# Error Control for Receiver-Driven Layered Multicast of Audio and Video

Philip A. Chou, *Senior Member, IEEE*, Alexander E. Mohr, Albert Wang, and Sanjeev Mehrotra, *Member, IEEE*

***Abstract*—We consider the problem of error control for receiver-driven layered multicast of audio and video over the Internet. The sender injects into the network multiple source layers and multiple channel coding (parity) layers, some of which are delayed relative to the source. Each receiver subscribes to the number of source layers and the number of parity layers that optimizes the receiver's quality for its available bandwidth and packet loss probability. We augment this layered FEC system with layered pseudo-ARQ. Although feedback is normally problematic in broadcast situations, ARQ can be simulated by having the receivers subscribe and unsubscribe to the delayed parity layers to receive missing information. This pseudo-ARQ scheme avoids an implosion of repeat requests at the sender and is scalable to an unlimited number of receivers. We show gains of 4–18 dB on channels with 20% loss over systems without error control and additional gains of 1–13 dB when FEC is augmented by pseudo-ARQ in a hybrid system. Optimal error control in the hybrid system is achieved by an optimal policy for a Markov decision process.**

***Index Terms*—ARQ, error correction, FEC, Markov decision process, multicast, multimedia communication, packet audio, packet video, rate-distortion optimization.**

## I. Introduction

THIS paper addresses the problem of error control for receiver-driven layered multicast. The application scenario is that of broadcasting live or prestored audio and video over the Internet to potentially millions of simultaneous receivers. This problem is important to address as the Internet moves to augment or supplant existing radio and television distribution systems and as it begins to provide a broadcast infrastructure for new smaller markets, such as the distance learning market.

Error control for the broadcast application is an easier problem than for other communications applications in one important way: delay. Broadcasting has rather loose delay requirements, possibly in the tens of seconds, compared to telephony and conferencing, which have delay requirements on the order of tenths of a second, and even compared to media-on-demand, which may have delay requirements on the order of seconds because of interactivity features such as fast forwarding, rewinding, and restarting.

However, error control for the broadcast application is more difficult than for other communications applications in another important way: heterogeneity. In a heterogeneous environment such as the Internet, the one-to-many nature of broadcast guarantees that the communication channels between the sender and its receivers are extremely diverse. This means that communication with each receiver must proceed over an essentially unknown channel. In the Internet, the heterogeneity is so large that error control targeted to the worst case channel (as is done in the case of terrestrial radio broadcast) is not feasible.

There are many sources of heterogeneity in the Internet. Bandwidth can vary by three or more orders of magnitude, from kilobits to tens or hundreds of megabits per second, across different links. Packet loss probabilities can also vary by many orders of magnitude, from near zero to tens of percents, across different routers. Likewise, both delay and jitter can vary by several orders of magnitude across different network paths. Moreover, all of these quantities can vary over time, as competing communication processes begin and end. In the future, heterogeneity will only increase, as both variable-bandwidth mobile units and ultra high speed links are added to the network.

Existing systems for multimedia broadcast deal with bandwidth heterogeneity in the Internet in one of two ways: either by simulcasting the content at a variety of bandwidths (e.g., at 28 Kbps and 56 Kbps) or by layered coding. Layered coding, also known as scalable, embedded, or progressive coding, is the basis of the Receiver-driven Layered Multicast (RLM) scheme, pioneered by McCanne [1] and others. McCanne's multicast backbone (MBONE) tool `vic` implements RLM by coding video in multiple layers and broadcasting (actually, multicasting) each layer to a different multicast group. Each receiver estimates its available bandwidth and joins a number of these multicast groups to fill that available bandwidth. The available bandwidth is estimated by measuring packet losses. Essentially, if the packet losses are frequent, then the transmission rate is deemed too high for the available bandwidth and the receiver leaves some of the multicast groups. If the packet losses are sufficiently rare, then the receiver joins some of the multicast groups. In this way the receiver can track the available bandwidth even if it is varying. Because each receiver determines its own transmission rate, the scheme is said to be *receiver-driven*. Commercial systems do not use RLM, primarily because efficient layered video coding is

Manuscript received May 31, 2000; revised December 1, 2000. A. E. Mohr was supported in part by the National Science Foundation Grant EIA-9973531. This work was presented in part at the IEEE Communication Theory Workshop, Aptos, CA, May 1999, and at the IEEE Data Compression Conference, Snowbird, UT, March 2000. The associate editor coordinating the review of this paper and approving it for publication was Dr. Sadaoki Furui.

P. A. Chou and S. Mehrotra are with the Microsoft Corporation, Redmond, WA 98052-6399 USA (e-mail: pachou@microsoft.com; sanjeevm@microsoft.com).

A. E. Mohr was with the Microsoft Corporation, Redmond, WA 98052-6399 USA. He is now with the Department of Computer Science, University of Washington, Seattle, WA 98195-2350 USA (e-mail: amohr@cs.washington.edu).

A. Wang was with Microsoft Corporation, Redmond, WA 98052-6399 USA. He is now with Rivio, Inc., Santa Clara, CA 95054 USA (e-mail: alwang@rivio.com).

Publisher Item Identifier S 1520-9210(01)01866-1.

difficult to achieve. (However, there are a number of recent papers suggesting that the performance gap between layered and monolithic coders is closing [2]–[5].) Commercial systems typically simulcast their content at a variety of rates. Because each receiver determines which multicast group it will join, this is also a receiver-driven scheme. However, it does not achieve the network bandwidth efficiencies of RLM, nor is it as bandwidth adaptive as RLM.

Existing research and commercial systems for multimedia broadcast also deal with packet loss heterogeneity in the Internet in a variety of ways. These will be discussed in the next section.

In our work, we simultaneously address both packet loss heterogeneity and bandwidth heterogeneity in a unified receiver-driven framework, using layered channel coding as well as layered source coding. Layered source coding is accomplished with any of the usual techniques, while layered channel coding is accomplished using a systematic rate-compatible Reed–Solomon style erasure code. The sender multicasts all of the source coding layers and all of the channel coding (i.e., parity) layers to separate multicast groups. Each receiver subscribes and unsubscribes to the multicast groups to adapt to changing bandwidth and packet loss conditions. After estimating the bandwidth and packet loss probability of its channel, each receiver computes the optimal allocation of the available transmission rate between the source and channel codes (which generally results in unequal error protection for the different source layers) and joins the multicast groups for the optimal collection of source and channel layers. This FEC system can be hybridized with a pseudo-ARQ system, in which ARQ is simulated by the sender continuously transmitting delayed parity packets to additional multicast groups. The receivers can join and leave these multicast groups to retrieve information lost in previous transmissions, up to a given delay bound. The optimal algorithm for a receiver joining and leaving multicast groups is equivalent to the optimal policy in a finite horizon Markov decision process, which contains the optimal allocation for pure FEC as a special case. For channels with 20% ambient packet loss, compared to standard RLM (without error control), receiver-driven layered FEC gains 4–18 dB and receiver-driven layered hybrid FEC/pseudo-ARQ gains an additional 1–13 dB, for a total of 5–31 dB.

In contrast to error control, congestion control is outside the scope of this paper. The purpose of congestion control at a receiver is determining the transmission rate at the receiver so that the receiver's traffic does not cause congestion, and subsequent packet loss and delay, in the network. Examples of receiver-driven congestion control mechanisms can be found in the original RLM work [1] and in more recent work on TCP-friendly audio and video transmission [6]–[10]. However, because congestion control at a receiver cannot reduce ambient packet loss caused by other applications' traffic, error control is needed in addition to congestion control.

Thus error control is the focus of this paper. The purpose of error control is to use the available transmission rate, as determined by the congestion control mechanism, to mitigate the effects of packet loss. Error control is generally accomplished by transmitting some amount of redundant information to compensate for the loss of potentially important packets. It is critical to note that such redundancy is sent *in lieu of* packets containing less important source information and hence the use of error control does not increase congestion. Rather, when congestion control mechanisms are in place, error control simply changes the source/redundancy mix, so that the information that is transmitted is more robust to ambient packet loss. In this work, we assume the existence of a congestion control mechanism and solve the problem of optimal error control given a transmission rate and packet loss probability.

The remainder of this paper proceeds as follows. Section II describes related work. Section III describes our source coding and packetization model. Section IV describes our channel coding (FEC) model, while Section V describes our algorithm for locally optimal selection of source and parity information in this model. Section VI describes our pseudo-ARQ and hybrid FEC/ARQ models, while Section VII describes our algorithm for optimal error control in these models. Section VIII describes how overall rate control could be performed in a receiver, Section IX presents analytical and simulation results, and Section X summarizes and concludes.

## II. Relation to Previous Work

Our multicast system can be regarded in various ways. Those familiar with RLM can regard the system as an error control extension to RLM. Those familiar with joint source-channel coding (JSCC)—in particular layered source coding with unequal error protection—can regard the system as adaptive JSCC with receiver feedback. Those familiar with hybrid FEC/ARQ can regard the system as an extension of hybrid FEC/ARQ to layered coding. Those familiar with reliable multicast techniques can regard the system as an adaptation of some of these techniques to real-time multicast.

Given the many possible interpretations of our work, the amount of related previous work is far too great to cover here. We cite here only a personally motivating subset. Unequal error protection (UEP) using rate-compatible codes is an old idea, going back at least as far as the 1970s. It was popularized by Hagenauer [11] and subsequently was used extensively in coding speech for wireless communication. It was rediscovered and adapted to packet networks by Albanese *et al.* [12] in their priority encoding transmission (PET) scheme. PET achieves UEP by varying the source blocklength $K$ across the different source layers while keeping the channel blocklength $N$ fixed. This permits PET to have a packetization scheme that ensures that the source layers in a channel block can only be lost in order. Optimization of the rate allocation in such a scheme was addressed by [13]–[17]. UEP can also be achieved by fixing the source blocklength $K$ and varying the channel blocklength $N$ across the different source layers. Optimization of the rate allocation in this second scheme, in which the source layers can be independently lost, was solved approximately by [18], [19] and exactly using dynamic programming by [20]. We solve the problem without approximation, using an iterative descent algorithm. An asymptotic analysis of the optimal source-channel rate allocation was performed in [21].

Automatic repeat request schemes have a long history in data communication. However, joint source-channel coding with feedback has not been examined until recently. In [22], the receiver can indicate to the sender whether or not to send additional packets of parity information generated by a rate-compatible convolutional code. In that work, Chande *et al.* identify the optimal receiver strategy as the optimal policy in a finite horizon Markov decision process. We generalize this to the case where the receiver can ask for a variable number of parity packets, hence we owe a particular debt to their work. The problem of joint source-channel coding with feedback is also addressed in [23]–[27].

