Peer-Driven Video Streaming: Multiple Descriptions versus Layering

Yanming Shen†, Zhengye Liu†, Shivendra S. Panwar†, Keith W. Ross‡, Yao Wang†
Department of Electrical and Computer Engineering†
Department of Computer and Information Science‡
Polytechnic University

Abstract— **We propose a peer-driven video streaming system that provides a reliable and high-performance video on-demand service. We encode each video into multiple low bit-rate sub-streams and distribute copies of the sub-streams from the participating peers. Each client receives the constituent multiple sub-streams from different serving peers, and combines them into the original video stream. When a server peer disconnects, the system attempts to find a replacement peer that has a copy of the missing sub-stream and has sufficient available uplink bandwidth. We consider using multiple description coding and layered coding to generate the multiple sub-streams. We develop a traffic theory for peer-driven video streaming, and examine the system performance analytically and via simulation. We show that peer-driven video distribution can provide excellent performance even if peers disconnect from the network frequently. We also compare the performance of layered coding with multiple description coding.**

I. Introduction

Today, much of the streamed video in the Internet is managed and distributed by Content Distribution Networks (CDNs) such as Akamai. Recall that a CDN deploys dedicated servers for storing and distributing content to clients. When a user requests a video, the CDN redirects the client to one or more of its dedicated servers, which stream the stored video to the client. CDNs have significant infrastructure costs, not only for the dedicated servers they deploy, but also for the access bandwidth they consume.

In this paper we propose and study **peer-driven video streaming**, which is an alternative architecture to CDNs for video distribution. In peer-driven streaming, peers – that is, nodes owned and operated by ordinary users – are commissioned to do the streaming. As an example, a news company (such as CNN.com) may directly commission thousands of peers to act as servers and stream video to clients on the behalf of the news company. As a second example, there may be a third-party company that sells video distribution to numerous publishers but delegates the actual streaming to commissioned peers. In this second example, the third-party company has a role similar to a CDN; the primary difference is that a CDN owns and operates the streaming servers, whereas here the third-party distribution company outsources the video streaming to the hired peers. In both examples, the hired peers might receive micropayments for their streaming services. Because there is an abundant supply of potential server peers with excess unused bandwidth and storage, peer-driven architectures should have fixed and recurring costs that are significantly less than those of CDNs.

Although peer-driven video streaming potentially provides huge cost savings over CDNs, it presents a new set of design challenges. First, in a peer-driven video streaming system, the server peers are not permanently connected to the Internet, as with CDN servers, but are instead intermittently connected. Second, although some of these server peers might be connected via high-speed Ethernet access, many peers would have DSL or cable access with relatively low upstream bandwidth. These two salient characteristics of the server peers need to be carefully taken into consideration when designing a peer-driven video streaming system. Other issues for peer-driven video streaming, which are beyond the scope of the current paper, include payment systems, dispute resolution, digital rights management and security.

One strategy for peer-driven streaming is, for each client request, to stream the complete video from one of the server peers to the client. The problem with this "one-server" approach is twofold. First, because of limited upstream access rates, many peers would be incapable of streaming the video at its full rate. Second, when the server disconnects, the client will lose service until a new server peer can be found and begin streaming. It may be possible to partially mitigate this last problem by prefetching video into a client buffer before and during playback; however, not all clients have the potential for significant buffering (e.g., many hand-held devices), and a server's limited upstream bandwidth and a client's limited downstream bandwidth may hinder or fully preclude prefetching.

We introduce and explore a promising sub-stream design for peer-driven video streaming. In this design, each video is encoded into multiple sub-streams, and copies of the sub-streams are stored in the server peers. When a client wants to see a video, it receives multiple sub-streams, each from a different server peer. As the sub-streams arrive to the client, the client combines the sub-streams, decodes and displays the video. Because each sub-stream typically has a rate that is a fraction of the combined stream, the server peers can more easily accommodate sub-streams with their limited upstream bandwidth. Furthermore, if the system is designed properly, the loss of one stream due to a server failure or disconnect will not seriously impair video quality while waiting for a replacement sub-stream.

In this paper, we examine two sub-stream approaches: **Multiple Description (MD) video system**, for which the sub-streams (called descriptions) have equal importance; and **layered video system**, for which the sub-streams (called layers) have varying importance. For both of these approaches, we examine coding, distribution and substream placement. This paper makes several contributions, including:

- It proposes peer-driven video streaming with multiple sub-streams as an alternative to CDN video streaming.
- For MD video coding, we use MD-FEC for creating the substreams. We develop a traffic theory for MD video coding. In particular, we formulate and solve the problem of finding the optimal video parameters (number of descriptions and their rates). We prove (for a somewhat simplified model) that it is optimal to deploy and stream the maximum number of descriptions for each video. We also develop a dynamic programming scheme to determine the optimal number of sub-stream replicas for each video, and propose simple mechanisms for placing the replicas in the server, for selecting servers, and for admission control.
- For layered coding, we use Fine-Grained Scalable (FGS) video to create the layers, and we again consider the problems of video coding, optimal replication, sub-stream placement, server selection and admission control.
- We simulate peer-driven video streaming for both MD and layered coding, using the optimal coding and optimal replication strategies. In our simulations, we use rate distortion functions derived from real videos. We compare the MD and layered approaches, gaining insight into which scheme provides better performance for different peer behaviors and traffic conditions.

We briefly mention that as an alternative to streaming video, a user can download a video and watch it at his or her convenience. Many P2P file-sharing architectures can be employed for video downloading, including BitTorrent, FastTrack, Gnutella and the DHT architectures. However, there will be a continued demand for video streaming solutions. Users who either wish to browse videos or see a video instantly prefer streaming. Furthermore, many hand-held devices will continue to have modest storage capabilities and will be incapable of downloading entire videos.

This paper is organized as follows. In Section II, we describe the overall system design. In Section III and Section IV, we discuss the MD and layered coding systems respectively. Section V compares the performance of the MD and layered designs. The related work is described in Section VI and Section VII concludes this paper.

II. Overview of System Design

Streaming is performed from a pool of server peers. Server peers are typically non-NATed and have been "hired" to stream video to client peers. Although peers in the pool connect and disconnect from the Internet, the total number of peers in the pool (either connected or disconnected) is assumed to be approximately constant over the time frame of interest (say, one day). Client peers, which watch the videos, are arbitrary Internet-connected hosts, some of which may be handheld wireless devices. Client peers may be non-NATed or NATed.

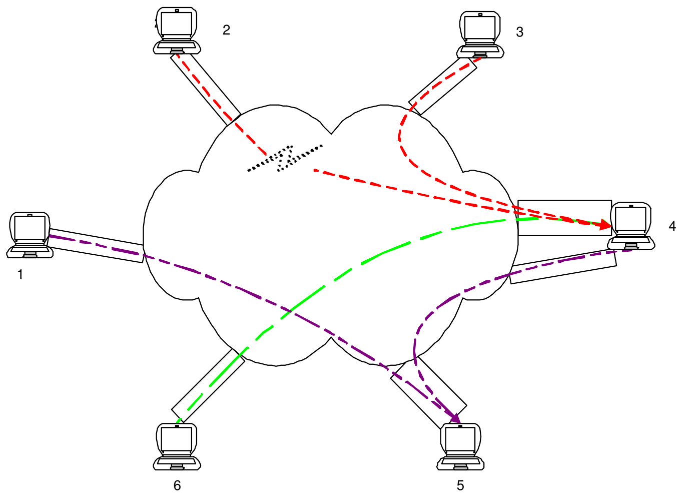


Fig. 1. In this example, peers 4 and 5 are each receiving a video. Initially, peer 4 receives sub-streams from peers 2 and 3, and peer 5 receives from peers 1 and 4. Then peer 2 disconnects, and the system recovers by assigning peer 6 as a replacement. While locating and establishing a replacement, visual quality at peer 4 is degraded. We use a fat pipe to indicate the downlink of each peer, and a thin pipe to illustrate the uplink of each peer. Generally, a peer can function as a client only, a server only, or simultaneously as a server and a client (e.g., peer 4).

Our approach to peer-driven video streaming is to encode each video into multiple sub-streams and store copies of the sub-streams in the pool of server peers. When a client wants to see a video, multiple server peers are selected, each currently connected to the Internet and having sufficient available upstream bandwidth. Each of these server peers sends a different sub-stream to the client. After a short initial delay, the client combines, decodes, and displays the video as the multiple sub-streams are being delivered. When a server peer disconnects in the middle of a streaming session, the client peer loses one sub-stream until a replacement peer begins to send the missing sub-stream. Figure 1 illustrates the system architecture.

