Loss-resilient On-demand Media Streaming Using Priority Encoding[*]

Cheng Huang   Ramaprabhu Janakiraman   Lihao Xu

Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130-4899, USA
E-mail: {cheng, rama, lihao}@cse.wustl.edu

ABSTRACT

A novel solution to the reliable multicast problem is the "digital fountain" approach, in which data is encoded with an erasure protection code before transmission, and receivers can recover the original data after receiving enough distinct encoded data. This solution, however, is not desirable for streaming media schemes in which it is preferable for parts of a movie to be available for consumption before the entire movie is received. Earlier work has proposed the use of Unequal Error Protection (UEP) codes, which permit some parts of the movie to be recovered before others. Unfortunately, a straightforward implementation of this solution can incur prohibitive coding complexity.

We outline an on-demand media streaming scheme involving a combination of segmentation and rateless encoding. Our solution reduces the coding complexity to feasible levels, while guaranteeing the least bandwidth consumption for a given playout delay and number of segments. We propose an efficient algorithm to find the optimal segmentation for single-layered and multi-layered transmissions, and analyze its performance under network packet loss. Through analysis, numerical examples, and simulations, we demonstrate the feasibility and performance of the proposed scheme.

Categories and Subject Descriptors: C.2.4 [Distributed Systems]: Distributed Applications
General Terms: Algorithms, Design, Theory
Keywords: Video-on-demand, Priority Encoding Transmission, Fountain Codes, Multicast, Loss Resilience

1. INTRODUCTION

Content distribution networks are often beset by the "flash mob" effect, where the latest release of a popular software application or a movie trailer triggers a flurry of accesses by clients, saturating the connection bandwidth of the server. Researchers have proposed several ways to ameliorate the problem by load-balancing the work between multiple servers or using efficient distribution methods like multicast.

One of the stumbling blocks in large scale multicast distribution is implementing a reliable transport for one-to-many distribution schemes. A novel solution is to use a "digital fountain" approach [2] in which the data are protected using *erasure codes* and transmitted cyclically over a multicast channel. Any number of clients can subscribe to the channel at any time and receive these transmissions without consuming additional server resources. The use of appropriate erasure codes ensures that clients receiving enough unique transmitted packets can fully recover the original data with low computational and communication costs.

The trouble in using a digital fountain for *streaming* media distribution on the other hand is that the encoding makes no promises about *partial* availability of data while the download is in progress. This forces clients to wait until the entire file is downloaded before commencing playout.

For streaming media distribution, several schemes (perhaps too many, Hu [5] provides a nice survey) have been proposed in which the data is segmented into pieces, and each piece is transmitted at a different rate. The rate assignment is such that clients listening to these transmissions can commence playout after a short delay while still receiving portions due for playout later. All these schemes involve multicasting the *original* data unprotected by erasure codes and hence do not inherit the desirable properties of a digital fountain.

The earliest work to reconcile the two approaches [10] proposes a segmentation and loss-resilient encoding scheme for on-demand streaming. The proposed segmentation scheme is not bandwidth-optimal. A later work [16] proposes the use of Unequal Error Protection (UEP) codes for specifying a different degree of erasure protection on each frame to obtain a stream of UEP encoded packets, which are multicast cyclically. Clients receiving these packets can progressively recover the original media data after a small delay even as they continue to receive data due for later playout. This scheme has theoretically optimal bandwidth utilization, but incurs a prohibitively high encoding/decoding complexity.

Since the latter scheme [16] has provably optimal performance, we take that as a starting point to propose an encoding and transmission scheme for efficient and loss-resilient on-demand media streaming to a large and heterogeneous client population. Our work extends and generalizes this work in several ways:

Practicality: Our first result is a general method to *partition* a movie by grouping individual frames or symbols into a few segments with minimal bandwidth overhead while preserving the guar-

[*]This work was supported in part by NSF Grants CCR-TC-0209042 and ANI-0322615

antee on initial delay, so that it becomes practicable to use existing erasure codes to protect the segments from network loss. This generalizes several existing segmentation and transmission schemes. We obtain analytical results on segmenting the optimal transmission scheme in [16] and provide a method to optimally segment *any* broadcast transmission scheme.

Efficiency: Traditional erasure codes like the well-known *Reed-Solomon codes* require high computational overhead for large data sizes. We suggest a way to use the recently-discovered *fountain codes* as a feasible way to implement our broadcast scheme, and outline the design of a rateless priority encoder.

Supporting Client Heterogeneity: In prior work [8], we propose a scheme to support heterogeneous clients using layered encoding. We extend the optimal broadcast segmentation technique to a layered transmission scheme, obtaining initial results on reducing the latencies of a heterogeneous client population.

Loss Resilience: We analyze the performance of our scheme under both random and burst network loss. We also propose an extension in which the segmentation is tailored according to a targeted network packet loss rate.

Reducing Latency: To keep the bandwidth utilization small, the initial delay before playout in most proposed schemes is significant, perhaps as much as a few minutes. We propose a “fast start” option for media streaming in which the initial portions of a movie are encoded at a lower quality than the rest. This is analogous to the progressive decoding of still images. We provide an algorithm to maximize the average quality of the movie for a given (bandwidth, delay) pair, while taking into account smooth variation in quality.

The paper is organized as follows: we start by describing the general broadcast framework used in the remainder of the paper. Next, we summarize the bandwidth-optimal scheme [16] and describe our broadcast segmentation technique in Section 3. In Section 4, we extend this method to layered broadcasts for supporting heterogeneous clients. In Section 5, we describe a *pruning* method that greatly reduces the redundant transmissions received by clients, and study the effect of the broadcast segmentation on pruning performance. We study the loss resilience of our scheme in Section 6. In Section 7, we propose a “fast start” option for reducing the latency for low-bandwidth clients. Conclusions and an outline of ongoing work end the paper. For readability, we state all theorems without proof, and in some cases restate theorems from our previous work [8].

2. BROADCAST FRAMEWORK

Before we describe our scheme, we will formalize the terminology used in the rest of the paper:

Symbols are fixed-size atomic units of data. All encoding, decoding and transmission is done on symbols. A symbol could be a bit or a block of bits. A ***broadcast*** $\langle C, M, S, T\rangle$ consists of:

A *channel* C with *bandwidth* $|C|$ symbols per unit time. Clients can subscribe and unsubscribe to C at any time. A set of X of $|X|$ symbols transmitted exclusively on C can be recovered in time $\frac{|X|}{|C|}$ by clients subscribed to C. Multiple such sets can be transmitted concurrently on C, and set X has rate r on C if X can be recovered entirely after listening to C for a time $\frac{|X|}{r}$. This refers to the *ideal* or achievable performance of the channel under no-loss conditions.

