Efficient Substream Encoding for P2P Video on Demand

Zhengye Liu†, Yanming Shen†, Shivendra Panwar†, Keith W. Ross‡ and Yao Wang†
†Department of Electrical and Computer Engineering
‡Department of Computer and Information Science
Polytechnic University, Brooklyn, NY, USA 11201

***Abstract*— In a P2P VoD system, the rate at which peers receive video fluctuates due to peer churn. Although scalable video coding has the potential to cope with rate fluctuation, existing scalable video schemes have not been designed for P2P systems for which substreams emanate from different supplier peers. In this paper we propose a new multi-stream coding and transmission scheme, Redundancy Free Multiple Description (RFMD) Coding and Transmission, that has been specifically designed for P2P VoD systems. Unlike layered video, with RFMD all substreams have equal importance. Thus, video quality gracefully degrades as substreams are lost, independently of which particular substreams are lost. Furthermore, only the source bits are collectively transmitted by the supplying peers, allowing more substreams to be simultaneously transmitted in the system. Finally, RFMD can be used to create any number of descriptions. We conduct an extensive and fair simulation study, comparing single-layer coding with high-rate erasure coding, scalable layered encoding, multiple-description encoding, and RFMD. The simulations show that RFMD performs best in a variety of representive scenarios.**

I. Introduction

Video-on-demand (VoD) over the Internet has become very popular in recent years. VoD is currently being provided by numerous television networks and news sites, as well as by video sharing sites such as YouTube [1] and Google Video [2]. Most of the VoD being delivered today is short-length, low bit-rate clips; for example, YouTube videos today are typically less than 10 minutes in length and have a bit rate under 200 kbps. In the near future, we expect a high demand for higher-bit rate (potentially DVD quality) and longer videos (including full length movies) streamed over the Internet.

The vast majority of the VoD being delivered today over the Internet emanates from dedicated infrastructure servers. But as the demand, bit-rates, and video lengths increase, it will become costly to meet the demand, both in terms of bandwidth costs and server hardware costs. Thus, much of the Internet VoD will likely be streamed via P2P architectures, in which the consumers of the VoD content are also the suppliers of the content. Since, in P2P VoD architectures, each peer contributes its own storage and network bandwidth resources to the system, the upload capacity of the system scales as the demand increases.

Most P2P file distribution systems today, including BitTorrent [3] and eDonkey [4], employ *swarming* for downloading. With swarming, the downloader obtains different pieces of the file in parallel from multiple peer sources and reassembles the entire file once it has all the pieces. Swarming can significantly improve download performance, particularly if the downloader's download bandwidth exceeds the upload bandwidths of individual uploading peers. P2P live video streaming systems, such as PPLive [5] [6] and ppstream [7], also employ swarming. Swarming is critical for live video since the video rate often exceeds the upload capacity of a residential peer.

Swarming is also useful for P2P VoD. When a peer makes a request for a video, it can receive video substreams from multiple peers in parallel, reassemble and decode the substreams, and playback the video to the user. Just as in P2P live streaming systems, swarming is critical in P2P VoD, since individual supplying nodes may not be able to upload the video at the video rate. But in order to receive multiple streams, at the time of request there must be multiple peers that (i) have portions of the requested video, and (ii) have available upload capacity. Indeed, even though a peer may store a portion of the requested video, all of its upload capacity may currently be exhausted by other ongoing video streaming sessions.

In a P2P VoD system, the receiving peer would like to receive the video from all of its supplying peers at an aggregate rate (averaged over a short time scale) that exceeds the compressed video rate, which we denote by r. However, the rate at which a peer receives video will fluctuate and may drop below r. These fluctuations will not likely be due to congestion in the upper-tier ISPs, but instead to the peer churn in the system [8]. Indeed, every active peer in the system is both a consumer and supplier of upload bandwidth, with different peers (residential peers, institutional peers, and so on) contributing different amounts of upload bandwidth and different peers making available portions of different videos. As peers come and go, the ratio of the upload-bandwidth-supply to the upload-bandwidth-demand for a given video fluctuates. Thus the rate at which a peer can receive a video will fluctuate because of the churn.