The multicast community has also done a fair amount of work on error control mechanisms for both reliable and real-time multicast. For reliable multicast, which is used for broadcast of bulk data (e.g., distributed database updates, software updates, and the like) ARQ-based error control mechanisms are usually proposed. To avoid the implosion of repeat requests upon the sender, most of these schemes (SRM, RMTP, RAMP, RMP, …) use explicit hierarchical re-transmitters scattered throughout the network or local recovery techniques [28]. Using hybrid FEC/ARQ, Nonenmacher *et al.* [29] show that hybrid FEC/ARQ can dramatically reduce the network traffic for retransmissions by answering multiple NAKs from different receivers with a single parity packet. Byers *et al.* [30] take this idea to an extreme in an ingenious scheme that uses pure FEC for reliable multicast with no retransmissions, by continuously multicasting a revolving stream of FEC packets. Interested receivers join the multicast group to "drink" from this "digital fountain" just long enough to receive enough FEC packets to recover the original data. We borrow from these ideas for real-time multicast by multicasting FEC packets continuously from the server. This technique simultaneously satisfies a large range of retransmission requests and obviates the need for explicit retransmissions.

For real-time multicast, simple error resilience is used in the earliest MBONE tools (e.g., the network video tool `nv`), while a form of FEC is used in later MBONE tools (e.g., the robust audio tool `rat`). A more recent proposal, layered video multicast with retransmission (LVMR) [28], borrows its hierarchical retransmission strategy from various reliable multicast schemes. Commercial systems can be found in all three categories: Real Networks uses ARQ to the server for small multicast applications and uses error resilience for scalable (large) multicast applications. Microsoft Windows Media uses FEC, whose Reed–Solomon parameters are fixed for all receivers in a session, but are adjustable by the content creator. We think that our work goes significantly beyond the state of the art in error control for real-time multicast.

In contemporaneous independent work, Tan and Zakhor developed a system very similar to ours. They first presented their system in [31], [9], [32]; we first presented our system in [33]–[35]. Tan and Zakhor do not explicitly propose pseudo-ARQ. However, in a clever trick, they delay some parity information to allow receivers to trade delay for redundancy. Most significantly, they also implement TCP-friendly congestion control and deploy it in a real system (though in a limited way) over the MBONE. In contrast, we focus on the receivers' optimization algorithms, on the extension of the FEC system to hybrid FEC/ARQ, and on distortion-rate performance analysis. We limit our experimental results to analysis and simulation. We believe that our work and the work of Tan and Zakhor are mutually supportive.

## III. Preliminaries

In this section we outline our background assumptions related to source coding and packetization. Readers requiring background on multicast are referred to [28], [36].

We assume that the source is encoded by a layered source coder, such as three-dimensional (3-D) SPIHT [2] for video or one–dimensional (1-D) SPIHT [37] for audio, which produces a set of elementary bit streams, or source layers, with unbounded duration, bounded bit rate, and a common, finite decoding delay, such that nested subsets of these source layers are decodable to a level of quality commensurate with the total bit rate of the subset. Such a set of source layers can be produced by blocking the audio or video sequence into groups of frames (GOFs), independently encoding and packetizing each GOF, and assigning one packet from each GOF per source layer. One way to do this is to encode each GOF as an embedded bit string, partition each embedded bit string into a sequence of packets, and assign the $l$th packet to the $l$th source layer, $l = 1, \cdots, L$, as shown in Fig. 1. Typical parameters for such a construction might be the following, for a 16 fps QCIF $(144 \times 176)$ video sequence: 16 frames per group of frames, 50 packets per GOF, and 1000 bytes per packet payload. This implies that a GOF has a duration of 1 s; that there are 50 source layers; that each source layer has a bit rate of 8000 bps (not including packet header information); and that a receiver that subscribes to all 50 source layers can decode the video to about 1 bit per pixel (400 Kbps). With this choice of parameters, the packet sizes are big enough that the packet header information is a small fraction of the total data rate (about 5%). In addition, each GOF has a long enough duration so that the bit rate of each source layer is sufficiently small for relatively fine-grained rate adjustments (8 to 400 Kbps in steps of 8 Kbps) and sufficiently long so that independent coding of each GOF is not unduly inefficient. Finally, each GOF has a short enough duration so that delays equal to small multiples of this duration, required for error control methods, are not objectionable to the end user. The optimal choice of such parameters is complicated; we do not address it here.

There may also be other methods of producing such a set of source layers. For example, after blocking a video sequence into groups of frames, one could independently encode each GOF into I, P, and B frames using a standard video coder, packetizing I frames in a set of base layers, P frames in a subsequent set of enhancement layers, and B frames in a further set of enhancement layers.

Regardless of how the source layers are produced, we assume that they have a dependency relationship that can be characterized by a directed acyclic graph. More specifically, we assume that there exists a single directed acyclic graph that for every GOF characterizes the dependencies between the packets of the GOF. For example, if the frames of the GOF are encoded by a standard video coder as IBBPBBPBBP frames and packetized with one packet per encoded frame, then the dependencies between packets can be characterized by the graph illustrated in

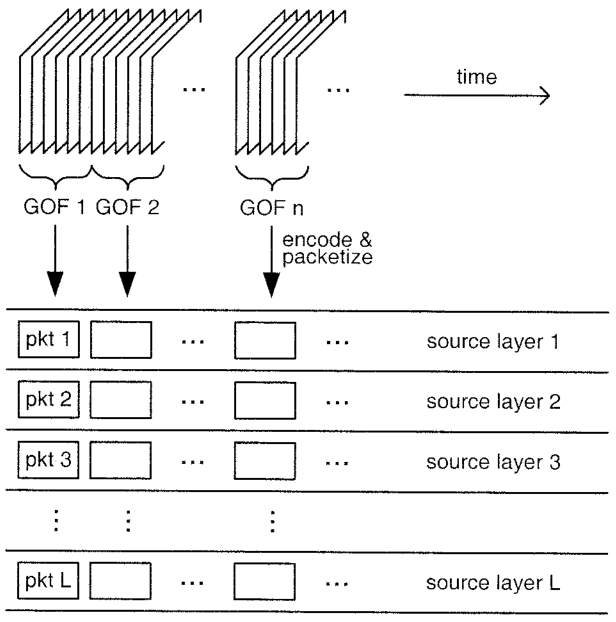

Fig. 1. Source packetized into layers.

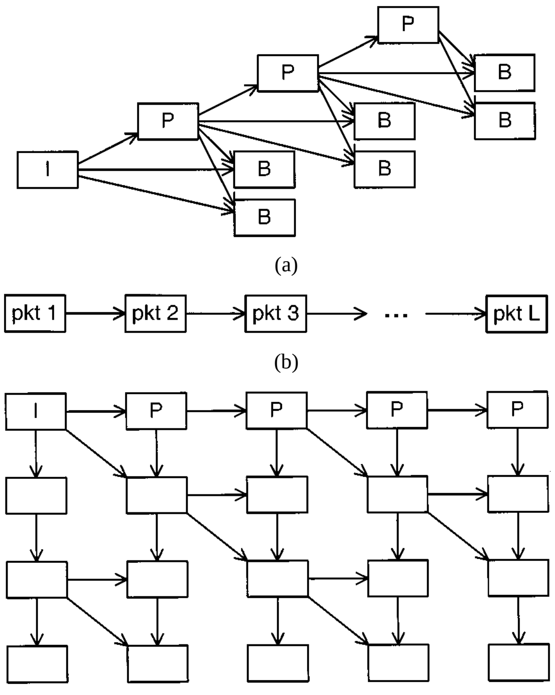

Fig. 2. Dependency graphs between packets in a GOF.

Fig. 2(a). Similarly, if every GOF is encoded into an embedded bit string that is partitioned into a sequence of packets, then the dependencies between packets could be characterized by the sequential dependency graph illustrated in Fig. 2(b). Hybrids of these are also possible. For example, typical dependencies for MPEG-4 fine grain scalability mode are illustrated in Fig. 2(c). We simply assume that the dependency graph is known at the decoder, along with the quantities $\Delta R_l \geq 0$ and $\Delta D_l \geq 0$ at each node $l$ of the graph, which respectively represent the expected increase in rate (per group of frames) if packet $l$ is transmitted and the expected decrease in distortion (per group of frames) if packet $l$ is decoded.

We now derive an expression for the expected rate and distortion corresponding to any pruned subgraph of a given dependency graph. We say that $G$ is a *pruned subgraph* of $G_{\max}$, written $G \preceq G_{\max}$, if $G$ is a subgraph of $G_{\max}$ and for every node $l$ in $G$, all of its ancestors in $G_{\max}$ are also its ancestors in $G$. We write $l' \prec l$ if $l'$ is an ancestor of $l$ (or equivalently if $l$ is a descendant of $l'$). Let $G_{\max}$ be the dependency graph for all available source layers, i.e., all source layers multicast by the sender, and let $G$ be the pruned subgraph of $G_{\max}$ corresponding to the source layers to which a receiver has subscribed. (We assume that if a receiver subscribes to a source layer then it also subscribes to all source layers on which it depends.) The transmission rate requested by the receiver is thus the sum of $\Delta R_l$ over all the nodes $l$ in $G$:

$$R(G) = \sum_{l \in G} \Delta R_l.$$

Similarly, the expected distortion experienced by the receiver is

$$D(G) = D_0 - \sum_{l \in G} P_{l' \preceq l} \Delta D_l$$

where $D_0$ is the expected distortion when the rate is zero (i.e., when the receiver does not subscribe to any source layers) and $P_{l' \preceq l}$ is the probability that packet $l$ and all its ancestors are received. Note that $P_{l' \preceq l}$ depends on the channel model and possibly on packet transmission sequence. If the packet losses are independent with probability $\epsilon$, then $P_{l' \preceq l} = (1-\epsilon)^{|\{l' : l' \preceq l\}|}$.

It is an interesting problem to find a pruned subgraph $G$ of a given graph $G_{\max}$ that minimizes $D(G)$ subject to a constraint on $R(G)$. By considering only those graphs $G$ on the lower convex hull of the set $\{(R(G), D(G)) \colon G \preceq G_{\max}\}$, this problem can be simplified to finding a $G$ that minimizes $D(G) + \lambda R(G)$ for some Lagrange multiplier $\lambda \geq 0$ corresponding to the rate constraint. It can be shown that the sequence of graphs $G_{\max}, G_{\lambda_1}, G_{\lambda_2}, \cdots$ minimizing $D(G) + \lambda R(G)$ for an increasing sequence of Lagrange multipliers $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots$ is nested: $G_{\max} \succeq G_{\lambda_1} \succeq G_{\lambda_2} \succeq \cdots$. Furthermore, if $G_{\max}$ is a tree, dynamic programming can be used to find $G_\lambda$ [38]. Unfortunately, if $G_{\max}$ is not a tree, dynamic programming cannot be used. However, it is possible to find $G_\lambda$ using a simple iterative method, which will be described in Section V. This iterative method will also be able to solve the problem of optimally allocating rate between source and channel codes on such graphs.

## IV. FEC Coding Model

In this section, we outline our model for channel coding and packetization. We assume that channel coding for each source layer is performed as follows. Each source layer is partitioned into coding blocks having $K$ source packets per coding block. The blocksize $K$ is constant across all source layers and in fact the block boundaries are synchronized across the layers so that $K$ represents the overall coding delay. For example, $K = 8$ implies a coding delay of 8 s, if a group of frames has a duration of 1 second.