A *movie* M, which is simply a set of symbols $\{f_i : i = 1 \text{ to } |M|\}$, each played out in unit time. Thus the playout time of M is simply $|M|$ time units. A segment s of M is a set of symbols played out continuously in time $|s|$.

A *segmentation* S of M, which is an ordered partition of M into segments s_i; segment s_i is played out entirely before s_{i+1}, and $\cup_{s_i \in S} s_i = M$.

A *transmission* T mapping elements s_i of S to rates $T(s_i)$ such that s_i has rate $T(s_i)$ on C and $\sum_{s_i \in S} T(s_i) = |C|$.

Definition 1
Given w, a broadcast $\langle C, M, S, T\rangle$ is w-feasible if a client subscribed to C can begin watching movie M uninterruptedly after an initial delay of at most w. A w-feasible broadcast is optimal if no T' exists for which a broadcast $\langle C, M, S, T'\rangle$ is w'-feasible for C, M, S and $w' < w$. ■

Theorem 1
Optimal Broadcast [8]: For a segmentation S, denote the prefix $\wp(s_i)$ for $s_i \in S$ as $\cup_{j=0}^{i} s_j$ and its size $|\wp(s_i)|$ as $\sum_{j=0}^{i} |s_j|$*. Broadcast $\langle C, M, S, T\rangle$ is w-feasible iff:

$$T(s_i) \geq \frac{|s_i|}{w + |\wp(s_{i-1})|} \tag{1}$$

and optimal if the equality holds. ■

Note that an optimal broadcast given C, M, and S is neither necessarily optimal for any segmentation given C and M, nor is an optimal schedule according to (1) necessarily feasible when additional constraints like storage limits are introduced.

2.1 Implementing broadcasts using PET

In this section we touch briefly on how the preceding abstract formulation of a broadcast $B = \langle C, M, S, T\rangle$ may be realized in practice.

A naïve way to do it is to just transmit the source symbols of M over C, with segment s_i transmitted at rate $T(s_i)$. This is the approach taken by traditional on-demand streaming protocols. However, this performs poorly over a loss-prone channel. To recover a lost symbol a client has to wait until much later than its scheduled playout, by which time it is useless.

A better way is to encode symbols using an *erasure-resilient* code and transmit the encoded symbols on the channel. Since some parts of the transmission must be recoverable before others, a Priority Encoded Transmission (PET) [1] naturally suggests itself. Given a file M of $|S|$ segments and a priority function $\{\rho_i, i = 1 \text{ to } |S|\}$, a PET system produces n encoded packets such that segment s_i can be recovered from any fraction ρ_i of these n packets. Note that the packets may be larger than source symbols in this case.

One way to realize broadcast B using PET is to encode each segment s_i using an $(n_i, |s_i|)$ *maximum distance seperable* (MDS) code [13]. MDS codes have optimal erasure recovery capability: any $|s_i|$ out of n_i code symbols encoded using an $(n_i, |s_i|)$ MDS code suffice to recover the original $|s_i|$ source symbols of s_i. Then, some n packets of a suitable length ℓ are constructed by choosing k_i out of n_i encoded symbols per segment s_i per packet without replacement, where:

$$n \geq \frac{|C|}{\ell} \max\left\{\frac{|s_j|}{T(s_j)}\right\};\ n_i = \frac{T(s_i) n \ell}{|C|};\ k_i = \frac{n_i}{n}. \tag{2}$$

These n packets are transmitted repeatedly on C.

Theorem 2
Broadcasts $\equiv$ ***PET*** [8]: This scheme represents a priority encoded transmission with $\rho_i = \frac{|s_i|}{n_i}$ and rate $T(s_i)$ for segment s_i on C. ■

*We assume a dummy zero-sized segment s_0 exists

In principle, the above-mentioned scheme implements a loss-resilient broadcast using PET. However, for large media files, using traditional MDS erasure codes can result in prohibitive encoding and decoding complexity.

In practice, we envision the use of codes with more efficient encoding and decoding. Specifically, we propose the use of the recently discovered *rateless codes* like *Raptor codes* [14]. Rateless codes, instead of having a fixed encoding length, use *random* encoders to generate code symbols in a *stateless* manner, and are ideally suited for large-scale data distribution applications. The encoding process is extremely simple: to generate a code symbol, the encoder randomly picks a degree d from a carefully constructed degree distribution. It then picks d random source symbols, and computes the encoded symbol as a sum (usually bit-wise XOR) of these source symbols. This simple structure leads to extremely fast encoding and decoding. A small disadvantage is that these codes are not MDS. To recover the original k source symbols, some $k(1+\epsilon)$, $(1 >> \epsilon > 0)$ encoded symbols are needed.

This approach solves all the problems associated with broadcast implementation, without the prohibitive complexity of using regular MDS codes. For example, the encoder for a Raptor code does $O(\log \frac{1}{\epsilon})$ work on average per transmitted symbol regardless of the number of segments or their rate assignment. Decoding complexity is equally low, $O(|s_i| \log \frac{1}{\epsilon})$ to recover segment s_i.

In the rest of this paper, we will continue to work with broadcasts rather than the underlying PET framework, and will make the simplifying assumption that an arbitrary w-feasible broadcast can be implemented efficiently by a suitable PET-based system.

3. BROADCAST SEGMENTATION

In this section, we address the issue of devising a segmentation S of a movie M over a channel C, and assigning transmission rates T such that the resulting broadcast $B = \langle C, M, S, T\rangle$ is optimal.

First, we start with an ideal segmentation. It is known [4] that when there is no limit on the number of segments $|S|$, the optimal solution is to split the movie into as small segments as possible; in the extreme case, each symbol is transmitted in its own segment.