In this paper, we examine two sub-stream approaches: MD coding and layered video coding. With MD coding, all sub-streams have equal importance; if a client loses any one of the many sub-streams, then video quality should not severely degrade before a replacement stream arrives at the client. With layered coding, the sub-streams have varying importance; the layered system must be designed so that the more important layers are delivered with high probability. An MD system design will be simpler but, in the absence of packet loss, its source codec requires a higher bit rate to reach the same target decoding quality as with a layered system. Thus, in the absence of packet loss, the layered approach is more efficient in utilizing the peer resources, i.e., it can serve more requests for the same target decoding quality. But this higher efficiency is obtained at higher design and operational cost.

The design of a peer-driven video streaming system has four interacting components: video coding, sub-stream replication and placement, server selection, and admission control. Each of these components has important design issues:

- **Video coding:** Should the system employ MD coding or layering? What MD codec or layer codec should be used? How many sub-streams should a video have? What should be the rates of each of the sub-streams?
- **Sub-stream replication and placement:** Given that each server peer has limited storage and upstream bandwidth, how should the copies of the sub-streams be distributed over the pool of server peers? How many copies of each sub-stream should be stored in the pool?
- **Server selection:** When establishing a new streaming session, which server peers should be selected to stream the sub-streams? When a client loses a sub-stream due to a server disconnect, which peer should be chosen as a replacement peer?
- **Admission control:** When a client makes a request, should we admit the request and establish the streaming session? Or will an additional streaming session significantly compromise the quality of the existing streaming sessions?

In the subsequent sections, we examine all four of these components, for both MD and layered systems.

An additional component of a peer-driven video distribution is sub-stream search, both during initial set-up and also in response to server peer disconnects. Sub-stream search can be done in a variety of different ways, depending on the underlying architecture of the peer management system. For a decentralized architecture, distributed hash tables (DHTs) can be used to locate sub-streams [1]–[4]. The sub-stream search problem is orthogonal to the coding/streaming/placement problem and is beyond the scope of the current paper.

We also mention that one possible design strategy is to use large client buffers, and prefetch during playback - that is, transmit the sub-streams at a rate that is higher than the encoded playback rate. In this manner, when a substream is lost, the client may have a sufficient "reservoir" for that sub-stream, so that playback continues without any quality degradation [5], [6]. The disadvantages of this strategy include: (i) the need for large client buffers, which may not be feasible for many handheld devices; and (ii) the need to transmit at rates higher than the encoded rate, which can produce wasted traffic when the user ceases to watch the video before completion or jumps forward into the video. We feel that strategies with and without client prefetching during playback merit investigation. In this paper we only allow for a small amount of prefetching before the start of playback and no prefetching during playback. We shall consider client prefetching in greater detail in a subsequent paper.

A. Overall Design Criterion

In a peer-driven video distribution system, users request videos for immediate viewing. After a user makes such a request, the system attempts to set up a *session*, which consists of multiple sub-streams sent from different server peers to the user's device. From the user's perspective, there are two critical performance measures:

- **Acceptance probability:** the likelihood the system will locate the necessary sub-streams and establish a session.
- **Video quality:** the visual quality of the session, from start to finish.

Because each admitted session consumes peer upstream bandwidth resources, and because the number of commissioned server peers is roughly constant over the time frame of interest (say, 24 hours), as more sessions are admitted into the system it becomes increasingly more difficult to provide each on-going session with sufficient sub-streams for adequate visual quality. Furthermore, once a server's upstream pipe becomes full, new session requests no longer have access to any of the video sub-streams stored at that server. These simple observations lead to our high-level design goal:

> *Design in an integrated fashion the components of the system – the coding, sub-stream placement, server assignment, and admission control – to maximize the acceptance probability subject to the constraint that each on-going video session meets a target video quality constraint.*

Currently, the end-to-end bandwidth bottleneck is in access and not in the Internet core [7], [8]. We expect this trend to continue for the foreseeable future. Furthermore, in most residential broadband connections today (including cable modem and ADSL), the upstream rate is significantly less than the downstream rate. Thus, it is not unreasonable to assume that the bandwidth bottleneck between server and client is the server's upload rate.

Throughout this paper we use N to denote the number of peers in the server pool and M to denote the number of sub-streams in a video.

III. MD System

Recall that peer-driven video streaming has four interacting components: video coding, video placement, server selection, and admission control. We examine each of these components for MD coding in this section.

A. Video Coding

MD video coding has become an active research area in the past few years. Although various coders have been proposed [9]–[11], most coders generate a fairly small number of descriptions, with $M = 2$ being the most common case. However, to fully explore the load balancing and error-resilience benefit in peer-driven video streaming, a larger M is desired.

Instead of designing the source encoder to yield multiple descriptions directly, one can apply unequal FEC to different parts of a scalable bitstream, commonly known as MD-FEC [12], [13]. One of the advantages of MD-FEC is that it can be used to generate any number of sub-streams M. For peer-driven video distribution with MD coding, we use MD-FEC throughout this paper.

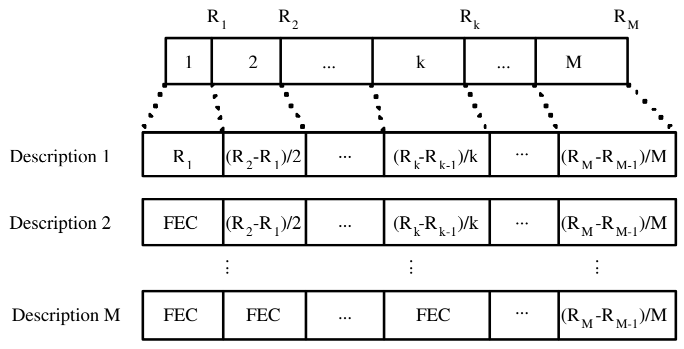


Fig. 2. MD-FEC algorithm.

We now briefly explain MD-FEC coding. As shown in Figure 2, the original scalable bitstream from each group of frames (GoF) is partitioned into M layers, with layer k allocated $R_k - R_{k-1}$ bits. For $k = 1, \dots, M$, the kth layer is further divided into k equal-length groups. An (M, k) Reed-Solomon (RS) code is then applied to k groups to yield M groups. Description m is formed by packing bits from group m from all layers. At the receiver, if any m of the M descriptions are received, the decoder can recover the first m layers of the original bitstream. The rate and receiver distortion for MD-FEC is controlled by varying the layer partition $(R_1, R_2, ..., R_M)$. (Throughout the paper, we use mean square error (MSE) as the distortion measure.)

In this paper, we will use M to denote the number of descriptions and r the bit rate of each description; thus the total rate of a video with all descriptions is $R = Mr$. We now briefly summarize the problem of optimally determining the layer partition $(R_1, R_2, ..., R_M)$ for a given M, r and the description loss distribution [14]–[16]. To this end, suppose that a client is receiving the M descriptions from M different server peers, and suppose that these server peers can fail or disconnect from the network. Let P_m denote the probability of receiving m out of M descriptions, and let $\mathbf{P}$ denote the probability mass function P_m, $m = 0, \dots, M$. Let $D_m(R_1, \dots, R_M)$, $m = 1, \dots, M$, denote the distortion when m descriptions are received for partition $(R_1, R_2, ..., R_M)$. Then the expected distortion of the received video is

$$D(R_1, \dots, R_M) = \sum_{m=0}^{M} P_m D_m(R_1, \dots, R_M). \qquad (1)$$

Using the above notation, the MD-FEC optimization problem can be formulated for given M, r, and $\mathbf{P}$ as follows: determine the optimal layer partition $(R_1, R_2, ..., R_M)$ for

$$\begin{array}{ll} \min & D(R_1, \dots, R_M) \\ s.t. \sum_{m=1}^{M} \frac{(R_m - R_{m-1})}{m} M = \sum_{m=1}^{M} R_m \frac{M}{m(m+1)} = Mr \\ R_0 = 0. \end{array} \qquad (2)$$

This is a non-linear optimal resource allocation problem. Fast algorithms for solving this optimization problem have been presented in [14]–[16]. Throughout this paper we suppose that the multiple descriptions are created with MD-FEC, and that the partition $(R_1, R_2, ..., R_M)$ for a given M, r and $\mathbf{P}$ is obtained by solving the optimization problem (2).