As with traditional client-server streaming, buffering and playback delays can be used to mitigate the effects of short-term variations in bandwidth availability. However, if the demand for upload bandwidth exceeds the supply for a long period of time, buffering is ineffective. Even with buffering, one or more of the receiving peers will receive the compressed video at a rate less than r during the bandwidth deficit period.

As with traditional client-server streaming, to deal with

long-term fluctations in available bandwidth, it is natural to consider multi-stream coding techniques where receiving a few substreams can lead to an acceptable quality, and receiving more streams can lead to better quality. Such multi-stream coding techniques include layered coding and multiple description coding. However, because the receiving node receives video from multiple sources (and not just from one, as in client-server systems), the design of multi-stream coding and delivery schemes for P2P VoD brings forth many new challenges. In particular, layered coding is vulnerable to peer disconnects due to the recursive dependency of layers. And MDC, while giving equal importance to each stream, typically introduces significant redundancy across streams.

In this paper we propose a new multi-stream coding and delivery scheme that has been specifically designed for P2P VoD systems. We refer to this scheme as Redundancy-Free Multiple Description Coding and Transmission, or more simply as RFMD. With the RFMD, all substreams have equal importance, unlike layered video. Thus, video quality gracefully degrades as substreams are lost, independently of which particular substreams are lost. Furthermore, with RFMD, only source bits contributing to reducing video distortion are transmitted by the supplying peers, so that there is no redundancy as is in conventional MDC. This lack of redundancy can significantly increase the streaming capacity of the P2P VoD system.

After describing and analyzing RFMD, we carry out an extensive simulation study that compares single layer coding with high-rate erasure codes, layered coding, a traditional MDC scheme known as MD-FEC, and RFMD. Careful attention is placed on making the comparisons as fair as possible. We find that RFMD provides the best overall performance as compared to other coding schemes. The contribution of this paper is twofold:

- A new multi-stream coding and transmission scheme tailored for P2P streaming systems, dubbed RFMD.
- A fair simulation study involving heterogeneous peers with peer churn, which compares single layer coding with high-rate erasure codes, layered coding, MD-FEC, and RFMD.

This paper is organized as follows. In Section II, we briefly describe the P2P VoD context. Section III describes the traditional substream schemes. Section IV proposes the redundancy-free MDC design. We study the performance of our proposed design via simulations in Section V, and Section VI concludes this paper.

II. The P2P VoD context

There are many possible business models for P2P VoD, and for each of those models the distribution, replication, and search for video substreams could be done differently. Here, we describe a broad P2P VoD context that focuses on the swarming and streaming, and should be applicable to most P2P VoD designs.

As indicated in the Introduction, our focus is on multi-stream video schemes, such as layered encoded video and multiple-description video. Each video is coded into a number of substreams and the substreams (or portions of the substreams) are scattered over all the peers. (Depending on the VoD design, the substreams may either be pushed into the various peers in a coordinated fashion [9]–[13], or may be pulled into the peers in an on-demand fashion [14], [15].) Since a peer has limited storage, it does not store every substream of every video.

When a peer wants to see a video (the receiving peer), the "system" provides the peer with a list of peers containing one or more substreams (supplier peers). The receiving peer then requests and receives substreams from one or more of the supplier peers. A supplying peer can service a receiving peer if (i) it has the requested substream (or at least portions of it) and (ii) it has sufficient available upload bandwidth to send the substream at the substream rate. The receiving peer caches the substreams it receives (or the portions of the substreams), so that it can be a supplier of that video in the future. After a small playback delay, the receiving peer decodes, assembles, and playbacks the substreams it receives.

While the user is watching the video, the receiving peer may lose or gain new substreams. It may gain new substreams because it discovers a supplier peer that stores a missing substream and also has sufficient available upload bandwidth for delivery. It may lose a substream because the supplier peer providing the substream may stop delivering it. Generally, as the ratio of the available upload capacity of the supplier peers to the demand of the receiving peers increases, the receiving peers should be able to gain new substreams; similarly, as this ratio decreases, the receiving peers should lose substreams.

