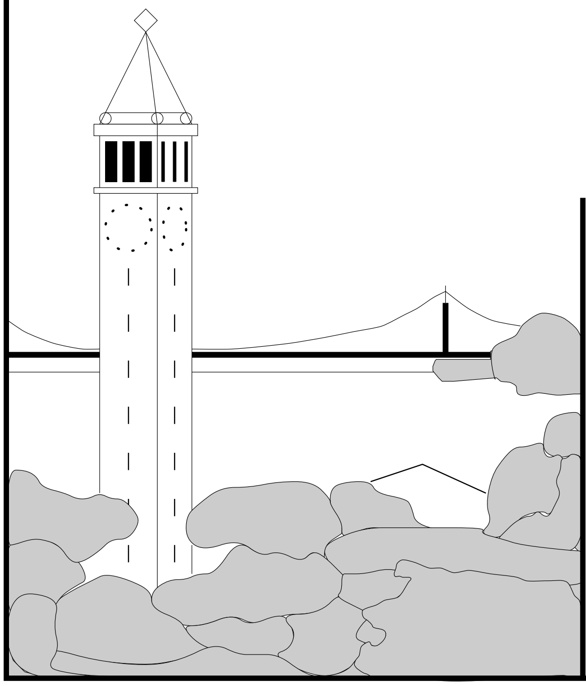
Soft ARQ for Layered Streaming Media

Matthew Podolsky[1], *Steven McCanne*[1], *and Martin Vetterli*[1,2]

[1] *Dept. of Electrical Eng. & Comp. Sci.*
University of California, Berkeley
Berkeley, CA, USA

[2] *Laboratoire de Communications Audiovisuelles*
École Polytechnique Fédérale de Lausanne
Lausanne, Switzerland

E-mail: `m.podolsky@ieee.org`, `{mccanne,martin}@eecs.berkeley.edu`



Report No. UCB/CSD-98-1024

November 1998

Computer Science Division (EECS)
University of California
Berkeley, California 94720

Work supported by DARPA Grant
66001-96-C-8508 and the
California State MICRO Program.

Abstract

A growing and important class of traffic in the Internet is so-called "streaming media," in which a server transmits a packetized multimedia signal to a receiver that buffers the packets for playback. This playback buffer, if adequately sized, counteracts the adverse impact of delay and reordering suffered by packets as they traverse the network. If large enough, that buffer can additionally provide adequate delay for the receiver to request that the source retransmit lost packets before their playback deadline expires. We call this framework for retransmitting lost streaming-media packets "Soft ARQ" since it represents a relaxed form of Automatic Repeat reQuest (ARQ). While schemes for streaming media based on Soft ARQ have been previously proposed, no work to date systematically addresses two important questions induced by Soft ARQ: (1) at any given point in time, what is the optimal packet to transmit? And, (2) when and how does a receiver generate feedback to the source? In this paper, we address both of these questions with a framework for streaming media retransmission based on *layered* media representations, in which a signal is decomposed into a discrete number of layers and each successive layer provides enhanced quality. In our approach, the source chooses between transmitting (1) older but lower-quality information and (2) newer but higher-quality information using a decision process that minimizes the expected signal distortion at the receiver. To this end, we develop a model of our streaming media system based on a binary erasure channel with instantaneous feedback and use Markov-chain analysis to derive the optimal strategy. Based on this analysis, we propose a practical transmission protocol for streaming media that performs close-to-optimal retransmission and can adapt to dynamic network conditions. To demonstrate the efficacy of this protocol, we simulate our system and present results that illustrate significant performance benefits both from layering the media signal and adaptively estimating a retransmission deadline.

1 Introduction

A common class of traffic on the Internet is so-called "streaming media," where real-time signals like audio and video are delivered from a server somewhere in the network to a human user that interactively views the material. Unlike human-to-human communication, which requires relatively tight and consistent end-to-end delays for good interactive performance [6], server-to-human communication can afford a certain level of artificial delay. As a result, streaming media applications often have sufficient time to recover from lost packets through retransmission and thereby avoid unnecessary degradation in reconstructed signal quality. We refer to this delay-constrained Automatic Repeat/reQuest system as "soft ARQ," because it represents a relaxed form of ARQ in which the successful on-time delivery of every packet is not guaranteed.

Soft ARQ has been exploited in research protocols like STORM [30] and MESH [17] and in commercial products like RealNetworks clients and servers. These prior works have focused on how to choose the playout delay and how to decide if retransmissions will arrive in time. However, they assume that when the sender wants to retransmit a packet it will be able to. This is not necessarily true when the there are rate constraints on the sender. In this case, the sender has to consider not only whether a (re)transmitted packet will arrive in time, but if that packet is more beneficial than other unsent packets. There is no existing solution to the problem of how a sender optimally chooses what available data to retransmit when the receiver indicates loss.

In this paper, we propose a framework to solve this optimization problem. In our scheme, the sender represents its signal in a layered format and at any given time transmits the "most important" information conditioned on receiver feedback and constrained by the available bit rate (which is either pre-configured or inferred from a companion congestion control algorithm [25]). We assume and study the case that packet loss is high and the source rate of the highest-quality version of the signal exceeds the available capacity. If the packet loss rate is sufficiently low that the effective channel capacity is larger than the source rate, then simple ARQ would more or less suffice and our problem would be solved.

Figure 1 illustrates our model for a streaming layered multimedia transmission system. The transmission processes begins with a multimedia signal X at the sender. We assume that the entire signal is not available prior to the start of transmission—in other words, it is either generated or retrieved from storage concurrently while the the transmission process is going on. The signal is segmented in time into equal length segments or "frames"; these frames are produced periodically as the signal is generated. We denote frame n as X^n. The signal is also encoded into a hierarchy of N layers $\{X_1, X_2, \ldots, X_N\}$, where X_1 is the most "important" layer and X_N is the least "important" layer, and we assume that all layers have the same bit-rate. We assume that the importance of a layer can be quantified, so that successfully transmitting the most important layer of a frame results in a greater benefit (*e.g.*, a greater increase signal quality or decrease in distortion) than a less important layer of that frame. We denote the ith frame of layer l by X_l^i. These layer/frame segments form the basic transmission units or "messages" that are sent across

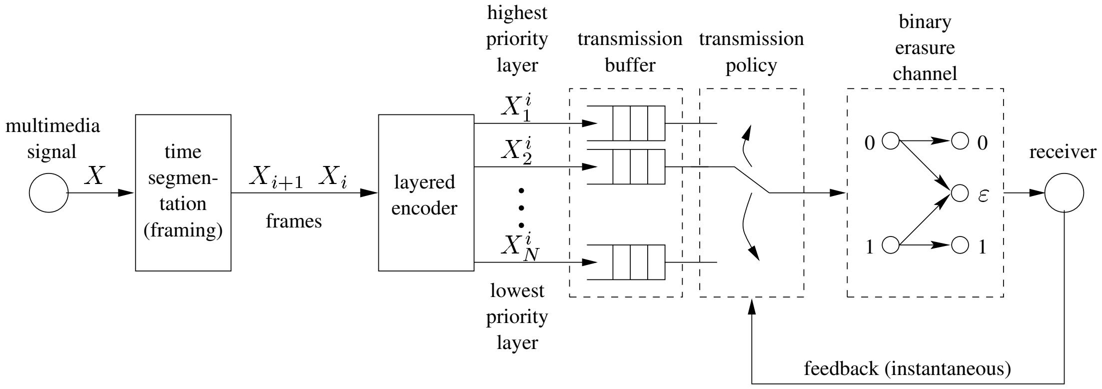


Figure 1: System diagram of layered transmission over a binary erasure channel with feedback.

the network (*e.g.*, contained in packets). The sender operates under a transmission rate constraint, which manifests itself as a lower bound on the minimum time between message transmissions.

To capture the effect of network packet losses, each message passes through a binary erasure channel (BEC) on its way to the receiver. The BEC either erases (drops) a packet with probability ε or successfully transmits the packet with probability $1-\varepsilon$. The BEC, in conjunction with an instantaneous feedback path, serves as an idealized model for the network. We assume a positive acknowledgment (ACK-based) scheme is used for retransmission requests. Messages which successfully reach the receiver are used to reconstruct the signal. Because we have assumed a "streaming" multimedia scenario, the receiver starts playback of the signal even while it is still being generated and transmitted at the source. At some fixed time after frame i is produced at the source, it is reconstructed from whatever layers X_l^i have arrived at the receiver and played back.

An important component of our model is the transmission policy, located at the sender. This policy dictates which message (frame and layer) the source should transmit (or retransmit) for any possible situation. For every feasible set of unsent (or sent but dropped) messages and their corresponding playback deadlines (*i.e.*, the latest time they can be sent before they are no longer useful to the receiver), the transmission policy contains a rule indicating which message the sender should choose to transmit next. Our problem is to find the transmission policy which optimizes the quality of the delivered media signal. We define the optimal policy π^* as the transmission policy that minimizes the distortion of the signal reconstructed from the successfully received messages.

The need for a policy stems from the fact that messages can have both different priorities (due to the layering) and different time constraints (due to the framing and streaming playback). Decisions of what to transmit so as to maximize signal quality are simple when the choice is restricted to messages from within a single frame: the sender should (re-)transmit the most important layer of that frame that has not been successfully transmitted yet. Likewise, decisions what message to transmit among all of the messages of a single layer are also clear: (re-)transmit the oldest message of that layer will still arrive in time for playback. However, the decision is not necessarily clear when choosing between messages from different frames *and* different layers. Specifically, how do you decide between sending an older, lower priority message O_L and a newer, high priority message N_H? (With the above terminology, if $L_0 = X_l^i$ and $N_H = X_m^j$, then we must have $l > m$ and $i < j$.) There are fundamental tradeoffs between the data's importance and its time-constraints. A reason to favor O_L is because it has an earlier playback point than N_H, so there is less time and hence fewer opportunities in which to successfully transmit it. However, an argument for choosing message N_H is that it is part of a more important layer and hence provides a greater distortion reduction than O_L. Choosing to send the less important O_L leaves fewer transmission opportunities for the more important N_H; if the loss rate is high it may take all of those opportunities to successfully transmit N_H. Thus it is not immediately clear which choice is better—*i.e.*, which choice results in a higher average signal quality. As a result, the sender relies on the transmission policy to tell it what choice to make.

A transmission policy consists of a set of decisions like the one above, a set which covers all possible choices that may need to be made. In other words, given all possible sets $\mathcal{L} = \{l_1, \ldots, l_n\}$ of layers which may be transmitted and a set $\mathcal{T} = \{t_1, \ldots, t_n\}$ corresponding to how much time remains before they expire, the transmission policy dictates which layer $l_i \in \mathcal{L}$ will be transmitted. In Section 2 we consider ways of finding the optimal transmission policy π^* that results in the lowest average distortion in the reconstructed signal. We develop a Markov chain analysis of this layered transmission system and show that the optimal policy π^* depends on many factors, such as the frame lifetime (a function of the one-way network and playout delays), the frame rate (time between frames), the erasure rate, and the relative importance of the layers. We perform the Markov analysis for the case that these factors are fixed and do not vary over time, and we use the resulting analysis to compute the average distortion incurred by a given policy. The optimal policy π^* is found by searching the distortions of all possible policies. A key result is that transmission decisions of π^* are time-invariant and thus do not change as the layers approach their playback times. For the transmission tradeoff mentioned above, this result means that if the best policy dictates O_L should be sent instead of N_H at some time t, and this attempt is erased, then O_L would be chosen again and retransmitted. It might seem logical to "give up" on O_L at some point and concentrate on the more important N_H. However, our results indicate that this is not true, and O_L should continue to be chosen and retransmitted in lieu of N_H until it either successfully reaches the receiver, or enough time passes that transmissions of O_L can no longer reach the receiver in time for playback.