We now proceed to develop a traffic theory for peer-driven video streaming using descriptions created with MD-FEC.

1) Optimal Coding Parameters: For each video, we would like to to determine the optimal coding parameters, that is, the optimal values of M and r. Unfortunately, we cannot solve this problem in isolation, but must instead solve this problem in the context of our overall design criterion (see Section II-A).

To this end, we measure video quality in terms of the distortion between the original video and the reconstructed video at the client, and translate the target quality into a target distortion $D_{\max}$. Let Q denote the number of ongoing video sessions (with each session consisting of multiple sub-streams). Let $D(M, r, \mathbf{P})$ denote the optimal expected distortion for a given number of descriptions M, a given description rate r, and given probability mass function $\mathbf{P}$. This probability in general depends on the rate r, the number of on-going sessions Q and the video substream placement over peers, collectively denoted by ϕ. To make this dependency explicit, we use the notation $P(m, M; r, Q, \phi)$ instead of P_m and write $D(M, r, Q, \phi)$ for $D(M, r, \mathbf{P})$. Recall that our overall design problem is to maximize the number of sessions subject to the constraint that on-going video sessions meets the target quality constraint. This problem can now be restated as find M, r, Q and ϕ that solves:

$$\begin{array}{ll} \max & Q \\ s.t. & D(M, r, Q, \phi) \le D_{\max} \end{array} \qquad (3)$$

The challenge in solving this problem lies in the fact that the distortion is affected by the video encoding parameters M, r and video sub-stream placement ϕ. To shed some insight on this complex problem, we now formulate and solve a simplified and idealized model. After gaining some insights, we will then return to the original problem.

Consider a homogeneous system with N peers, each with B_u bps of uplink bandwidth. Each peer is connected with probability μ and peer connectivity is independent from peer to peer. Each peer can simultaneously send $\gamma = B_u/r$ descriptions (for this illustrative analysis, we ignore the integer constraint). In this simplified model, we do not consider the storage limitation at the peers and assume that all descriptions of all videos are stored in every peer. We can therefore remove ϕ from $P(m, M; r, Q, \phi)$ and $D(M, r, Q, \phi)$ in this section. With these simplifications, the problem (3) reduces to choosing M and r to maximize Q while meeting the distortion constraint $D(M, r, Q) \le D_{\max}$. The solution of this problem determines the optimal coding parameters *along with* the optimal admission policy.

To solve this problem, we need to calculate the average distortion $D(M, r, Q)$ for a given M, r, and Q. To this end, we first calculate $P(m, M; r, Q)$ and then apply the optimization problem (2) to get the optimal average distortion $D(M, r, Q)$. So we now turn our attention to calculating $P(m, M; r, Q)$.

Let X be a random variable denoting the total number of peers in the network that are up at any time; X is binomial with parameters N and μ. The total number of descriptions that the system can send at any time is γX. For a given X, the number of descriptions that can be sent to each session is $m = \gamma X/Q$, with m ranging from 0 to $m_{\max} = \gamma N/Q$. In other words, m descriptions are available to each of the Q sessions when the number of connected peers is $X = k = Qm/\gamma$ with $\gamma = B_u/r$. This occurs with probability

$$P(X = k) = \binom{N}{k}\mu^k(1-\mu)^{N-k},\ k = 0, \ldots, N.$$

The number of descriptions available to any one session is at most M. When $m < M$, we need to use all the available servers to send m descriptions to each client. When $m \geq M$, we only need to send M descriptions to each client by selecting M out of possible m servers. Therefore, the probability of having m out of M descriptions available for a client is

$$P(m, M; r, Q) = \begin{cases} P(X = Qm/\gamma), \\ m = 0, 1, \ldots, M-1 \\ \sum_{l=M}^{\gamma N/Q} P(X = Ql/\gamma), \\ m = M \end{cases} \quad (4)$$

In summary, under the assumptions of this subsection, for each given M, r, and Q we can determine if a given parameter triple (M, r, Q) is *feasible* as follows:

1) Calculate $P(m, M; r, Q)$, $m = 1, \ldots, M$, using (4).
2) From $P(m, M; r, Q)$, $m = 1, \ldots, M$, calculate the optimal MD-FEC partition $(R_1, \ldots, R_M)$ and the corresponding average distortion $D(M, r, Q)$ from the sub-optimization problem (2).
3) If $D(M, r, Q) \leq D_{\max}$, then the triple (M, r, Q) is feasible.

To find the maximum number of sessions, we can search over all triples (M, r, Q) to find the feasible triple with the largest value of Q. Because the number of such triples is very large, we now show that it is optimal to set $M = M_{\max}$, where $M_{\max}$ is the largest number of descriptions permitted. Define $Q^*(M, r)$ as the maximum number of sessions that the network can support to meet the distortion criterion D_{max} given the video parameters (M, r).

Theorem 1: For any given r, $Q^*(M, r)$ is non-decreasing with M.

Proof: Fix r and M. Let $Q_1^* := Q^*(M, r)$ and $Q_2^* := Q^*(M+1, r)$. It suffices to show that $Q_2^* \geq Q_1^*$.

We first prove that the optimized average distortion $D(M, r, Q_1^*)$ is greater than or equal to the optimized average distortion $D(M+1, r, Q_1^*)$. For M descriptions, the optimal MD-FEC structure is given in the Figure 3(a). Each row corresponds to one description. The white rectangles represent the amount of source bits. The shaded rectangles represent the amount of channel bits. We construct a MD-FEC structure for $M+1$ descriptions in Figure 3(b) as follows:

1) The first M rows in Figure 3(a) are copied to the first M rows in Figure 3(b);
2) The last row in Figure 3(b) is filled with channel bits only.

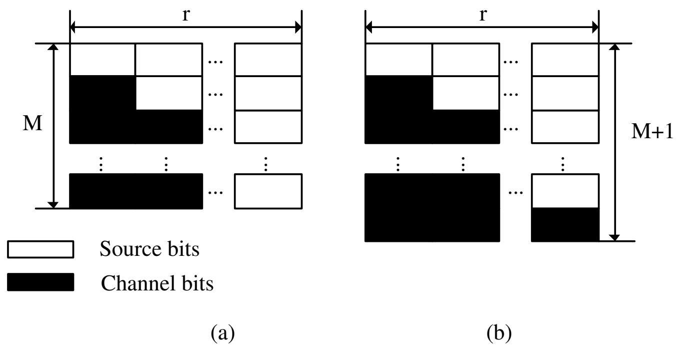


Fig. 3. MD-FEC bit allocation

Based on this construction, the distortion of receiving m descriptions out of M in Figure 3(a) is equal to the distortion of receiving m descriptions out of $M+1$ in Figure 3(b) for $m \leq M$. Let $\widetilde{D}(m, M+1, r)$ denote the distortion when receiving m descriptions for this construction (which is not optimized). Thus, we have

$$\widetilde{D}(m, M+1, r) = D(m, M, r),\ m = 1, \ldots, M. \quad (5)$$

Also, from Figure 3(b), we have

$$\widetilde{D}(M+1, M+1, r) = \widetilde{D}(M, M+1, r). \quad (6)$$

The probability in equation (4) implies:

$$\begin{cases} P(m, M; r, Q_1^*) = P(m, M+1; r, Q_1^*), \\ m = 1, \ldots, M \\ P(M, M; r, Q_1^*) = P(M, M+1; r, Q_1^*) \\ +P(M+1, M+1; r, Q_1^*). \end{cases} \quad (7)$$

The average distortion for $M+1$ descriptions, using relationships (5), (6) and (7), is given by:

$$\begin{aligned} &\widetilde{D}(M+1, r, Q_1^*) \\ &= \sum_{m=1}^{M+1} P(m, M+1; r, Q_1^*)\widetilde{D}(m, M+1, r) \\ &= \sum_{m=1}^{M-1} P(m, M+1; r, Q_1^*)\widetilde{D}(m, M+1, r) + \\ &[P(M, M+1; r, Q_1^*) + P(M+1, M+1; r, Q_1^*)] \\ &\cdot\widetilde{D}(M, M+1, r) \\ &= \sum_{m=1}^{M-1} P(m, M; r, Q_1^*)D(m, M, r) \\ &+P(M, M; r, Q_1^*)D(M, M, r) \\ &= \sum_{m=1}^{M} P(m, M; r, Q_1^*)D(m, M, r) \\ &= D(M, r, Q_1^*). \end{aligned}$$

Therefore, the optimized average distortion for M descriptions is equal to the average distortion for $M+1$ descriptions using the above construction.

