, while if there is a NAK, then the expected error returns to 1 for policy $[0,0]$. However, the expected error for the other policies in Fig. 5(c) is generally lower than in Fig. 5(b) because, as the NAK for time $t_i - 2T$ indicates, the client possesses a systematic packet that can dramatically decrease the expected error when the next parity packet is sent, especially when the BER is high. Fig. 5(a) also shows, in contrast to Fig. 4, that the error-cost performances for the policies $[1,0]$ and $[0,1]$ are not identical. As long as it is still possible to receive an ACK for the previous transmission, it is preferable to delay sending the next packet. In fact, the policy $[1,0]$ does not even lie on the lower convex hull. The error-cost performances of the policies $[1,0]$ and $[0,1]$ collapse again in Fig. 5(b) and (c).

Fig. 5(d)–(f) show error-cost functions for a history of *two* transmissions: a NAK'd systematic packet and a parity packet, respectively, at times $t_i - 2T$ and $t_i - T$ for Fig. 5(d), and, respectively, at times $t_i - 3T$ and $t_i - 2T$ for Fig. 5(e) and (f). Again, the latter two figures are distinguished by their acknowledgment. In Fig. 5(e), the parity packet is NAK'd, while in Fig. 5(f), it is not NAK'd, by time t_i. The figures are similar to Fig. 5(a)–(c), except that the error-cost performances in Fig. 5(d)–(f) are generally lower than in Fig. 5(a)–(c), because the NAK'd systematic packet in the histories of Fig. 5(d)–(f) indicates that the client possesses a systematic packet that can dramatically decrease the expected error when the next parity packet is sent. It is interesting to note that the performances of policies $[0,0]$ and $[0,1]$ from Fig. 5(d) match the performances of policies $[1,0]$ and $[1,1]$ from Fig. 5(c).

C. Distortion-Rate Optimized Streaming

Here we investigate the end-to-end distortion-rate performance for streaming one minute of packetized audio content using different methods. The audio content, the first minute of Sarah McLachlan's *Building a Mystery*, is compressed using a scalable version of the Windows Media Audio codec. The codec produces a group of 12 500-byte data units every 0.75 seconds for a maximum data rate of 64 kb/s. All twelve data units in the mth group receive the same decoding timestamp, equal to $0.75m$.

We compare two streaming systems. Both of them perform rate-distortion optimized scheduling of the packet transmissions at the sender, with the Lagrange multiplier λ fixed for the entire presentation. In each one of them, the server uses its history of previous transmissions, as well as its history of acknowledgment to determine which packets to transmit (or retransmit) at a transmission opportunity. The first system, *ACK-only*, is the system

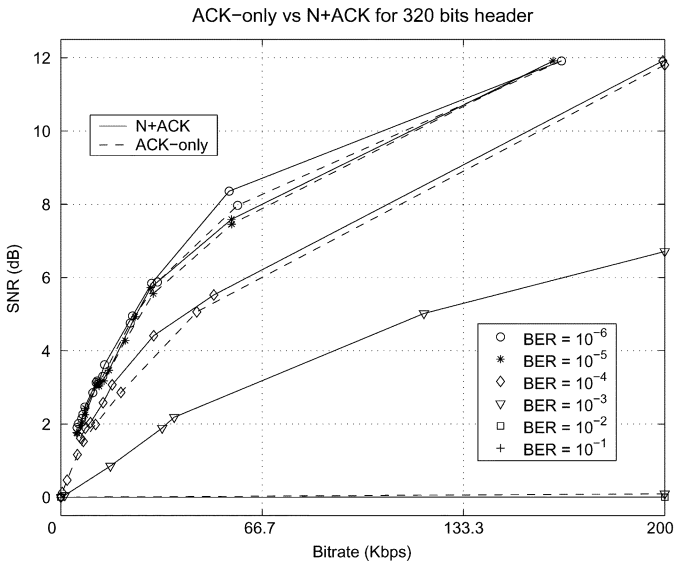

Fig. 6. R–D performance for *ACK-only* (dashed) versus ACK + NAK (solid). Header size $= 320$ b, and BER $= 10^{-6} : 10^{-1}$.

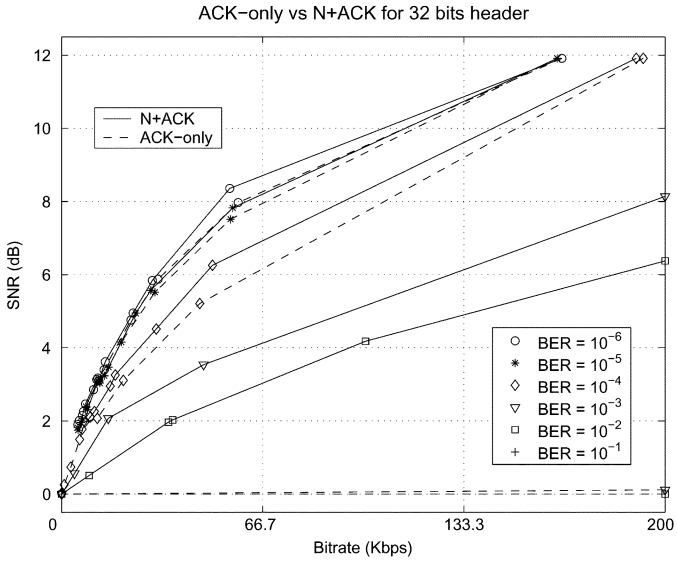

Fig. 7. R–D performance for *ACK-only* (dashed) versus ACK + NAK (solid). Header size $= 32$ b, and BER $= 10^{-6} : 10^{-1}$.

introduced in [24] and [25]. Here the server can transmit only data packets to the client, and only ACK can be sent back to the server upon receipt of packets by the client. The second system, ACK + NAK, is the system described in this paper. In addition to data packets, here the server can also transmit parity packets. Furthermore, the client can also respond with a NAK upon receipt of a packet.