For each block of $K$ source packets in a source layer, we assume that $N - K$ parity packets are produced using a systematic $(N, K)$ Reed–Solomon style erasure correction code. The "code length" $N$ is determined by the maximum amount of redundancy that will be needed by any receiver to protect the source layer. ($N$ may vary across source layers). The $N-K$ parity packets are generated bytewise from the $K$ source packets, using for each byte the generator matrix from a systematic Reed–Solomon style code over the finite Galois field $GF(2^8)$. That is, if the source packets $\mathbf{s_1}, \cdots, \mathbf{s_K}$ are interpreted as fixed-length column vectors of bytes and $P$ is the $K \times (N-K)$ matrix for generating the parity portion of a systematic Reed–Solomon type code, then the parity packets $\mathbf{p_1}, \cdots, \mathbf{p_{N-K}}$ are computed by

$$[\mathbf{s_1} \quad \cdots \quad \mathbf{s_K}]\,P = [\mathbf{p_1} \quad \cdots \quad \mathbf{p_{N-K}}].$$

Since $P$ is constructed such that any $K$ columns of $G = [P \;\; I]$ are linearly independent, any received subset of $K$ source or parity packets can be used to reconstruct the original $K$ source packets, by multiplying the received packets by the inverse of $G_r$, where $G_r$ consists of the $K$ columns of $G$ corresponding to $K$ of the received packets.

Each of the parity packets so generated is placed in its own multicast stream, so that each source layer (at rate 1 packet per GOF) is accompanied by $N-K$ parity layers, each at rate $1/K$ packets per GOF, as illustrated in Fig. 3. Thus a receiver now has many layers to which it can subscribe. It can subscribe to any collection of source layers and any collection of parity layers associated with those source layers.

The coding delay illustrated in Fig. 3 is $K = 3$ GOFs. To see that, assume that the delays due to computation and transmission are zero, so that the end-to-end delay (which in general is equal to the sum of coding, computation, and transmission delays) is equal to the coding delay only. The sender buffers frames as they arrive. When a group of frames is accumulated, the sender instantly encodes, packetizes, and transmits the group of frames. After $K$ such groups of frames, the sender instantly computes the $N-K$ parity packets and transmits them. The receiver instantly recovers as many source packets as possible from the received source and parity packets, and decodes them. Playback begins after exactly $K$ GOFs of coding delay. Of course in a real system computation and transmission delays are nonzero and the end-to-end delay may be significantly more than $K$ GOFs.

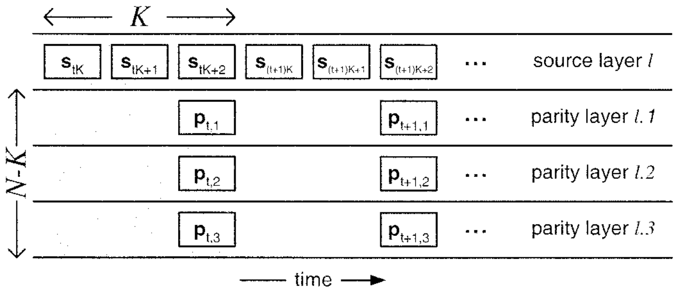

Fig. 3. Generation of parity packets: block each source layer into $K$ packets per block; apply a systematic Reed–Solomon type code (bytewise) to packets in block to produce $N-K$ parity packets; send each parity packet to a different multicast group.

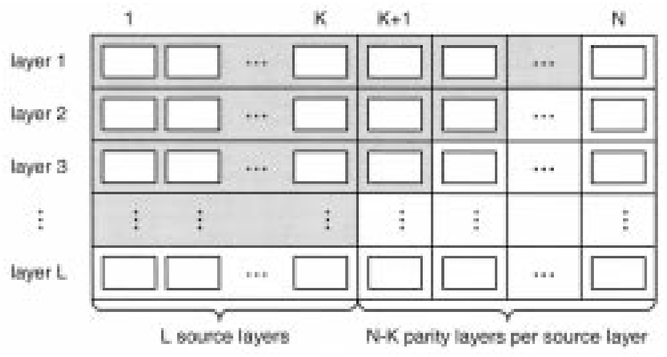

Fig. 4. Source and parity packets multicast by the sender every code block, identified by the layers they are in. The packets in the shaded area are those to which a hypothetical receiver subscribes.

## V. Optimal Selection of Source and Parity Layers

In this section, we solve the problem of finding the collection of source and parity layers to which the receiver should subscribe in order to minimize its expected reconstruction error, given that the receiver's channel from the sender has a fixed packet loss probability and maximum transmission rate. Formally, this is equivalent to the problem of optimal allocation of transmission rate between source and channel codes.

We assume that all packets transmitted within the time period of a $K$-GOF code block are lost independently with some probability $\epsilon$. Although in the Internet packet losses appears to have a very long term correlation structure [39], over relatively short periods of time (e.g., $K$ GOFs) the packet loss probabilities appear to be approximately independent. Indeed, a common model of packet loss in the Internet is the two-state Gilbert–Elliot hidden Markov model, according to which the channel changes infrequently between "good" and "bad" states, while within each state packet losses are independent and identically distributed. Accordingly, in our system, we assume that the underlying packet loss probability $\epsilon$ is constant over each code block. It is the responsibility of the receiver to update its estimate of this quantity after each code block, as well as to update its estimate of the available transmission rate. This responsibility is described further in Section VIII.

Fig. 4 depicts the source and parity packets multicast by the sender for every code block, identified by the layers they are in. In the figure, $L$ is the number of source layers, $K$ is the number of source packets per code block per source layer, and $N-K$ is the number of parity packets per code block per source layer. Thus there are $N-K$ parity layers associated with each source layer. The packets/layers in the shaded area are those to which a hypothetical receiver subscribes. Let $N_l$ be the number of source plus parity packets in the code block for source layer $l$ to which the receiver subscribes. Then $N_l$ takes the value 0 if the receiver does not subscribe to source layer $l$ or any of its associated parity layers; it takes the value $K$ if the receiver subscribes to source layer $l$ but to none of its associated parity layers; and it takes the values $K+1, \cdots, N$ if the receiver subscribes to source layer $l$ and $1, \cdots, N-K$ of its associated parity layers, respectively. Let the redundancy $\rho_l = N_l/K$ be the number of packets per GOF transmitted to the receiver in layer $l$. The redundancy $\rho_l$ is the inverse of the rate $K/N_l$ of the $(N_l, K)$ erasure correction code that the receiver will use to recover any missing source packets in layer $l$.

The vector $\boldsymbol{\rho} = (\rho_1, \cdots, \rho_L)$ is called the *rate allocation vector* in this paper. The rate allocation vector specifies which source layers to subscribe to, and also which parity layers to subscribe to for each source layer. In this way, the rate allocation vector specifies the allocation of the total transmission rate between source and parity layers.

Any given rate allocation vector $\boldsymbol{\rho}$ induces a total transmission rate and an expected distortion. The total transmission rate, in terms of transmitted packets per GOF, is given by

$$R(\boldsymbol{\rho}) = \sum_l \rho_l. \tag{1}$$

This is just the total number of packets in a block to which the receiver subscribes (i.e., the number of packets in the shaded area in Fig. 4) divided by $K$.

The expected distortion for a given rate allocation is somewhat more complicated to express. For each source packet $k = 1, \cdots, K$ in layer $l$ to which the receiver subscribes let $I_k^{(l)}$ be the indicator random variable that is 1 if the receiver can recover the source packet after channel decoding, and is 0 otherwise. Then the probability that the receiver can recover the source packet after channel decoding is given by $EI_k^{(l)}$. Actually this quantity does not depend on $k$, because by symmetry, the random variables $I_k^{(l)}$, $k = 1, \cdots, K$, are exchangeable. Hence

$$P\left\{I_k^{(l)} = 1\right\} = EI_k^{(l)} = \frac{1}{K} E\left[\sum_k I_k^{(l)}\right] = \frac{1}{K} M(N_l, K)$$

where $M(N_l, K)$ is the expected number of source packets that can be recovered after channel decoding with a $(N_l, K)$ code. If the source and parity packet losses are independent with probability $\epsilon$, then when $N_l \geq K$

$$M(N_l, K) = \sum_{i=0}^{N_l} \binom{N_l}{i} \epsilon^{N_l - i}(1-\epsilon)^i M(N_l, K|i)$$

where $M(N_l, K|i)$ equals $K$ if $i \geq K$ and equals $iK/N_l$ if $i < K$. When $N_l = 0$ (i.e., when the receiver does not subscribe to layer $l$), then $M(N_l, K) = 0$. Now, the $k$th source packet in layer $l$ can be decoded if and only if it and all of its ancestors can be recovered after channel decoding, i.e., if and only if the product $\prod_{l' \preceq l} I_k^{(l')}$ equals 1. When this packet is decoded, then the reconstruction error reduces by a random quantity, say $\Delta d_k^{(l)}$. Otherwise the reduction in reconstruction error is zero. Since this random reduction in reconstruction error is independent of the packet loss process, the *expected* reduction in reconstruction error is given by $E[\prod_{l' \preceq l} I_k^{(l')}] \times \Delta D_l$, where $\Delta D_l = E\Delta d_k^{(l)}$. That is, the expectation factors. Furthermore, if the packet losses across layers are independent, then the expected reduction in distortion factors still further: $\prod_{l' \preceq l} EI_k^{(l')} \times \Delta D_l$. Hence the expected distortion (per GOF) for the given rate allocation vector $\boldsymbol{\rho}$ is

$$D(\boldsymbol{\rho}) = D_0 - \sum_l \left(\prod_{l' \preceq l} (1 - \epsilon(\rho_{l'}))\right) \Delta D_l \tag{2}$$

where $\epsilon(\rho_l) = 1 - EI_k^{(l)} = (1 - M(K\rho_l, K)/K) = P\{I_k^{(l)} = 0\}$ is the residual probability of packet loss after channel decoding.

With (1) and (2) for the transmission rate and expected distortion for any given rate allocation vector now in hand, we are able to optimize the rate allocation vector to minimize the expected distortion subject to a transmission rate constraint. By restricting ourselves to solutions on the lower convex hull of the set of rate-distortion pairs $\{(R(\boldsymbol{\rho}), D(\boldsymbol{\rho})\}$, we can solve the problem by finding the rate allocation vector $\boldsymbol{\rho}$ that minimizes the Lagrangian

$$\begin{aligned} J(\boldsymbol{\rho}) &= D(\boldsymbol{\rho}) + \lambda R(\boldsymbol{\rho}) \\ &= D_0 + \sum_l \left[\left(-\prod_{l' \preceq l} (1 - \epsilon(\rho_{l'}))\right) \Delta D_l + \lambda \rho_l\right]. \end{aligned} \tag{3}$$

The solution to this problem is completely characterized by the set of distortion increments $\Delta D_l$ (which are determined by the source, source code, and source packetization) and the residual loss probability function $\epsilon(\rho)$ (which is determined by the channel, channel code, and channel packetization). This simplifies the problem of determining the quantities needed for the optimization. However, the minimization itself is complicated by the fact that the expression for the distortion cannot be split into a sum of terms each depending on only a single $\rho_l$, as is usually the case in bit allocation problems. By approximating the Lagrangian, the optimal rate allocation was solved (approximately) by [18], [19] in the case of sequential dependence [shown in Fig. 2(b)]. Quite recently, Chande and Farvardin showed how to solve the problem exactly, using dynamic programming [20], also for the case of sequential dependence. Here, we solve the problem using an iterative descent algorithm. The advantage to the iterative descent algorithm is that it handles the case of packet dependencies given by a general directed acyclic graph. It also easily generalizes to the case where error control is performed by a hybrid of FEC and ARQ, which we consider in Section VII.