When a receiving peer loses a substream, it will naturally attempt to find a replacement supplier peer that can provide the lost substream. Even if such a supplier peer is present, there will be a delay until receiver peer receives the replacement substream, since the appropriate substitute peer must be found and then instructed to deliver the substream to the receiving peer. Depending on the length of this delay, and whether the receiving peer has a reservoir of pre-fetched substream content, there may be a "gap" in the playback of the substream. We note that, in some designs, it may be possible to find the substitute peer almost immediately, as the system may be able to continuously track the content and availability of all active peers. The searching and tracking of peers is orthogonal to the video delivery problem, and will not be considered further in this paper.

In designing a multi-stream coding scheme for P2P VoD, we therefore have the following objectives:

- The coding scheme should allow for smooth quality variation as the peers churn (causing the supply and demand for upload bandwidth to evolve over time).
- When a supplier peer suddenly stops providing a substream, and the receiving peer does not have a sufficient reservoir of that substream buffered locally, quality degradation should be minimal until a substitute supplier with the same substream begins to deliver the substream.
- The coding should be efficient, that is, it should carry

little or no redundancy so that the system as a whole can provide as many substreams as possible of the different videos to all the receiving peers.

A. Related work

To date, a few different coding schemes have been proposed and compared for P2P video streaming [9], [14], [16]–[18]. CoopNet uses MD-FEC to code the source into several sub-streams [9]. It builds multiple multicast trees from sources to receivers, with each tree disseminating a separate description of the media content. CoopNet is a multicast service rather than an encoding scheme, with all the multicast trees rooted at a single server. The authors suggest employing MDC in P2P multimedia streaming, but they do not really design an MD coding scheme specifically for P2P VoD. In PeerStreaming [14], the author proposes to use high-rate erasure coding to generate parity substreams instead of replicating directly. The media stream is broken up into "data units", and each "data unit" is encoded using a high-rate Reed-Solomon code. Specifically, a data unit is divided into K blocks, then RS coding is employed to generate $N - K$ parity blocks. The receiving peer can recover the original data unit from any K blocks. In PeerStreaming, the original media stream is non-scalable, so that a data unit is non-decodable if less than K blocks are received. In [16], the authors compare the performance of multiple description streaming in P2P network and CDNs. But they only consider using two descriptions. In our previous work [17], [18], we compare the performance of MD-FEC and layered coding under different scenarios. To our knowledge, the current paper is the first paper to develop an multi-stream video coding and transmission scheme for P2P VoD.

III. Traditional Substream Generation

In this paper we consider a number of different mechanisms for generating substreams for P2P VoD systems. Throughout this paper, let M denote the number of substreams of a video. We now briefly review layered video and MD-FEC.

A. Layered coding

Layered coding generates multiple layers with recursive dependency. Specifically, layer $k+1$ can only be decoded if layers 1 through k are available. MPEG-4 Fine Grain Scalable (FGS) [19] is a popular scheme for creating layered coding. An FGS encoder encodes the video into a base layer and a scalable enhancement layer. The enhancement layer is then sliced into $M-1$ substreams, creating a total of M substreams. The latest scalable coding standard, known as SVC [20], also has a base-layer and a successfully refinable enhancement layer. With FGS or SVC, the rate of the base layer must be sufficiently high so that an acceptable quality can be recovered from the base layer only. Compared to a single-layer coder, the distortion achievable by a scalable coder at the same rate (above the base layer) is typically higher. For example, FGS (resp. SVC) has a significantly lower coding efficiency than the MPEG-4 single layer coding (resp. H.264).