This is the approach proposed in [16], and corresponds to a loss-resilient version of the well-known *harmonic broadcasting* [12] protocol. In other words, the optimal broadcast to guarantee a delay of w time units for a movie M is $B^* = \langle C^*, M, S^*, T^*\rangle$, where the parameters bear the following relation:

$$|S| = |M|; s_i = f_i;\;\; T^*(s_i) = \frac{1}{w+i-1};\; |C^*| \approx \log\big(1 + \frac{|M|}{w}\big)$$

Example 1 Consider a 2-hour, 1 mbps constant bitrate, 30 frames-per-second Mpeg-4 movie transmitted using this scheme. Consider three classes of clients: DSL (1.5 mbps downlink), Cable Modem (4 mbps downlink), and Ethernet (10 mbps downlink). For simplicity, we assume clients can download data at their maximum downlink capacity with negligible protocol overhead. The performance of the above-mentioned scheme tailored for each class of clients is shown below:

Class	Downlink capacity	Initial Delay
DSL	1.5 mbps	34 minutes
Cable Modem	4 mbps	135 seconds
Ethernet	10 mbps	1/3 second

Note that the server bandwidth usage matches the client's usage in each case. Thus it is feasible for a server to stream several tens of movies concurrently, with unlimited number of clients streaming each movie on demand. Also note that in this scheme, a little bandwidth goes a long way: going from 1.5 to 10 mbps reduced the initial delay of clients from 34 minutes to one third of a second. ■

This approach guarantees the least bandwidth utilization $|C^*|$ to guarantee a given initial delay w. However, there are practical difficulties: to keep the encoding and decoding complexities from being prohibitively high, a movie must be segmented into a few segments $|S| \ll |M|$; it is not feasible to construct a few hundred thousand encoders per movie broadcast. Even when efficient rateless codes are used, the number of symbols in a segment must be sizable to ensure good performance.

At first glance, it might appear that splitting the movie into $|S|$ equal-sized segments would suffice to solve this problem. This is non-optimal. To be precise, this corresponds to a *uniform* partition of symbols into segments. On the other hand, we seek an *optimal partition* of symbols into segments, as we explain below:

Definition 2
Given n and k, an (n,k)-partition is a partition of the interval $[1..n]$ into k intervals.[†] Given arbitrary costs $\boldsymbol{\sigma} = \{\sigma_1, \sigma_2, \cdots, \sigma_k\}$ mapping intervals in $[1..n]$ to real numbers, a mincost (n,k)-partition $\mathfrak{K}(n,k,\boldsymbol{\sigma}) = \{s_1, s_2, \cdots, s_k\}$ is one that minimizes the sum of the interval costs of its members: $||\mathfrak{K}(n,k,\boldsymbol{\sigma})|| = \min \sum_{i=1}^{k} \sigma_i(s_i)$
■

Example 2 $\{[1..3], [4..5], [6..9], [10]\}$ is a $(10,4)$-partition. If $\sigma_i(a,b) = \frac{b}{a}$ for $i = 1$ to 4, then a mincost $(10,4)$-partition is $\mathfrak{K} = \{[1], [2], [3..4], [5..10]\}$ with cost $||\mathfrak{K}|| = \frac{16}{3}$. ■

The mincost (n,k)-partition problem has an efficient $O(k(n-k)^2)$ dynamic programming solution [8].

With this in place, we can now formulate the broadcast segmentation in terms of a mincost (n,k)-partition as follows: We know that the optimal scheme puts each symbol in a separate segment. Restricting the number of segments $|S|$ to fewer than $|M|$, while guaranteeing the wait to remain w, requires the transmission rate of some symbols to exceed their rate under the optimal broadcast. Given a maximum wait w for movie M and a limited number of segments $|S| < |M|$, our goal is to determine a w-feasible segmented broadcast $B = \langle C, M, S, T\rangle$ with minimal channel bandwidth $|C|$.

Theorem 3
Optimal segmentation: Consider a w-feasible segmented broadcast $B = \langle C, M, S, T\rangle$ with $|S| < |M|$. $|C|$ is minimized by the segmentation $S = \{s_i : i = 1 \text{ to } |S|\}$ if $\mathfrak{K}(|M|, |S|, \boldsymbol{\sigma}) = \{s_i : i = 1 \text{ to } |S|\}$ is a mincost $(|M|, |S|)$-partition for $\boldsymbol{\sigma} = \{T(s_i) : i = 1 \text{ to } |S|\}$. The bandwidth usage of the segmented broadcast B is the value of this mincost partition: $|C| = ||\mathfrak{K}(|M|, |S|, \boldsymbol{\sigma})||$. ■

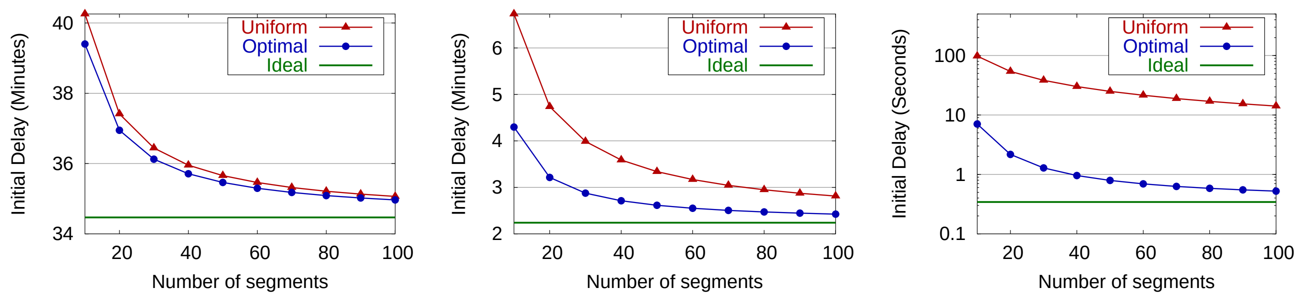
3.1 Single-layered case: analytical solution

In the case of the idealized protocol described in the previous section, we can explicitly obtain an optimal partition as follows:

Given a wait w, consider an optimal segmented broadcast $\langle C, M, S, T\rangle$ with segmentation $S = \{s_i : i = 1 \text{ to } |S|\}$. From (3), we know that

$$\begin{aligned} T(s_i) &= \frac{|\wp(s_i)| - |\wp(s_{i-1})|}{w + |\wp(s_{i-1})|} = \frac{w + |\wp(s_i)|}{w + |\wp(s_{i-1})|} - 1 \\ ||\mathfrak{K}(|M|, |S|, \boldsymbol{\sigma})|| &= \sum_{i=1}^{|S|} \Big(\frac{w + |\wp(s_i)|}{w + |\wp(s_{i-1})|} - 1\Big) \end{aligned} \tag{3}$$

[†]An interval $[a..b]$ where $a \le b$ is the set of integers from a to b, inclusive.