Both streaming systems use the same playback delay (750 ms). The communication channel is specified with the following parameters: $\epsilon_F = \epsilon_B = 10\%, T = 100$ ms, $\kappa_F = \kappa_B = 50$ ms $(0.5T), n_F = n_B = 2$ nodes, and $1/\alpha_F = 1/\alpha_B = 25$ ms $(0.25T)$, where as the delay density $p_{F(B)}$, we use the shifted Gamma distribution with parameters $(n_{F(B)}, \alpha_{F(B)})$ and right shift $\kappa_{F(B)}$. Transmitted packets are dropped at random, with the modified drop rate, ϵ'_F, to account for the nonzero BER. Those packets that are not dropped receive a random delay, and their payload is corrupted with the given BER of the channel, using a pseudorandom number generator. The pseudorandom number generator is initialized to the same seed for each of the systems under comparison. Two cases for the packet header size are studied: regular (320 bits) and compressed (32 bits). A compressed IP/UDP/RTP header size of 32 bits is typical using header compression schemes such as [33].

We examine the signal-to-noise ratio (SNR) in decibels of the end-to-end perceptual distortion, averaged over the one-minute-long audio clip, as a function of the available transmission rate (kb/s). Fig. 6 illustrates the SNR performance of the two systems for the case of regular header size. It can be seen from the figure that for low BERs, both systems perform similarly, with the ACK + NAK system doing a bit better than the *ACK-only* system. The difference in performance becomes more exaggerated when the BER starts increasing. It is interesting to note that for BER $= 10^{-3}$ the performance of the *ACK-only* system collapses, while the ACK + NAK system is still able to maintain a reasonable QoS. Here, the difference in performance increases as the transmission rate increases. The performance gains reach up to 6.7 dB at transmission rate of 200 kb/s. Finally, for BER $> 10^{-3}$, even the performance of the ACK + NAK system collapses. This can be explained by the fact that at such high BERs, every single packet is received with a corrupted header, and thus is dropped by the IP layer at the client side. Hence, the client never gets a chance to see a transmitted packet and exploit the benefits of the IR transmission scheme.

A similar situation is observed for the compressed header case, as shown in Fig. 7. The difference in performance is minimal for low BERs, with the gap in performance becoming more significant as the BER increases. The major difference between the two header size cases is in the performance of the ACK + NAK system for BER $= 10^{-2}$. Due to the smaller header size here, the IR scheme can still be exploited by the client, even at this BER. In other words, for header size of 320 b, the probability that the header will be corrupted at BER $= 10^{-2}$ is near one, while for header size of 32 b, this is significantly less than one. Therefore, some packets actually make it to the client. Maximum performance gains of 8 and 6.4 dB are observed at transmission rate of 200 kb/s for BERs of 10^{-3} and 10^{-2}, respectively. Lastly, for the same reasons as in the first header-size case, even the ACK + NAK system performs poorly at extremely high BERs $(\geq 10^{-1})$.

VII. Conclusion

A system for distortion-rate optimized streaming to wireless clients over lossy packet networks has been presented. The system consists of an IR transmission scheme, which enables the clients to exploit the information in corrupted payloads, and a rate-distortion optimization algorithm for scheduling the transmission of the systematic and parity packets. By taking advantage of the unequal sensitivity of a multimedia presentation to the loss of different components, and the fact that the client can observe corrupted payloads, our system obtains superior performance over existing solutions, and thus uses the available bandwidth in a most cost-effective way. However,

for extremely high BERs (above 10^{-2}), header corruption limits the effectiveness of our system. Further improvement requires that the IP/UDP/RTP headers are protected below the application layer, i.e., at the link, network, and transport layers. Finally, the novel contributions of this paper can be summarized with the following.

- Introduction of an IR error-control scheme for media streaming. The scheme is implemented at the application layer and operates in an end-to-end fashion.
- Derivation of a decision framework that optimizes in a rate-distortion sense the packet transmissions of the IR scheme.

APPENDIX

In this appendix, we derive expressions for the error-cost performances of the four policies, $[a_i, a_{i+k}] = [0,0], [1,0], [0,1]$, and $[1,1]$ at transmission opportunities t_i and t_{i+k}, for $t_{i+k} < t_{\rm DTS}$. For $t_{i+k} \geq t_{\rm DTS}$, the error-cost performances of the policies $[a_i] = [0]$ and $[1]$ are, respectively, equal to those of the policies $[a_i, a_{i+k}] = [0,0]$ and $[1,0]$. To simplify notation in the following, without loss of generality (since t_i and t_{i+k} are arbitrary), we take $k = 1$.

In the next three subsections, we evaluate the error-cost performances of the four policies for the three cases of transmission history: 1) when there are no previous transmissions, i.e., $a_0 = a_1 = \cdots = a_{i-1} = 0$; 2) when there are previous transmissions but no previous NAKs, i.e., $o_0 = o_1 = \cdots = o_{i-1} = \emptyset$; and 3) when there are previous transmissions, as well as previous NAKs.

A. No Previous Transmissions

If there are no previous transmissions of a data unit, then the expected error and cost for the trivial policy $[a_i, a_{i+1}] = [0,0]$ are simply

$$\epsilon([0,0]) = 1, \quad \rho([0,0]) = 0.$$

This is because if a data unit is never transmitted, then the data unit will surely not arrive on time at the client, and no cost will be incurred.