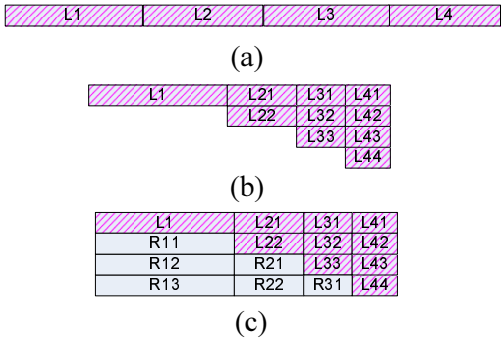

Fig. 1: MD-FEC encoding procedure.

B. Multiple Description FEC

MD-FEC is a popular scheme for multiple description encoding with many descriptions. Throughout this paper, we will use MD-FEC for the multiple description video. We now explain MD-FEC coding [21], as it forms the basis of our novel scheme, described in the next section.

The first step of MD-FEC is to encode each Group of Pictures (GOP) into M layers, which can be accomplished with a scalable video coder such as MPEG4 FGS or SVC. This is shown in Fig. 1 (a) for the case of 4 layers. Denote by $L1$, $L2$, $L3$, and $L4$ for the bits in these 4 layers. These layers are in general of varying size. The kth layer is then further divided into k equal-length groups. Thus, as shown in Fig. 1 (b), layer two is broken into two equal-size groups $L21$ and $L22$; layer three is broken into three equal-size groups $L31$, $L32$ and $L33$; and layer four is broken into four equal-size groups $L41$, $L42$, $L43$ and $L44$. Then a (M, k) Reed-Solomon (RS) code is applied to the k groups from layer k to yield M groups. The M^2 groups are then arranged as in Fig. 1 (c). For layer 1, the RS step creates three redundant groups $R11$, $R12$, and $R13$; for layer 2 it creates two redundant groups $R21$ and $R22$; and for layer 3 it creates 1 redundant group $R31$. (For layer 1, $L1 = R11 = R12 = R13$.) Thus, due to the RS code, if any k of the M groups is received for layer k, then layer k can be decoded.

After creating the M^2 groups, the substreams are generated by combining the groups across the rows in Fig. 1 (c). For example, the first substream is created by combining $L1$, $L21$, $L31$ and $L41$; the fourth substream is created by combining $R13$, $R22$, $R31$ and $L44$. From this construction, it is easy to see that the substreams have the following desirable properties:

- Each substream has the same bit-rate;
- In order to recover k layers from the original layer encoded video, the receiver needs to receive any k of the M substreams. Thus each stream is of equal importance.

The number of bits assigned to each layer (in the first step of the procedure) can be optimized to minimize distortion if the characteristics of peer availability are known in advance. One nice feature of MD-FEC as a multiple description technique is that it can generate any number of substreams (any M) from a scalable stream generated by any scalable coder, which is desirable for P2P VoD. When a supplying peer disconnects,

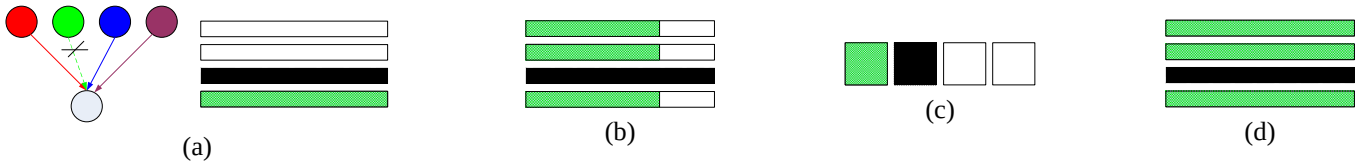

Fig. 2: Comparison of different coding schemes for P2P VoD system (a) Layered coding; (b)MD-FEC; (c)Single layer; (d) The desired scheme.

the video quality is not severely degraded while waiting for a substitute peer to supply the substream.

Note that with MD-FEC, when a receiver receives k of the M substreams, only a fraction of bits are used (for all values of k). For example, in the case $M = 4$, suppose the receiver receives only the first substream, which contains groups $L1$, $L21$, $L31$ and $L41$; then only $L1$ is of use (providing the base layer of the layered video) and the remaining groups are wasted transmissions. Similarly, suppose both substreams 1 and 4 are received, giving groups $L1$, $L21$, $L31$, $L41$, $R13$, $R22$, $R31$ and $L44$; then only $L1$, $L21$ and $R22$ are used and the remaining groups are wasted. Thus, another property of MD-FEC is that it is inefficient in that it wastes significant upload bandwidth resources.