(a) DSL users (1.5 mbps) (b) Cable modem users (4 mbps) (c) Ethernet users (10 mbps)

Figure 1: **Performance of optimal segmentation**

Differentiating $||\mathfrak{K}(|M|, |S|, \boldsymbol{\sigma})||$ partially w.r.t. $|\wp(s_i)|$ and setting it to zero for minimal cost gives the recurrence:

$$(w + |\wp(s_i)|)^2 = (w + |\wp(s_{i-1})|)(w + |\wp(s_{i+1})|), \text{ or} \\ |s_i|^2 = |s_{i-1}||s_{i+1}| \tag{4}$$

Thus, the segmentation that minimizes total bandwidth consumption divides the movie into segments of geometrically increasing size. Imposing the constraint $\sum_{i=1}^{|S|} |s_i| = |M|$ gives the result:

$$|C| = |S|\left(\{1+\frac{|M|}{w}\}^{\frac{1}{|S|}} - 1\right)$$

As $|M| \to \infty$ and $|S| \to |M|$, $|C^*| \to \log(1+\frac{|M|}{w})$, and

$$|C| \to \lim_{|S|\to\infty} |S|\left(e^{\frac{1}{|S|}\log(1+\frac{|M|}{w})} - 1\right) \to \log(1+\frac{|M|}{w}) \to |C^*| \tag{5}$$

Thus we see that for a large number of segments and symbols, the bandwidth consumption of the segmented broadcast approaches that of the ideal broadcast scheme. Note that the geometric segmentation above is known in the literature in the context of harmonic broadcasting [5]; our contribution is a general method to segment any such periodic broadcast scheme (including VBR media broadcasts), and its use in implementing a loss-resilient broadcast using rateless priority encoding.

The baseline case we use for comparison is the straightforward one of segmenting the movie into $|S|$ equal segments of approximately $|M|/|S|$ each, which, for a delay w, gives the following bandwidth utilization:

$$|C| = \sum_{i=1}^{|S|} \frac{1}{\frac{w|S|}{|M|} + i - 1} \tag{6}$$

Example 3 Consider the same movie and the same three classes of clients. The following table summarizes the initial delay for each class of clients, achieved by splitting the movie into 10 segments and 100 segments. Thus we see that with as few as 100

Class	**Initial Delay**		
	Ideal	**10 segs**	**100 segs**
DSL(1.5 mbps)	34 min	40 min	35 min
Cable Modem(4 mbps)	135 sec	258 sec	145 sec
Ethernet(10 mbps)	1/3 sec	7 sec	1/2 sec

segments, the performance is acceptably close to the optimal one, while making the encoding/decoding process much simpler (only 100 encoders are needed instead of a few hundred thousand.)

Figure 1 shows the performance of the optimal segmentation scheme compared with the "ideal" performance and that of the uniform (equal segment sizes) segmentation scheme. ■

4. SUPPORTING HETEROGENEITY

Using a single broadcast $\langle C, M, S, T\rangle$ limits all clients to receive at the same rate, and thus have identical initial delays regardless of their bandwidth constraints.

One way to support heterogeneous clients with varying bandwidth constraints is to split the bandwidth available for C into multiple channels of smaller bandwidth. A naïve approach is to partition C into multiple channels of varying bandwidth, and let clients choose their preferred one. This is wasteful use of channel bandwidth.

A more efficient way is to *layer* the broadcast cumulatively over multiple channels. The details of the layered broadcast we propose are as follows: suppose we want to provide for α classes of clients, with a set of delays $\mathbf{w} = \{w_1, w_2, \cdots, w_\alpha\}$ in decreasing order. This can be provided by α cumulative layers over α channels. Subscribing to the first channel provides a delay of w_1, the first and second together provide a delay w_2, and similarly subscription to all channels provides a delay w_α. In this scheme, clients will subscribe to as many channels as their bandwidth constraints allow to get the minimal delay possible before beginning playout.

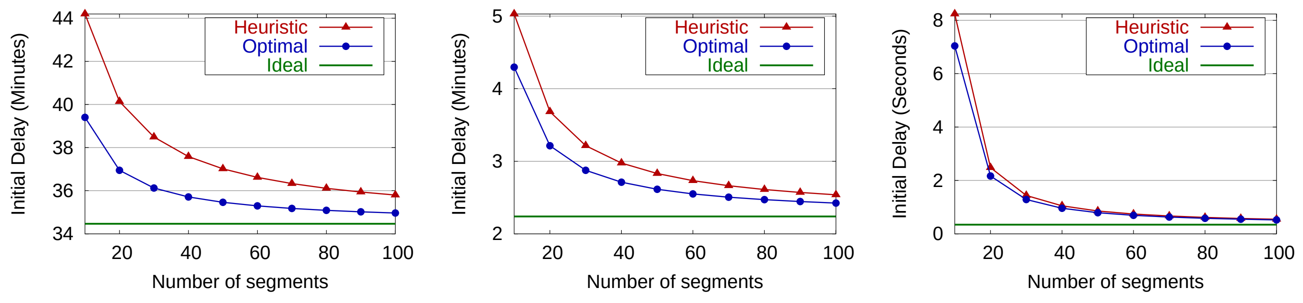
Clearly, we can extend the framework in the previous section in a straightforward manner by *vectorizing* all parameters. A layered broadcast $\mathbf{B} = \langle \mathbf{C}, M, S, \mathbf{T}\rangle$ implements a set of $|\mathbf{B}|$ *virtual* broadcasts $B_i = \langle C_i, M, S, T_i\rangle$, $i = 1$ to $|\mathbf{B}|$, where subscribers at layer i will get a virtual broadcast B_i by cumulatively subscribing to the first i channels. Thus the *physical* broadcasts, i.e., the broadcasts going out on the individual channels, are just the *deltas* between successive virtual broadcasts. Note that the bandwidth of the channel with least delay, $|C_{|\mathbf{B}|}|$ is the total bandwidth used by this layered scheme.