To express the expected error and cost for the other policies, we define $P\{\mathrm{FTT}'' > \tau\}$ to be the probability that a packet transmitted from the server to the client at time t either is not received by the client application by time $t+\tau$ or else its payload is corrupted. That is

$$P\{\mathrm{FTT}'' > \tau\} = P\{\mathrm{FTT}' > \tau\} + P\{\mathrm{FTT}' \leq \tau\}\epsilon_P.$$

Then the expected error-cost performances for the policies $[a_i, a_{i+1}] = [1,0]$ and $[0,1]$ are

$$\epsilon([1,0]) = P\{\mathrm{FTT}'' > t_{\rm DTS} - t_i\}, \quad \rho([1,0]) = 1$$
$$\epsilon([0,1]) = P\{\mathrm{FTT}'' > t_{\rm DTS} - t_{i+1}\}, \quad \rho([0,1]) = 1.$$

This is because if only a single packet for a data unit is transmitted, then the data unit will not arrive on time at the client, uncorrupted, if FTT'' of the packet is greater than the amount of time before the deadline. Furthermore, the cost incurred will be exactly one, i.e., a redundancy of exactly one transmitted byte per information byte.

The expected error-cost performance for the policy $[a_i, a_{i+1}] = [1,1]$ requires a little more work. After the systematic packet is transmitted at time t_i, the acknowledgment observed in the interval $(t_i, t_{i+1}]$ are either $o = \{\mathrm{ACK}_i\}, \{\mathrm{NAK}_i\}$, or $\emptyset$ with respective probabilities $P_{\rm ACK} = P\{\mathrm{RTT}' \leq t_{i+1} - t_i\}(1-\epsilon_P)$, $P_{\rm NAK} = P\{\mathrm{RTT}' \leq t_{i+1} - t_i\}\epsilon_P$, and $P_\emptyset = P\{\mathrm{RTT}' > t_{i+1} - t_i\}$. If $o = \{\mathrm{ACK}_i\}$, then the conditional expected error is zero. If $o = \emptyset$, then another systematic packet is transmitted at time t_{i+1}, and the conditional expected error is the probability that both systematic packets are lost or else have corrupted payloads, i.e., $P\{\mathrm{FTT}'' > t_{\rm DTS} - t_i | \mathrm{RTT}' > t_{i+1} - t_i\} P\{\mathrm{FTT}'' > t_{\rm DTS} - t_{i+1}\}$. However, if $o = \{\mathrm{NAK}_i\}$, then a parity packet is transmitted at time t_{i+1} and the conditional expected error is $P\{\mathrm{FTT}''^{(1)} > t_{\rm DTS} - t_{i+1}\}$, where

$$P\{\mathrm{FTT}''^{(1)} > \tau\} = P\{\mathrm{FTT}' > \tau\} + P\{\mathrm{FTT}' \leq \tau\}\frac{\epsilon_P^{(1)}}{\epsilon_P}$$

is the probability that a parity packet transmitted from the server to the client at time t either is not received by the client application by time $t+\tau$, or else it cannot be successfully decoded using the previously received systematic packet. As explained earlier in Section III, $\epsilon_P^{(1)}/\epsilon_P$ is the conditional probability that the first parity packet that arrives at the client will fail to decode, given that the systematic packet has been received with a corrupted payload. Hence, the error-cost performance for the policy $[a_i, a_{i+1}] = [1,1]$ is

$$\begin{aligned}\epsilon([1,1]) &= P_\emptyset \cdot P\{\mathrm{FTT}'' > t_{\rm DTS} - t_i | \mathrm{RTT}' \\ &\qquad > t_{i+1} - t_i\} P\{\mathrm{FTT}'' > t_{\rm DTS} - t_{i+1}\} \\ &\qquad + P_{\rm NAK} \cdot P\{\mathrm{FTT}''^{(1)} > t_{\rm DTS} - t_{i+1}\} \\ \rho([1,1]) &= P_{\rm ACK} \cdot 1 + P_\emptyset \cdot 2 + P_{\rm NAK} \cdot 2 = 1 + (1 - P_{\rm ACK}).\end{aligned}$$

B. Previous Transmissions Without NAKS

Let $(a_0, o_0) \circ (a_1, o_1) \circ \cdots \circ (a_{i-1}, o_{i-1})$ be the history leading up to the current state q_i at the current transmission opportunity t_i. In this subsection, we assume that the server has received no ACK/NAK up to time t_i, that is, $o_j = \emptyset, j < i$. Thus, whenever $a_j = 1$ for $j < i$, the server has transmitted a systematic packet rather than a parity packet at time t_j. Furthermore, if $a_i = 1$, the server will transmit a systematic packet at time t_i as well.

First let us evaluate the transition probabilities from state q_i at time t_i to state q_{i+1} at time t_{i+1}. Regardless of whether $a_i = 0$ or $a_i = 1$, these transition probabilities have identical expressions (as functions of a_i), so we consider both cases simultaneously.

Observe that either o_i contains an ACK, or else it does not contain an ACK. If o_i contains an ACK, then q_{i+1} is a final state. This happens whenever it is not true that every previously transmitted packet has either had no acknowledgment or else has been NAK'd, by time t_{i+1}. That is

$$P_{\rm ACK} = 1 - \prod_{j \leq i : a_j = 1} P\{\mathrm{RTT}'' > t_{i+1} - t_j | \mathrm{RTT}' > t_i - t_j\} \quad (4)$$

where $P\{\mathrm{RTT}'' > \tau\}$ is the probability that a packet transmitted at time t has either had no acknowledgment or else has been NAK'd by time $t+\tau$, that is

$$P\{\mathrm{RTT}'' > \tau\} = P\{\mathrm{RTT}' > \tau\} + P\{\mathrm{RTT}' \le \tau\}\epsilon_P.$$

In the event that o_i contains an ACK, the conditional expected error (given the final state q_{i+1}) is 0.