Our iterative approach is based on the method of alternating variables for multivariate minimization [40]. The objective function $J(\rho_1, \cdots, \rho_L)$ in (3) is minimized one variable at a time, keeping the other variables constant, until convergence. To be precise, let $\boldsymbol{\rho}^{(0)}$ be any initial rate allocation vector and let $\boldsymbol{\rho}^{(t)} = (\rho_1^{(t)}, \cdots, \rho_L^{(t)})$ be determined for $t = 1, 2, \cdots$, as follows. Select one component $l_t \in \{1, \cdots, L\}$ to optimize at step $t$. This can be done round-robin style, e.g., $l_t = (t \bmod L)$. Then for $l \neq l_t$, let $\rho_l^{(t)} = \rho_l^{(t-1)}$, while for $l = l_t$, let

$$\begin{aligned} \rho_l^{(t)} &= \arg\min_{\rho_l} J\left(\rho_1^{(t)}, \cdots, \rho_{l-1}^{(t)}, \rho_l, \rho_{l+1}^{(t)}, \cdots, \rho_L^{(t)}\right) \\ &= \arg\min_{\rho_l} S_l^{(t)} \epsilon(\rho_l) + \lambda \rho_l \end{aligned} \tag{4}$$

where (4) follows from (3) with

$$S_l^{(t)} = \sum_{l' \succeq l} \prod_{\substack{l'' \preceq l' \\ l'' \neq l}} \left(1 - \epsilon\left(\rho_{l''}^{(t)}\right)\right) \Delta D_{l'}.$$

The factor $S_l$ can be regarded as the sensitivity to losing a source packet in layer $l$, i.e., the amount by which the distortion will increase when a source packet in layer $l$ cannot be recovered, given the current amount of erasure protection on the other layers. Another interpretation of $S_l$ is as the partial derivative of (2) with respect to $\epsilon_l = \epsilon(\rho_l)$, evaluated at $\boldsymbol{\rho}^{(t)}$. See [34], [35].

Now, the solution in (4) is simple: set $\rho_l$ equal to the redundancy at which the line at slope $-\lambda$ supports the graph of $S_l^{(t)}\epsilon(\rho)$, as shown in Fig. 5. This is equivalent to the redundancy at which the line at slope $-\lambda/S_l^{(t)}$ supports the graph of $\epsilon(\rho)$. In this way, the rate allocation vector $\boldsymbol{\rho}^{(t)}$ is determined and the process is repeated until $J(\boldsymbol{\rho}^{(t)})$ converges. Convergence is guaranteed, because $J(\boldsymbol{\rho}^{(t)})$ is nonincreasing and bounded below. By adjusting $\lambda$, the overall rate constraint can be met.

The algorithm is a descent algorithm, which in principle may get stuck in a nonglobal minimum. However, we have never (knowingly) observed convergence to such a nonglobal minimum. In our experiments, we have always started with the initial rate allocation vector equal to $(1, 1, \cdots, 1)$ and cycled through the components, beginning with the component associated with the root of the dependency graph (i.e., the base layer) and ending with the components associated with the leaves.

The resulting rate allocation $\rho_1^*, \cdots, \rho_L^*$ in general will not be equal across layers, because the packet loss sensitivities $S_1, \cdots, S_L$ in general are not equal across layers. Thus the layers are provided with unequal amounts of protection.

## VI. Pseudo-ARQ Coding Model

Even with unequal error protection, FEC does not achieve the capacity of the packet erasure channel, except in the limit of large blocksizes. ARQ, on the other hand, makes optimal use of the forward channel by transmitting only as many redundant packets as lost packets. In addition, it adapts automatically to the packet loss probability. For these reasons, ARQ is used extensively for data transmission and even for real-time media transmission, such as video-on-demand. But in the broadcast case, ARQ is usually regarded as impractical because of the feedback (negative acknowledgment, NAK, or repeat-request) implosion problem. However, we observe that for broadcast of real-time media, the delay is bounded, so that the number of repeat-requests has to be limited to a finite number. In addition we observe that for very large numbers of receivers, each packet will be lost by at least one receiver. With these observations, it makes sense for the sender to repeatedly retransmit all of the packets all of the time, up to the delay bound.

This leads to the idea of pseudo-ARQ, illustrated in Fig. 6, in which the sender transmits delayed versions of the source to different multicast groups. If a receiver loses a packet, it can obtain a repeated packet by joining (and soon leaving) the multicast group to which the delayed version of the source is transmitted. If it loses the repeated packet, then it can obtain a re-repeated packet by joining and leaving the multicast group to which a further delayed version of the source is transmitted, and so on, until the packet is received or the delay bound is reached. In this way, a receiver effects a repeat request by sending a pair of join and

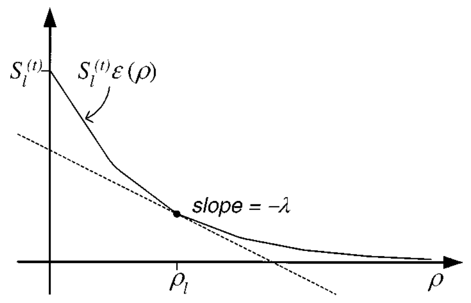

Fig. 5. Solution to $\arg\min_{\rho_l} S_l^{(t)}\epsilon(\rho_l) + \lambda\rho_l$.

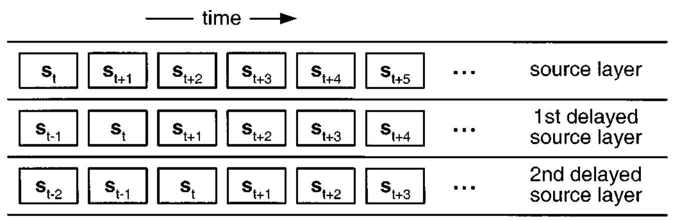

Fig. 6. Pseudo-ARQ for a single source stream. Delayed versions are multicast to different multicast groups.

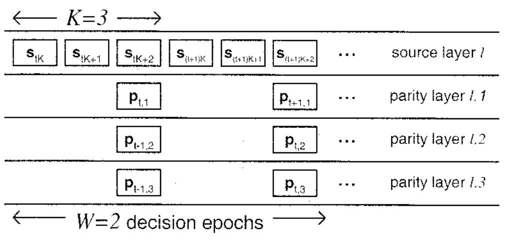

Fig. 7. Hybrid FEC/pseudo-ARQ for a single source stream. Delayed versions of the parity packets are multicast to different multicast groups.

leave messages into the network. To the sender, pseudo-ARQ looks like ordinary multicast, while to the receiver it looks like ordinary ARQ. Thus "pseudo-ARQ," as it is referred to in this paper, uses the existing multicast protocols to avoid the problem of the repeat-requests imploding upon the sender or upon other designated retransmitters, but in other ways it is completely equivalent to ARQ. The mechanism is completely scalable, to an unlimited number of receivers, with no explicit retransmitters or additional servers.

In pure pseudo-ARQ, the number of delayed versions that must be multicast by the sender is proportional to the delay bound. Because this number can be excessively high, we actually advocate a hybrid of FEC and Pseudo-ARQ techniques, in which the sender delays the parity packets rather than the source packets themselves, so that the "repeated" information is actually parity information. Fig. 7 illustrates hybrid FEC/Pseudo-ARQ for one source layer, blocked as before for $K = 3$ source packets, each block producing $N - K = 3$ parity packets, two of which are delayed by one code block into a second "epoch" of parity packets for the block. The coding delay for this system (six) is the blocklength (three) times the number of epochs (two). Other hybridizations are also possible, each with an arbitrary number of epochs, and an arbitrary number of parity packets per epoch. The coding delay is always the product of the blocklength and the number

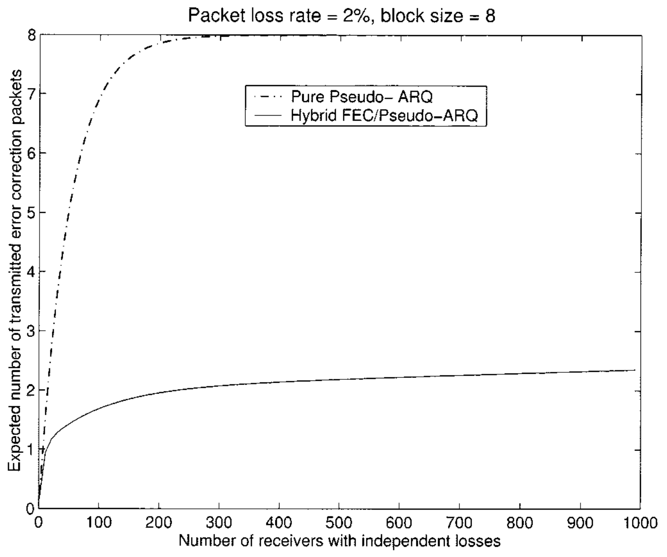

Fig. 8. Expected number of redundant packets transmitted per block as a function of the number of independent receivers. Solid line is hybrid FEC/pseudo-ARQ; dashed line is pure pseudo-ARQ.

of epochs. Source packets are sent only during the first epoch. However, if a source packet is not transmitted during the first epoch, then a subsequent parity packet can be transmitted in its stead, to recover the missing source packet.

Hybridizing FEC with pseudo-ARQ reduces network traffic and multicast router overhead compared to pure pseudo-ARQ. If the receivers have independent losses, they will lose different packets in a block. Yet only one retransmitted parity packet will satisfy them all. Fig. 8 shows the expected number of retransmitted packets as a function of the number of independent receivers, under 2% packet loss. With a blocksize of eight, even when there are a thousand receivers with independent losses, the expected number of redundant packets transmitted per block by hybrid FEC/pseudo-ARQ is only about two, whereas with pure pseudo-ARQ, essentially all eight packets are retransmitted when there are only two or three hundred receivers with independent losses. In addition hybrid FEC/pseudo-ARQ increases the amount of time between retransmission requests (i.e., the minimum time between leaves and joins) from one GOF to $K$ GOFs between requests. Thus network traffic is reduced and scalability is enhanced, in the hybrid system.

## VII. Optimal Error Control Using Hybrid FEC/Pseudo-ARQ

In hybrid FEC/Pseudo-ARQ, the control process at each receiver for joining and leaving the various multicast groups (as a function of which packets have been received) is stochastic, depending on the packet loss process. Indeed, this stochastic control process can be regarded as a finite horizon Markov decision process.

A Markov decision process with finite horizon $W$ is a $W$-step stochastic process through a state space in which an action can be taken at each state in a corresponding trellis of length $W$ to influence the outgoing transition probabilities and thereby maximize an expected reward or minimize an expected cost. The assignment of actions to trellis states is called a *policy* and the optimal policy, in our context, is the one that minimizes the expected cost $D+\lambda R$ of traversing the trellis in $W$ steps starting from a known initial state.