The optimal design of MD-FEC for $M+1$ descriptions has an average distortion no greater than the design just constructed for $M+1$. Therefore,

$$D(M+1, r, Q_1^*) \le D(M, r, Q_1^*) \le D_{max}. \tag{8}$$

Finally, by definition of Q_2^* we have

$$D(M+1, r, Q) \le D_{max} \quad \text{iff} \quad Q \le Q_2^*. \tag{9}$$

Thus, from (8) and (9), we have $Q_1^* \le Q_2^*$. ■

It follows from the theorem that we should use the highest possible value of M. In a real network, since we stream M descriptions from different peers, the total number of descriptions should be less than the total number of peers in the network. Furthermore, as M becomes larger, we need to set up more connections for each streaming session, which consumes more network and nodal computation resources. Therefore, within system constraints, there is a practical upper limit on M.

In summary, in the simplified model, we set M to its maximum possible value for each video. We then find the optimal sub-stream rate r and the maximum number of sessions Q^* by searching over tuples (r, Q) as described earlier.

2) Average Distortion Analysis for a Gaussian Source: In this subsection, to gain further insight into fundamental design issues, we consider the performance limit for an i.i.d Gaussian source with variance σ^2. It is well known that this source has the following rate-distortion (R-D) function [17]:

$$D_m(R_1, \ldots, R_M) = D(R_m) = \sigma^2 \cdot 2^{-2R_m}. \tag{10}$$

For simplicity, let P_m denote $P(m, M; r, Q)$. Then the average distortion of the Gaussian source is :

$$\begin{aligned} D(R_1, \ldots, R_M) &= \sum_{m=0}^{M} P_m \sigma^2 \cdot 2^{-2R_m} \\ &= P_0 \sigma^2 + \sum_{m=1}^{M} P_m \sigma^2 \cdot 2^{-2R_m}. \end{aligned}$$

Recall the MD-FEC optimization problem in equation (2). The optimal solution can be found using the theory of Lagrange Multipliers. Introducing the Lagrange multiplier λ, we obtain

$$\begin{aligned} L(R_1, \ldots, R_M, \lambda) &= P_0 \sigma^2 + \sum_{m=1}^{M} P_m D(R_m) \\ &\quad + \lambda\Big(\sum_{m=1}^{M} \alpha_m R_m - Mr\Big), \end{aligned}$$

where

$$\alpha_m = \frac{M}{m(m+1)}, \qquad m = 1, \ldots, M-1$$

and

$$\alpha_M = 1.$$

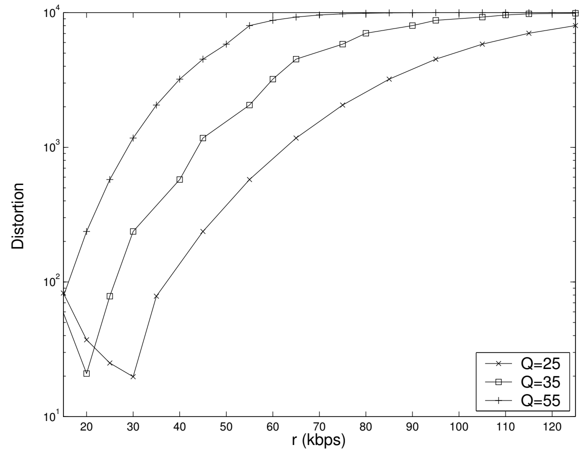


Fig. 4. Average distortion vs. r with fixed M=32 for a Gaussian source

At optimality, the partial derivatives of the Lagrangian function with respect to R_m, $m = 1, \ldots, M$, and λ equal zero. This yields the optimal bit allocation:

$$\frac{P_m}{\alpha_m} \frac{dD(R_m)}{dR_m} + \lambda = 0. \tag{11}$$

Substituting (10) in (11), we can explicitly determine the optimal values for MD-FEC partition:

$$R_m = -\log_2\left(\frac{\lambda \beta_m}{c}\right)/2, \tag{12}$$

where

$$\lambda = \left[\frac{2^{-2Mr}}{\prod_{m=1}^{M} (\frac{\beta_m}{c})^{\alpha_m}}\right]^{\frac{1}{\sum_{m=1}^{M} \alpha_m}},$$

and $\beta_m = \alpha_m / P_m$, $c = 2\sigma^2 \ln 2$.

Based on (12), we can get an explicit expression for (10), then calculate the average distortion as follows:

$$D(M, r, Q) = \sigma^2 \left[P_0 + \frac{\lambda}{c} \sum_{m=1}^{M} \alpha_m\right]. \tag{13}$$

From (13), the average distortion can be calculated for a given (M, r, Q). As an example, consider a network with $N = 100$ peers. Suppose the uplink bandwidth at each peer is $B_u = 250$ kbps and the peer connectivity probability is $\mu = 0.9$. We investigate how the average distortion changes as a function of Q and r for a given M. (We have already proved that the average distortion decreases with increasing M.) We search over rates r from 15 kbps to half of the peer uplink bandwidth 125 kbps, and the number of sessions Q from 10 to 100.

From Figure 4, we see that the average distortion is minimized at an intermediate rate r. (The actual optimal rate depends on the number of sessions Q.) This is because the rate r has two opposite effects: increasing r will increase the quality improvement due to each received description, but it will also decrease the number of deliverable descriptions (Recall that $m = XB_u/Qr$). The optimal r depends on the actual R-D curve of the source. Typically this occurs at the point when the R-D curve is in the flat tail region.

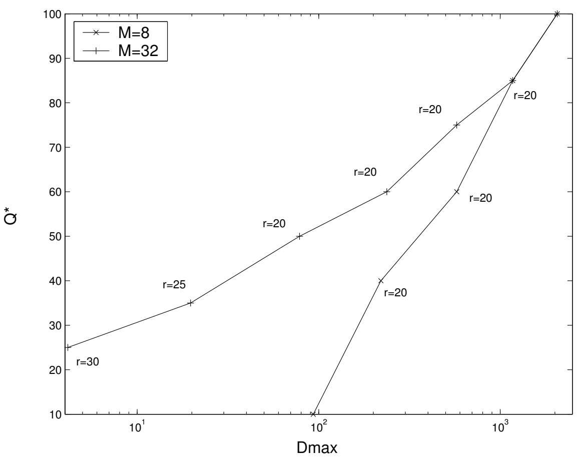


Fig. 5. D_{max} vs. Q^* for a Gaussian source

Recall that Q^* denotes the maximum number of sessions for a given target distortion D_{max}. For given D_{max} and M, we can search the space of r and Q together to find the maximum Q. In Figure (5) we plot Q^* as a function of the distortion constraint. As we would expect, Q^* increases with D_{max}. For a given D_{max}, when M is larger, the Q^* is also greater. This is consistent with the theorem we proved earlier. In Figure 5, when $D_{max} = 10^2$, for $M = 8$, Q^* is around 10, but for $M = 32$, Q^* is around 55. Thus the number of descriptions, M, can have a dramatic impact on performance. However, when D_{max} is greater than about 10^3, the curves for different M converge.

B. Video Placement

Video placement is another important component for peer-driven streaming system. Given that each peer has limited storage, one needs to decide how many copies of a sub-stream should be made and how to distribute these copies. Intuitively, the more popular videos should be replicated more aggressively. Next, we present an optimization procedure to find the optimal number of copies for a video. Then, we propose a natural heuristic to place the sub-stream copies in the server peers.

Assume a homogeneous system with N server peers, each with uplink bandwidth B_u and connectivity probability μ. Now, each peer also has a limited storage capacity S. There are J videos in the network and each video is encoded into M descriptions. Video j has a size b_j (thus a description has a size of $r_j = b_j/M$) and a demand rate λ_j. Assuming that the server selection and placement policy is such that the load on each server is roughly the same, the load on a server peer is such that each peer is evenly loaded, thus the average request rate for a peer is

$$\bar{\lambda} = \frac{\sum_j \lambda_j}{N}$$

Given the request rate $\bar{\lambda}$ for a peer, peer uplink bandwidth B_u and the rate of a sub-stream r, we can model a server peer with an Erlang-B model, where the number of channels in the model is B_u/r. From this model, we can calculate the probability that no channels are available at the server, denoted by P_{block}. Let $\hat{\mu}$ denote the probability that a peer is connected and also has sufficient uplink bandwidth, then $\hat{\mu} = \mu(1 - P_{block})$.