On the other hand, if o_i does not contain an ACK, then for each $j \le i$ such that $a_j = 1$, there are only two possibilities: either $\mathrm{NAK}_j \in o_i$ or $\mathrm{NAK}_j \notin o_i$. Thus, there are 2^n possible values of o_i not containing ACKs, where $n = \sum_{j\le i} a_j$ is the number of previously transmitted packets. Each of the 2^n corresponding transitions ends in a nonfinal state q_{i+1} with probability

$$P(q_{i+1}|q_i) = \prod_{j\le i: a_j=1, \mathrm{NAK}_j \notin o_i} A_j \prod_{j\le i: a_j=1, \mathrm{NAK}_j \in o_i} B_j \tag{5}$$

where

$$A_j = P\{\mathrm{RTT}' > t_{i+1} - t_j | \mathrm{RTT}' > t_i - t_j\} \tag{6}$$

is the probability that the packet transmitted at time t_j does not result in any kind of acknowledgment by time t_{i+1}, and

$$B_j = P\{\mathrm{RTT}' \le t_{i+1} - t_j | \mathrm{RTT}' > t_i - t_j\}\epsilon_P \tag{7}$$

is the probability that the packet does result in an acknowledgment, but the acknowledgment is negative.

Now, the conditional expected error (given the nonfinal state q_{i+1}) depends on whether $a_{i+1} = 0$ or $a_{i+1} = 1$. If $a_{i+1} = 0$, then the conditional expected error, given q_{i+1}, is the probability that not one of the non-NAK'd systematic packets transmitted up to and including time t_i reaches the client application with an uncorrupted payload by time t_{DTS}, that is

$$\epsilon_0(q_{i+1}) = \prod_{j\le i: a_j=1, \mathrm{NAK}_j \notin o_i} C_j \tag{8}$$

where

$$\begin{aligned} C_j &= P\{\mathrm{FTT}' > t_{\mathrm{DTS}} - t_j | \mathrm{RTT}' > t_{i+1} - t_j\} \\ &\quad + P\{\mathrm{FTT}' \le t_{\mathrm{DTS}} - t_j | \mathrm{RTT}' > t_{i+1} - t_j\}\epsilon_P \\ &\equiv P\{\mathrm{FTT}'' > t_{\mathrm{DTS}} - t_j | \mathrm{RTT}' > t_{i+1} - t_j\}. \end{aligned} \tag{9}$$

It is worth noting, at this point, that any of the expressions [such as (9)] of the form $P + (1-P)\epsilon$ can be re-expressed as either $1 - (1-P)(1-\epsilon)$ or $\epsilon + (1-\epsilon)P$. Hence, C_j can also be written

$$\begin{aligned} C_j = {} & \\ & \epsilon_P + (1-\epsilon_P)P\{\mathrm{FTT}' > t_{\mathrm{DTS}} - t_j | \mathrm{RTT}' > t_{i+1} - t_j\}. \end{aligned} \tag{10}$$

On the other hand, if $a_{i+1} = 1$, then the conditional expected error, given q_{i+1}, varies slightly depending on whether a systematic packet or a parity packet is transmitted at time t_{i+1}. If $o_i = \emptyset$, then a systematic packet is transmitted at time t_{i+1}, and in that case, the expected error is the expression in (8) times the probability that the last packet either is not received by the client application or else its payload is corrupted. That is

$$\epsilon_1(q_{i+1}) = \epsilon_0(q_{i+1})P\{\mathrm{FTT}'' > t_{\mathrm{DTS}} - t_{i+1}\}. \tag{11}$$

On the other hand, if $o_i \ne \emptyset$, then a parity packet is transmitted at time t_{i+1}, and in that case, the expected error is the expression in (8) times the probability that the parity packet either is not received by the client application or else is not successfully decoded. That is

$$\epsilon_1(q_{i+1}) = \epsilon_0(q_{i+1})P\{\mathrm{FTT}''^{(1)} > t_{\mathrm{DTS}} - t_{i+1}\}. \tag{12}$$

Clearly, (11) and (12) differ only by the use of either ϵ_P or $\epsilon_P^{(1)}$. We can now compute the expected error for the four policies $[a_i, a_{i+1}] = [0,0], [1,0], [0,1]$, and $[1,1]$. Putting together (5) and (8), the expected error for $a_{i+1} = 0$ can be computed as

$$\epsilon([a_i, 0]) = \sum_{q_{i+1}} P(q_{i+1}|q_i)\epsilon_0(q_{i+1}) \tag{13}$$

$$= \sum_{o_i} \prod_{j\le i: a_j=1, \mathrm{NAK}_j \notin o_i} A_j C_j \prod_{j\le i: a_j=1, \mathrm{NAK}_j \in o_i} B_j \tag{14}$$

$$= \prod_{j\le i: a_j=1} (A_j C_j + B_j). \tag{15}$$

In (13), the sum is over all 2^n nonfinal states q_{i+1}, and correspondingly, in (14), the sum is over all 2^n observations o_i not containing an ACK. In (15), the product is over all n previously transmitted packets. Equation (15) follows from (14), because, in general, $\prod_{j=1}^{n}(x_j + y_j)$ is the sum of the 2^n terms containing all possible products $x_1x_2\cdots x_n + y_1x_2\cdots x_n + x_1y_2\cdots x_n + y_1y_2\cdots x_n + \cdots + y_1y_2\cdots y_n$. The factors $(A_jC_j + B_j)$ can be simplified as follows. Putting together (6) and (10), we have

$$\begin{aligned} & A_jC_j \\ & = P\{\mathrm{RTT}' > t_{i+1} - t_j | \mathrm{RTT}' > t_i - t_j\}(\epsilon_P + (1-\epsilon_P) \\ & \quad \times P\{\mathrm{FTT}' > t_{\mathrm{DTS}} - t_j | \mathrm{RTT}' > t_{i+1} - t_j\}) \qquad (16) \\ & = P\{\mathrm{RTT}' > t_{i+1} - t_j | \mathrm{RTT}' > t_i - t_j\}\epsilon_P \\ & \quad + (1-\epsilon_P)P\{\mathrm{FTT}' > t_{\mathrm{DTS}} - t_j | \mathrm{RTT}' > t_i - t_j\}. \qquad (17) \end{aligned}$$