Fig. 9 illustrates the trellis for the FEC/Pseudo-ARQ example of Fig. 7, in which the blocklength is three, and the number of epochs (or steps) is two. Starting from a known initial state for each block, the receiver can subscribe to 0, 3, or 4 packets in the first epoch and 0, 1, or 2 packets in the second epoch. These subscriptions are the receiver's actions. If the receiver subscribes to no packets in the first epoch, then it cannot receive any packets and ends in a final state without receiving any source or parity packets. However, if it subscribes to 3 (source) packets, then it can receive 0, 1, 2, or 3 source packets (and 0 parity packets), while if it subscribes to 4 (3 source plus 1 parity) packets, then it can receive 0, 1, 2, or 3 source packets and 0 or 1 parity packets. In these cases, if it receives a total of three or more packets, then it can reconstruct all three source packets and reach a final state. Otherwise, if it receives fewer than three packets, then it can subscribe to 0, 1, or 2 parity packets in the second epoch with the associated possible outcomes. Each path through this state space has a cost in terms of $D+\lambda R$ and each receiver must determine the optimal policy that minimizes this expected cost.

More generally, let $s$ be the number of source packets received, let $c$ be the number of parity packets received, and let $w$ be the decision epoch. These quantities define the state in the trellis of the Markov decision process for a given code block of a given source layer $l$. Initially, $s = c = w = 0$. At the beginning of each decision epoch $w = 0, \cdots, W-1$, the receiver may request the server to transmit $a$ packets. In the multicast scenario, the receiver accomplishes this by subscribing to the appropriate number of layers during the decision epoch. The value of $a$ may take on any value in an action alphabet $A_{s,c,w}$, possibly depending on the trellis state. A policy $\pi$ assigns to each trellis state an action in the appropriate alphabet [41]. We seek the optimal policy $\pi_l$ for each layer $l$ that minimizes

$$\begin{aligned} J(\boldsymbol{\pi}) &= D(\boldsymbol{\pi}) + \lambda R(\boldsymbol{\pi}) \\ &= D_0 + \sum_l \left[ \left( - \prod_{l' \preceq l} (1 - \epsilon(\pi_{l'})) \right) \Delta D_l + \lambda R(\pi_l) \right]. \quad (5) \end{aligned}$$

Here, $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_L)$ is a "policy allocation vector," $\epsilon(\pi_l)$ is the probability that a particular packet is not recovered in the code block for layer $l$ (after up to $W$ retransmissions) under error control policy $\pi_l$, and $R(\pi_l)$ is the expected number of packets transmitted for the code block under the policy, normalized by the blocklength $K$. As in Section V, (5) can be solved by starting with an initial policy allocation vector $\boldsymbol{\pi}^{(0)}$ and for $t = 1, 2, \cdots$, iteratively minimizing $J(\boldsymbol{\pi}^{(t)})$ over component $l = l_t$ by choosing

$$\pi_l^{(t)} = \arg\min_{\pi_l} S_l^{(t)} \epsilon(\pi_l) + \lambda R(\pi_l) \quad (6)$$

where

$$S_l^{(t)} = \sum_{l' \succeq l} \prod_{\substack{l'' \preceq l' \\ l'' \neq l}} \left(1 - \epsilon\left(\pi_{l''}^{(t)}\right)\right) \Delta D_{l'}$$

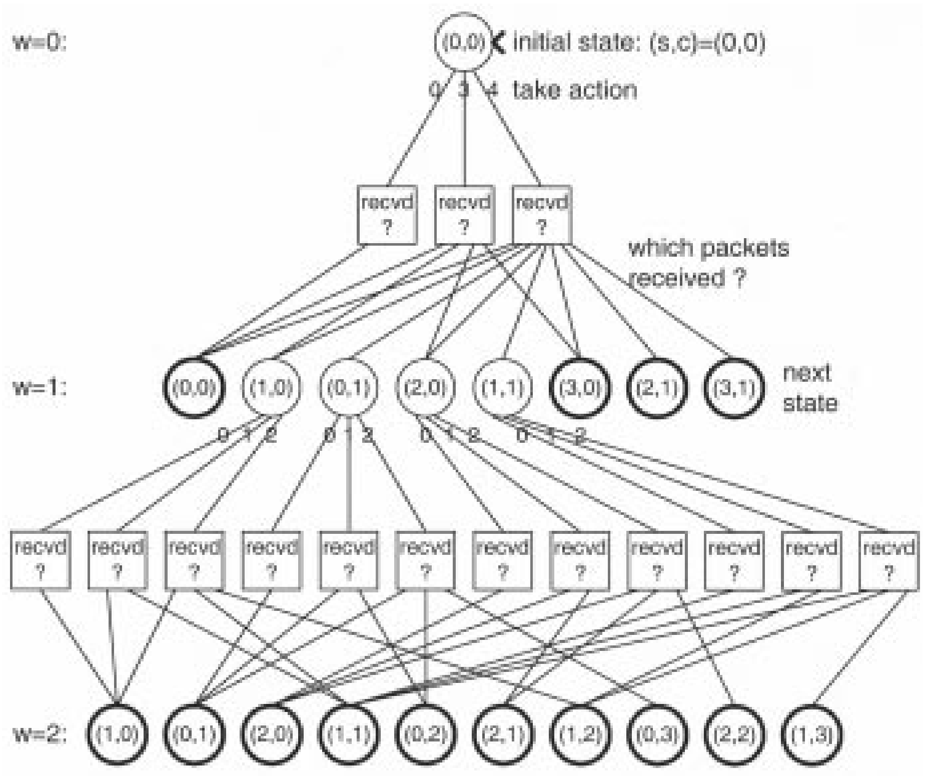

Fig. 9. Trellis for a Markov decision process. Round nodes are states; bold nodes are final states. At each state an action can be taken, leading to a different set of transition probabilities. States are indexed by epoch $w$ and by $(s, c)$, the number of received source and parity packets, respectively.

is the sensitivity of losing a packet in layer $l$ under the policy allocation vector $\boldsymbol{\pi}^{(t)}$. That is, $S_l^{(t)}$ is the expected increase in distortion when a packet is not recovered, or equivalently, the expected decrease in distortion when a packet is recovered. In Section V, the component minimization (6) could be solved simply by exhaustive search. Here, since the number of policies may be quite large, we require a more clever algorithm. Indeed, (6) can be solved by a dynamic programming algorithm, as follows.

For a given code block in a given layer, Let $M_\pi$ be the expected number of source packets recovered under policy $\pi$ and let $N_\pi$ be the expected number of total (source plus parity) packets transmitted under policy $\pi$. Then $\epsilon(\pi) = (K - M_\pi)/K$ and $R(\pi) = N_\pi/K$. Thus, minimizing $J(\pi) = S\epsilon(\pi) + \lambda R(\pi)$ in (6) is equivalent to minimizing

$$\tilde{J}_\pi = -SM_\pi + \lambda N_\pi. \tag{7}$$

We now minimize the latter.

For each trellis state $q = (s, c, w)$ and next state $q' = (s', c', w')$, let $P(q'|q, a)$ be the probability of transitioning from $q$ to $q'$ given action $a$. In addition, let $\Delta M(q'|q, a)$ be the increase in source packets recovered during the transition from $q$ to $q'$ given action $a$. That is

$$\Delta M(q'|q, a) = \begin{cases} K - s & \text{if } s + c < K \text{ and } s' + c' \geq K, \\ s' - s & \text{otherwise.} \end{cases}$$

In other words, $\Delta M(q'|q, a)$ equals the increase $s' - s$ in source packets received during epoch $w$, unless the total number of source plus parity packets received reaches $K$ during the epoch, in which case all remaining $K - s$ source packets can be recovered. Similarly let $\Delta N(q'|q, a)$ be the increase in packets transmitted during the transition from $q$ to $q'$ given action $a$. That is

$$\Delta N(q'|q, a) = a$$

where the action $a$ is identified with the number of packets requested for transmission in epoch $w$.

Now let $M_\pi(q)$ be the expected number of source packets recovered beginning in state $q$ under policy $\pi$ and let $N_\pi(q)$ be the expected number of source plus parity packets transmitted beginning in state $q$ under policy $\pi$. Then it is simple to see that $M_\pi(q) = N_\pi(q) = 0$ if $q = (s, c, w)$ is a final state (i.e., $w = W$), while if $q$ is not a final state (i.e., $w < W$)

$$M_\pi(q) = \sum_{q'} P(q'|q, \pi(q)) \left[\Delta M(q'|q, \pi(q)) + M_\pi(q')\right]$$

and

$$N_\pi(q) = \sum_{q'} P(q'|q, \pi(q)) \left[\Delta N(q'|q, \pi(q)) + N_\pi(q')\right].$$

To put these together, let $\tilde{J}_\pi(q) = -SM_\pi(q) + \lambda N_\pi(q)$ and let $\Delta\tilde{J}(q'|q, a) = -S\Delta M(q'|q, a) + \lambda\Delta N(q'|q, a)$. Then $\tilde{J}_\pi(q) = 0$ if $q$ is final, while if $q$ is not final,

$$\tilde{J}_\pi(q) = \sum_{q'} P(q'|q, \pi(q)) \left[\Delta\tilde{J}(q'|q, \pi(q)) + \tilde{J}_\pi(q')\right].$$

Finally, for each nonfinal state $q = (s, c, w)$, define recursively in terms of next states $q' = (s', c', w + 1)$,

$$\tilde{J}_{\pi^*}(q) = \min_a \sum_{q'} P(q'|q, a) \left[\Delta\tilde{J}(q'|q, a) + \tilde{J}_{\pi^*}(q')\right], \tag{8}$$

$$\pi^*(q) = \arg\min_a \sum_{q'} P(q'|q, a) \left[ \Delta\tilde{J}(q'|q, a) + \tilde{J}_{\pi^*}(q') \right]. \tag{9}$$

It is simple to see by induction that the policy $\pi^*$ so defined satisfies $\tilde{J}_{\pi^*}(q) \le \tilde{J}_\pi(q)$ for all $\pi$ and $q$. In particular, $\tilde{J}_{\pi^*} \equiv \tilde{J}_{\pi^*}(0, 0, 0)$ is less than or equal to $\tilde{J}_\pi \equiv \tilde{J}_\pi(0, 0, 0)$ for all $\pi$, and hence $\pi^*$ minimizes (7). The optimal policy $\pi^*$ can be efficiently computed using (8) and (9).