Having found the optimal policy when network conditions are static, in Section 3 we consider methods of adapting the transmission protocol when the erasure rate and playback point (data lifetime) vary over time. We describe an algorithm for estimating the erasure rate which is based on the technique in the Real-time Transport Protocol (RTP) [26], and we also present a novel algorithm for estimating the data lifetime. We conclude Section 3 with a protocol for changing the transmission policy according to the current estimates of the data lifetime and channel erasure rate.

We evaluate our protocol through network simulation and present the results in Section 4. By comparing our protocol's performance to algorithms which do not use layering, do not adaptively estimate the lifetime, or do not change the transmission policy as the erasure rate changes, we quantify the importance of each of these techniques to the overall protocol performance. We find that layering the data and accurately estimating the data lifetime provide substantial performance improvements. We also find that only marginal improvements are had by adapting the transmission policy (*i.e.*, the decisions to favor older, less important layers over newer ones, and vice versa) as the erasure rate and lifetime changes.

Section 5 describes related work on areas that include soft ARQ, alternative methods for increasing reliability of time-limited data transmissions, and mathematical analysis of problems similar to this one. Section 6 describes areas for future work, and concluding remarks are given in Section 7.

2 Analysis

We now present a formal analysis for the layered transmission system described above and illustrated by Figure 1. The problem is to find the best transmission policy (*i.e.*, set of transmission decisions) for a given set of known parameters, such as the packet erasure probability and data lifetime. In order to solve this problem, we break it down into parts. First, we formalize the parameters of our layered transmission system and define a state space which captures its dynamics—what layers of what frames have already been transmitted, how long before each frame expires, *etc.* We then apply Markov chain analysis to find the steady-state behavior of the transmission system. From the steady-state analysis we obtain a distribution on the number of layers per frame that are successfully received before the frame expires. We then combine this information with a cost function (*e.g.*, a rate-distortion curve) to find the average cost associated with a specific transmission policy. Finally, we obtain the optimal policy by searching over all possible policies to find the one with the lowest cost.

2.1 Variable and State Space Definitions

Our transmission model consists of a multimedia signal transmitted from the source to a receiver over a binary erasure channel with feedback. We make the following assumptions and definitions with regard to this model:

- The channel erases a packet sent from sender to receiver with probability ε. Erasures are independent.
- We ignore transmission delay in both directions between sender and receiver. The sender has instantaneous feedback and knows if the packet just transmitted was erased and needs retransmission.
- The multimedia signal is segmented in time into frames that are generated periodically every T time units.
- Each frame is further encoded into a hierarchy of N layers.
- Unit time is the length of time it takes to transmit one message (one layer of one frame), and one second denotes a one time unit.
- $T \geq N$, so that there is at least one chance to transmit each layer of every frame.
- Each frame has a lifetime at the *sender* of L seconds; any messages sent more than L seconds after

Variable	Meaning
L	frame lifetime
T	period of frame production
N	number of layers per frame
K	maximum number of frames "alive"

Table 1: Summary of transmission model variables.

Variable	Meaning	Eqn
S_t	transmission state at time t	2
$\phi^{(t)}$	phase within a T-length cycle	3
$\mathbf{n}^{(t)}$	K-tuple of the transmission state of the currently live frames $n_i^{(t)}$	4
$n_i^{(t)}$	number of frame i's layers successfully sent by time t	–

Table 2: Summary of the state space variables' definitions and relevant equation numbers.

the frame is produced will arrive too late for playback at the receiver. We say that a frame produced at time t "expires" at the sender at time $t + L$. Because we have assumed there is no network delay, this lifetime is solely a function of delays at the receiver: specifically, L is the playback delay less any processing delays.

- $L > T$, so that there is at least some overlap in the lifetimes of consecutive frames. This leads to situations requiring a non-obvious decision between transmitting a less important message of a older frames and a more important message of a newer frame.
- The maximum number of frames "alive" at any time is K. A live frame is one that has already been generated but has not yet expired. K is further explained below.

Because all of the frames of the multimedia signal are not available to the sender at the start of the transmission (the signal's frames are produced periodically), and because each frame only has L seconds after it is produced to be sent to the receiver, there is a finite limit on the number of frames whose layers can be considered valid candidates for transmission. This maximum number of frames K alive at any given time is a function of how long they live (L) and how frequently they are produced (T), and is given by:

$$K = \left\lceil \frac{L}{T} \right\rceil . \tag{1}$$

Table 2.1 summarizes the definitions of the above variables.

After the first K frames have been produced, a new frame is produced and an old frame expires once every T seconds. Let ϕ be the phase (position) within a T-length cycle, so that $\phi \in \{0, 1, \ldots, T-1\}$, and let the cycle start at $\phi = 0$ when a new frame is produced. Note that if the lifetime L is not an exact multiple of T, the oldest frame will expire at phase $\phi = L - (K-1)T$, *before* the next new frame is produced. In this case there will be only $K - 1$ frames alive during the last $(KT - L)$ seconds of a cycle.

In deciding which message to transmit next at any given time t, the sender must consider not only which messages of the K current live frames have been transmitted, but how much time remains before each of these frames expires. However, the sender does *not* need to consider (and hence, remember) any information about the older expired frames in order to make its decision. Because these frames have expired, there is no point in sending any of their untransmitted messages, and thus there is no need to remember their specific expiration times. Also, although we may be able to infer the channel erasure rate through knowledge of how many layers of of these frames were successfully transmitted, we have assumed that we already know the erasure rate and hence this knowledge is not needed to make the current transmission decision.[1]

We can now define a state S_t that summarizes the information the sender needs to make a transmission choice at time t. Let S_t be defined as:

$$S_t = (\phi^{(t)}, \mathbf{n}^{(t)}), \tag{2}$$
$$\phi^{(t)} = t \bmod T, \tag{3}$$
$$\mathbf{n}^{(t)} = \left[n_1^{(t)}, n_2^{(t)}, \ldots, n_K^{(t)} \right], \tag{4}$$

where $n_i^{(t)}$ is the number of successfully transmitted layers of the ith-oldest live frame at time t (*i.e.*, frame 1 is the oldest, frame K is the newest). We omit the t superscript from ϕ, $\mathbf{n}$, and n_i when its context is clear. Because there are N layers, $0 \le n^{(t)} \le N$. These state space components are summarized in Table 2.1.

The phase ϕ tells us how much time is left before each frame expires. For example, at the beginning of a cycle ($\phi = 0$) the frame K is produced, and so we know it expires in L seconds. More generally, let ttl_i be frame i's "time-to-live," *i.e.*, how much time remains before it expires. It is calculated as:

$$\mathrm{ttl}_i = L - \phi - iT. \tag{5}$$

[1] Because the sender typically does not know the erasure rate, we consider ways of estimating the erasure rate in Section 3.3; however, our goal here is to try to find the optimal policy when the sender possesses all information relevant to making transmission decisions.

The K-tuple $\mathbf{n}$ tells us exactly what layers of the K frames have already been transmitted, and, conversely, which layers remain for each frame. At any time t, the $n_i^{(t)}$-most important layers of frame i have been transmitted, and so there are $N - n_i^{(t)}$ layers remaining. At the beginning of a cycle ($\phi = 0$) a new frame is produced, so $n_K = 0$. Also at this time, all of the frames "age" one position in the K-tuple $\mathbf{n}$. To see this, suppose that at a time t at the start of one cycle ($t \bmod T = 0$), we have a state

$$S_t = \left(0, \mathbf{n}^{(t)} = \left[n_1^{(t)}, n_2^{(t)}, \ldots, n_K^{(t)}\right]\right).$$

Now suppose that the next T transmission attempts are all erased, so that no frame gets any more messages across. For this case the next T states are independent of our transmission policy—regardless of which messages the policy dictated we tried to (re-)transmit, they were all erased—and our state evolves with time as:

$$\begin{aligned} S_{t+1} &= (1, \mathbf{n}^{(t)}) \\ S_{t+2} &= (2, \mathbf{n}^{(t)}), \\ &\vdots \\ S_{t+T-1} &= (T-1, \mathbf{n}^{(t)}). \end{aligned}$$

At time $t + T$, immediately following the Tth erasure, a new frame arrives and a new cycle begins. Because the oldest frame of the previous cycle has expired by this time, we no longer track its state. The new state at time $t + T$ is $S_{t+T} = (0, \widetilde{\mathbf{n}^{(t)}})$, where

$$\widetilde{\mathbf{n}^{(t)}} \equiv \left[n_2^{(t)}, n_3^{(t)}, \ldots, n_K^{(t)}, 0\right]. \tag{6}$$

The $\widetilde{\ }$ operator left shifts each frame's state one position to reflect how each frame ages one position per cycle as one frame expires and a new frame arrives. In this simple example, the values $n_i^{(t)}$ did not change (except for the position shifts) because all of the transmission attempts were failures. Next, we consider how to analyze the state evolution for the more general case when some transmissions succeed and some are erased.

2.2 Markov Chain Analysis

In this section we present an analysis of the process $\mathcal{S} = \{S_0, S_1, S_2, \ldots\}$, which illustrates how the state space evolves with time. We perform this analysis so we can find the steady-state behavior of $\mathcal{S}$; with this knowledge we can calculate the expected distortion of incurred by a particular policy π. The steady-state behavior of $\mathcal{S}$ depends on both the erasure rate of the channel, which determines the chance of a successful transmission, and our policy π, which dictates what layers of which frames should be transmitted, or retransmitted, at any given time. To illustrate this dependency we first examine how $\mathcal{S}$ can change in a single time step.