An optimal broadcast scheme to implement a set of delays $\mathbf{w}$ is implemented by setting B_i to be the optimal broadcast for delay $w_i \in \mathbf{w}$ (using the mapping from the previous section), and using the above relationship to determine the per-channel broadcasts.

Example 4 Consider the movie in the previous examples. Suppose the server aims to provide a layered broadcast to satisfy all three classes of clients simultaneously. There will be three layers with bandwidth 1.5 mbps, 4 mbps, 10 mbps transmitted over three channels of bandwidth 1.5 mbps, $4 - 1.5 = 2.5$ mbps, and $10 - 4 = 6$ mbps. Note that the server bandwidth requirement with layering is just 10 mbps, rather than 15.5 mbps if separate channels are used.

4.1 Multi-layered broadcast segmentation

In this section, we study the problem of segmenting a *layered* broadcast. From the previous section, we know that for any given delay w, there exists an optimal segmented broadcast with geometric segment sizes that are functions of w. The problem with



(a) DSL users (1.5 mbps) (b) Cable modem users (4 mbps) (c) Ethernet users (10 mbps)

Figure 2: **Performance of heuristic multi-layered segmentation**

segmenting a layered broadcast is that its component broadcasts share data transmissions, and hence must follow a common segmentation, but individually they must have different segmentations to minimize per-layer delays.

The problem, then, is to determine a common segmentation which will minimize the maximum *delay inflation* over the entire client population. If $w^*(C)$ is the delay with optimal segmentation tailored for channel C, and $w(C, S)$ is the minimum delay for C under segmentation S, then our goal is as follows:

$$\min\max \frac{w(C_i, S) - w^*(C_i)}{w^*(C_i)}, \quad i = 1 \text{ to } |\mathbf{B}| \tag{7}$$

The problem can be approximately solved using the following heuristic: The idea is to assume a common geometric segmentation that is *targeted for a virtual delay* w_v. Such an assignment is not optimal since the segmentation is not tailored to the individual channel bandwidth. Hence the problem can be treated as a univariate minimization problem on w_v, the objective being to minimize the maximum delay inflation across the entire user population. This solution is still not optimal, since it constrains the segmentation to follow a geometric progression. However, as the following example shows, the heuristic performs adequately well for a reasonable number of segments:

Example 5 Consider the movie and 3 classes of clients from previous examples. The following table shows the initial delay incurred by each class of clients when heuristic multi-layered segmentation is used with 100 segments. For comparison, the delay under *optimal* per-layer segmentation is shown.

Class	**Initial Delay with 100 segments**	
	Optimal	**Heuristic**
DSL(1.5 mbps)	35 min	36 min
Cable Modem(4 mbps)	145 sec	152 sec
Ethernet(10 mbps)	1/2 sec	1/2 sec

Figure 2 shows the resultant initial delay of the heuristic multi-layered segmentation scheme compared to the per-layer optimal segmentation. ■

5. REDUCING RECEPTION OVERHEAD

In this section, we study the reception overhead of the proposed segmented broadcast scheme. One drawback when using priority encoding in a broadcast setting is that while clients are able to recover portions of the broadcast earlier than others at the cost of increased bandwidth consumption, they lose the utility from further transmissions of the recovered portions. For example, once the first segment is recovered, the client does not require the broadcast to contain symbols for that segment anymore. However, the server must keep repeating the broadcast containing all segments to enable later clients to start watching from the beginning.

One way to reduce this cost somewhat, as described in [8], is to use *progressive pruning*. First consider the single channel case. In this case, the channel is split into a small number of sub-channels. Clients have the ability to "tune in" to and "tune out" of sub-channels. In the context of Internet multicast, for example, each sub-channel corresponds to a separate multicast group to which each client can subscribe and unsubscribe independently. The idea of progressive pruning is to split the movie into *chapters* of contiguous segments, one per sub-channel, and transmit each chapter on its own sub-channel, the goal being to split the movie so that a client, by unsubscribing greedily from each sub-channel, can receive the minimal wasteful symbols over the course of the movie.

Consider a layered broadcast $\mathbf{B} = \langle \mathbf{C}, M, S, \mathbf{T} \rangle$. Assume that a total of β channels are available, where $\beta > |\mathbf{B}|$, and that the channel C_i for each broadcast B_i can be split into *pruning* sub-channels dropped progressively by clients during the course of playout, with the total number of pruning sub-channels over all layers not exceeding β. Our goal is to jointly devise an assignment of β sub-channels to $\mathbf{B}$ channels and, for each channel, an assignment of $|S|$ segments to that channel's sub-channels in a way that minimizes the reception overhead of clients listening to these channels.

While other objectives like minimizing network bandwidth overhead, etc., can be considered, we focus on strategies that minimize the average reception overhead η of $\mathbf{B}$, where the averaging is done over the entire client population listening to $\mathbf{B}$:

Reception overhead $=$ **#symbols received** $-$ **#symbols in movie**

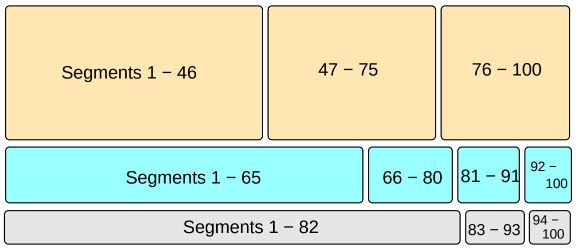
Theorem 4
Optimal progressive pruning [8]: Given a receiver rate distribution $F(.)$ and a layered broadcast $\langle \mathbf{C}, M, S, \mathbf{T} \rangle$, with β channels assigned to $|\mathbf{B}|$ layers, average reception overhead is minimized by the partition $\mathfrak{K}(\beta, |\mathbf{B}|, \boldsymbol{\sigma})$ where:

$$\boldsymbol{\sigma} = \{\sigma_i : i = 1 \text{ to } |\mathbf{B}|\}$$
$$\sigma_i([a..b]) = ||\mathfrak{K}(|S|, b - a + 1, \boldsymbol{\sigma_i})||(1 - F(|C_i|)) \text{ , where}$$
$$\boldsymbol{\sigma_i} = \{\sigma_{ij} : j = 1 \text{ to } \beta - |\mathbf{B}| + 1\}$$