Suppose C_j copies of each description of video j are stored on different server peers. The probability p_j that a server is available with a specific description for video j is

$$p_j = 1 - (1 - \hat{\mu})^{C_j}.$$

The probability that m of M descriptions for video j are available is

$$P_{jm} = \binom{M}{m} p_j^m (1 - p_j)^{(M-m)}.$$

Let D_j denote the average distortion for video j, then

$$D_j = \sum_{m=0}^{M} P_{jm} D_{jm},$$

where D_{jm} is the distortion of video j when receiving only m descriptions. Our objective is to determine $C_1, \ldots, C_J$ in order to minimize the average distortion, that is, solve the following optimization problem:

$$\begin{aligned} \min \quad & \sum_{j=1}^{J} \lambda_j D_j \\ s.t. \quad & \sum_{j=1}^{J} b_j C_j \leq NS \end{aligned} \tag{14}$$

We can solve this optimization problem using dynamic programming. Let $f_i(s) = \sum_{j=1}^{i} \lambda_j D_j$, where s is the total storage capacity allocated to video 1 to i. Let $f_i^*(s)$ denote the minimum value of $f_i(s)$. Then we have

$$f_i^*(s) = \min_{0 \leq C_i \leq \frac{s}{b_i}} [\lambda_i D_i + f_{i-1}^*(s - b_i C_i)].$$

The detailed procedure is as follows:

1) At step i, for all values of C_i within $[0, s/b_i]$, calculate D_i.
2) Calculate $\lambda_i D_i + f_{i-1}^*(s - b_i C_i)$, where $f_{i-1}^*(s - b_i C_i)$ is obtained in the previous step ($f_0^*(s) = 0$).
3) Choose the minimum, which is $f_i^*(s)$. Go to the next step.
4) The last step is $f_J^*(NS) = min_{0 \leq C_J \leq \frac{NS}{b_J}} [\lambda_J D_J + f_{J-1}^*(NS - b_J C_J)]$, from which, we can get C_J^*, which is the optimal number of copies for video J.
5) Go backward to get the optimal value $C_{J-1}^*, C_{J-2}^*, \ldots, C_1^*$.

As described above, the optimal profile $(C_1^*, C_2^*, \ldots, C_J^*)$ solves the optimization problem (14).

After we obtain the number of copies of a sub-stream, we sort the sub-streams in descending order by λ_i/C_i^*. We assign an index $I(j, m, i)$ to the ith copy of the mth description for the jth video. The index is calculated as follows:

$$I(j, m, i) = \sum_{k=1}^{j-1} M \cdot C_k^* + C_j^* \cdot (m-1) + i.$$

We put the description indexed by $I(j, m, i)$ on the peer $I(j, m, i) \ mod \ N$.

C. Admission Control and Server Selection

Recall that we assume a fixed number of peers in the server pool, and these peers are intermittently connected. When the number of on-going sessions becomes very large, then the target video quality for each session cannot be sustained. To meet the distortion constraint, we set the number of connections to Q_{max}. If the total number of sessions in the network reaches Q_{max}, then new requests are blocked. A new request is admitted into the system only if the current number of on-going sessions is less than Q_{max}. For an admitted request, the system will search for the sub-streams and establish as many connections (up to M) as possible.

In our peer-driven streaming system, each sub-stream is stored on a different peer. When a peer requests a video with M sub-streams, the system will try to find M serving peers that have the M sub-streams of the video, with each peer also having sufficient surplus uplink bandwidth to serve one additional sub-stream. Recall that our overall design objective is to maximize the acceptance probability while meeting the distortion constraint. As more sessions are admitted into the system, more peer upstream bandwidth is consumed. Once a server peer's uplink bandwidth is used up, new requests have no access to any of the sub-streams stored at that server. Thus, it is desirable to a newly admitted request be assigned servers in order to even out the server transmission load. Following this logic, in our simulations we apply the following server selection policy: for a newly admitted client streaming session, for each sub-stream in the video, if there are multiple servers available, the system selects the server that has the most available uplink bandwidth. We also apply this policy when replacing a sub-stream after a server disconnect.

This server selection policy, although natural and intuitive, admittedly leaves room for refinement in future research. In particular, the selection policy should also take into account current CPU and disk utilization of each server peer. It may also take into account estimated future availability of the server peers, in order to avoid choosing a peer that may, with high probability, disconnect in the near future. Arguably, it may also take into account the distance between the client and the server peers, where distance could be defined in terms of RTT, one-way delays, number of AS hops, and so on. However, because we assume a small client buffer, delay variations from the different servers, each contributing a different substream, can be smoothed out.

D. Simulation Studies

The previous sections presented analytical models for the MD system. We proved that the system performance will improve as the number of descriptions for each video increases and discussed an optimal scheme to find the number of copies for each video. In this section, we perform extensive simulations to study the performance of the peer-driven MD scheme for *real* video sources.

1) Video Data: We have $J = 50$ videos. Each video has the same size but not the same popularity. Generally, the popularity of on-demand videos follows a heavy-tailed distribution. In our simulation, we assume the video popularity follows a Zipf distribution. Suppose the J videos are sorted in descending order of their popularities, and the popularity of video j is λ_j, where λ_j is now normalized by setting $\sum_{j=1}^{J} \lambda_j = 1$. Then $\lambda_j = j^{-(1-\rho)}/I$, where I is the normalization factor with $I = \sum_{j=1}^{J} j^{-(1-\rho)}$, and ρ is a control parameter. In our simulations, we chose $\rho = 0.27$ which is a commonly used factor for video on-demand services [18]. The new requests are modeled as a Poisson process; we change the rate to get different network loadings. The length of each video is 2 hours. The number of copies C_j for each video is obtained as in Section III-B. Each peer stores at most one description for a particular video.

2) Video Coding: We coded the "Foreman" video sequence in CIF (352x288) resolution into a scalable bit stream using the MPEG-4 FGS codec [19], at a base layer rate of 150 kbps. Each Group of Frames(GOF) has a duration of $T = 1$ second and comprises 15 frames. The output bits from each GOF are converted to M descriptions using the MD-FEC method described in Section III-A, where M is varied from 4 to 32. The total rate of a video after MD-FEC is set to be either 512 kbps, 576 kbps or 640 kbps. We precompute the operational rate-distortion function for "Foreman" and assume that all the 50 videos have the same characteristics as the "Foreman" sequence.

3) Peer Parameters: In our simulation, we assume a homogeneous network with 300 peers. Each peer has the same connectivity probability, uplink bandwidth and storage capacity. We set the uplink bandwidth of each peer to 250 kbps and the storage contributed by a peer to 1 GB. Another important characteristic is the connectivity probability of the peer. Each peer in the network alternates between "connect" and "disconnect" status. We model the connected time as an exponentially distributed random variable with mean α. Similarly, the disconnected time is another exponentially distributed random variable with mean β. Then the probability that a node is connected is $\frac{\alpha}{\alpha+\beta}$.

4) Performance Metrics: We use video quality to represent the performance of service perceived by the requesting peers. The video quality refers to the average peak signal to noise ratio (PSNR) over all streaming sessions occurred during our simulation time. To determine this value, we first determine the loss probability P_m based on our network simulation for given M, r, and Q. Then we compute the average distortion in terms of MSE using the precomputed rate-distortion function for "Foreman" and the loss probabilities using (1). Finally we convert the MSE to PSNR with $PSNR = 10 \log_{10}(255^2/MSE)$.