Consider the possible transitions from a state S_t to S_{t+1}. The transition of the phase component ϕ of the state is completely deterministic:

$$\phi^{(t+1)} = (\phi^{(t)} + 1) \bmod T. \tag{7}$$

As a result, we focus our attention on the transitions of the transmission state vector $\mathbf{n}$. There are K components n_i of $\mathbf{n}$, each of which can take on any of $N + 1$ values ($0 \le n_i \le N$), so the maximum number of possible values $\mathbf{n}$ may take on is $M = (N+1)^K$. However, there are only two values that $\mathbf{n}^{(t+1)}$ may take on for a given value of $\mathbf{n}^{(t)}$. To see this, first assume that at time t we are not at the end of a cycle: $\phi^{(t)} \neq T - 1$. The transmission policy π contains a rule for every state $S_t = (\phi^{(t)}, \mathbf{n}^{(t)})$ which dictates what frame's layer should next be transmitted, or retransmitted if a previous attempt has failed. If $\pi(S)$ is the frame that the policy dictates be chosen for a state S, then at time t the most important layer of frame $\pi(S_t)$ not yet successfully transmitted would be sent. This layer is $(n_{\pi(S_t)}^{(t)} + 1)$-most important layer, since the first $n_{\pi(S_t)}^{(t)}$ layers of frame $\pi(S_t)$ have already been transmitted. This transmission can either succeed or be erased. If it is erased then $\mathbf{n}$ does not change; if the transmission succeeds then $\mathbf{n}^{(t+1)}$ differs from $\mathbf{n}^{(t)}$ in only one component:

$$n_{\pi(S_t)}^{(t+1)} = n_{\pi(S_t)}^{(t)} + 1. \tag{8}$$

Because the probability of an erasure is ε, the one-step transition probability is:

$$P\left(\mathbf{n}^{(t+1)} \middle| \mathbf{n}^{(t)}\right) = \begin{cases} \varepsilon & \text{if } \mathbf{n}^{(t+1)} = \mathbf{n}^{(t)} \\ 1-\varepsilon & \text{if } n_{\pi(S_t)}^{(t+1)} = n_{\pi(S_t)}^{(t)} + 1, \\ & \quad n_j^{(t+1)} = n_j^{(t)},\ j \neq \pi(S_t) \\ 0 & \text{else.} \end{cases} \tag{9}$$

We encapsulate the one-step transition probabilities of all M-possible values of $\mathbf{n}^{(t)}$ in an $M \times M$ state transition matrix P_ϕ, where $\phi = t \bmod T$. Assume that we have a function f which maps each possible value of $\mathbf{n}$ to a unique index $i \in 1, \ldots, M$; for example, $f([1,0,0]) = 2$ and $f^{-1}(2) = [1,0,0]$. With this mapping function, the components of P_ϕ are defined as:

$$[P_\phi]_{i,j} = P\left(\mathbf{n}^{(t+1)} = f^{-1}(j) \middle| \mathbf{n}^{(t)} = f^{-1}(i)\right), \tag{10}$$

where the conditional probability can be found using Equation 9. Each row i of P_ϕ contains two non-zero elements: ε in column i, and 1-ε in column j, where j is determined by the policy π.

In our analysis so far we assumed that we were not at the end of a cycle at time t. However, if we are at the end of a cycle ($\phi^{(t)} = T - 1$) the state transition matrix given by Equation 10 is not quite correct. It fails to account for the arrival of a new frame and the aging of each frame of the previous cycle by one position. This is corrected by right-multiplying the matrix P_{T-1} by a matrix P_a, which left-shifts each state by one position. Letting $\tilde{\mathbf{n}}$ denote $\mathbf{n}$ shifted left by one (see Equation 6), the elements of P_a are defined as:

$$[P_a]_{i,j} = \begin{cases} 1 & \text{if } f^{-1}(j) = \widetilde{f^{-1}(i)} \\ 0 & \text{else.} \end{cases} \quad (11)$$

The new state transition matrix P'_{T-1} to describe transitions from $\mathbf{n}^{(t)}$ to $\mathbf{n}^{(t+1)}$ when $\phi^{(t)} = T - 1$ is simply the product $P_\phi P_a$.

We can now use the one-step state transition probabilities in order to find the steady-state behavior of $\mathcal{S} = \{S_0, S_1, S_2, \ldots\}$. Because erasures are independent, $\mathcal{S}$ is a discrete-time Markov chain. In other words, the probability of being at some state s_{t+t_1} in the future does not depend on any past knowledge of the process s_{t-t_2} if we know the current state s_t. The only factors that affect transitions from s_t to s_{t+t_1} are the transmission policy and the erasure rate. Their influence can be summarized as follows: εaffects the chance that $\mathbf{n}$ will change, and π determine how it changes.

Because $\mathbf{n}$ includes phase information, $\mathcal{S}$ is also cyclostationary with period T. This is because it is not possible to go from a state $\mathbf{n}$ at time t to the same state $\mathbf{n}$ in less than T steps. The process $\mathcal{S}_\phi = \{S_\phi, S_{\phi+T}, S_{\phi+2T}, \ldots\}$, $\phi \in \{0, \ldots, T-1\}$, is a stationary process, however. Its $M \times M$ state transition matrix $P^{(\phi)}$ is derived from Equations 10 and 11:

$$P^{(\phi)} = P_\phi P_{\phi+1} \cdots P_{T-1} P_a P_0 P_1 \cdots P_{\phi-1}. \quad (12)$$

A stationary distribution $\{S_\phi, S_{\phi+T}, S_{\phi+2T}, \ldots\}$ can be found analyzing the matrix of Equation 12. Let η be the stationary distribution when the oldest live frame expires, *i.e.*, $\phi = L - (K-1)T$. The probability ν_i of transmitting the i most important layers of a frame by the time it expires is calculated by summing out the possible states of the other $K - 1$ frames:

$$\nu_i = \sum_{n_2=0}^{N} \sum_{n_3=0}^{N} \cdots \sum_{n_K=0}^{N} \eta_{f(i,n_2,n_3,\ldots,n_K)} \quad (13)$$

Note that although there are $M = (N+1)^K$ possible values of the K-tuple $\mathbf{n}$, the number of *feasible* values may actually be lower. Which states are unfeasible will depend on the policy π. For example, if π dictates that the most important layer of the oldest live frame is always chosen, then it is not possible to have $n_3 \neq 0$ if $n_2 < N$, since transmission of any message of the third oldest frame would not commence until all messages from the second oldest frame had been sent. The transmission policy will have no rule associated with these states. To get around this problem, we can remove each unfeasible state $\mathbf{n}_u$ from the analysis, and thus have $M' < M$ states. Alternatively, we can still keep M states and assign a probability of 1 to the $[f(\mathbf{n}_u), f(\mathbf{n}_u)]$ entries of each P_ϕ matrix defined by Equation 9. The stationary probability of these states $\nu_{f(\mathbf{n}_u)}$ will then 0 because they are null-recurrent, and thus their presence will change the result of Equation 13.

Finally, given a rate-distortion function $D(R)$ such that $D(i)$ is the distortion incurred in reconstructing a frame from its i highest priority layers, $0 \leq i \leq N$, we can compute the average distortion per frame for a transmission policy π by

$$D_\pi = \sum_{i=0}^{N} \nu_i D(i). \quad (14)$$

Equation 14 can be interpreted as a weighted sum: the distortion $D(i)$ of reconstructing a frame from i layers is weighted by the probability ν_i that only i layers of the frame are successfully sent in time for playback. One constraint on these weights is that the expected number of layers transmitted, N_{avg}, can not exceed either the channel capacity C or the raw transmission rate R:

$$N_{\text{avg}} = \sum_{i=1}^{N} \nu_i i \leq \min(C, R) = \min\left((1-\varepsilon)T, N\right), \quad (15)$$

where $C = (1 - \varepsilon)T$ is a basic information theoretic result on the capacity of a binary erasure channel [9]. Even when the rate R is less than the channel capacity C, the bound of Equation 15 may still be unachievable because the data is time-constrained. Thus although *on average* there may be enough channel capacity to send the entire multimedia signal, in the short term there may be a sequence of many consecutive erasures so that data expires before it is successfully transmitted. The choice of policy can affect both N_{avg} and the distribution of the rate-distortion weights (ν_i). For example, a policy always favoring the most important message of the oldest live frame will maximize the average number of expected layers N_{avg} and the chance of sending all layers across (ν_N). Comparatively, with a policy that sends any messages belonging to the most important layer ahead of all others, the chance of sending at least one message in a frame (ν_1) is maximized by reducing the chances of both getting none (ν_0) and getting all of them (ν_N). Which policy is better will depend not only how they can affect ν (which is also dependent on T and L), but also on the shape of the rate distortion curve $D(R)$.

2.3 Analytical Results

Given D_π and our Markov chain model, we can formulate an optimization that computes the best transmission policy:

$$\pi^* = \arg\min_{\pi \in \Pi} D_\pi, \tag{16}$$

where Π is the set of all possible policies.

In this section we apply our Markov chain analysis to determine π^* for a given set of static network conditions. Because our analysis depends on the erasure rate ε, the relative importance of different priority layers (determined by a rate-distortion function $D(R)$), the lifetime L, and inter-frame period T, the optimal policy π^* will depend on these factors as well. The steps for finding π^* can be summarized as follows: first, fix the four aforementioned parameters; next, perform a calculation of every possible policy's distortion; and finally, find the policy which produced the minimum distortion.

In our analysis we focus on the case that there are two layers ($N = 2$) and that T and L are such that there is a maximum of two frames alive at any time ($K = 2$). The reasons for this choice are two-fold: first, it simplifies the discussion and interpretation of our results; and second, it simplifies the computational complexity of our analysis, since the size of our state space is exponential in N and K. With $N = 2$ and $K = 2$ there are at most 4 messages to choose from at any time: the two layers X_1^i and X_2^i of an older frame i, and the two layers X_1^{i+1} and X_2^{i+1} of the next, newer frame $i + 1$. In order to emphasize the priority and age of these 4 messages, we introduce the following variable names that will be used throughout this discussion:

- $O_H = X_1^i$: the high priority layer of the older frame
- $O_L = X_2^i$: the low priority layer of the older frame
- $N_H = X_1^{i+1}$: the high priority layer of the newer frame
- $N_L = X_2^{i+1}$: the low priority layer of the newer frame

In the introduction we explained that non-obvious transmission decisions arise when we must choose between older, less important messages and newer, more important messages. For the two-layer, two-frame case, this situation arises in only one of the 9 possible values of $\mathbf{n}$: $\mathbf{n} = [1, 0]$. This is the case that O_H was successfully transmitted, and so either O_L or N_H must be chosen next. Each policy that we consider in this section consists of a distinct choice of O_L or N_H for each phase in the cycle such that the older frame has not yet expired. In other words, $0 \le \phi < L - T$, so there are 2^{L-T} possible policies to consider.

We first present results when illustrating the average distortion of various policies as a function of the erasure rate ε, when other parameters T, L, and the rate-distortion function $D(R)$ are all held fixed. A non-intuitive result is that the optimal policy π^* always belongs to a subset of two of the possible policies, and these two do not change their message choice for a state $\mathbf{n}$ as the frames get closer to their expiration times, *i.e.*, as ϕ changes. There is a threshold value of ε at which π^* switches from one of these policies to the other. Another key result is that the best policy on one side of this threshold was also the worst policy on the other side. Next, we illustrate how the shape of $D(R)$ can affect the value of this threshold; this tells us the best policy as a function of both $D(R)$ and ε. Finally, we examine how changing the values of L and T can also affect the best policy.

2.3.1 Effect of the erasure rate

The erasure rate ε affects the probability of successfully transmitting a message. In this section we examine how it affects the choice of π^*. For fixed values of L and T, the stationary distribution ν of a particular policy depends only on the erasure rate ε. However, the *optimal* policy depends not only ν, and hence ε, but also on the rate-distortion function $D(R)$.