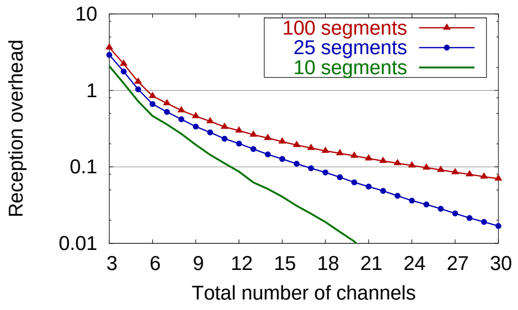
$$\sigma_{ij}([a..b]) = \sum_{k=a}^{b} (T_i(s_k) - T_{i-1}(s_k))(|\wp(s_{b-1})| - |\wp(s_{k-1})|)$$

Using precomputed values of $\mathfrak{K}(|S|, j, \boldsymbol{\sigma_i})$ for $i = 1$ to $|\mathbf{B}|$ and $j = 1$ to $\beta - |\mathbf{B}| + 1$, this can be solved to give an allocation of sub-channels to each broadcast. The total running time for the entire procedure can be shown to be $O(|S|^2|\mathbf{B}|(\beta - |\mathbf{B}|)^2)$.

Example 6 Consider the movie in the previous examples, along with the three classes of users with the arbitrary distribution of 30%



(a) Splitting 3 layers with 10 channels (Each row represents a channel and each block within it a pruning sub-channel)



(b) #channels vs. Reception overhead

Figure 3: **Progressive pruning and performance for 3-layered broadcast**

1.5 mbps DSL users, 40% 4 mbps cable modem users, and 30% 10 mbps ethernet users.

Assume that a 3-layered broadcast is used, and the heuristic multi-layered segmentation of the previous section is used with 100 segments. Now suppose that 10 channels are available, and can be used as pruning channels over the three physical channels implementing the 3-layered broadcast. The allocation that minimizes average reception overhead is shown in Figure 3(a). It results in an average reception overhead of about 40% of the movie size. Assuming the same classes of users and their distribution, Figure 3(b) shows the average reception overhead as a function of number of channels. ■

Note that the reception overhead is actually *higher* when more segments are used since in this case more segments are recoverable earlier and must be tolerated until they can be collectively dropped.

6. LOSS RESILIENCE

In a loss-free environment, both the ideal scheme and the segmented scheme proposed in this paper guarantee that segment s_i is available to the client when segments s_1 through s_{i-1} have been fully consumed. Thus, ideally these schemes will *not* suffer from the so-called "re-buffering" effect, once playout begins after a predefined initial delay w. However, when operating under lossy networks, clients have to tolerate an extra delay to ensure smooth playout under packet loss.

Effectiveness in combating packet loss is the most powerful feature of an encoding-based scheme compared to traditional on-demand streaming schemes. In this section, we quantify the effect of both random and burst network losses on the performance of our segmented broadcast scheme. The former is easier to analyze, and provides useful insight for the more realistic latter case.

6.1 Resilience to Random Loss

We first address the following problem: given a random symbol loss rate p, how many symbols n do we expect to receive until we get *exactly* k symbols? Let $Pr(n, k)$ be the probability of receiving k symbols with $n - k$ ($n \geq k$) symbols lost. Then,

$$Pr(n,k) = \binom{n-1}{k-1} p^{n-k}(1-p)^k \tag{8}$$

The $\binom{n-1}{k-1}$ term comes because the k^{th} symbol has to be received and other received symbols occupy $(k-1)$ out of the remaining $(n-1)$ positions. From this, the expected value can be determined using probability generating functions as:

$$\bar{n} = \frac{k}{1-p} \tag{9}$$

In a random loss environment, instead of receiving all $|s_i|$ encoded symbols of segment s_i by the time segment s_{i-1} is consumed, it is now desirable to receive $|s_i|/(1-p)$ encoded symbols. This can be achieved by simply increasing the transmission rate of each segment by a factor of $1/(1-p)$: as

$$T(s_i) = \frac{1}{1-p}\frac{|s_i|}{w + |\wp(s_{i-1})|} \tag{10}$$

Clearly, the optimal segmentation S from the no-loss case still yields minimum broadcast bandwidth and thus remains optimal.

Example 7 Consider the movie broadcast and 3 classes of users in previous examples. The increased initial delay for these users as a result of random losses in the network is tabulated below. Assume that the broadcast is tailored to each individual loss rate as described above and that 100 segments are used.

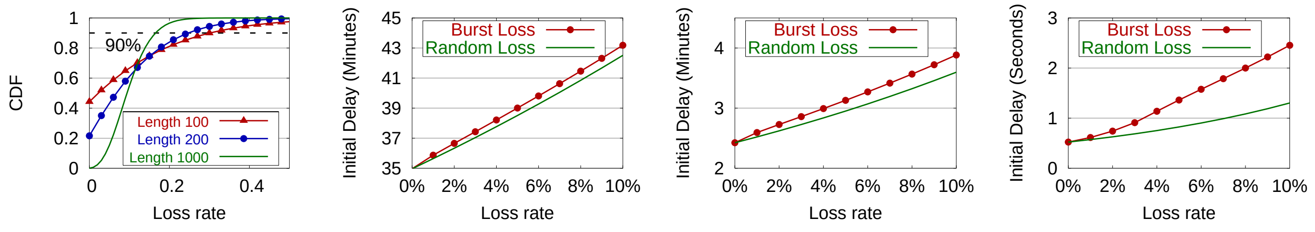
Class	**Initial Delay with random loss**		
	No loss	**5% loss**	**10% loss**
DSL(1.5 mbps)	35 min	38.5 min	42.5 min
Cable Modem(4 mbps)	145 sec	177 sec	216 sec
Ethernet(10 mbps)	1/2 sec	0.8 sec	1.3 sec

Figures 4(b) through 4(d) show the performance of segmented broadcast tailored to different random loss rates.

6.2 Resilience to Burst Loss

Wide-area packet loss on the Internet is typically bursty [15]. Burstiness can be modeled reasonably well with a two-state Gilbert model [6] defined by two parameters: average burst length b and average loss rate p. The model suggests that loss is not uniform and burstiness is more prominent at shorter time scales. In our segmentation scheme, earlier segments in the movie are geometrically smaller than later ones. Therefore they are more subject to burst loss than later ones and warrant greater protection from packet loss. With apriori knowledge of model parameters, we now describe how our segmentation scheme can be modified to tolerate burst loss such that each segment is equally well protected, while achieving minimum broadcast bandwidth.