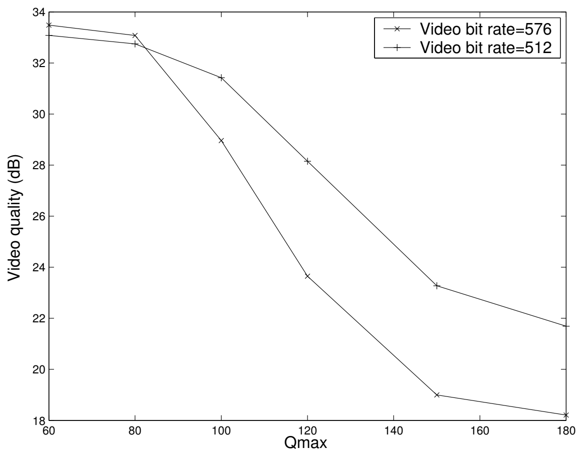


Fig. 6. Video quality vs. Q_{max} for $M = 32$

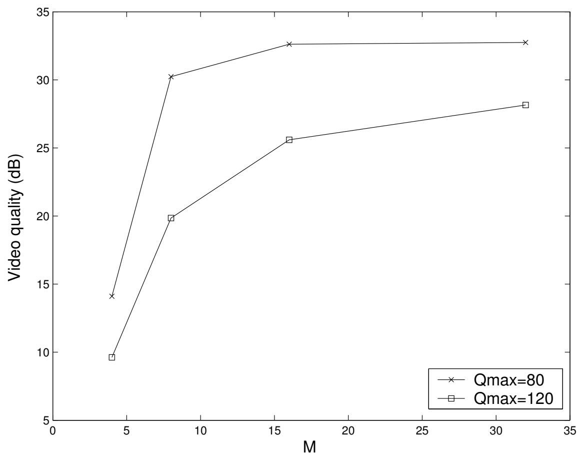


Fig. 7. Video quality vs. M with a fixed total rate of 512 kbps

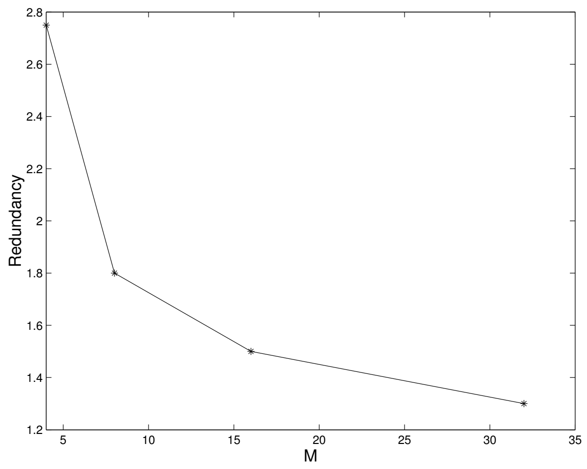


Fig. 8. Redundancy vs. M

5) Simulation Results: We first examine how the admission control parameter Q_{max} affects the system performance. As expected, the video quality will decrease as Q_{max} increases. Figure 6 shows how the video quality changes as Q_{max} increases for $M = 32$ descriptions. We simulated with two different total rates: 512 kbps and 576 kbps. From the figure, we can see that when Q_{max} is small, a higher total rate gives a better video quality. This is because when Q_{max} is small, the network is lightly loaded, which means that the descriptions are less likely to be lost, and the increase in total rate gives a better video quality. However, when Q_{max} is large, as seen from Figure 6, a smaller total rate provides a better video quality. As the total rate of each video increases, the overall video size increases as well. Since the total storage is fixed, the number of copies for each video will decrease. Also, the number of sub-streams served per uplink goes down for a fixed M. Therefore, the overall serving capacity for each video decreases. If during the streaming interval, a peer goes down, it is less likely to find a replacement peer. Although the increase in total rate gives a better video quality, for a higher rate, each session receives, on the average, a smaller number of descriptions. Therefore, when the network is heavily loaded, a smaller total rate has a better video quality. This result is consistent with the result shown earlier for Gaussian sources.

Figure 7 shows how the video quality varies with the number of descriptions for a fixed total rate of 512 kbps. We can see that the PSNR improves as the number of descriptions increases, but the improvement gradually levels off as M becomes very large. Note that this is consistent with the theorem in Section III-A.1, which assumed an idealized model. One reason that increasing the number of descriptions is always better is that the MD-FEC redundancy (defined as the ratio of the total bits divided by data bits) is optimally determined based on the description loss distribution determined from the network simulation. When M is large, MD-FEC can apply unequal error protection with finer granularity (because the same number of bits is split into more layers), thus requiring lower total redundancy and leaving more bits for source coding. Figure 8 shows the total redundancy determined by MD-FEC for different values of M. Another reason is that when M is small, the rate r is large; since the uplink bandwidth is limited, some of the bandwidth will be wasted. For example, in our simulations with uplink bandwidth 250 kbps, when the total rate is 512 kbps and $M = 4$, the r is 128 kbps. Thus each peer can only serve one streaming session and the rest of the bandwidth 250 - 128 = 122 kbps is wasted.

IV. Layered System

In the previous section, we focused on peer-driven video streaming system using MD-encoded video. With MD coding, each sub-stream has the same importance. One alternative to MD coding is layered video. With layered video, the sub-streams (called layers) have varying importance. For the client to be able to make use of sub-stream $m + 1$, the client must also be receiving sub-streams 1 through m. Thus, with layer-encoded video, the lower layers are more important than the upper layers.

As with MD peer-driven streaming system, the design of a peer-driven streaming system with layered video has four interacting components: video coding, sub-stream placement, admission control, and sub-stream server selection. However, the design for the layered-case is significantly different. We examine each of these components for layered video in this section.

A. Video Coding

We encode each video into multiple layers using a MPEG-4 FGS Coder [20]. The FGS coding technique encodes a video into two layers, a base layer and a scalable enhancement layer. The base layer contains the most essential quality information. The enhancement layers provide quality enhancements. FGS allows the user to adjust the relative sizes of the base layer and enhancement layer and further allows the enhancement layer to be broken up into an arbitrary number of hierarchical layers. As with MD system simulations, we set the base layer rate to 150 kbps and vary the rate of the enhancement layer. Given M, we divide the video bits from each GOF into M identically-sized hierarchical layers. When decoding up to m layers, denote the associated distortion by $D_m = D(R_1 + \cdots + R_m)$, where $D(R)$ is the rate distortion function of the underlying scalable coder. Let q_m denote the probability of receiving all layers up to layer m. Then the expected distortion of the received video is

$$D = \sum_{m=0}^{M} D_m q_m. \tag{15}$$

B. Video Placement

In section III-B, we presented the video placement scheme for the MD system. With the MD system, since each description has the same importance, the same number of copies should be created for all descriptions of a particular video. However, for the layered system, since the layers have varying importance, we should create more copies of the lower layers than the upper layers in order to increase the likelihood of delivering the more important lower layers. This way, when a peer serving a more important lower layer disconnects from the network, it is more likely to find another peer having the same layer that is connected to the network and has available uplink bandwidth to serve this layer. Therefore, the problem of video placement for the layered system is fundamentally different as compared to the MD system. In this section, we describe how to replicate the layered video and place these copies.

1) Single Video Case: First, we consider the problem of finding, given the storage capacity constraint, how many copies should be created for each layer of a particular video. Consider video j, encoded into M layers with each layer having size b_j. The total storage capacity allocated to this video is S_j, that is,

$$\sum_{m=1}^{M} b_j C_{jm} \le S_j, \tag{16}$$

where $C_{jm}, m = 1, 2, \ldots, M$, is the number of copies of layer m.

For the layered video, the probability of receiving first m layers but not the $m+1$ layer q_{jm} is,

$$q_{jm} = \prod_{k=1}^{m} p_{jk}(1 - p_{j(m+1)}), \quad m = 0, \ldots, M-1,$$

and

$$q_{jM} = \prod_{k=1}^{M} p_{jk},$$

where p_{jm} is the probability of receiving layer m and, as in section III-B, is given by $p_{jm} = 1-(1-\hat{\mu})^{C_{jm}}$. Then the video placement problem for a single layered video is to decide the number of copies for each layer given the constraint in (16), such that the average distortion is minimized:

$$\begin{aligned} \min \quad & D_j = \sum_{m=0}^{M} D_{jm} q_{jm} \\ s.t \quad & \sum_{m=1}^{M} b_j C_{jm} \le S_j \end{aligned} \tag{17}$$

This optimization problem can also be solved by dynamic programming. Actually, we can rewrite $D_j = \sum_{m=0}^{M} D_{jm} q_{jm}$ as

$$\begin{aligned} D_j &= \Delta_0 + \Delta_1 p_{j1} + \Delta_2 p_{j1} p_{j2} + \ldots + \Delta_M p_{j1} p_{j2} \ldots p_{jM} \\ &= \Delta_0 + p_{j1}[\Delta_1 + p_{j2}[\Delta_2 + \ldots] \ldots], \end{aligned}$$

where $\Delta_m = D_{jm} - D_{j(m-1)}$. Let $f_m(s) = p_{jm}[\Delta_m + p_{j(m+1)}[\Delta_{m+1} + p_{j(m+2)}[\Delta_{m+2} + \ldots] \ldots]$, where s is the storage capacity allocated to layer m up to M, $0 \le s \le S_j$. Let $f_m^*(s)$ denote the minimum value of $f_m(s)$, then we have

$$f_m^*(s) = \min_{0 \le C_{jm} \le \frac{s}{b_j}} p_{jm}[\Delta_m + f_{m+1}^*(s - b_j C_{jm})],$$

and

$$f_M^*(s) = \Delta_M p_{jM}(\frac{s}{b_j}),$$

where $0 \le s \le S_j$. The last step gives us

$$f_1^*(S_j) = \min_{0 \le C_{j1} \le \frac{S_j}{b_j}} p_{j1}[\Delta_1 + f_2^*(S_j - b_j C_{j1})]. \tag{18}$$

From (18), we obtain C_{j1}^*, which is the optimal number of copies of the first layer. Then we can go backward to get $C_{j2}^*, C_{j3}^*, \ldots, C_{jM}^*$. The optimal profile $(C_{j1}^*, C_{j2}^*, \ldots, C_{jM}^*)$ solves the optimization problem.