D_π, the distortion associated with a policy given by Equation 14, is a linear function of $D(R)$. As a result, translating and/or positively scaling $D(R)$ does not change which policy is optimal (*i.e.*, has minimum distortion), since by Equation 14 all policies' average distortions will be equally scaled and translated. Therefore, we can find and apply a scaling $a > 0$ and translation b to any N-layer rate-distortion function to normalize it so that the resulting $D'(R) = aD(R) + b$ satisfies $D'(0) = 1$ and $D'(N) = 0$. This new distortion function can be completely characterized by the $N - 1$ values of $d_i = D'(i)$, $0 < i < N$, subject to the convexity constraints $(d_i - d_{i+1}) \ge (d_{i+1} - d_{i+2})$, $0 \le i < N - 1$. For the two-layer case, this means that the form of any rate-distortion function can be completely summarized by d_1. We will refer to d_1 as the "layer gap" for the two-layer case because it measures the gap in importance between the two layers. The convexity constraint $0 \le d_1 \le 0.5$ is necessary so that the high priority layer actually is more important than the low priority layer (or, at the minimum, equally important). If d_1 is close to 0.5 then both layers are of near equal importance; if d_1 is near 0 the high priority layer has much more benefit (distortion reduction) than the low priority layer.

To illustrate the effect of the erasure rate on the distortion of different policies, we fix the layer gap at $d_1 = 0.1$, the inter-frame period at $T = 4$, and the frame lifetime at $L = 8$. In this case the overlap in consecutive frames' lifetimes lasts $L - T = 4$ seconds, and so there are a total of $2^4 = 16$ possible transmission policies. We found that of all possible policies, π^* is always one of the two "phase-invariant" policies, which either always choose O_L or always choose N_H throughout the entire 4-second overlap window, regardless of the phase. In other words, if the chosen message (O_L or N_H) is erased, these two policies always retransmit it until it succeeds or it expires, whichever comes first. The optimality of the phase-invariant policies can be seen in Figure 2, which displays the average distortion as a function of ε incurred by these two transmission policies and a third, phase-varying "hybrid" policy. The hybrid policy shown favors O_L for the first two transmissions in a cycle ($\phi = \{0, 1\}$) and then switches to favor N_H for the last two transmissions ($\phi = \{2, 3\}$). The scale of the y-axis can be interpreted as follows: assuming that the distortion function is mean squared error, a one-decade decrease in distortion corresponds to a 10 dB increase in signal-to-noise ratio.

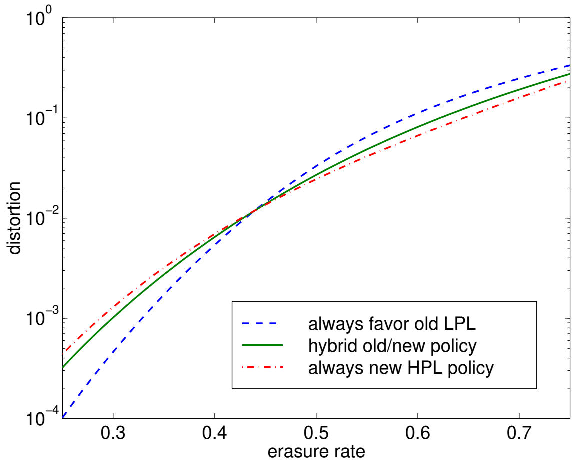


Figure 2: Distortion versus erasure rate for 3 different decision policies, for $T = 4$, $L = 8$, and $d = 0.1$.

Figure 2 shows that for low values of ε, the best policy always favors the O_L, for high values of ε, the best policy always favors the N_H; and there is a value of ε where the two policies have equal distortion (≈ 0.44 here). When the erasure rate is low, sending O_L instead of N_H is better because O_L will expire sooner, and although this choice reduces the number of possible transmission attempts for the more important N_H, it is unlikely to need them all. However, when the erasure increases it increases the average number of attempts needed to get N_H to the receiver, and hence sending N_H before O_L becomes more beneficial, even though O_L may expire before the sender succeeds with N_H.

An equally important finding is that for values of ε in which the O_L phase-invariant policy is optimal, the N_H phase-invariant policy is not only suboptimal, but it is also the *worst* possible policy. The converse holds true as well. In general, the two optimal policy distortion curves form the upper and lower boundaries of a performance envelope between which all other policies' performance curves must lie. Note that although we have only shown results from 3 of the 16 possible policies, we did find that the distortion curves of other 13 do all lie between the envelope formed by the curves of the two phase-invariant policies. Also, although we have not proven this optimal/worst nature of the two phase-invariant policies (it was identified through exhaustive search of all possible policies), we found that this property held true for all other combinations of d_1, T, and L that we examined.

2.3.2 Effect of the layer gap

In the preceding section we found that when all parameters except the erasure rate were fixed, the optimal policy could be characterized by the threshold value of the erasure rate: if the erasure rate is below this threshold, π^* always chooses O_L; if above, it chooses N_H. In this section we examine how the layer gap affects the value of this threshold. Figure 3 illustrates the location of this threshold (shown on the x-axis) as the layer gap is varied between 0 and 0.5 (y-axis), for an inter-frame period of 3 and a frame lifetime of 5 ($T = 3$, $L = 5$). Area A to the left of the curve indicates when the phase-invariant policy favoring O_L is optimal; area B to the curve's right indicates that the N_H phase-invariant policy is optimal. The curve was obtained by analytically solving for the average distortions D_π of the two policies as a function of ε and d_1, setting them equal and solving for ε as d_1 was numerically varied. We verified the correctness of the curve by sampling the ε-d_1 plane, finding π^* through exhaustive search, and confirming that the O_L-favoring policy was indeed optimal for all points lying in area A, and likewise for area B.

Figure 3 shows that as the layers become more equal in importance (d_1 increases), the erasure rate threshold moves to the right. This makes intuitive sense: if there is a small disparity between the layers' importance, then O_L is almost as beneficial as N_H, and thus unless the erasure rate is high it is better to send O_L because it expires sooner. Conversely, if the high priority layer is much more important, then the erasure rate does not have to be as high before it makes sense to start favoring N_H over O_L; this increases the chance of successfully transmitting this more important message.

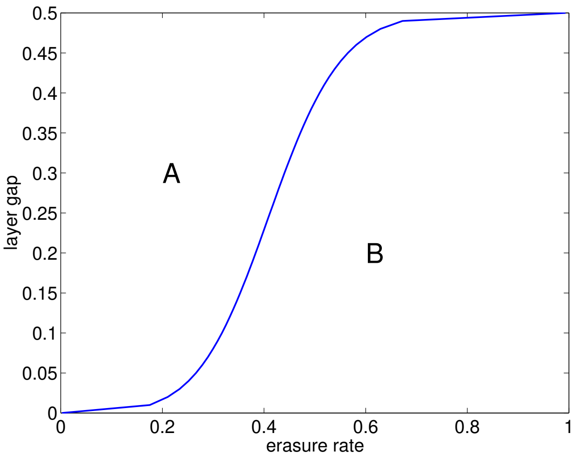


Figure 3: Optimal decision policy as a function of ε and d, for $T = 3$ and $L = 5$.

2.3.3 Effect of the inter-frame period

We also examined the effect of the inter-frame period on the erasure rate threshold, and found that increasing T tends to increase the threshold. Figure 4 illustrates this effect; it shows threshold curves (as described in the preceding section) for each value of T between 3 to 6; L is set to $T+2$ for all cases. Increasing T causes the curve's knee to move further to the right; because increases in T provide more transmission opportunities per frame, the erasure rate must also increase before sending N_H over O_L becomes more beneficial. We also see that as T increases, the spacing between the curves becomes smaller, but the slope of the knees remains relatively fixed. This indicates that changing T does not change the amount of influence the layer gap has on the erasure rate threshold value.

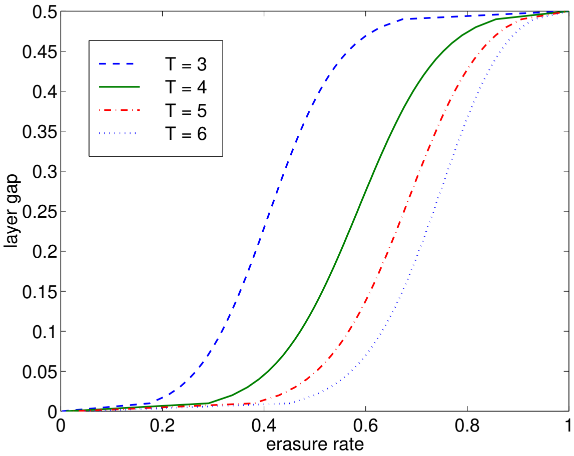


Figure 4: Effect of T on the optimal decision policy.

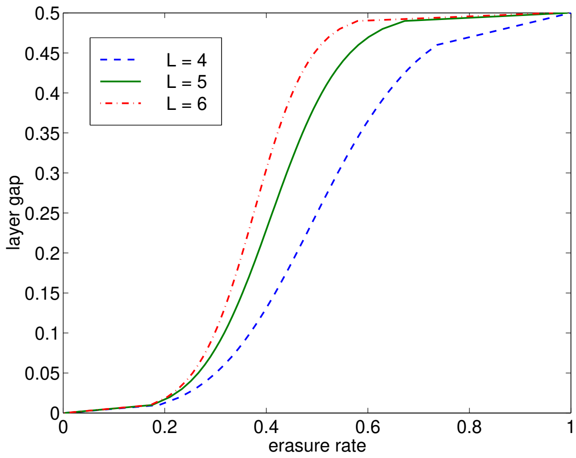


Figure 5: Effect of L on the optimal decision policy. $T = 3$ for all curves.

2.3.4 Effect of the frame lifetime

Finally, we examined how the frame lifetime L affects the location of the erasure rate threshold, and hence the optimal policy pi^*. In general, we found that the lifetime has the following effects:

- Increasing the lifetime moves the threshold curve to the *left*, so that it becomes more beneficial to send the N_H over the O_L at even lower erasure rates. We hypothesize that this is because if the sender chooses to send the more important N_H before the older O_L, increasing the lifetime increases the chance that N_H is successfully transmitted *before* O_L expires, thereby increasing the chance that O_L can be sent as well.

- Increasing the lifetime decreases the impact of the layer gap on the choice of π^*. This is reflected by a steeper threshold curve.

- The magnitude of the difference in average distortion between the best and worst policies increases as the lifetime increases. This is because a longer lifetime results in a longer overlap between consecutive frames' lifetimes, and hence the transmission policy can influence a the sender's behavior over a larger fraction of the T second transmission cycle, for better or worse.

The first two properties are illustrated Figures 5 and 6, which show the threshold curves for various L-values when T equals 3 and 4, respectively. In each figure L was varied between $T+1$ and $2T$. The latter property was confirmed by examining $D_\pi(\varepsilon)$ graphs (like Figure 2) for various lifetimes.