For completeness, it remains to specify $P(q'|q, a)$. We assume that in epoch $w = 0$, the receiver may request up to $K$ source packets and, additionally, up to $n_0$ parity packets. In epochs $w = 1, \cdots, W-1$, the receiver may request up to $n_w$ parity packets. Thus in epoch 0, $q = (0, 0, 0)$, $q' = (s', c', 1)$, $a = 0, \cdots, K + n_0$, $s' = 0, \cdots, \min\{K, a\}$, and $c' = 0, \cdots, \max\{0, a-K\}$, whence

$$P(q'|q, a) = \begin{cases} \binom{a}{s'}(1-\epsilon)^{s'}\epsilon^{a-s'}, & \text{if } a \le K, \\ \binom{K}{s'}(1-\epsilon)^{s'}\epsilon^{K-s'} \\ \quad \binom{a-K}{c'}(1-\epsilon)^{c'}\epsilon^{a-K-c'}, & \text{if } a > K. \end{cases}$$

Similarly, in epochs $w = 1, \cdots, W-1$, $q = (s, c, w)$, $q' = (s', c', w+1)$, $a = 0, \cdots, n_w$, $s = 0, \cdots, K$, $c = 0, \cdots, \sum_{i=0}^{w-1} n_i$, $s' = s$, and $c' = c, \cdots, c + n_w$, whence

$$P(q'|q, a) = \binom{a}{c'-c}(1-\epsilon)^{c'-c}\epsilon^{a-(c'-c)}.$$

## VIII. Receiver Operation

In this section we comment briefly on the overall receiver operation. As we mentioned in the Introduction, the focus of this paper is error control, rather than congestion control. However, in any real multicast receiver one must have some means of congestion control to determine the allowable rate of transmission to the receiver at any given moment. This constrains the cumulative rate of all the source and parity layers to which the receiver can subscribe. The congestion control algorithm must update its estimate of the allowable transmission rate, as well as its estimate of the packet loss probability $\epsilon$, for every block of $K$ GOFs. Congestion control algorithms for receiver-driven layered multicast are explored in [1], [42], [43], [6].

Rate control is the problem of maintaining the transmission rate specified by the congestion control algorithm. Generally, this is accomplished by adjusting the Lagrange multiplier $\lambda$ until the specified transmission rate is achieved. Higher $\lambda$ will result in a lower transmission rate; lower $\lambda$ will result in a higher transmission rate. The Lagrange multiplier for the current group of frames can be determined open loop, by predicting the Lagrange multiplier from the previous transmission history, or it can be determined closed loop, by repeatedly running the optimal error control algorithm until the target transmission rate over a given window is exactly achieved. In either case, the rate control algorithm must update its estimate of the Lagrange multiplier for every block of $K$ GOFs. Use of the Lagrange multiplier for rate control is explained in [44]–[46].

The optimal error control algorithm runs for each block of $K$ GOFs, using the current values of $\epsilon$ and $\lambda$, to determine the layers to which the receiver should subscribe or unsubscribe. Indeed, the algorithm determines, for each block of $K$ GOFs, the optimal subscription/unsubscription policy to follow over the course of $W$ waves of transmission epochs. These $W$ waves of epochs overlap in time with waves transmitted for other $K$-GOF blocks, as illustrated in Fig. 10. Although the waves of epochs from different blocks overlap in time, from an error control point of view the waves of epochs from one block of $K$ GOFs can be treated independently of the waves of epochs from another block of $K$ GOFs. The rate control algorithm, however, may try to limit the total number of packets transmitted within any given window of time. As a result the rate control algorithm may change $\lambda$ significantly during the lifetime of the $W$ waves of a block. The error control algorithm must then adapt by recomputing the optimal policy to be followed *from the current state* in each layer of each block, for the new value of $\lambda$. Ultimately this serves to coordinate the error control across blocks of $K$ GOFs.

The channel decoding algorithm is also run for each block of $K$ GOFs, after sufficient delay to accommodate all retransmission epochs for the block. The channel decoder recovers any missing source packets, if possible, from all source and parity packets received during the $W$ epochs for the block, and forward the recovered source packets to the source decoder. The decoding process runs independently of the control process. The overall receiver operation is illustrated in Fig. 11.

Note that the end-to-end delay includes not only the coding delay of $W$ epochs (containing $K$ GOFs each) but also includes the delays between epochs. The delays between epochs must be large enough to accommodate running the optimization algorithm and joining or leaving the necessary multicast groups. Except for possible effects due to rate control, increasing this delay between waves does not impact the distortion-rate performance of the system. However, when comparing systems with the same end-to-end delay, increasing the delay between waves decreases the allowable coding delay and therefore does indeed impact distortion-rate performance. In the next section we will see that this impact can be relatively low.

## IX. Results

We now present the results of our analyzes and simulations. In this section, unless otherwise noted, we model the channel as having independent 20% packet losses and we model the network as having no delay, jitter, resequencing, or join and leave latencies. We concentrate on a single receiver and do no analysis of aggregate receiver behavior beyond that shown in Fig. 8.

### *A. Analyzes*

For the analyzes in this subsection, we model the source as having an operational distortion-rate function $D(R) = A2^{-2R}$ for an arbitrary constant $A$ and sequential packetization dependencies. Thus in this subsection we do no actual source coding, channel coding, or network transmission.

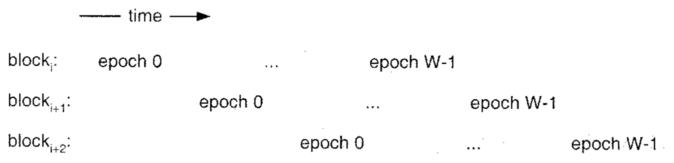
time
block$_i$: epoch 0 ... epoch W-1
block$_{i+1}$: epoch 0 ... epoch W-1
block$_{i+2}$: epoch 0 ... epoch W-1

Fig. 10. Overlapping waves of epochs from different blocks of $K$ GOFs.

We compare the performance of a number of systems of increasing complexity, all based on Receiver-driven layered multicast with multiple source layers. The first is equivalent to RLM, with no error protection. The second employs equal error protection across source layers, with the level of redundancy determined *a priori* (for all receivers and all layers, regardless of transmission rate), for blocksize $K = 8$ and different levels of redundancy, $N = 8, 11, 14, 17, 20$. When $N = 8$, this is equivalent to RLM with no error protection. The third is our FEC system: unequal error protection with redundancy up to $\rho = 2.5$ determined optimally for each layer by each receiver, again for blocksize $K = 8$. Last is our hybrid FEC/Pseudo-ARQ system, in which the coding delay is 8 GOFs, as in the previous systems, but the number of epochs $W$ ranges from one to eight (in powers of two) by reducing the blocksize accordingly, so that $WK = 8$. When $W = 1$ and $K = 8$, this is equivalent to our FEC system. Similarly, when $W = 8$ and $K = 1$, this is equivalent to a pure pseudo-ARQ system.

Fig. 12 shows the signal-to-noise ratio of the end-to-end reconstruction error as a function of the packet transmission rate in packets per GOF. With original RLM, which has no redundancy $((N, K) = (8, 8))$, as the receiver subscribes to more layers its performance saturates, because with high probability there is a loss within the first few layers (rendering the subsequent layers useless) regardless of the number of layers to which the receiver subscribes. With a fixed amount of FEC determined *a priori* $[(N, K) = (11, 8), (14, 8), (17, 8)$, and $(20, 8)]$, performance still eventually saturates for a similar reason, although at a higher level. Putting a very large amount of redundancy on each layer [e.g., $(N, K) = (20, 8)$] and subscribing to only a few layers also penalizes performance somewhat relative to the convex hull (dotted) of the performances of all equal error protection systems. This convex hull can be achieved if the amount of redundancy is matched optimally to the transmission rate, though generally this is not possible in a heterogeneous multicast environment unless the amount of redundancy as well as the transmission rate are receiver-driven. Of course, receiver-driven unequal error protection $((W, K) = (1, 8))$ can perform still better than receiver-driven equal error protection, by up to 3 dB. However, when this is combined with Pseudo-ARQ $[(W, K) = (2, 4), (4, 2)$, and $(8, 1)]$, even when the delay is held constant, performance quickly approaches the optimal performance (the operational distortion-rate function evaluated at the channel capacity) as the number of epochs increases. Indeed, the performance of Pseudo-ARQ for $W = 8$ is indistinguishable from $D((1-\epsilon)R)$, where $(1-\epsilon)R$ is the information-theoretic capacity (in packets per GOF) of the channel with transmission rate $R$ and independent packet loss probability $\epsilon$.

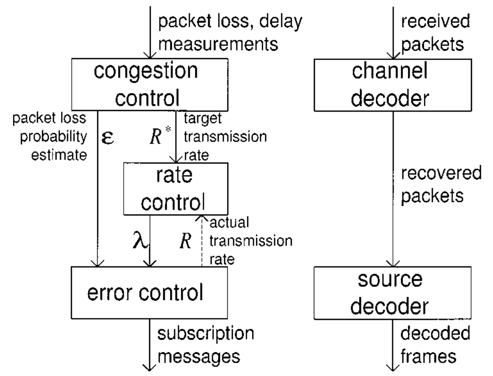

Fig. 11. Overall receiver operation: control and decoding processes.

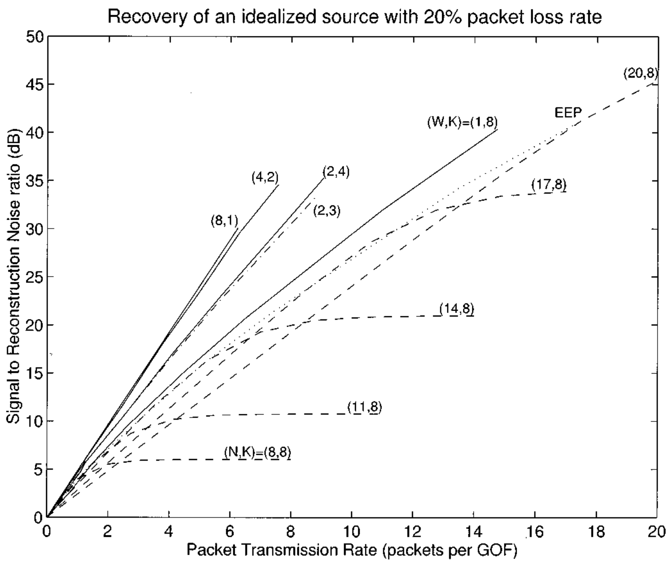

Fig. 12. Analytical results for 20% packet loss, when the leave latency is zero.

The dot-dashed line in Fig. 12 shows the distortion-rate performance of a system with $W = 2$ and $K = 3$. This is the expected performance of a system with two epochs of three GOFs each, without rate control, so that the epochs can be separated by an arbitrary delay, e.g., a delay of two GOFs between epochs for an overall delay of eight GOFs. Compared to the system with $W = 2$ and $K = 4$ having the same overall delay, it is clear that there is a performance penalty to be paid for the two-GOF delay between epochs. However, this penalty is minor compared to the gains that can be had by using multiple epochs.

Fig. 13 shows the same results as Fig. 12, but for a packet loss probability of 5%. The impact of error control coding in this case is slightly smaller, but still quite sizeable.

Fig. 14 shows the same results as Fig. 12, but with a large multicast group leave latency. Join latencies in most multicast protocols are quite short—on the order of a round trip time. But leave latencies are still long in most multicast protocols—on the order of seconds—owing to polling that takes place when downstream receivers leave a group. For example, Internet Group Management Protocol (IGMP) introduces a two-second leave latency between the host and the last-hop router, while Dense Mode Protocol Independent Multicast (PIM) introduces a 3-s

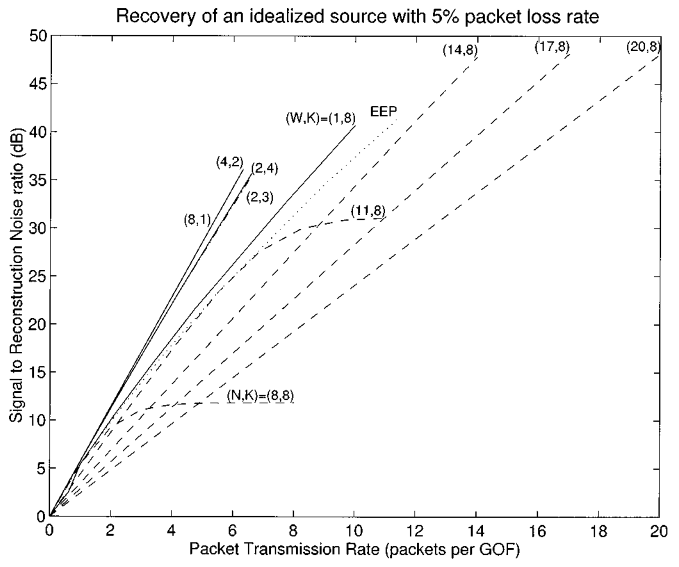

Fig. 13. Analytical results for 5% packet loss, when the leave latency is zero.