Equation (17) follows from (16) because $\{\mathrm{FTT}' > t_{\mathrm{DTS}} - t_j\}$ implies $\{\mathrm{RTT}' > t_{i+1} - t_j\}$, which, in turn, implies $\{\mathrm{RTT}' > t_i - t_j\}$. Hence

$$\begin{aligned} & P\{\mathrm{RTT}' > t_{i+1} - t_j | \mathrm{RTT}' > t_i - t_j\} \\ & \quad \times P\{\mathrm{FTT}' > t_{\mathrm{DTS}} - t_j | \mathrm{RTT}' > t_{i+1} - t_j\} \\ & \quad = P\{\mathrm{RTT}' > t_{i+1} - t_j | \mathrm{RTT}' > t_i - t_j\} \\ & \quad \quad \times P\{\mathrm{FTT}' > t_{\mathrm{DTS}} - t_j | \mathrm{RTT}' > t_{i+1} - t_j, \\ & \quad \quad \quad \mathrm{RTT}' > t_i - t_j\} \\ & \quad = P\{\mathrm{FTT}' > t_{\mathrm{DTS}} - t_j, \\ & \quad \quad \mathrm{RTT}' > t_{i+1} - t_j | \mathrm{RTT}' > t_i - t_j\} \\ & \quad = P\{\mathrm{FTT}' > t_{\mathrm{DTS}} - t_j | \mathrm{RTT}' > t_i - t_j\}. \end{aligned}$$

Now, putting together (17) and (7), we have

$$\begin{aligned} &A_jC_j + B_j \\ &\quad = P\{\mathrm{RTT}' > t_{i+1} - t_j | \mathrm{RTT}' > t_i - t_j\}\epsilon_P \\ &\qquad + (1-\epsilon_P)P\{\mathrm{FTT}' > t_{\mathrm{DTS}} - t_j | \mathrm{RTT}' > t_i - t_j\} \\ &\qquad + P\{\mathrm{RTT}' \le t_{i+1} - t_j | \mathrm{RTT}' > t_i - t_j\}\epsilon_P \\ &\quad = \epsilon_P + (1-\epsilon_P)P\{\mathrm{FTT}' > t_{\mathrm{DTS}} - t_j | \mathrm{RTT}' > t_i - t_j\} \\ &\quad = P\{\mathrm{FTT}'' > t_{\mathrm{DTS}} - t_j | \mathrm{RTT}' > t_i - t_j\}. \end{aligned} \tag{18}$$

In summary, combining (18) and (15), we have

$$\epsilon([a_i, 0]) = \prod_{j \le i: a_j = 1} P\{\mathrm{FTT}'' > t_{\mathrm{DTS}} - t_j | \mathrm{RTT}' > t_i - t_j\}.$$

The expected error for $a_{i+1} = 1$ can also be computed, by building on the above results with some care. When $a_{i+1} = 1$, according to (11) and (12), the one term in (14) corresponding to $o_i = \emptyset$ is multiplied by $P\{\mathrm{FTT}'' > t_{\mathrm{DTS}} - t_{i+1}\}(\dagger)$, while the $2^n - 1$ remaining terms are multiplied by $P\{\mathrm{FTT}''^{(1)} > t_{\mathrm{DTS}} - t_{i+1}\}(\ddagger)$. Hence, to get $\epsilon([a_i, 1])$, we multiply all 2^n terms in $\epsilon([a_i, 0])$ by $(\ddagger)$ and then adjust by adding and subtracting the first term (corresponding to $o_i = \emptyset$) multiplied by $(\dagger)$ and $(\ddagger)$, respectively. That is

$$\begin{aligned} &\epsilon([a_i, 1]) \\ &\quad = \epsilon([a_i, 0])P\{\mathrm{FTT}''^{(1)} > t_{\mathrm{DTS}} - t_{i+1}\} \\ &\qquad + \left(\prod_{j \le i: a_j = 1} A_jC_j\right) P\{\mathrm{FTT}'' > t_{\mathrm{DTS}} - t_{i+1}\} \\ &\qquad - \left(\prod_{j \le i: a_j = 1} A_jC_j\right) P\left\{\mathrm{FTT}''^{(1)} > t_{\mathrm{DTS}} - t_{i+1}\right\} \\ &\quad = \epsilon([a_i, 0])P\{\mathrm{FTT}''^{(1)} > t_{\mathrm{DTS}} - t_{i+1}\} \\ &\qquad + \left(\prod_{j \le i: a_j = 1} A_jC_j\right) P\{\mathrm{FTT}' \le t_{\mathrm{DTS}} - t_{i+1}\} \\ &\qquad \times \left(\epsilon_P - \frac{\epsilon_P^{(1)}}{\epsilon_P}\right). \end{aligned}$$

The expected costs of the four policies are far simpler to describe. Using (4), we can compute $\rho([a_i, a_{i+1}])$ as

$$\rho([a_i, a_{i+1}]) = a_i + a_{i+1}(1 - P_{\mathrm{ACK}}).$$

It can be shown that these expressions for the expected error and cost reduce to those in the Appendix, section A.

C. Previous Transmissions With NAKS

In this section, we derive expressions for the expected error and cost in the case where the history includes previous transmissions and at least one NAK. As explained earlier, these expressions are, in general, analytically intractable. Using the "two-delta" model introduced in Section V, they simplify to something tractable, as we shall see. It should be noted that in a "two-delta" model, the forward, backward, and round-trip times are essentially deterministic, except in the event of loss.