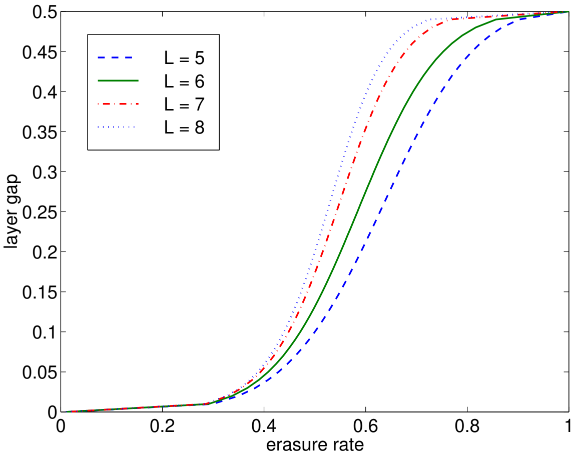


Figure 6: Effect of L on the optimal decision policy. $T = 4$ for all curves.

3 Adapting to Varying Network Conditions

Having shown that an optimal transmission policy can be found for a static set of model parameters, we now address the problem of casting our optimization onto a real streaming multimedia application. Typical streaming media applications do not change the frame length over time, so we continue to assume T is fixed and known to the sender. Similarly, the encoding process does not typically vary with time with a typical layered multimedia stream. As a result, the layer gap also remains fixed, and we presume it is known to the sender.[2] On the other hand, the lifetime L of the frames at source depends on the delay in the network and at the receiver (playback buffering, processing, *etc.*), and these conditions change with time. Likewise, in the best-effort Internet, the erasure rate ε varies with time. Consequently, our transmission policy must adapt to changes in L and ε.

3.1 Generalizing the Static Case

Our goal is to leverage our results from the analysis of Section 2 and develop a more general transmission protocol that adapts the transmission policy to changes in L and ε. For the two-layer ($N = 2$), two-frame overlap ($K = 2$) case, our results indicated that the optimal policy always gives preference to transmitting the less important message of the older frame (O_L) over the more important message of the newer frame (N_H) if ε is less than a threshold value, and vice versa when ε exceeds this value. The value of this threshold depends on the values of d_1, T, and L. Because we assume that d_1 and T are fixed throughout the transmission process, the only unknown factor influencing the threshold is the lifetime L. Let this threshold be $th(L)$. Assume that $th(L)$ is pre-computed for various values of L, and that the sender estimates the current values of the lifetime and erasure rate; we denote these estimates $\hat{L}$ and $\hat{\varepsilon}$, respectively. Using these estimates, the sender can adaptively determine its transmission policy by comparing $\hat{\varepsilon}$ to $th(\hat{L})$. Sections 3.2 and 3.3 describe specific methods for calculating $\hat{\varepsilon}$ and $\hat{L}$. These methods we describe are by no means the only techniques for estimating these quantities.

Because we assumed zero network delay in either direction in the analysis of Section 2, we evaluate our protocol's performance in Section 4 under the same assumption. We do so because we wish to judge how well we can adapt the transmission policy when we are given a pre-computed table of the optimal policy under various sets of fixed network conditions. And from Section 4 we know the optimal policy for static conditions when there is instantaneous feedback; we do not know it for the case of delayed feedback in part because the delay causes an explosion in the state space of the system.[3] Thus we restrict our investigation to the zero-network-delay case so as to accurately compute the optimal policy for a static L and ε. A result of this is that the lifetime L is solely a function of delays at the receiver; for example, L could be the playback delay (defined as the difference between the time a frame is played out and when it first arrived at the source transmission buffer) less any processing delays at the receiver. We note, however, that our lifetime estimation technique described below in Section 3.2 works regardless of this zero-network-delay assumption. To measure its effectiveness without varying the zero network delay, we varying the size of the receiver's playback buffer; this has the same effect on L as variations in

[2] In reality, the true relative importance of the layers may change with time since multimedia signals are not stationary, especially over large time scales. Furthermore, since it is difficult to characterize perceptual distortion, it is difficult to know which value of d_1 which accurately corresponds to subjective quality. With a real signal and today's cost measures, d_1 is probably at best an estimate of the average relative importance of the layers.

[3] For example, with instantaneous feedback the K-tuple $\mathbf{n}$ of how many layers of each frame had been transmitted is enough to completely characterize the message transmission state, because we can correctly assume that the first i successful messages of a frame corresponded to the i most important layers. Feedback delay means that lower priority messages may be successfully transmitted before high priority messages of the *same* frame, and so the overall system state must describe the state of every message of every live frame. These descriptions must indicate whether each message has been successfully transmitted, is awaiting transmission, or whether it is in transit. Furthermore, if it is in transit we must keep track of whether it was successful or erased, and how much time is left before that information is sent back to the source. Such a state is needed for every possible combination of every possible component of each live layer. The result is an extremely large exponential increase in the total size of the state space necessary to fully describe the transmission process.

network delays would.

The question still remaining is how to estimate L and ε in order to compute the transmission policy. The next two sections contain our approaches to these estimation problems.

3.2 Estimating the Lifetime

The lifetime L is defined as the length of time after the arrival of a frame at the input buffer during which a transmission of that frame will arrive at the receiver in time for playback. Our purpose in estimating L is two-fold:

1. The erasure rate threshold value that determines the optimal transmission policy depends on L; knowing L lets us accurately choose the best policy.
2. An accurate estimate $\hat{L}$ not only prevents the source from transmitting layers of frames that would arrive too late to be useful (thus wasting bandwidth and possibly preventing transmission of other layers), it also maximizes retransmission opportunities by preventing the source from "giving up" too soon on data the source incorrectly believes will not reach the receiver in time.

Suppose that packet n contains a layer of frame i which arrives at the source buffer at time $t_s[i]$, is transmitted at time $t_x[n]$, arrives at the receiver at time $t_r[n]$, and is scheduled for playback at time $t_p[i]$. The source lifetime $L[n]$ of the data can be written as the following:

$$\begin{aligned} L[n] &= t_p[i] - \delta_{\text{proc}}[i] - (t_r[n] - t_x[n]) - t_s[i] && (17) \\ &= (t_p[i] - t_s[i]) - \delta_{\text{proc}}[i] - (t_r[n] - t_x[n]) && (18) \\ &= \delta_{\text{play}}[i] - \delta_{\text{proc}}[i] - \delta_{\text{net}}[n], && (19) \end{aligned}$$

where $\delta_{\text{proc}}[i]$ is the receiver's processing delay for the ith frame, $\delta_{\text{net}}[n]$ is the one-way network delay from source to receiver experienced by packet n, and $\delta_{\text{play}}[i]$ is the delay in playback of the ith frame, defined as the absolute time difference between the playback of frame i and its arrival at the source's transmission buffer. When there are no network or processing delays, δ_{play} is simply the amount of playback buffering done by the receiver.

One way to estimate $L[n]$ is to estimate each of the delay components of Equation 19 individually. However, computing $\delta_{\text{play}}[i]$ and $\delta_{\text{net}}[n]$ at the source requires either synchronization of the source's and receiver's clocks or knowledge of the offset between them. Fortunately, we can avoid both of these requirements. When the source transmits a layer of frame i in packet n, it time stamps the packet with its estimate $\hat{t}_{\text{tl}}[n]$ of that frame's time-to-live at the source:

$$\hat{t}_{\text{tl}}[n] = \left(t_s[i] + \hat{L}[n]\right) - t_x[n], \qquad (20)$$

where $\hat{L}[n]$ is the current estimate of the source's lifetime. The first term in the above equation is the latest possible time the source believes it can transmit a layer of frame i so that it arrives in time for playback. The actual time at which the packet is transmitted is subtracted off from this sum to result in an estimate of the remaining time-to-live for that frame. Besides stamping packet n with its estimate of the layer's time-to-live, the source also stores the current lifetime estimate $\hat{L}[n]$ used to compute this estimate. At the receiver, the latest time the packet could arrive for playback is $t_p[i] - \delta_{\text{proc}}[i]$; hence the true remaining time-to-live $t_{\text{tl}}[n]$ of packet n when it arrives at the receiver $t_r[n]$ is:

$$t_{\text{tl}}[n] = t_p[i] - \delta_{\text{proc}}[i] - t_r[n]. \qquad (21)$$

Upon receiving packet n, the receiver examines the header and determines which layer of which frame the packet contains, the frame's scheduled playback time, and the current processing delay. If the receiver does not want to continuously estimate the processing delay (which may vary with the system load), it could set this delay to a constant or 0 to ignore it. After determining the above information, the receiver calculates the frame's actual time to live using Equation 21, and computes a residual error:

$$e[n] = t_{\text{tl}}[n] - \hat{t}_{\text{tl}}[n]. \qquad (22)$$

The receiver sends the residual back to the sender in its ACK for packet n, and upon receiving this ACK, the sender calculates the actual lifetime:

$$\begin{aligned} L[n] &= \hat{L}[n] + e[n] && (23) \\ &= \hat{L}[n] + (t_p[i] - \delta_{\text{proc}}[i] - t_r[n]) \\ &\quad -(t_s[i] + \hat{L}[n] - t_x[n]) && (24) \\ &= t_p[i] - \delta_{\text{proc}}[i] - (t_r[n] - t_x[n]) - t_s[i] && (25) \end{aligned}$$

Equation 25 agrees exactly with the definition of $L[n]$ given by Equation 17. Because the source and receiver exchange only *relative* times (differential as opposed to absolute times), clock synchronization is unnecessary. Receiver computations may be reduced by including the absolute times $(t_p[i] - \delta_{\text{proc}}[i])$ and $t_r[n]$ in the ACK instead of $e[n]$, and letting the source calculate $t_{\text{tl}}[n]$ and $e[n]$ itself.

Because the source does not receive the correction $e[n]$ for the estimate $\hat{L}[n]$ until at least one round trip time after packet n was initially sent, it may have sent other packets in the interim. Suppose packet n's ACK is received by the source before packet m is sent, but after packet $m-1$ is sent ($m > n$). We could simply update the lifetime estimate to immediately account for the measurement error, *i.e.*

$$\hat{L}[m] = L[n] = \hat{L}[n] + e[n]. \qquad (26)$$

However, if any of the delay components that define the lifetime (*cf.* Equation 19) vary rapidly (*i.e.*, faster than one round trip time), then the actual lifetime $L[n]$ of packet n may not be an accurate estimate for the lifetime $\hat{L}[m]$ of packet m. This could cause undesired behavior, like incorrect switching of the transmission policy in response to short term fluctuations in the network delay. To avoid this and to capture underlying long-term characteristics of the delays, we smooth the lifetime estimate as follows:

$$\begin{aligned} \Delta[m] &= L[n] - \hat{L}[m-1] && (27) \\ \hat{L}[m] &= \hat{L}[m-1] + \beta\Delta[m] && (28) \\ &= (1-\beta)\hat{L}[m-1] + \beta L[n]. && (29) \end{aligned}$$

In this simple IIR filtering process, β determines the amount of smoothing performed: lower values result in more smoothing and a slower reaction to changes in L, and vice versa. The source uses the value of $\hat{L}$ defined by Equation 29 in order to estimate a layer's time-to-live $\hat{t}_{\text{tl}}$ via Equation 20. The source must also use its lifetime estimate to determine the transmission policy and decide the expiration times of frames in the input buffer. However, because the estimate $\hat{L}$ may lag behind the actual lifetime L due to feedback delay and the IIR smoothing process, we do not base these decisions directly on $\hat{L}$, but instead on a value L_{dec} on $\hat{L}$ that includes a safety margin. Much as TCP sets its retransmission timers not to the actual round trip time estimate but to a larger value dependent on the variance in the estimate, we compute L_{dec} in an analagous fashion:

$$L_{\text{dec}}[m] = \mu\hat{L}[m] - \phi\sigma_{\hat{L}}[m], \qquad (30)$$

where μ and ϕ are design parameters and $\sigma_{\hat{L}}$ is a measure of the deviation in the sequence of $\hat{L}[n]$ values. We calculate $\sigma_{\hat{L}}$ as in TCP:

$$\sigma_{\hat{L}}[m] = \sigma_{\hat{L}}[m-1] + \beta\left(|\Delta[m]| - \sigma_{\hat{L}}[m-1]\right). \qquad (31)$$

After the source computes $L_{\text{dec}}[m]$, it updates the transmission policy and re-determines the expiration times of any layers still in its input buffer. To update the transmission policy, the source calculates a new erasure rate threshold as $th\left(L_{\text{dec}}[m]\right)$. The source computes the new expiration time of frame i as $t_s[i] + L_{\text{dec}}[m]$. Because we *subtract* off a fraction of the deviation from $\hat{L}[m]$ in Equation 30, L_{dec} is more likely to be smaller than L than than it is larger, especially when the L varies rapidly. This in turn increases the chance that layers of an old frame that is close to its expiration time are treated as already expired, even though they could still reach the receiver in time. However, we chose to subtract the deviation in Equation 30 because it reduces the chance that we overestimate the lifetime and send useless, expired data.