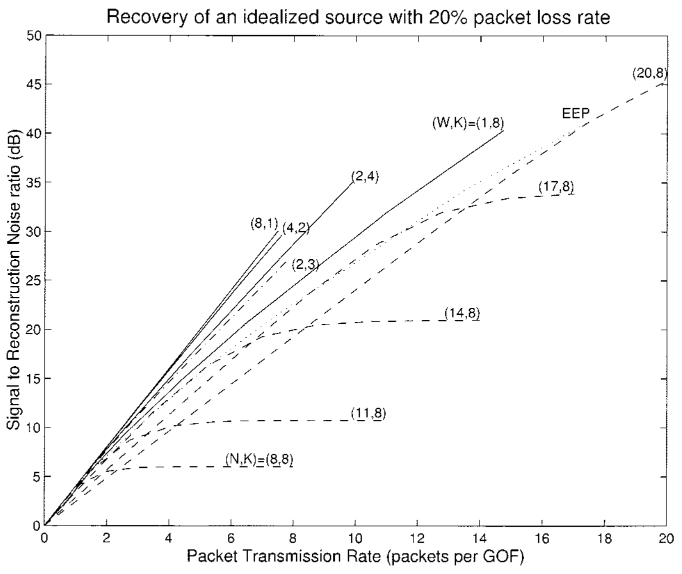

Fig. 14. Analytical results for 20% packet loss, when the leave latency is equal to $K$ GOFs.

leave latency between every two routers connected by a LAN (but no leave latency between directly connected routers) [47]. Thus when a receiver wishes to join a multicast group to receive a (pseudo) retransmission, it can subscribe to a multicast group containing the information and quickly obtain the relevant packet. However, when the receiver subsequently wishes to leave the multicast group, it may take several seconds before the stream flowing to the receiver is actually shut off. This may be enough time for an unwanted packet, from the next block of $K$ GOFs, to flow through, using up bandwidth that could otherwise be better used. We need to solve the problem of minimizing the distortion subject to a constraint on the *actual* number of packets transmitted to the receiver, whether they are wanted or not. If each packet intentionally transmitted after the first epoch incurs one additional undesired packet to be transmitted, then the optimal policy can be found using the algorithm of Section VII by artificially setting the size of all packets past the first epoch to be twice as large as usual. This will penalize the use of retransmissions. Only when the receiver is missing a particularly important packet will it venture to request a retransmission. The results of this modification are shown in Fig. 14, where the leave latency is equal to $K$ GOFs (e.g., eight seconds when $K = 8$ and the duration of a GOF is one second). The performance gains of pseudo-ARQ and hybrid FEC/pseudo-ARQ are reduced, but are still substantial. Moreover, performance still improves with an increasing number of epochs, for the same delay.

## B. Simulations

To test the validity of our analytical results, we run simulations of our system on real data transmitted from a server to a client over a simulated network.

As source data we use the standard 25 fps color QCIF (176 $\times$ 144) sequence *foreman* concatenated with a time-reversed version of itself, to obtain an 600-frame sequence from which we extract the first 576 frames for testing. We partition the test sequence into 18 GOFs containing 32 frames per GOF, and independently code each GOF using 3-D-SPIHT [2] to obtain an embedded bit string for each GOF. We then partition each embedded bit string into sequentially dependent packets containing 1000 bytes per packet. We assign the $l$th packet from each embedded bit string to the $l$th source layer. Thus, the duration of a GOF is 32/25 = 1.28 s, while the transmission rate of each source layer is 8000/1.28 = 6.25 Kbps. We produce up to 32 source layers for a total of up to 200 Kbps for the source.

Encoding and packetization are done off line, with the results written into a file. The server reads the file, computes parity packets on-line, and streams all source and parity layers into the network simulator. The server inserts no delay between successive waves of parity information. The network simulator conveys a subset of the layers to the client with zero transmission delay and jitter and 20% independent packet loss. The network simulator allows the client to reliably subscribe to and unsubscribe from the layers with zero join and leave latency. The client recomputes all optimal policies prior to receiving each wave, to ensure that the total number of packets to which the client subscribes during each block of $K$ GOFs never exceeds the rate constraint. Thus the client obeys an absolute rate constraint rather than just an average rate constraint. (However, unused rate from the current block is allowed to carry over to the next.) The client estimates the parameters $\Delta D_l$ for each layer by starting with a crude initial estimate and averaging over time the actual values from decoded source packets when recoverable. The client recovers as many of the source packets for each GOF as possible, given the received source and parity packets, and writes the recovered packets to a file. Depacketization and decoding of the video sequence are done off line. The overall setup is illustrated in Fig. 15.

We compare five systems: RLM with no error protection, receiver-driven unequal error protection with no retransmission requests $((W, K) = (1, 8))$, and receiver-driven unequal error protection with a varying number of retransmission requests $[(W, K) = (2, 4)$, (4, 2), and (8, 1)] such that the delay is fixed at eight GOFs. In every wave $w = 0, 1, \cdots, W-1$, the number of parity packets transmitted by the server is set to $n_w = K$.

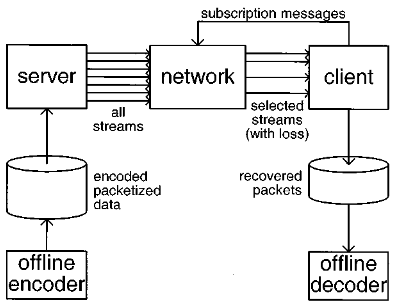

Fig. 15. Simulation setup.

We evaluate each of these systems at four transmission rates: 50, 100, 150, and 200 Kbps. The end-to-end distortion is computed by averaging the distortion over each of the 576 frames of the test sequence, and averaging again over three independent transmission trials (with different random packet loss patterns).

Fig. 16 shows the peak signal to reconstruction noise ratio of each system at each rate. The bottom line representing RLM is 4–9 dB worse than any of the other lines, all of which use protection of some kind. The middle line representing pure FEC performs substantially better. The cluster of lines at the top representing the hybrid approaches provide nearly identical performance and exceed pure FEC by 0.5–1.5 dB, confirming that retransmissions can allow significant quality improvements. The fact that the simulation results differ quantitatively from the analytical results is due to several factors. First, the operational distortion-rate function of real encoded video is not precisely $D(R) = A2^{-2R}$. Indeed the operational distortion-rate function varies from GOF to GOF. In our simulation, for GOF layers $l = 1, \cdots, L$, the quantity $\Delta D_l$ is transmitted as side information in the header of each packet (using up to 32 bits per packet), and the receiver performs backward adaptive estimation of $\Delta D_l$ for the current GOF using previously recovered packets. However, this estimate could be erroneous, particularly if a layer has not been received in a long time, or had never been received. A second reason for the discrepancy between simulation and analytical results is that the simulations use fewer parity packets than the analyzes, restricting for complexity reasons the number of parity packets in each epoch to $n_w = K$. A third reason is that our simulations use absolute rate control, whereas our analyses assume no rate control.

Finally, a note on computational complexity: the server, network simulator, and client in the simulations run in separate Java threads on the same 366 MHz Pentium II machine. Nominally, the simulations runs in real time. However, for simplicity of programming, the optimal error control equations (8) and (9) are implemented in the client using recursive calls instead of dynamic programming. This effectively expands the trellis in the Markov decision process into a tree, and explodes the computational complexity exponentially in the number of epochs $W$. Thus, in the simulations, when $W$ is large (four or more), the real-time simulation must be slowed down by a factor of 16 or 32 to accommodate the policy computation, which is scheduled to run in the second half of the last GOF preceeding each wave. We believe that with careful implementation the policy computation can be made to run in real time under most scenarios of interest.

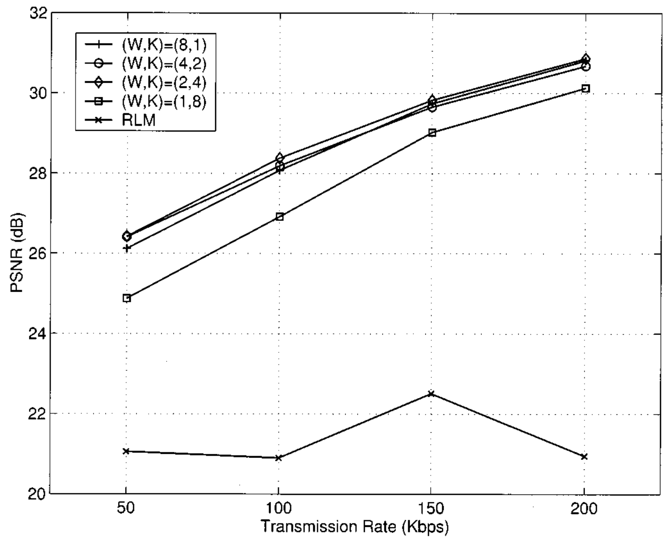

Fig. 16. Simulation results for 20% packet loss.

## X. Conclusion

In summary, we have presented a FEC and hybrid FEC/pseudo-ARQ error control framework for use in receiver-driven layered multicast systems. Pseudo-ARQ uses only existing multicast mechanisms to process the repeat requests and hence is scalable to an unlimited number of receivers without causing repeat request implosions. We also presented algorithms for optimizing each receiver's reconstruction quality at a given transmission rate and packet loss probability. For the hybrid FEC/pseudo-ARQ scheme, this optimization involves finding the optimal policy for a finite horizon Markov decision process. Analytical results show that on a channel with 20% packet loss, receiver-optimized FEC with moderate delay can gain up to 18 dB over systems without explicit error control and receiver-optimized pseudo-ARQ can gain up to an additional 13 dB. Simulation results show corresponding gains of up to 9 dB and 1.5 dB.

As astoundingly high as these gains are, they can be had for free, without the delay and processing load introduced by application-level error control mechanisms, if the routers simply dropped packets in order of priority. Thus our work quantitatively justifies the implementation of network protocols that support prioritized routing, such as relative differentiated services using a proportional loss model [48].

Indeed, our application-level error control essentially creates different qualities of service for transmission of different source layers. Each quality of service provides a packet loss rate $\epsilon(\rho)$ at the cost of $\rho$ transmitted packets per source packet. Our error control algorithms determine how best to use the available budget to transmit multimedia signals using different qualities of service, in order to reconstruct the signal at the receiver with the highest quality. In this sense, our error control algorithms quantify the gains possible with differentiated services, as well as prescribe the optimal use of these services. We leave this aspect of our algorithms to future publications.