Let $N_s \ge 1$ be the number of systematic packets transmitted (at least one of which is NAK'd) and let $N_p \ge 0$ be the number of parity packets subsequently transmitted, prior to the current transmission opportunity t_i. Furthermore, let N_{old} be the number of "old" parity packets transmitted prior to $t_i - \tau_R$, and let $N_{\mathrm{new}} = N_p - N_{\mathrm{old}}$ be the number of "new" parity packets transmitted at or after $t_i - \tau_R$. For the "old" parity packets, by time t_i, the server receives all the acknowledgment it is ever going to get. For the "new" parity packets, by time t_i, the server can receive no acknowledgment. However, by time t_{i+1}, for the new parity packets (including, if any, the parity packet transmitted at time t_i), the server receives all the acknowledgment it is ever going to get. Here, we are assuming that $t_{i+1} \ge t_i + \tau_R$, since we are only interested in evaluating those policies $[a_i, a_{i+1}]$ where there is a chance that the packet transmitted at time t_i can be acknowledged by time t_{i+1}.

Consider the last packet for which the server receives a NAK by time t_i. This packet could be a systematic packet, if there are no NAK'd parity packets, or it could be an "old" parity packet. Let t_{j_0} be the transmission time of this last NAK'd packet. Now partition the "old" parity packets into three groups. Let N_0 be the number of "old, NAK'd" parity packets for which the server receives NAKs. Let N_1 be the number of "old, unacknowledged, pre-t_{j_0}" parity packets for which the server receives no acknowledgment, but are transmitted before t_{j_0}. Finally, let N_2 be the number of "old, unacknowledged, post-t_{j_0}" parity packets for which the server receives no acknowledgment, but are transmitted after t_{j_0}.

Of the N_1 "old, unacknowledged, pre-t_{j_0}" packets, let $n_1 \in \{0, 1, \ldots, N_1\}$ be the number that eventually arrives at the client application. Next, of the N_2 "old, unacknowledged, post-t_{j_0}" packets, let $n_2 \in \{0, 1, \ldots, N_2\}$ be the number that eventually arrives at the client application. It can be seen that these numbers are probabilistically independent, and have probability mass functions (pmfs)

$$p(n_j) = \binom{N_j}{n_j}\left(1 - \frac{\epsilon'_F}{\epsilon'_R}\right)^{n_j}\left(\frac{\epsilon'_F}{\epsilon'_R}\right)^{N_j - n_j}, \quad \text{for } j = 1, 2.$$

This is because

$$\frac{\epsilon'_F}{\epsilon'_R} = \frac{P\{\mathrm{FTT}' > \tau_F\}}{P\{\mathrm{RTT}' > \tau_R\}} = P\{\mathrm{FTT}' > \tau_F | \mathrm{RTT}' > \tau_R\}$$

is the probability that a packet does not eventually arrive at the client application, given that the server receives no acknowledgment for it.

Similarly, of the $N_{\mathrm{new}} + a_i + a_{i+1}$ "new" parity packets, including, if any, the parity packets transmitted at t_i and t_{i+1}, let $n_3 \in \{0, 1, \ldots, N_{\mathrm{new}} + a_i + a_{i+1}\}$ be the number that arrives at the client by time t_{DTS}. It can be seen that n_3 is independent of n_1 and n_2, and that the pmf of n_3 is

$$\begin{aligned} p_{a_i + a_{i+1}}(n_3) = &\binom{N_{\mathrm{new}} + a_i + a_{i+1}}{n_3} \\ &\times (1 - \epsilon'_F)^{n_3}(\epsilon'_F)^{N_{\mathrm{new}} + a_i + a_{i+1} - n_3}. \end{aligned}$$

Here, we are assuming that $t_{\mathrm{DTS}} \ge t_{i+1} + \tau_F$, since we are only interested in evaluating those policies $[a_i, a_{i+1}]$ where there is a

$$p(k|n_1,n_2,n_3)=\begin{cases}1-\epsilon_P^{(N_0+n_1+n_2+1)}/\epsilon_P^{(N_0+n_1)}, & \text{if } k=0\\ \epsilon_P^{(N_0+n_1+n_2+k)}/\epsilon_P^{(N_0+n_1)}-\epsilon_P^{(N_0+n_1+n_2+k+1)}/\epsilon_P^{(N_0+n_1)}, & \text{if } 1\le k<n_3\\ \epsilon_P^{(N_0+n_1+n_2+n_3)}/\epsilon_P^{(N_0+n_1)}, & \text{if } k=n_3.\end{cases}$$

chance that the packet transmitted at time t_{i+1} can arrive at the client by time $t_{\rm DTS}$.

We know that successful decoding does not occur for any of the N_0+n_1 "pre-t_{j_0}" parity packets that arrive at the client application, because the server receives a NAK for the last of these. However, we do not know *a priori* whether successful decoding occurs for any of the remaining n_2+n_3 "post-t_{j_0}" parity packets that arrive at the client application. The probability that none of the latter n_2+n_3 "post-t_{j_0}" packets are successfully decoded, given that none of the former N_0+n_1 "pre-t_{j_0}" packets are successfully decoded, is $\epsilon_P^{(N_0+n_1+n_2+n_3)}/\epsilon_P^{(N_0+n_1)}$. Thus, the probability that none of the parity packets that eventually arrive at the client application (however many there are) can be successfully decoded is

$$\sum_{n_1=0}^{N_1}p(n_1)\sum_{n_2=0}^{N_2}p(n_2)\sum_{n_3=0}^{N_{\rm new}+a_i+a_{i+1}}p_{a_i+a_{i+1}}(n_3)\times\frac{\epsilon_P^{(N_0+n_1+n_2+n_3)}}{\epsilon_P^{(N_0+n_1)}}.$$