The source updates $\hat{L}[m]$, L_{dec}, and the transmission policy whenever it receives an ACK for a packet n. If n's ACK is lost then no update is performed, because the source cannot accurately correct its estimates $\hat{t}_{\text{tl}}[n]$ and $\hat{L}[n]$.

3.3 Estimating the Erasure Rate

Since the best transmission policy depends on the packet erasure rate, we must accurately estimate this parameter. In order to do this, we employ a measurement protocol similar to the window-based measurement technique specified by the Real-time Transport Protocol (RTP) [26]. Each packet is time-stamped with a sequentially increasing packet number. Within a fixed window of packets, the receiver records the number of packets received. At the end of each interval defined by the window size, the receiver computes how many packets were expected (the difference between the sequence numbers of the last successful packet and the first packet of in the window), and how many it actually received. The receiver then calculates the loss rate over the interval, ε_{win}, and updates the overall loss rate estimate $\hat{\varepsilon}$ via an IIR low-pass filter:

$$\hat{\varepsilon}_{\text{new}} = (1-\alpha)\hat{\varepsilon}_{\text{old}} + \alpha\varepsilon_{\text{int}}. \qquad (32)$$

The level of smoothing caused by the filter depends on the parameter α: higher values result in less smoothing and a faster response to changes in ε, but they also produce a noisier estimation process than higher levels of smoothing do.

One problem with this window-based scheme is if there is a loss burst that starts near the end of an estimation interval, $\hat{\varepsilon}$ will not be updated until the first success after the burst reaches the receiver. A solution is to add a timer at the receiver. The receiver sets the timer for the expected end of a measurement interval and calculates the number of packets that could have arrived. When the timer expires it updates $\hat{\varepsilon}$ accordingly. Besides adding computational complexity to the receiver, this scheme can also miscount losses if the network delay changes in the middle of a measurement interval. Another alternative is to use overlapping windows, which results in more frequent measurement updates at the expense of increased receiver complexity.

The receiver informs the source of the erasure rate by putting the current value of $\hat{\varepsilon}$ in every ACK. This adds robustness to ACK loss. Alternatively, the source can compute the loss rate in an analogous based on ACKs. To prevent overestimated loss rates when ACKs are lost, the receiver could send a bit vector indicating the loss/success status of the most recently received packets.

4 Performance Evaluation

To evaluate the performance of our adaptive protocol, we simulated transmission of a multimedia signal over the system of Figure 1. Before testing our full-fledged protocol, we first focused on two of its components: the lifetime and erasure rate estimation techniques. We evaluated these techniques and tuned their parameters by measuring their response to step changes. We then simulated the complete protocol and compared its performance to other protocols. To evaluate the efficacy of our lifetime estimation technique, we compare our adaptive protocol to various fixed-lifetime transmission protocols. To evaluate the benefits of a layered encoding, we compared our protocol to non-layered transmission schemes. And finally, to evaluate the importance of adapting the transmission policy, we compared our protocol with others that adapt to lifetime changes but which keep the transmission policy static.

We simulated changes in the lifetime L and erasure rate ε by randomly changing their values at exponentially distributed intervals of rate λ_L and λ_ε, respectively. At the end of an interval we chose a new value of L from a uniform distribution $[\bar{L} - \sigma_L, \bar{L} + \sigma_L]$. We varied the erasure rate in a similar way, except that we set the erasure rate to $\max(0, \min(e, 1))$, where e is uniformly distributed in the interval $[\bar{\varepsilon} - \sigma_\varepsilon, \bar{\varepsilon} + \sigma_\varepsilon]$, in order to restrict it to the $[0, 1]$ interval. Thus $\bar{\varepsilon}$ is the median value of the erasure rate.

Figure 7 illustrates the adaptivity of the control algorithm that governs the evolution of L_{dec}. The x-axis denotes time in terms of transmission slots, while they y-axis indicates the value of both the the actual lifetime L and protocol lifetime L_{dec}. When L changes, L_{dec} converges to the new value in about 10 transmission slots. For multimedia signal with 20 ms frames and $T = 4$ transmission opportunities/frame, this would corresponds to a 50 ms convergence time. Figure 7 also shows marks that indicate when packets are transmitted, and if they are successful or erased. L_{dec} does not change if a packet is erased.

Figure 8 shows the adaptation of the estimated erasure rate $\hat{\varepsilon}$ to a step change in the true value of ε. A comparison of Figure 8 to Figure 7 reveals that $\hat{\varepsilon}$ is a much noisier estimate than $\hat{L}$. Unlike the lifetime, the erasure rate cannot be measured directly, and instead must be estimated from packet loss. Figure 8 shows that for that while $\hat{\varepsilon}$ does correctly respond to the change in ε, its accuracy is only about ± 0.05. The accuracy of the estimate and its quickness to respond to change can be traded off via choice of α and the size of the loss window.

In choosing values for the adaptation parameters $(\alpha, \beta, \phi, \mu)$ to use in our adaptive protocol, we tried to achieve a balance between speed of response and accuracy. Table 3 lists the parameter values we chose for our simulations. We ran these simulations for a range of median erasure rates ranging from 0.05 to 0.95. We compared our layered adaptive protocol to three other types of transmission schemes:

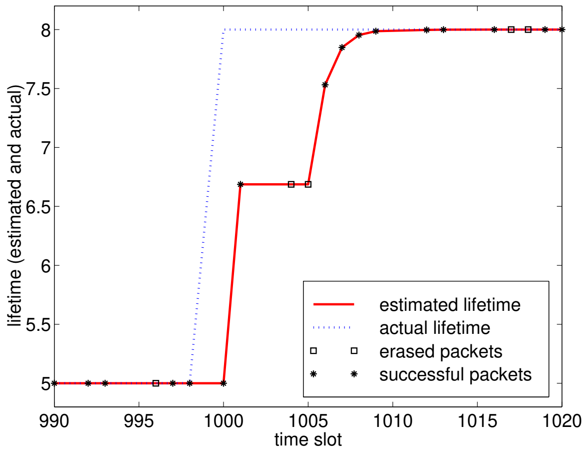


Figure 7: Lifetime adaptation to step response for adaptation parameters $\beta = 0.75$, $\phi = 1$, and $\mu = 1$. The frame period is $T = 4$ and the erasure rate $\varepsilon = 0.45$.

- Layered transmissions with a fixed transmission policy which always favored older messages over newer messages, and which used fixed, non-adaptive values for L_{dec}. (Results are shown in Figure 9).

- Non-layered transmissions with lifetime adaptation. These policies always sent the oldest live frame. We simulated two different policies: one with the same frame size T and one with the frame size half as small. With the former, messages are twice as large as the two-layered case, and with the latter they are equal sized. (Figure 10)

- A layered transmission with a fixed phase-invariant transmission policy (always favoring the older O_L over the newer N_H, or vice versa) but with full lifetime adaptation. (Figure 11)

We used a positive acknowledge (ACK) feedback scheme for all of the protocols. Losses were detected by the source via gaps in the sequence of ACKs. In addition, protocols adaptively estimating the lifetime and/or the erasure rate included time-to-live corrections and/or the current erasure rate estimate in the ACKs.

Figure 9 illustrates the benefits of adaptively estimating the data lifetime. The adaptive policy performs better at all erasure rates than the three fixed policies that do not adapt to changes in L. At low erasure rates, the

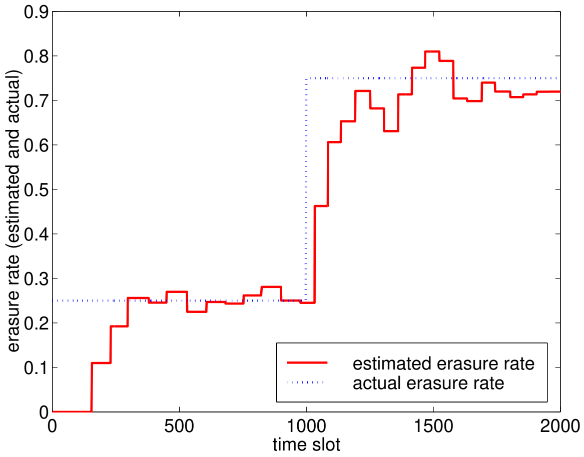


Figure 8: Erasure rate adaptation to step response for $\alpha = 0.125$ and a loss window of 50 packets. The frame period is $T = 4$ and the frame lifetime is set to $L = 6$.

fixed policy with $\hat{L} = 4$ is worst because it underestimates the lifetime and hence prevents retransmissions of messages because it thinks they have expired, even if they actually have not. At high erasure rates more retransmissions are necessary, and so the worst fixed policy is $\hat{L} = 8$ because it overestimates the data lifetime and transmits expired messages.

The advantages of layering the data are illustrated by Figure 10. The adaptive scheme is demonstrably better than the non-layered schemes. For the non-layered schemes, Figure 10 also shows there is an advantage to halving the frame size in order to send smaller packets more frequently. When the frame length is halved packets are sent twice as often, but their lifetime is unchanged. As a result the short-term loss rate observed during a frame's lifetime has a smaller variance, and this reduces the chance that more packets than average are lost and not retransmitted in time, and, conversely, that fewer than average packets are lost, which can result in the transmission channel being idle. As the erasure rates increases, the smaller-frame benefit decreases because the variation in the short-term loss rate becomes less significant.