We first introduce the concept of *loss rate bound* p_b and determine its connection with segment length. As in the previous section, let $Pr(n, k)$ denote the probability of $n - k$ losses out of n



(a) Burst Loss Distribution (b) DSL (1.5 Mbps) (c) Cable Modem (4 Mbps) (d) Ethernet (10 Mbps)

Figure 4: **Loss resilience of segmented broadcast (100 segments)**

transmitted symbols to recover a segment of length k. With given Gilbert model parameters, $Pr(n,k)$ can be computed using a recursive equation as described in [3]. Considering all possible loss rates $(n-k)/n$, we can obtain a cumulative density function (CDF) for a particular k. The loss rate bound $p_b(\ell)$ is defined as the maximum loss rate under which most (say 90%) losses happen for a segment of length ℓ. The concept of loss rate bound is now illustrated with a numerical example.

Example 8 Prior work [7] suggests the average burst length is of the order of 100 ms, so we choose $b = 500$ ms as a worst-case example. Assume the average loss rate $p = 0.1$. Figure 4(a) shows the CDFs corresponding to different segment lengths. It is clear that for large segment length, the loss probability is clustered around the mean (0.1 in this example), but the distribution is more widely spread for smaller segment lengths. We choose loss rate bound for each ℓ such that 90% cases have loss rate below the bound. From Figure 4(a), we can see that $p_b = 0.31$ for segment length 100 symbols and $p_b = 0.166$ for segment length 1000. Clearly, $p_b(\ell) \to p$ as $\ell \to \infty$. Enumerating all possible values of ℓ, we can determine the mapping from segment length to loss rate bound. ■

The analysis for random loss extends easily to burst loss. Assume that we require smooth playout after an initial delay w. Similar to the random loss case, it is now desirable to receive $|s_i|/(1-p_b(t))$ symbols from segment s_i by time t at which segment s_{i-1} is fully consumed. With an initial delay w, segment s_{i-1} is consumed in time $t = w + |\wp(s_{i-1})|$, and the loss bound for this interval is just $p_b(w + |\wp(s_{i-1})|)$. Hence we set:

$$T(s_i) = \frac{1}{1 - p_b(w + |\wp(s_{i-1})|)} \times \frac{|s_i|}{w + |\wp(s_{i-1})|} \tag{11}$$

Example 9 The same calculation as in the previous example was done for burst loss, with average burst length $b = 500$ ms. The increased initial delay for these users as a result of factoring for burst losses in the network is tabulated below.

Class	**Initial Delay with burst loss**		
	No loss	**5% loss**	**10% loss**
DSL(1.5 mbps)	35 min	39 min	43.2 min
Cable Modem(4 mbps)	145 sec	188 sec	233 sec
Ethernet(10 mbps)	1/2 sec	1.4 sec	2.5 sec

Figures 4(b) through 4(d) show the performance of segmented broadcast tailored for different burst loss rates.

7. REDUCING LATENCY

The primary trade-off in on-demand streaming is between broadcast bandwidth and initial delay. This dictates that receivers with moderate bandwidth constraints have to tolerate relative long delays. For the 2-hour movie partitioned into 100 segments, the peak broadcast bandwidth is 5.6 mbps to ensure a delay of 30 seconds and 4.9 mbps for 60 seconds. Compared to the few seconds of delay typically experienced in unicast-based streaming applications, these delays are still relatively long.

Moreover, on-demanding streaming schemes broadcast earlier symbols more frequently than later ones to guarantee playback continuity. Thus, earlier symbols occupy a significant portion of the broadcast bandwidth. These symbols, on the other hand, often contain data of not much interest to receivers (such as previews, ads or cast information). Therefore, it is possible to effectively reduce bandwidth by broadcasting these initial symbols at a reduced quality (reduced bitrate). From another perspective, such a "fast start" option enables the use of the same broadcast bandwidth with a smaller startup latency, improving the user experience.

The *optimal fast start* problem studied in this section is to find a segmentation *and* corresponding per-segment bitrate such that maximum possible quality (on the average) is achieved with the new (smaller) startup delay while maintaining the same broadcast bandwidth.

7.1 Problem Formulation

In this section, we formalize the problem and derive an optimal solution. With no quality reduction and delay w, we know that the peak bandwidth requirement of an optimal segmented broadcast $B = \langle C, M, S, T\rangle$ is

$$|C| = |S|\left(\left\{1 + \frac{|M|}{w}\right\}^{\frac{1}{|S|}} - 1\right) \tag{12}$$

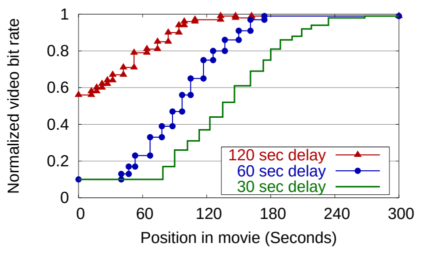
Now suppose we wish to reduce the initial delay by reducing the quality in the initial portion of the movie. We can effectively split the movie into two parts, a prefix M_1 with partially reduced quality and a suffix M_2 at full quality, and define two broadcasts $B_1 = \langle C_1, M_1, S_1, T_1\rangle$ and $B_2 = \langle C_2, M_2, S_2, T_2\rangle$. Based on our earlier results, the suffix is full quality and can be broadcast optimally with bandwidth

$$|C_2| = |S_2|\left(\left\{1 + \frac{|M_2|}{w + |M_1|}\right\}^{\frac{1}{|S_2|}} - 1\right)$$

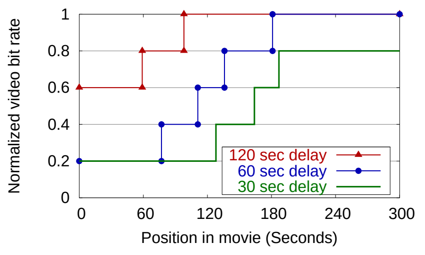
The bandwidth difference $|C| - |C_2|$ is used to transmit the prefix M_1 at a reduced quality. Now let r_i denote the normalized bitrate of segment $s_i \in S_1$. Also define w_f as the delay with fast startup. Then the broadcast bandwidth of all $|S_1|$ segments must satisfy:

$$\sum_{i=1}^{|S_1|} \frac{r_i |s_i|}{w_f + |\wp(s_{i-1})|} \le |C_1| = |C| - |C_2| \tag{13}$$