Since this event is independent of the received systematic packets, the probability of error can be expressed

$$\epsilon([a_i,a_{i+1}])=P_{\rm sp}\sum_{n_1=0}^{N_1}p(n_1)\sum_{n_2=0}^{N_2}p(n_2)\times\sum_{n_3=0}^{N_{\rm new}+a_i+a_{i+1}}p_{a_i+a_{i+1}}(n_3)\frac{\epsilon_P^{(N_0+n_1+n_2+n_3)}}{\epsilon_P^{(N_0+n_1)}}$$

where $P_{\rm sp}=(P\{{\rm FTT}''>\tau_F|{\rm RTT}'>\tau_R\})^{N_{s1}}$ is the probability that none of the post-t_{j_0} systematic packets arrives at the client with an uncorrupted payload. These are systematic packets transmitted after t_{j_0} and there are (if any) N_{s1} of them. Due to the two-delta channel model, $P_{\rm sp}$ attains this form, and can be calculated as

$$P_{\rm sp}=\left(\epsilon_P+(1-\epsilon_P)\frac{\epsilon_F'}{\epsilon_R'}\right)^{N_{s1}}.$$

Finally, consider the expected cost. As in the previous subsection, the expected cost can be expressed

$$\rho([a_i,a_{i+1}])=a_i+a_{i+1}(1-P_{\rm ACK})$$

where $P_{\rm ACK}$ (which depends on a_i) is the probability that an acknowledgment will be received by time t_{i+1}. Therefore, it suffices to derive an expression for $P_{\rm ACK}$.

As above, of the N_1 "old, unacknowledged, pre-t_{j_0}" parity packets, let n_1 be the number that arrives at the client application; of the N_2 "old, unacknowledged, post-t_{j_0}" parity packets, let n_2 be the number that arrives at the client application; and of the $N_{\rm new}+a_i$ "new" parity packets, including, if any, the parity packet transmitted at time t_i, let n_3 be the number that arrive at the client application. Of the latter, let $k\in\{0,1,\ldots,n_3\}$ be the number that are not successfully decoded. That is, assume that the first k "new" packets arriving at the client application are not successfully decoded, while the remaining n_3-k are successfully decoded. By definition, $\epsilon_P^{(l)}$ is the probability that it takes more than l parity packets for successful decoding. That is, $\epsilon_P^{(0)}=\epsilon_P$ is the probability that a systematic packet cannot be successfully decoded (i.e., its payload is corrupted), $\epsilon_P^{(1)}$ is the probability that a systematic packet and one parity packet cannot be successfully decoded, $\epsilon_P^{(2)}$ is the probability that a systematic packet and two parity packets cannot be successfully decoded, and so forth. Hence, the probability that k is at least one, given that the first N_0+n_1 parity packets are not successfully decoded, is $\epsilon_P^{(N_0+n_1+n_2+1)}/\epsilon_P^{(N_0+n_1)}$. Likewise, the probability that k is at least two, given that the first N_0+n_1 parity packets are not successfully decoded, is $\epsilon_P^{(N_0+n_1+n_2+2)}/\epsilon_P^{(N_0+n_1)}$, and so on. Finally, the probability that k is n_3 (i.e., that none of the parity packets is successfully decoded), given that the first N_0+n_1 parity packets are not successfully decoded, is $\epsilon_P^{(N_0+n_1+n_2+n_3)}/\epsilon_P^{(N_0+n_1)}$. Hence, the pmf for k is shown in the equation at the top of the page. The client sends an ACK for each of the n_3-k successfully decoded packets, but the ACK never reaches the server with probability ϵ_B'. Hence, the probability that at least one ACK is received is

$$p({\rm ACK}|n_3,k)=1-(\epsilon_B')^{n_3-k}$$

and thus $P_{\rm ACK}$ can be expressed as

$$P_{\rm ACK}=\sum_{n_1=0}^{N_1}p(n_1)\sum_{n_2=0}^{N_2}p(n_2)\sum_{n_3=0}^{N_{\rm new}+a_i}p_{a_i}(n_3)\times\sum_{k=0}^{n_3}p(k|n_1,n_2,n_3)P({\rm ACK}|n_3,k).$$

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for the constructive comments that helped to greatly improve the quality of the manuscript.

REFERENCES

[1] Third Generation Partnership Project [Online]. Available: http://www.3gpp.org
[2] Third Generation Partnership Project 2 [Online]. Available: http://www.3gpp2.org
[3] Third Generation Partnership Project General Packet Radio Service (GPRS) and EDGE General Packet Radio Service (EGPRS) [Online]. Available: http://www.3gpp.org