2) Multiple Videos Case: In this section, we consider how to obtain the optimal number of replicas of a sub-stream for the multiple videos case. Using the same notation as in the previous sections, the storage constraint is:

$$\sum_{j=1}^{J} \sum_{m=1}^{M} b_j C_{jm} \le NS.$$

The average distortion over all videos is

$$D = \sum_{j=1}^{J} \lambda_j D_j.$$

where $D_j = \sum_{m=0}^{M} D_{jm} q_{jm}$. Like before, our objective is to minimize the average distortion while satisfying the storage constraint,

$$\begin{aligned} \min \quad & D = \sum_{j=1}^{J} \lambda_j D_j \\ s.t. \quad & \sum_{j=1}^{J} \sum_{m=1}^{M} b_j C_{jm} \leq NS \end{aligned} \qquad (19)$$

Similarly to the MD system, we can solve this optimization problem using dynamic programming. The only difference is that for the MD system, given the storage capacity allocated to video j, we can calculate D_j directly. But for the layered system, D_j should be optimized as in the single video case.

Since the layers have varying importance, the placement policy for the layered video is different from the MD system. In order to balance the load, clearly the more important lower layers should not be stored on the same peer. Therefore, at the beginning, we place the first layer of all videos onto different peers, then the second layer, and so on. To this end, we sort the videos in descending order by λ_j, and assign an index to the ith copy of the mth layer for the jth video. The index is given by

$$I(j,m,i) = \sum_{k=1}^{J} \sum_{t=1}^{m-1} C^*_{kt} + \sum_{k=1}^{j-1} C^*_{km} + i.$$

The layer indexed by $I(i,m,j)$ is placed on the peer $I(i,m,j) \mod N$. However, note that we have the natural restriction that no two layers from the same video are stored on the same peer, for a given layer m, if the peer $I(i,m,j) \mod N$ already stores a lower layer of video j, then we go to the next peer that does not store a layer for video j.

C. Admission Control and Server Selection

As in MD system, we use the same admission control and server selection policy (described in Section III-C) for peer-driven streaming system with layered video in our simulations. When a new request arrived, if the current number of ongoing sessions is less than Q_{max}, the request is admitted. For an admitted request, the system searches for all layers of this video. However, for a layered video, if layer m is not available, then all layers high than this layer are useless, which means that the system does not need to search for layers higher than m. For each layer in the video, the system selects the server with the most available uplink bandwidth. During the streaming session, if a peer serving layer m disconnects and the system cannot find a replacement peer, then the connections streaming layers higher than m are terminated.

V. Comparison of the MD and the Layered System

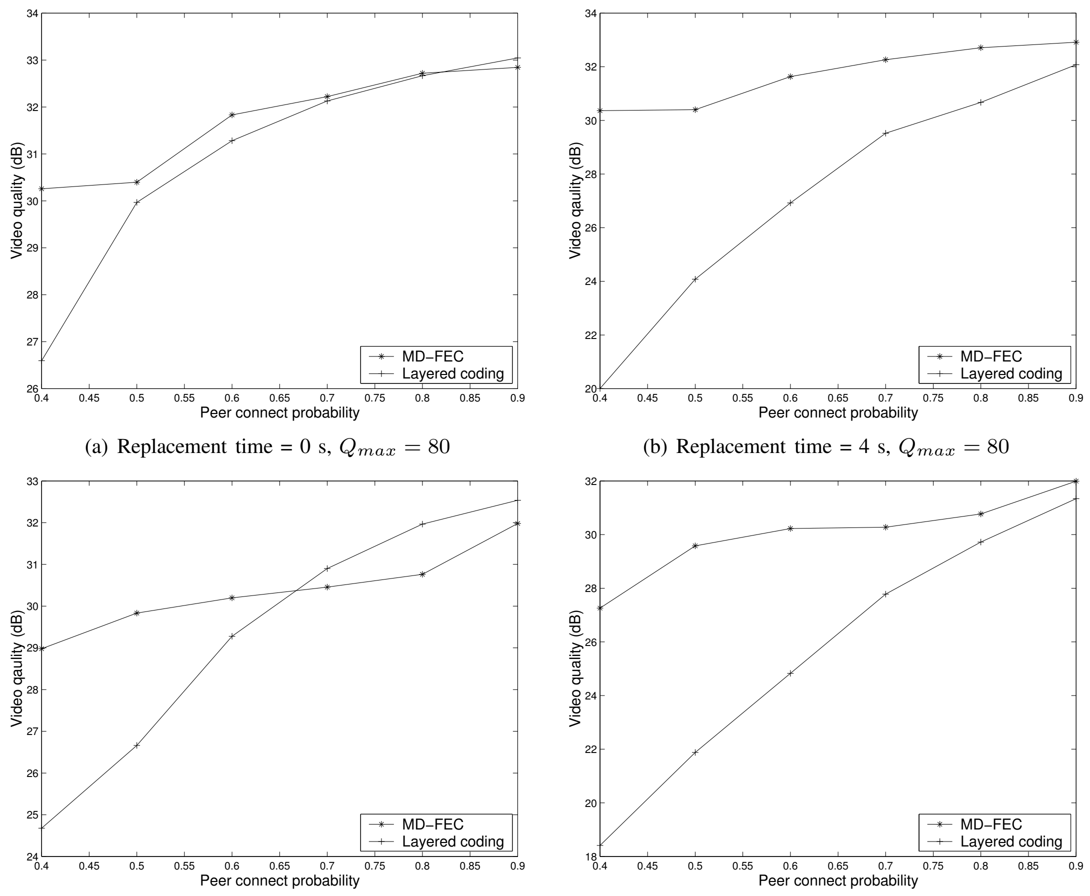
In the previous sections, we studied two classes of systems: systems which use multiple description coding in which all sub-streams (called descriptions) have equal importance; and systems which use layered coding, in which the sub-streams (called layers) have varying importance. As we have seen, the design for the layered-case is significantly different from the MD system. With the MD system, all sub-streams can be treated equally. This greatly simplifies the system design. On the other hand, the layered system must provide unequal treatment to the sub-streams, to increase the likelihood of delivering the more important sub-streams. For example, for sub-stream placement, for the MD encoded video, each description has the same number of replicas. By contrast, the optimal scheme replicates more aggressively the lower layers than the upper layers for the layered video. In this section, we compare the performance of the MD system using the MD-FEC codec and the layered system with the same simulation settings as described in Section III-D. The total rate for both MD and layered video is set to be 512 kbps.

Figure 9 shows the comparison between MD-FEC and layered system with varying peer connect probability, where each plot represents one combination of Q_{max} and replacement time. We fix the mean of the average connected time α to be 4 hours and obtain different peer connect probability by changing the mean of the average disconnected time β. The replacement time is defined to be the time to find a replacement peer if a server peer disconnects during the streaming session, and the video quality is degraded during this time. In both MD-FEC and layered systems, we use the video placement, admission control and server selection policies described in Section III and Section IV. Then we choose the optimal M which achieves the best video quality.