C. Inadequacies of existing schemes

Fig. 2 (a), (b) and (c) illustrate how layered coding, MD-FEC and single layer coding schemes, respectively, work in a P2P VoD system. In this comparison, M is set to 4 and we assume that each supplying peer stores at most one substream. For each scheme, we consider a scenario that only three supplying peers are available in the system. The rectangles with black background indicate that the substreams are not available. The rectangles with the white background indicate that the substreams are received but can not be decoded. The green shadow in the rectangle indicates the portion of data that can be used for decoding and has contribution to video quality.

For layered coding, as shown in Fig. 2 (a), although the receiving peers can find three peers, since layer 2 is unavailable, layer 3 and layer 4 are useless. As illustrated in Fig. 2 (b), for MD-FEC, if the receiving peer can find three supplying peers with different descriptions, no matter which descriptions they are, each is able to contribute some video source data. But since some redundancy is transmitted, only a portion of the received data can be used for video source decoding.

For the single layer scheme, we assume the video is encoded into a single layer, and each GOP is divided sequentially into four blocks, so that earlier blocks contain data from earlier frames in the GOP. Each peer holds one block from each GOP. As shown in Fig. 2 (c), the second block is not available, so that frames contained in following blocks are not decodable either (assuming the video is coded using temporal prediction), and only the frames contained in the first block can be decoded and frozen until the next GOP.

Fig. 2 (d) shows the ideal coding scheme. If the receiving peer can locate *any* m ($m < M$) peers holding the requested

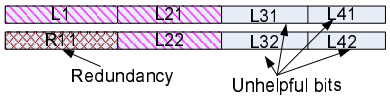

Fig. 3: The redundancy and unhelpful bits in MD-FEC when only two descriptions are available.

video, it can decode the first m layers. Moreover, all of the bits transmitted by the supplying peers are useful; so that there is no wasted bandwidth. In the next section, we design such an ideal coding scheme for P2P VoD.

IV. Redundancy-Free MD-FEC Scheme

Our RFMD scheme is based on three observations.

- First, for MD-FEC, to make each description have equal importance, *redundant bits* are transmitted. Also, some *unhelpful bits* are transmitted. For example, when $M = 4$, and only two descriptions are available for a receiving peer, the redundant and unhelpful bits are marked in Fig. 3.
- Second, the unhelpful bits are not useful for handling sudden peer disconnects. When one supplying peer disconnects, more unhelpful bits are introduced. Therefore, the unhelpful bits should not be transmitted until some additional supplying peers are connected.
- Third, although the redundant bits are useful when a supplying peer disconnects during the streaming session and the substitute streams can not be quickly found, it does not have to be always transmitted. VoD does not have as rigid delay constraints as interactive multimedia applications. When a supplying peer disconnects, the receiving peer can detect it quickly and then instruct the remaining supplying peers to adjust their transmissions.

In summary, in P2P VoD, which does not have stringent delay constraints, it is not necessary to always transmit all the encoded bits; the amount of redundancy and unhelpful bits that are transmitted should adapt to the number of available supplying peers.

We now describe new MDC scheme, which we dub Redundancy-Free MD coding and transmission (RFMD). This scheme is derived from traditional MD-FEC coding (see Section III-B). The coding procedure is as follows:

- We adopt MD-FEC coding to generate M descriptions.
- If m supplying peers are available, then each supplying peer only transmits a fraction k/m portion of the data for layer k, where $k = 1, \ldots, m$; each supplying peer transmits different portion of layer k data.