[4] A. Furuskar, S. Mazur, F. Muller, and H. Olofsson, "EDGE: Enhanced data rates for GSM and TDMA/136 evolution," *IEEE Pers. Commun. Mag.*, vol. 6, pp. 55–56, June 1999.
[5] S. Blake, D. Black, M. Carlson, E. Davies, and Z. Wang, "An architecture for differentiated services,", IETF, Tech. Rep. RFC-2475, Dec. 1998.
[6] A. Sehgal and P. A. Chou, "Cost-distortion optimized streaming media over DiffServ networks," in *Proc. IEEE Int. Conf. Multimedia and Exhibition*, vol. 1, Lausanne, Switzerland, Aug. 2002, pp. 857–860.
[7] J. C. de Martin, "Source-driven packet marking for speech transmission over differentiated-services networks," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, vol. 2, Salt Lake City, UT, May 2001, pp. 753–756.
[8] E. Masala, D. Quaglia, and J. de Martin, "Adaptive picture slicing for distortion-based classification of video packets," in *Proc. IEEE Workshop Multimedia Signal Processing*, Cannes, France, Oct. 2001, pp. 111–116.
[9] J. Shin, J. Kim, and C.-C. J. Kuo, "Relative priority based QoS interaction between video applications and differentiated service networks," in *Proc. IEEE Int. Conf. Image Processing*, vol. 3, Vancouver, BC, Canada, Sept. 2000, pp. 536–539.
[10] ——, "Quality-of-service mapping mechanism for packet video in differentiated services network," *IEEE Trans. Multimedia*, vol. 3, pp. 219–231, June 2001.
[11] D. Quaglia and J. de Martin, "Delivery of MPEG video streams with constant perceptual quality of service," in *Proc. IEEE Int. Conf. Multimedia and Exhibition*, vol. 2, Lausanne, Switzerland, Aug. 2002, pp. 85–88.
[12] L. Larzon, M. Degermark, and S. Pink, "UDP Lite for real time multimedia applications," HP Labs, Bristol, MA, Tech. Rep. HPL-IRI-1999-001, Apr. 1999.
[13] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
[14] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Commun.*, vol. 36, pp. 389–400, Apr. 1988.
[15] D. J. Costello, Jr., J. Hagenauer, H. Imai, and S. Wicker, "Applications of error-control coding," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2531–2560, Oct. 1998.
[16] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[17] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. IEEE Int. Conf. Communications*, vol. 2, Geneva, Switzerland, May 1993, pp. 1064–1070.
[18] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429–445, Mar. 1996.
[19] C. Heegard and S. Wicker, *Turbo Coding*. Norwell, MA: Kluwer, 1999.
[20] K. Narayanan and G. Stuber, "A novel ARQ technique using the turbo coding principle," *IEEE Commun. Lett.*, vol. 1, pp. 49–51, Mar. 1997.
[21] D. Rowitch and L. Milstein, "On the performance of hybrid FEC/ARQ systems using rate compatible punctured turbo (RCPT) codes," *IEEE Trans. Commun.*, vol. 48, pp. 948–959, June 2000.
[22] D. Chase, "Code combining—A maximum likelihood decoding approach for combining an arbitrary number of noisy packets," *IEEE Trans. Commun.*, vol. COM-33, pp. 385–393, May 1985.
[23] B. A. Harvey and S. B. Wicker, "Packet combining systems based on the Viterbi decoder," *IEEE Trans. Commun.*, vol. 42, pp. 1544–1557, Feb.-Apr. 1994.
[24] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Trans. Multimedia*, submitted for publication.
[25] "Rate-distortion optimized sender-driven streaming over best-effort networks," in *Proc. IEEE Workshop Multimedia Signal Processing*, Cannes, France, Oct. 2001, pp. 587–592.
[26] J. Chakareski, P. Chou, and B. Aazhang, "Computing rate-distortion optimized policies for streaming media to wireless clients," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, Apr. 2002, pp. 53–62.
[27] J. Chakareski and P. Chou, "Application layer error correction coding for rate-distortion optimized streaming to wireless clients," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, vol. 3, Orlando, FL, May 2002, pp. 2513–2516.
[28] J.-C. Bolot, "End-to-end packet delay and loss behavior in the Internet," in *Proc. ACM Data Communication Conf.*, Ithaca, NY, Sept. 1993, pp. 289–298.
[29] N. Seshadri and C. Sundberg, "List Viterbi decoding algorithms with applications," *IEEE Trans. Commun.*, vol. 42, pp. 313–323, Feb.-Apr. 1994.
[30] T. Ramabadran and S. Gaitonde, "A tutorial on CRC computations," *IEEE Micro*, vol. 8, pp. 62–75, Aug. 1988.
[31] L. C. Lee, *Convolutional Coding: Fundamentals and Applications*. Norwood, MA: Artech House, 1997.
[32] J. Chakareski and P. Chou. (2002, Aug.) Application layer error correction coding for rate-distortion optimized streaming to wireless clients. Microsoft Res., Redmond, WA. [Online]. Available: ftp://ftp.research.microsoft.com/pub/tr/TR-2002-81.ps
[33] Robust Header Compression (ROHC) (IETF proposed standard), C. Bormann *et al.*. (2001, July). [Online]. Available: http://www.ietf.org/rfc/rfc3095.txt

Jacob Chakareski received the B.S. degree from Saints Cyril and Methodius University, Skopje, Macedonia, in 1996 and the M.S. degree from Worcester Polytechnic Institute, Worcester, MA, in 1999, both in electrical engineering. He is currently working toward the Ph.D. degree in the Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA, after completing three years of Ph.D. degree study at Rice University, Houston, TX.

His research interests include multimedia streaming and networking, statistical signal processing, communication theory, and Macedonian history.

Philip A. Chou (S'82–M'87–SM'00–F'03) received the B.S.E. degree from Princeton University, Princeton, NJ, in 1980, and the M.S. degree from the University of California, Berkeley, in 1983, both in electrical engineering and computer science, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1988.

From 1988 to 1990, he was a Member of Technical Staff at AT&T Bell Laboratories, Murray Hill, NJ. From 1990 to 1996, he was a Member of Research Staff at the Xerox Palo Alto Research Center, Palo Alto, CA. In 1997, he was Manager of the Compression Group at VXtreme, Mountain View, CA, before it was acquired by Microsoft in 1997. From 1998 to the present, he has been a Senior Researcher with Microsoft Research in Redmond, Washington, where he currently manages the Communication and Collaboration Systems research group. He also served as a consulting Associate Professor at Stanford University in 1994-95, and has been an affiliate professor at the University of Washington, Seattle, since 1998. His research interests are data compression, information theory, communications, and pattern recognition, with applications to video, images, audio, speech, and documents.

Dr. Chou served as an Associate Editor in Source Coding for the IEEE TRANSACTIONS ON INFORMATION THEORY from 1998 to 2001, and served as a Guest Associate Editor for special issues in the IEEE TRANSACTIONS ON IMAGE PROCESSING and the IEEE TRANSACTIONS ON MULTIMEDIA in 1996 and 2004, respectively. From 1998 to 2004, he was a member of the IEEE Signal Processing Society's Image and Multidimensional Signal Processing technical committee (IMDSP TC). He is a member of Phi Beta Kappa, Tau Beta Pi, Sigma Xi, and the IEEE Computer, Information Theory, Signal Processing, and Communications societies, and was an active member of the MPEG committee. He is the recipient, with Anshul Seghal, of the 2002 ICME Best Paper award, and is the recipient, with Tom Lookabaugh, of the 1993 Signal Processing Society Paper award.