Finally, Figure 11 compares the adaptive protocol with two protocols which have fixed transmission policies (either always favoring O_L or N_H) but which do adaptively estimate L so as to know when the messages expire. The adaptive protocol's performance when it has perfect knowledge of ε (*i.e.*, it is not estimated) is also shown for comparison. Although the two adaptive protocols do better than the N_H-favoring scheme at low erasure rates and the O_L-favoring scheme at high erasure rates, the performance improvement is small. Furthermore, there are points at which the adaptive protocol (even with perfect ε knowledge) does worse than one or both of the fixed schemes. Two reasons for this are:

- The policy that is optimal below the threshold value of ε is the worst policy above that value, and vice versa. As a result the adaptive protocol relying on estimates of ε may incorrectly choose the worst possible policy if the current estimate of ε is on the wrong side of the threshold.

- The ε-threshold values which determine the adaptive protocols policy decisions are derived from a steady-state analysis of fixed L and ε—in other words, what is the best policy when L and ε are fixed over a long period of time. Since these parameters are changing throughout our simulation, the system never settles into steady-state, and so the optimal policy over each short interval of time during which ε and L are constant is not necessarily the same as the optimal steady-state policy for those values.

Our simulation results of Figure 11 illustrate the difficulty in adaptively determining the best transmission policy, as well as the marginal benefits of our attempt to do this. This only implies that one component of our streaming multimedia protocol: trying to optimally adapt the transmission policy. There are still substantial benefits to be had from the other pieces, however. Specifically, Figure 10 illustrates that layering the media signal improves signal quality by increasing the chance that the most important parts are successfully sent, and Figure 9 illustrates how adapting the message lifetime improves performance by maximizing the number of retransmission opportunities while still preventing useless transmissions of expired messages.

Table 3: Simulation parameters.

Parameter	Value
Simulation Length	2×10^5 frames
Number of Layers (N)	2/frame
T	4 s
$\bar{L}$	6 s
σ_L	2 s
λ_L^{-1}	100 s
β	0.75
μ	1
ϕ	0.5
σ_ε	0.25
λ_ε^{-1}	1000 s
α	0.5
d_1	0.1

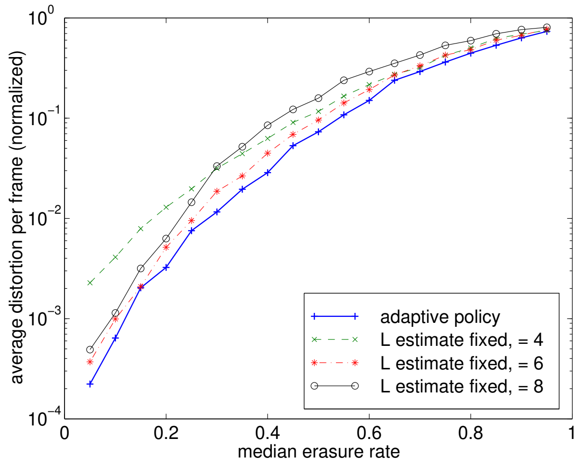


Figure 9: Comparison of fully adaptive scheme to three fixed policies (always favoring older layers) which do not estimate L.

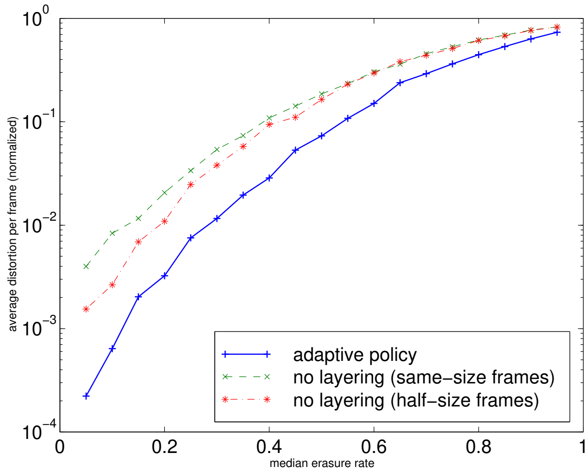


Figure 10: Comparison of the layered scheme with adaptive policy decisions to two non-layered schemes (which need no transmission policy).

5 Related Work

There has been a significant amount of work done on streaming multimedia, and much of this is oriented towards improving the performance of *interactive* multimedia streams. Because delay requirements of interactive multimedia (less than 200 ms by some measures [6]) typically preclude soft ARQ for error recovery, interactive multimedia research has explored alternative ways to improving signal quality. Before discussing related work on soft ARQ, we briefly review these alternative techniques.

One such technique for interactive media is to adjust the playback point at the receiver to compensate for variations in network delay (jitter). Many algorithms have been developed to automatically adjust the playback point [15, 8, 24, 14, 19]. The common goal of these algorithms is to minimize the playback delay small without causing signal dropouts stemming from frames that arrive past their scheduled playback times. Because they were developed for interactive multimedia, these algorithms do not try to increase the playback buffer enough to allow frames lost by the network (due to congestion) to be retransmitted. Instead, two techniques, error concealment (EC) and forward error correction (FEC) have been frequently advocated to control errors resulting from packet loss in streaming multimedia.

EC does not add any delay because it relies upon the receiver to patch up missing packet(s) by concealing the loss to the listener/viewer [28, 29, 13]. Perceptual models can be exploited to carry out effective concealment, but oftentimes simpler techniques are employed where the receiver. For example, in audio the receiver can replace missing frames with silence, white-noise, or a repeat of the last successfully received frame. And although EC requires no network or delay overhead, its performance is comparatively limited by the lack of side information about the missing data. Also, it is much harder to conceal consecutive losses ("bursts") than isolated ones. Hence EC is most useful in conjunction with an error-correcting technique (such as ARQ or FEC) which reduces the loss rate to a level low enough that EC can be effectively used to mask unavoidable losses that do occur.

Forward error correction, on the other hand, lowers the effective loss rate of streaming multimedia by adding redundant information to the original source data while incurring a small delay. This redundancy is commonly computed using algebraic block codes [27, 3, 7], but recent work in "signal processing-based FEC" (SFEC) has examined adding redundant highly compressed versions of the media signal [13, 5, 4]. In both cases, delay can be traded for robustness to error bursts. However, both techniques also result in increased load on the network due to the overhead of the redundancy. Since packet losses are usually the result of network congestion, the addition of FEC can actually cause losses that otherwise would not occur. [22, 23] showed that when many multimedia users simultaneously increase their rate by applying SFEC, their performance actually worsens.

One disadvantage of FEC is that the error correction is *forward*; because the source does not know *a priori* which packets will be lost, it sends redundant information even if it is not actually needed. ARQ retransmission schemes, on the other hand, only send extra infor-

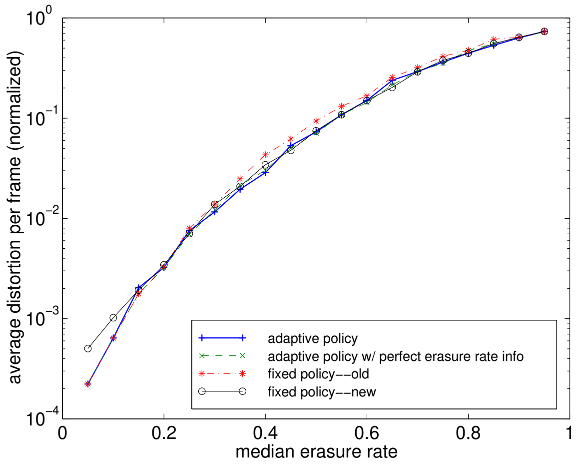


Figure 11: Comparison of adaptively changing the transmission policy to two fixed transmission policies. The data lifetime is adaptively estimated in all cases. Additionally, the performance of an adaptive policy in which the source has perfect information of ε is shown.

mation that the sender believes has been lost;[4] as a result they do not unnecessarily waste bandwidth when there is no packet loss, and they can easily adapt to changes in the loss rates. Thus studies have examined soft ARQ for both unicast [20, 11] and multicast [21, 31, 30] streaming multimedia. One way our work differs from all of these is that we assume there is an overall transmission rate limit, so that a retransmission of one message can come at the expense of the first transmission of another; these other works assume that enough bandwidth is available for any retransmissions the sender decides to send.

Methods for making traditional ARQ schemes such as TCP more suitable to delay constrained multimedia are given in [20]. They propose the following enhancements to a selective repeat-based retransmission scheme: using gap-based loss detection at the receiver, as opposed to timer-based techniques; playout buffering at the receiver to allow retransmission attempts; implicit expiration of data at the sender through knowledge of the receiver's playback buffer size (to avoid both waiting for ACKs and retransmitting packets that would arrive late); and conditional retransmission, whereby the receiver maintains an estimate of the RTT to the receiver and only requests retransmission if it is expected to arrive in time for playback. Our protocol incorporates all of these ideas; the only significant difference is that the sender controls retransmission decisions based on the data lifetime estimate rather than relying on the receiver to estimate the RTT and suppress retransmission requests.

Retransmission schemes for interactive unicast non-layered multimedia are also studied in [11], which focuses on the viability of impact of the playback delay on the effectiveness of streaming multimedia. Using an end-to-end voice transmission model and an empirical measurement study, the authors conclude that there are playback delays meeting the delay constraints of interactive audio which still allow for a high probability of successful retransmissions of single-packet losses. Unlike our work, the authors do not consider multiple retransmissions of frames that are lost multiple times.

A general examination of NACK-based retransmission-based schemes for multicast real-time media is given in [21]. The authors present an analysis which indicates that not only are retransmissions both useful and practical for real-time media, but in many situations it is optimal for the source to immediately multicast a retransmission upon the reception of a NACK from any receiver. Although the average number of packets sent is a factor in their optimality criteria, they do not account for the potential impact of the retransmissions on congestion, and hence the loss rate.

Two specific retransmission-based schemes that have been proposed for multicast streaming multimedia are STructure-Oriented Resilient Multicast (STORM) [30] and Layered Video Multicast with Retransmission (LVMR) [31]. STORM is a NACK-based technique that expands upon the Scalable Reliable Multicast (SRM) [12] approach by adding local recovery and a multi-leaf tree structure to a multicast non-layered multimedia stream. Receivers send retransmissions requests via NACKs to a parent node that is selected based on typical packet reception times and the receiver's own playback buffer, which is assumed to be fixed and preselected. However, because receivers request retransmissions of lost frames as long as their scheduled playback has not yet arrived; the source may retransmit lost frames that will arrive too late for playback at the receiver. LVMR, on the other hand, adds "smart retransmissions" to the Reliable Multicast Transport Protocol (RMTP) [16], so that a receiver sends a repair request to a designated receiver only if the receiver estimates the retransmission will arrive in time for playback. LVMR also uses a layering transmission scheme that builds upon Receiver-driven Layered Multicast (RLM) [18]. Unlike our work, in which layering is used to control the order in which data is sent, LVMR uses layering to control the overall transmission rate and to adjust the playback delay of each receiver. The idea is to allow more time for retransmission when the receiver gets a smaller number of frames per second.