As we would expect, the video quality improves as the peer connect probability increases, for both MD-FEC and layered system. When the peer connect probability is small, the performance of the MD-FEC system is much better than the layered system. As peer connect probability increases, the performance of the layered system increases at a rate faster than the MD system. With zero replacement time, as the peer connect probability increases beyond a certain point, the layered system outperforms the MD system. As we know, layered coding requires a lower bit rate to reach the same video quality as a MD-FEC codec. Therefore, when the network is more reliable, layered coding is more efficient than MD-FEC. However, when the replacement time increases (4 s), the performance of the MD system is always better than the layered system. Therefore, the time to find a replacement peer has a bigger impact on the layered system than the MD system. The reason is that MD-FEC has inherent protection against sub-stream loss. When a single sub-stream is lost for MD-FEC, the video quality is only slightly affected. But for layered coding, all layers higher than this sub-stream cannot be decoded at the receiver.

VI. Related Work

Several schemes have been proposed for peer-driven muticast streaming [21]–[26]. Most of these schemes build multicast trees over the peers for streaming the video content stored in a central server; the peers relay the video originating from a central server. Of these research efforts, CoopNet [22], which



(a) Replacement time = 0 s, $Q_{max} = 80$

(b) Replacement time = 4 s, $Q_{max} = 80$

(c) Replacement time = 0 s, $Q_{max} = 120$

(d) Replacement time = 4 s, $Q_{max} = 120$

Fig. 9. Comparison between MD-FEC and layered system with varying peer connect probability

uses MD coding to code the source into several sub-streams, has some similarities with our scheme. It builds multiple distribution trees from sources to receivers, with each tree disseminating a separate description of the media content. A receiver can receive all descriptions with best quality in the best case. A peer failure only causes its descendant peers to lose a few descriptions with graceful quality degradation. CoopNet is multicast service, with all the multicast trees rooted at a single server. Our system, which does not employ multicast trees, provides a fundamentally different video-on-demand service.

In P2Cast [27], [28], peers arriving close in time form a session or a multicast tree. For clients that arrive later than the first client in the session, the missing initial part can be retrieved or patched from the server or other peers that have already cached the initial part. Thus, peers help each other when they watch the same video. To join a session, a peer needs to traverse the tree peers downward from the source until finding one patch server and base stream server with enough bandwidth and/or small delay. Unlike our scheme, P2Cast does not employ sub-streams to distribute videos.

However, many of these multicast approaches, if applied using edge peers, may not feasible with the current Internet. Current residential Internet access is typically asymmetric in the upstream and downstream bandwidth, with a peer having more downstream than upstream bandwidth. Upstream bandwidth of a peer determines its out-bound bandwidth. Although this asymmetry is beneficial for downloading content, it severely limits the bandwidth of a server peer. Since these peers, which may be part of multicast trees, have large outgoing bandwidth requirements, they may not be able to satisfy the bandwidth requirements of video streaming.

The Peersreaming [29] video-on-demand system has some similarities to our system. In Peersreaming [29], the media stream is broken up into “data units,” and each “data unit” is encoded using a high-rate Reed-Solomon code. The client peer first searches for other peers that have a copy (or a portion of a copy) of the desired video. Then the client peer sets up TCP connections to the discovered peers and, for each data unit, the client determines which blocks to download from the connected peers. In our schemes, with MD coding and layered coding, there is graceful degradation of quality when substreams are lost, whereas with Peerstreaming there is complete loss of video if the client does not receive all blocks. Our work is complementary to Peerstreaming in that we explore different forms of coding, for which we develop a traffic and placement theory.

VII. Conclusion and Future Work

We have presented a novel video streaming scheme. Unlike an infrastructure-based architecture, our approach is based on a peer-driven architecture, where each peer stores and streams videos to the requesting client peers. We encode each video into multiple sub-streams and place each sub-stream on a different peer. If a serving peer disconnects in the middle of a streaming session, the system searches for a replacement peer that stores the same video sub-stream and has sufficient uplink bandwidth. We have developed a teletraffic and placement theory for optimizing the performance of both MD and layered systems, compared the performance via simulation of the two optimized schemes, and obtained insights into the range of parameters for which one system performs better than the other.

We have not considered the effect of pre-fetching on either the MD or the layered system. By pre-fetching, the system can avoid the loss of a sub-stream upon a peer disconnect. One could think of a system that uses a proactive buffering strategy where the streaming rate is increased to ensure that a minimum amount of buffer space is occupied at all times. By maintaining a certain buffer size clients could easily cope with churn since they have enough time to react in case of peer disconnects. But prefetching will increase the play-out delay, increase the memory requirement at the client, and complicate the overall system design. As an extension of the current work, we plan to investigate the design and performance evaluation of the system with pre-fetching.

Acknowledgment: This work is supported in part by the National Science Foundation under Grant CNS-0435228, and also in part by the New York State Center for Advanced Technology in Telecommunications (CATT).

References

[1] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proc. of 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, Heidelberg, Germany, November 2001.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proc.of ACM SIGCOMM'01*, San Diego CA, USA, August 2001.

[3] I. Stoica, R. Morris, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. of ACM SIGCOMM'01*, San Diego, CA, USA, August 2001.

[4] B. Y. Zhao, J. D. Kubiatowicz, and A. Joseph, "Tapestry: An infrastruture for fault-tolerant wide-area location and routing," UCB, Tech. Rep. CSD-01-1141, Apr. 2000.

[5] P. Decuetos and K. Ross, "Adaptive rate control for streaming fine-grain scalable video," in *International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV) 2002*, Miami, May 2002.

[6] D. Saparilla and K. Ross, "Optimal streaming of layered encoded video," in *IEEE Infocom 2000*, Tel Aviv, March 2000.

[7] [Online]. Available: http://ipmon.sprint.com/delaystat

[8] S. Bhattacharyya, C. Diot, J. Jetcheva, and N. Taft, "POP-Level and Access-Link-Level Traffic Dynamics in a Tier-1 POP," in *ACM SIGCOMM Internet Measurement Workshop (IMW)*, San Francisco, Nov. 2001.

[9] Y. Wang, R. Reibman, and S. Lin, "Multiple description coding for video delivery," *Proceedings of the IEEE*, vol. 93, pp. 57–70, 2005.

[10] A. Reibman, H. Jafarkhani, Y. Wang, and M. Orchard, "Multiple Description Coding for Video using Motion Compensated Prediction," *IEEE Trans. Circuit and System for Video Technology*, pp. 193–204, Mar. 2002.

[11] Y. Wang and S. Lin, "Error resilient video coding using multiple description motion compensation," *IEEE Trans. Circuit and System for Video Technology*, vol. 12, pp. 438–453, June 2002.

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

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

[14] R. Puri and K. Ramchandran, "Multiple description source coding through forward error correction codes," in *33rd Asilomar Conf. Signals, Systems and Computers*, Oct. 1999.

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

[16] V. Stankovic, R. Hamzaoui, and Z. Xiong, "Packet loss protection of embedded data with fast local search," in *Proc. IEEE Int. Conf. Image Processing*, Rochester, NY, September 2002.

[17] T. M. Cover and J. Thomas, *Elements of Information Theory*. John Wiley and Sons, Inc, 1991.

[18] J. Chu, K. Labonte, and B. Levine, "Availability and popularity measurements of peer-to-peer file systems, Tech. Rep. Technical report 04-36, June. 2004.

[19] M. code, "MPEG4 verfication model version 18.0," in *ISO/ IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio*, January 2001.

[20] H. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over ip," *IEEE Trans. on Multimedia*, March 2001.

[21] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over a peer-to-peer network," Stanford University, Tech. Rep., 2001.

[22] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proceedings of NOSSDAV*, 2002.

[23] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth content distribution in a cooperative environment," in *Proc. of 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Berkeley, CA, USA, February 2003.

[24] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proceedings of ACM Sigcomm*, August 2002.

[25] D. Tran, K. Hua, and T. Do, "ZIGZAG: an efficient peer-to-peer scheme for media streaming," in *Proceedings of IEEE INFOCOM 2003*, San Francisco, CA, USA, March 30-April 3 2003.

[26] X. Zhang, J. Liu, B. Li, and P. Yum, "DONet: A data-driven overlay network for efficient live media streaming," in *Proc. of IEEE INFOCOM*, 2005.

[27] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast: peer-to-peer patching scheme for VoD service," in *Proceedings of the 12th World Wide Web Conference (WWW-03)*, Budapest, Hungary, May 2003.

[28] ——, "A peer-to-peer on-demand streaming service and its performance evaluation," in *Proceedings of 2003 IEEE International Conference on Multimedia and Expo (ICME 2003)*, Baltimore, MD, July 2003.

[29] J. Li, "Peerstreaming: A practical receiver-driven peer-to-peer media streaming system," Microsoft Research, Tech. Rep. MSR-TR-2004-101, September 2004.