## ACKNOWLEDGMENT

The authors would like to thank J. Li for carrying out the initial work on this project as a summer intern at VXtreme in 1997. They would also like to thank E. Riskin and R. Ladner for their helpful discussions and support.

## REFERENCES

[1] S. R. McCanne, "Scalable compression and transmission of Internet multicast video," Ph.D. dissertation, Univ. California, Berkeley, Dec. 1996.

[2] B.-J. Kim and W. A. Pearlman, "An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT)," in *Proc. Data Compression Conference*. Snowbird, UT: IEEE Comput. Soc., Mar. 1997, pp. 251–260.

[3] B.-J. Kim, Z. Xiong, and W. A. Pearlman, "Very low bit-rate embedded video coding with 3D set partitioning in hierarchical trees (3D SPIHT)," *IEEE Trans. Circuits Syst. Video Technol.*, to be published.

[4] A. Wang, Z. Xiong, P. A. Chou, and S. Mehrotra, "Three-dimensional wavelet coding of video with global motion compensation," in *Proc. Data Compression Conference*. Snowbird, UT: IEEE Comput. Soc., Mar. 1999.

[5] S. Li, F. Wu, and Y.-Q. Zhang, "Experimental results with progressive fine granularity scalable (PFGS) coding,", Noordwijkerhout, The Netherlands, Tech. Rep. m5742, ISO/IEC JTC1/SC29/WG11, Mar. 2000.

[6] T. Turletti, S. F. Parisis, and J.-C. Bolot, "Experiments with a layered transmission scheme over the Internet," INRIA, Sophia Antipolis, France, Tech. Rep. 3296, Nov. 1997.

[7] D. Sisalem and H. Schulzrinne, "The loss-delay adaptation algorithm: A TCP-friendly adaptation scheme," in *Proc. Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, U.K., July 1998.

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

[9] W.-T. Tan and A. Zakhor, "Error control for video multicast using hierarchical FEC," in *Proc. Int. Conf. Image Processing*, Kobe, Japan, Oct. 1999.

[10] S. Floyd, M. Handley, and J. Padhye, "Equation-based congestion control for unicast applications," Int. Comput. Sci. Inst., Berkeley, CA, Tech. Rep. TR-00-03, Mar. 2000.

[11] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC Codes) and their applications," *IEEE Trans. Commun.*, vol. 36, pp. 389–400, Apr. 1988.

[12] A. Albanese, J. Blömer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1737–1744, Nov. 1996.

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

[14] A. E. Mohr, E. A. Riskin, and R. E. Ladner, "Graceful degradation over packet erasure channels through forward error correction," in *Proc. Data Compression Conference*. Snowbird, UT: IEEE Comput. Soc., Mar. 1999.

[15] ——, "Unequal loss protection: Graceful degradation of image quality over packet erasure channels through forward error correction," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 819–829, June 2000.

[16] A. E. Mohr, R. E. Ladner, and E. A. Riskin, "Approximately optimal assignment for unequal loss protection," in *Proc. IEEE Int. Conf. Image Processing* Vancouver, BC, Canada, Sept. 2000.

[17] R. Puri and K. Ramchandran, "Multiple description source coding through forward error correction codes," in *Proc. Asilomar Conference on Signals, Systems, and Computers* Asilomar, CA, Oct. 1999.

[18] M. J. Ruf and J. W. Modestino, "Operational rate-distortion performance for joint source and channel coding of images," *IEEE Trans. Image Processing*, vol. 8, pp. 305–320, Mar. 1999.

[19] J. Lu, A. Nosratinia, and B. Aazhang, "Progressive source-channel coding of images over bursty error channels," in *Proc. IEEE Int. Conf. Image Processing* Chicago, IL, Oct. 1998.

[20] V. Chande and N. Farvardin, "Progressive transmission of images over memoryless noisy channels," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 850–860, June 2000.

[21] B. Hochwald and K. Zeger, "Tradeoff between source and channel coding," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1412–1424, Sept. 1997.

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

[23] M. Podolsky, S. McCanne, and M. Vetterli, "Soft ARQ for layered streaming media," Comput. Sci. Div., Univ. California, Berkeley, Tech. Rep. UCB/CSD-98-1024, Nov. 1998.

[24] J. Lu, A. Nosratinia, and B. Aazhang, "Progressive joint source-channel coding in feedback channels," in *Proc. Data Compression Conference*. Snowbird, UT: IEEE Comput. Soc., Mar. 1999, pp. 140–148.

[25] C.-Y. Hsu, A. Ortega, and M. Khansar, "Rate control for robust video transmission over burst-error wireless channels," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 756–773, May 1999.

[26] S. D. Servetto, "Compression and reliable transmission of digital image and video signals," Ph.D. dissertation, Univ. Illinois, Urbana-Champaign, 1999.

[27] V. Chande, N. Farvardin, and H. Jafarkhani, "Image communication over noisy channels with feedback," in *Proc. Int. Conf. Image Processing*, Kobe, Japan, Oct. 1999.

[28] S. Paul, *Multicasting on the Internet and its Applications*. Norwell, MA: Kluwer, 1998.

[29] J. Nonenmacher, E. Biersack, and D. Towsley, "Parity-based loss recovery for reliable multicast transmission," in *Proc. SIGCOMM*. Cannes: ACM, Sept. 1997.

[30] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proc. SIGCOMM*. Vancouver, BC, Canada: ACM, Sept. 1998.

[31] W.-T. Tan and A. Zakhor, "Multicast transmission of scalable video using receiver-driven hierarchical FEC," in *Packet Video Workshop*, New York, Apr. 1999.

[32] ——, "Multicast transmission of scalable video using receiver-driven hierarchical FEC," *Proc. SPIE, Vis. Commun. Image Process.*, Jan. 2000.

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

[34] ——, "FEC and pseudo-ARQ for receiver-driven layered multicast of audio and video," Microsoft Research, Redmond, WA, Tech. Rep. MSR-TR-99-86, Nov. 1999.

[35] ——, "FEC and pseudo-ARQ for receiver-driven layered multicast of audio and video," in *Proc. Data Compression Conference*. Snowbird, UT: IEEE Comput. Soc., Mar. 2000.

[36] L. H. Sahasrabuddhe and B. Mukherjee, "Multicast routing algorithms and protocols: A tutorial," *IEEE Network Mag.*, pp. 90–102, Jan./Feb. 2000.

[37] Z. Lu and W. A. Pearlman, "An efficient, low-complexity audio coder delivering multiple levels of quality for interactive applications," in *Proc. IEEEWorkshop on Multimedia Signal Processing* Redondo Beach, CA, Dec. 1998, pp. 529–534.

[38] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Optimal pruning with applications to tree structured source coding and modeling," *IEEE Trans. Inform. Theory*, vol. 35, pp. 299–315, Mar. 1989.

[39] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of ethernet traffic," *IEEE/ACM Trans. Networking*, vol. 2, Feb. 1994.

[40] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. New York: Wiley, 1987.

[41] M. L. Puterman, *Markov Decision Processes*. New York: Wiley, 1994.

[42] S. R. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proc. SIGGCOM*. Stanford, CA: ACM, Aug. 1996, pp. 117–130.

[43] S. R. McCanne, M. Vetterli, and V. Jacobson, "Low-complexity video coding for receiver-driven layered multicast," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 983–1001, Aug. 1997.

[44] J. Choi and D. Park, "A stable feedback control of the buffer state using the controlled Lagrange multiplier method," *IEEE Trans. Image Processing*, vol. 3, pp. 546–588, Sept. 1994.

[45] T. Wiegand, M. Lightstone, D. Mukherjee, T. George, and S. K. Mitra, "Rate-distortion optimized mode selection for very low bit rate video coding and the emerging H.263 standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 182–190, Apr. 1996.

[46] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Mag.*, vol. 15, pp. 74–90, Nov. 1998.

[47] W. Fenner, "Re: [q] knowledge confirmation; leave latency,", IP Multicast Mailing List, IP Multicast Initiative, http://www.stardust.com/ipmulticast/hypermail/ipmulticast/0976.html, Dec. 1999.

[48] C. Dovrolis and P. Ramanathan, "A case for relative differentiated services and the proportional differentiation model," *IEEE Network Mag.*, pp. 26–34, Sept. 1999.

**Philip A. Chou** (SM'00) was born in Stamford, CT, on April 17, 1958. He received the B.S.E. degree from Princeton University, Princeton, NJ, in 1980, and the M.S. degree from the University of California, Berkeley, CA, in 1983, both in electrical engineering and computer science, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1988.

Since 1977, he has worked for IBM, AT&T Bell Laboratories, Princeton Plasma Physics Lab, Telesensory Systems, Speech Plus, Hughes, Xerox, VXtreme, and Microsoft, where he was involved variously in office automation, motion estimation in television, optical character recognition, LPC speech compression and synthesis, text-to-speech synthesis by rule, compression of digitized terrain, speech and document recognition, and video network communication. His research interests are pattern recognition, data compression, and speech, image, and video processing. In 1994–1995, he was a Consulting Associate Professor at Stanford University. Currently, he is with Microsoft Corporation, in Redmond, WA.

Dr. Chou serves on the Editorial Board of the IEEE TRANSACTIONS ON INFORMATION THEORY as an Associate Editor for source coding, and also serves on the IEEE Technical Committee for Image and Multidimensional Signal Processing (IMDSP). He is a Senior Member of the IEEE, 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 Tom Lookabaugh and Robert M. Gray, of the 1993 Signal Processing Society Paper award.

**Alexander E. Mohr** received the B.S.E. and M.S.E. degrees in bioengineering and the M.S.degree in computer science in 1994, 1999, and 2000, respectively, all from the University of Washington, Seattle, where he is currently pursuing the Ph.D. degree in the Department of Computer Science and Engineering.

His research interests include the design of algorithms and systems for transmitting compressed multimedia over computer communications networks.

**Albert Wang** received the B.S. degree in electrical engineering from Stanford University, Stanford, CA in 1995.

He was pursuing his graduate studies in electrical engineering at Stanford when he left to help form VXtreme, where he developed compression algorithms for streaming media applications. From 1997 to 2000, he worked at Microsoft, Redmond, WA, on wavelet coding, streaming media systems, and palettized video coding. He is now at Rivio, Santa Clara, CA, where he is a Senior Architect working on distributed computing applications.

Mr. Wang received first place in the 1994 IEEE National Programming Contest and third place in the 1992 ACM Scholastic Programming Contest.

**Sanjeev Mehrotra** (M'99) received the B.S. degree from Wright State University, Dayton, OH, in 1994 and the M.S. and Ph.D. degrees from Stanford University, Stanford, CA in 1995 and 2000 respectively, all in electrical engineering.

From 1996 to 1997, he worked at VXtreme on compression algorithms for streaming audio and video. Since 1997, he has worked in the audio/video codecs group in the Windows digital media division at Microsoft, Redmond, WA, working on video compression, multiple description coding, and palletized video coding. His research interests include data compression and streaming media.

Dr. Mehrotra was the recipient of the National Science Foundation, Tau Beta Pi, and Kodak graduate research fellowships during his studies at Stanford.