The MESH protocol [17] is another framework for ARQ-based error recovery of multicast streaming multi-

[4] Whether this belief is accurate depends on the specifics of the protocol and network.

media. MESH assumes the multicast group is partitioned into subgroups (which represent high performance local area networks), and each subgroup elects an active receiver (AR) to coordinate error recovery between subgroups. [17] focuses on an ARQ-based mechanism for error recovery between the ARs. An AR sends a repairs requests to the AR with the lowest RTT that has not recently experienced a similar loss pattern. Receivers also suppresses requests if the remaining time until playback is less than the RTT estimate. Our protocol differs from this approach (and LVMR) in that we use a sender-based suppression mechanism that relies on a time-to-live measurement based upon the one-way network delay. Because the network delays and traffic loads in the paths between the source and receiver need not be symmetric, the variance in those delays may not be the same either. A potential advantage of our source-based scheme is that its data lifetime estimate (in terms of the latest allowable time it could transmit a layer) needs only to account the forward transmission delay, and thus should have less variance than a receiver-based round trip time estimate which must account for delay in both directions.

[10] describes FLITT, a fast lossy Internet image transmission scheme. FLITT is a FEC-based scheme for transmission of layered images in a finite amount of time (this time is determined by a transmission rate that is fixed for the image). FLITT starts with a fixed-rate layered encoding of the image (a wavelet transform is used); the rate is unequally allocated to each layer (by adjusting the quantizer step sizes) so that the total image distortion is minimized. Then as the image is transmitted, FLITT dynamically allocates the total rate between the image (again adjusting the quantizer step sizes) and FEC (adjusting the amount of redundancy). More FEC and quantization bits are given to visually important layers. Results indicate that FLITT transmissions of a lossy version of the image were up to five times faster than TCP transmissions. Although FLITT is not an ARQ-based protocol, it is an example of a joint source-channel coding scheme that incorporates layering to adapt to changing network conditions.

The tradeoff we analyzed in choosing between messages differing in both priority and playback deadlines has analogies to delay-constrained class-based queuing, in which a switch must choose between packets of different priorities (classes) with different deadlines. Bhattacharya and Ephremides examined such queuing problems in [1] and [2]. An important distinction is that in their work, the arrival times of packets (*i.e.*, production times of layers) are random and geometrically distributed; in our case, we have known deterministic and periodic arrival times of messages.

6 Future Work

In our analysis of the 2-layer ($N = 2$), 2-frame overlap ($K = 2$) case of our transmission model in Section 2, we found that the optimal transmission policy was always one of the two phase-invariant policies, and that other was the worst policy. Although we have observed identical results for all other values of L and T when $K = 2$ and $N = 2$, the cases of multiple layers ($N > 2$) and multiple overlaps ($K > 2$), as well as delay in the feedback, still remain open to examination. With all of these issues the problem's state space grows exponentially. As a result other approaches to analyzing the problem should be explored, such as Markov decision analysis or approximations to simplify the analysis.

We also found that our fully adaptive protocol which switched the transmission policy based on the current lifetime and erasure rate estimates had few performance benefits over fixed transmission policies that adaptively found the data lifetime. Although this negligible benefit stems from the time-varying case's nonstationarity and the chance of using the worst policy when the network estimates, it remains to be seen if these factors have the same impact when there are multiple layers and multiple overlaps. Using more layers gives the sender finer control and granularity over what to transmit and how those choices affect distortion. And combining more layers with longer overlaps leads to many more possible policies to choose from. Analysis should be performed to determine if the set of optimal policies contains more than two extremes, and if so does this result in more significant performance benefits from adapting the transmission policy?

As discussed Section 3.1, one limitation of our analysis is the zero-network-delay assumption. Because this assumption clearly does not hold in the Internet, it could be eliminated in future work. A difficulty in accounting for network delay is that the delay leads to an explosion in the state space, and as a result the number of potential transmission policies. Another limitation of our work is our assumption that packet erasures are independent events; Internet losses are often very correlated. Future work could account for correlation by adding incorporating the network status into the state space (for example, a 2-state Gilbert model could model losses).

As mentioned in Section 3.2, a potential advantage of our data lifetime estimation is that it only need adapt to variations of the *forward* network delay, in contrast to RTT-based techniques which must adapt to variations in both the forward and reverse delays. For asymmetric connections this might be especially important. Another area of future work is to study how much performance advantage (*e.g.*, in terms of correctly suppressing requests that would arrive late and allowing requests that will arrive in time), if any, our one-way estimation provides.

With all ARQ-based schemes, the receiver or source must determine when a packet is lost. Gap-based detection schemes cause unnecessary retransmissions when packets arrive out of order and excessive delay when a burst of packets is lost; timer based schemes can cause excessive delay or unnecessary retransmissions when the network delay changes more quickly than the timer's estimate. We are currently investigating a hybrid scheme which measures the level and frequency of both reordering and packet loss and uses this information to adapt how long it waits before determining a packet has been lost. The idea is that if reordering is observed much less often than packet losses, retransmissions should be sent shortly after a packet gap because the gap corresponds to a loss with high probability. However, if the relative level of reordering rises, retransmissions after gap detection are delayed to allow time for reordered packets to arrive.

Finally, we have presented a scheme which can adapt to both changes in network delay and in receiver buffering. Prior works have either studied the performance of various static receiver playback buffer sizes for retransmission or looked at ways of minimizing the playback buffer for interactivity, but no work has looked at combining the two. It remains an open area of research to dynamically compute the playback buffer size as network conditions change. For example, an algorithm might be designed to compute the optimal receiver playback size based on the current erasure rate, delay, and two curves characterizing a media application's performance as a function of error rate and of delay. Playback delay could be traded off for error recovery according to the requirements of the application.

7 Conclusion

We have developed a model for ARQ-based recovery of streaming layered multimedia. A key result from our analysis is that it is not always beneficial to favor transmission of older, low priority layers over newer, higher priority layers. We have applied the results of our analysis to develop a retransmission protocol which adapts to changes in the network erasure rate and in delay components affecting the on-time delivery of the streaming data (link delays, receiver buffering, processing delays). We have introduced a novel scheme to prevent data transmissions that will not arrive at the receiver in time for playback. By relying on exchanges of "time-to-live" information between the source and receiver, this scheme can adapt to one-way network delay changes without need the source and receiver's clocks to be synchronized. Finally, although we did not find significant benefits from adapting the transmission policy, we did show that there are benefits from layering the media signal and from adapting to delay changes.

References

[1] P. Bhattacharya and A. Ephremides. Optimal scheduling with strict deadlines. *IEEE Trans. Automat. Control*, 34(7):721–728, July 1989.

[2] P. Bhattacharya and A. Ephremides. Optimal allocation of a server between two queues with due times. *IEEE Trans. Automat. Control*, 36(12):1417–1423, December 1991.

[3] Ernst W. Biersack. A simulation study of forward error correction in ATM networks. *Computer Communication Review*, 22(1):36–47, January 1992.

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

[5] J.-C. Bolot and A.V. Garcia. Control mechanisms for packet audio in the Internet. In *Proc. IEEE INFOCOM*, volume 1, pages 232–239, San Francisco, CA, March 1996.

[6] P. Brady. Effects of transmission delay on conversational behavior on echo-free telephone circuits. *Bell Syst. Tech. J.*, 50(1):115–134, January 1971.

[7] I. Cidon, A. Khamisy, and M. Sidi. Analysis of packet loss processes in high-speed networks. *IEEE Trans. Info. Theory*, 39(1):98–108, January 1993.

[8] David Clark, Scott Shenker, and Lixia Zhang. Supporting realtime applications in an integrated services packet network: Architecture and mechanism. In *Proceedings of SIGCOMM '92*, pages 14–26, Baltimore, Maryland, August 1992. ACM.

[9] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.

[10] J. Danskin, G. Davis, and X. Song. Fast lossy Internet image transmission. In *Proc. ACM Multimedia*, pages 321–331, San Francisco, CA, November 1995.

[11] B. Dempsey, J. Liebeherr, and A. Weaver. On retransmission-based error control for continuous media traffic in packet-switching networks. *Computer Networks and ISDN Systems Journal*, 28(5):719–36, March 1996.

[12] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Trans. Networking.*, 5(6):784–803, December 1997.

[13] V. Hardman, M. A. Sasse, M. Handley, and A. Watson. Reliable audio for use over the Internet. In *Proc. INET*, 1995.

[14] Van Jacobson. SIGCOMM '94 Tutorial: Multimedia conferencing on the Internet, August 1994.

[15] M. Lara-Barron and G. Lockhard. Speech encoding and reconstruction for packet-based networks. In *IEE Colloqium on Coding for Packet Video and Speech Transmission*, volume 199, pages 1–4, 1992.

[16] J. Lin and S. Paul. RMTP: A reliable multicast transport protocol. In *Proc. IEEE INFOCOM*, pages 1414–1424, San Francisco, CA, March 1996.

[17] M. Lucas, B. Dempsey, and A. Weaver. MESH: distributed error recovery for multimedia streams in wide-area multicast networks. In *Proc. IEEE Int. Conf. on Commun.*, volume 2, pages 1127–32, Montreal, Que., June 1997.

[18] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *Proc. ACM SIGCOMM*, pages 26–30, Stanford, CA, August 1996. ACM.

[19] S. Moon, J. Kurose, and D. Towsley. Packet audio playout delay adjustment: performance bounds and algorithms. *Multimedia Systems*, 6(1):17–28, January 1998.

[20] C. Papadopoulos and G. Parulkar. Retransmission-based error control for continuous media applications. In *Proc. NOSSDAV*, pages 5–12, April 1996.

[21] S. Pejhan, M. Schwartz, and D. Anastassiou. Error control using retransmission schemes in multicast transport protocols for real-time media. *IEEE/ACM Trans. Networking.*, 4(3):413–427, June 1996.

[22] M. Podolsky. A study of speech/audio coding on packet switched networks. Master's thesis, U.C. Berkeley, December 1996. Also published as UC-Berkeley/ERL Tech Report M96/96, Dec. 1996.

[23] M. Podolsky, C. Romer, and S. McCanne. Simulation of FEC-based error control for packet audio on the Internet. In *Proc. IEEE INFOCOM*, volume 2, pages 505–515, San Francisco, CA, March 1998.

[24] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *Proc. IEEE INFOCOM*, pages 680–8, Toronto, Ont., June 1994.

[25] R. Rejaie, M. Handley, and D. Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet. *To appear in INFOCOM '99*. Chapter 13.

[26] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. Internet Engineering Task Force, Audio-Video Transport Working Group, January 1996. RFC-1889.

[27] N. Shacham and P. McKenney. Packet recovery in high-speed networks using coding and buffer management. In *Proc. IEEE INFOCOM*, volume 1, pages 124–131, San Francisco, CA, June 1990.

[28] M. Wada. Selective recovery of video packet loss using error concealment. *IEEE Journal on Selected Areas in Communications*, 7(5):807–814, June 1989.

[29] O. Wasem, D. Goodman, C. Dvorak, and H. Page. The effect of waveform substitution on the quality of PCM packet communications. *IEEE Trans. Acoust. Speech Signal Proc.*, 26(3):342–348, March 1988.

[30] X. Xu, A. Myers, H. Zhang, and R. Yavatkar. Resilient multicast support for continuous-media applications. In *Proc. NOSSDAV*, pages 183–194, St. Louis, MO, May 1997.

[31] L. Xue, , S. Paul, P. Pancha, and M. Ammar. Layered video multicast with retransmission (LVMR): evaluation of error recovery schemes. In *Proc. NOSSDAV*, pages 161–172, St. Louis, MO, May 1997.