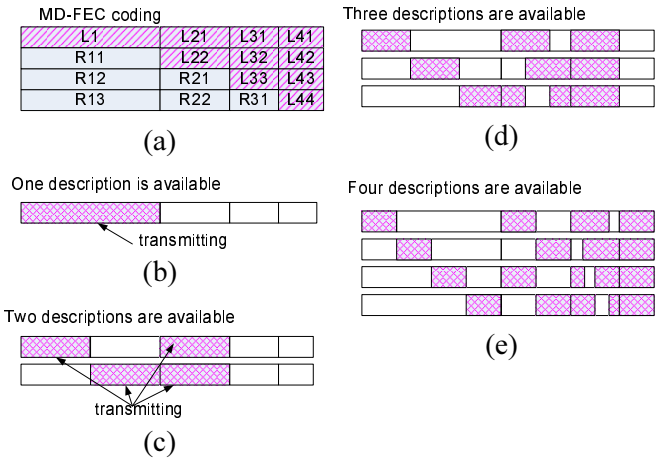

Fig. 4: Redundancy-free transmission of MD-FEC data (M=4): (a) shows the stored data in each description, with purple color representing the source bits and gray color the redundancy bits; (b-e) shows the portion of the data (purple) delivered by each supplying node.

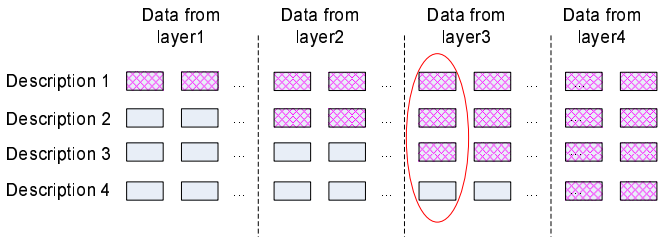

Fig. 5: Segmentation for redundancy-free transmission

- The receiving peer combines the received m substreams and obtains layer k ($k = 1, \ldots, m$) by (M, k) FEC decoding; hence, the lowest m layers are recovered.

Fig. 4 shows an example when $M = 4$. The shaded areas indicate the portion of data that will be transmitted. For example, if three descriptions are available, then for each description, one-third of the data from the first layer (original data or parity data of the first layer), two-thirds of the data from the second layer, and all data from the third layer are transmitted. Thus, all first three layers can be recovered. Note that each description has equal importance and the same bit rate, and neither redundancy nor unhelpful bits are transmitted.

Because the portion of data transmitted at each supplying peer depends on how many peers are available, generally, the transmission rate at each peer is not constant. Constant bit rate can be achieved with the appropriate rate partitioning for the different layers before MD-FEC encoding. Specifically, given a scalable stream, we truncate the encoded bits into M layers, each with equal rate $R_k - R_{k-1} = R$ for each $k = 1, \ldots, M$. Note that, in each MD-FEC encoded description, the bit rate for the data from layer k is R/k. Thus, no matter how many supplying peers are available in the system, the transmission rate for one substream is always constant:

$$\sum_{k=1}^{m} \frac{R}{k} \frac{k}{m} = R \qquad m = 1 \ldots M \tag{1}$$

Based on this rate partition, when the transmission rate for one substream is equal to R, the storage S used to store the substream is:

$$S = T \sum_{k=1}^{M} \frac{R}{k} = RT \sum_{k=1}^{M} \frac{1}{k} \tag{2}$$

where T is the length of the video.

As discussed above, for RFMD, when the number of available supplying peers changes, the portion of data that is transmitted from the remaining peers should also change correspondingly. This is achieved by segmentation and selecting different sets of segments for transmission at each of the supplying peers.

A. Implementation of RFMD

Fig. 5 shows the segmentation procedure. For a description, we segment the data generated from the same layer to several MD segments. One MD segment cannot cross the data from two different layers. We assign each MD segment a segment number (denoted as $SegNum$) and a layer number ($LayerIdx$), which indicates which layer the segment belongs to. The corresponding segments (covered with a red oval in Fig. 5) in different descriptions have the same $SegNum$.