Now the problem is to find an optimal segmentation S_1 on M_1 and also a bitrate assignment $\{r_i : i = 1 \text{ to } |S_1|\}$ such that maximum possible quality is achieved within constraint (13). To quantitatively measure quality, we adopt a suitable rate-distortion (R-D)



(a) 100 discrete rate levels (R = 100)



(b) 5 discrete rate levels (R = 5)

Figure 5: **Fast Startup with Reduced Quality** (While (a) can be achieved using scalable codecs, (b) can be easily implemented using separately coded streams of different bitrates for different segments.)

function as described in [9, 11]. Maximizing quality is equivalent to minimizing distortion. As an idealized example, we use a simplified R-D function $D(r) = 2^{-r}$, where r and D are rate and distortion, respectively. Thus, we summarize the problem as follows:

$$\min \sum_{i=1}^{|S_1|} \{|s_i| D(r_i) + \sigma(r_i - r_{i-1})^2\}, \text{ subject to (13)} \qquad (14)$$

Note that the second term in the objective function is added to ensure smooth bitrate transition between segments, σ being a tunable value. If the bitrate is discretized to take on integer values in $[1 : R]$ and bandwidth in $[1 : B]$, then this problem has a simple dynamic programming solution with complexity $O(BR|M_1|^2|S_1|)$. Since $|M_1|$ has a quadratic effect on the complexity, we can speed up things a little by defining symbols to be groups of frames (say, a second's worth), with negligible effect on optimality.

Example 10 We use the usual 2-hour movie in our simulation to demonstrate the performance of the proposed fast start scheme. For this example, we take the case of the cable modem user with 4 mbps access speed. With 100 segments, the minimal initial delay is about 145 seconds without quality reduction. We then allow a reduced quality in the first five minutes ($|M_1| = 900$), with $S_1 = 25$. Figure 5 shows the results of rate (quality) evolution with different fast startup delays. Without changing broadcast bandwidth, we can see that the startup delay can be reduced to as low as 30 seconds if we sacrifice quality moderately. ■

8. CONCLUSIONS

The "digital fountain" approach provides a powerful abstraction for loss-resilient bulk data distribution in which packet transmissions act like drops of water: any sufficient combination of unique erasure-coded packets suffices to recover the original data in its entirety. However, for on-demand media streaming, a "digital straw" abstraction is preferable in which the original data can be consumed progressively even as further transmissions arrive for later consumption.

Building on the theoretically optimal but practically infeasible solution in [16], we advance a scheme to implement a "digital straw". Our solution is based on a combination of optimal segmentation into a few segments and efficient priority encoding based on the recently discovered *rateless codes*. The proposed scheme allows the continuous trade-off of encoding/decoding complexity against bandwidth optimality.

We then study the applicability of our solution to 1) multi-layered broadcast, to simultaneously support a heterogeneous client population 2) progressive pruning, to reduce the reception overhead inherent to asynchronous broadcast schemes, 3) loss-adaptive broadcasting, where the segmentation and transmission is tailored for some maximum loss rate, and 4) quality adaptation for reducing startup latency for clients.

Our on-going work focuses on 1) incorporating flexible quality adaptation, so that clients can continuously trade-off quality against startup latency, 2) dynamic quality and rate control for reacting to network congestion, and 3) a prototype implementation capable of streaming several movies on a LAN-based media streaming testbed.

9. ACKNOWLEDGMENTS

We are grateful to Dr. Marcel Waldvogel for stimulating discussions, and to the referees for their in-depth reviews.

10. REFERENCES

[1] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan. Priority Encoded Transmission. In *Proceedings of IEEE FOCS 1994*, pages 604–612, 1994.

[2] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. In *Proceedings of ACM SIGCOMM 1999*, pages 56–67, Sept. 1999.

[3] C. Huang and L. Xu. Efficient FEC Codes for Data Loss Recovery. Technical report, Washington University in St. Louis, June 2004. Available at http://www.nisl.wustl.edu.

[4] L. Engebretsen and M. Sudan. Harmonic broadcasting is optimal. In *Proceedings of SODA 2002*, pages 431–432, 2002.

[5] A. Hu. Video-on-Demand Broadcasting Protocols: a Comprehensive Study. In *Proceedings of IEEE INFOCOM 2001*, pages 508–517, 2001.

[6] J.-C. Bolot, S. Fosse-Parisis, and D. Towdley. Adaptive FEC-Based Error Control for Internet Telephony. In *Proceedings of IEEE INFOCOM 1999*, 1999.

[7] J. G. Apostolopoulos. Reliable Video Communication Over Lossy Packet Networks Using Multiple State Encoding and Path Diversity. In *Proceedings of MMCN 2002*, 2002.

[8] R. Janakiraman and L. Xu. Layered Priority-encoded Transmission for Video Streaming to Heterogeneous Clients. In *Proceedings of IEEE ISIT 2004*, July 2004.

[9] M. Dai, and D. Loguinov. Analysis of Rate-Distortion Functions and Congestion Control in Scalable Internet Video Streaming. In *Proceedings of ACM NOSSDAV 2003*, June 2003.

[10] A. Mahanti, D. L. Eager, M. K. Vernon, and D. Sundaram-Stukel. Scalable on-demand media streaming with packet loss recovery. In *Proceedings of ACM SIGCOMM 2001*, pages 97–108, 2001.

[11] N. Jayant, and P. Noll. *Digital Coding of Waveforms*. Prentice Hall, Englewood Cliffs, NJ, 1984.

[12] J.-F. Pâris, S. W. Carter, and D. D. E. Long. A Low Bandwidth Broadcasting Protocol for Video on Demand. In *Proceedings of IC3N 1998*, pages 690–697, Oct. 1998.

[13] V. Pless. *Introduction to the Theory of Error-Correcting Codes*. Wiley-Interscience, 1998.

[14] M. A. Shokrollahi. Raptor codes. In *Proceedings of IEEE ISIT 2004*, July 2004.

[15] V. Paxson. End-to-End Internet Packet Dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277 – 292, 1999.

[16] L. Xu. Resource-Efficient Delivery of On-Demand Streaming Data Using UEP Codes. *IEEE Transactions on Communications*, 51(1):63–71, Jan. 2003.