During a video session, the receiving peer informs each of its supplying peers the number of currently available supplying peers ($NumPeer$). Furthermore, the receiving peer assigns a description number ($DespIdx$) for each of its supplying peers. For example, if a receiving peer has $NumPeer$ available supplying peers, then the $DespIdx$ for each supplying peer ranges from 1 to $NumPeer$. When one supplying peer goes down, if the receiving peer can find a replacement supplying peer quickly, the receiving peer simply assigns that supplying peer the same index used for the one that just disconnected, while keeping the same index for other peers; if the receiving peer can not find a replacement supplying peer in time, then the receiving peer re-assigns $NumPeer$ as $NumPeer - 1$ and re-assigns the $DespIdx$ to its supplying peers from 1 to the new $NumPeer$. For a supplying peer, it can use the following algorithm to determine whether a MD segment should be transmitted or not.

SegmentSelect(NumPeer,DespIdx,SegIdx,LayerIdx)
if (NumPeer<=LayerIdx)
    && ((SegNum+DespIdx)mod NumPeer<LayerIdx)
        the segment will be transmitted
else
        the segment will not be transmitted

Note that as compared with traditional MD-FEC, RFMD trades off storage for upload bandwidth usage. In order to

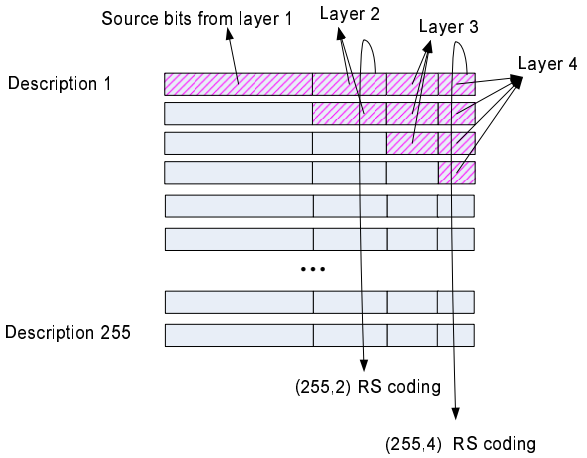

Fig. 6: Coding instead of replicating to generate substreams ($M = 4, N = 255$): collision avoidance

make each substream have equal importance and eliminate transmission redundancy, the RFMD scheme shifts the transmission redundancy of MD-FEC to the storage. Since today upload bandwidth is a scarcer resource than storage (and will likely remain so for the foreseeable future), this tradeoff is appropriate. Compared with the trivial solution of having each peer hold M layers, RFMD consumes much less storage. In the case of the trivial solution, the storage used in a peer is MRT; but in the case of RFMD, the storage is $RT\sum_{k=1}^{M} 1/k$. As M increasing, the relative savings increases.

B. Distinctive RFMD

In general, with a substream coding scheme (layered coding, MD-FEC, and so on), the peers will collectively store multiple copies of the substreams. Consider a peer that wants to view a particular video. At any instant of time, there will be a number of potential supplier peers that have both substreams for this video and available upload bandwidth. However, not all of these suppliers can be used, since some of them may contain identical copies of the substreams. When this situation occurs, we say there are "collisions". In this subsection, we show RFMD can be extended so that collisions are virtually eliminated.

The idea is illustrated in Fig. 6. To generate the parity block from layer k, instead of applying an (M, k) Reed-Solomon (RS) code to k groups of source data to yield $M-k$ groups of parity data, we propose to apply an (N, k) RS code to generate $N-k$ groups of parity data, where N is much larger than M. Therefore, even if M is small, we can obtain N distinctive descriptions. Instead of replicating the M descriptions to peers as with traditional MD-FEC, we distribute the N distinctive descriptions, so that if *any* m (with m not larger than M) descriptions can be found, the first m layers can be decoded. We refer to this technique as *distinctive RFMD*.

With a large N, this distinctive RFMD can greatly reduce the collision probability. If we choose a very large N (equal to the maximum number of supplying peers), we can avoid collision completely. But a larger N also means higher complexity in RFMD encoding and decoding.

TABLE I: Distribution of peer bandwidth

Network type	Uplink bandwidth	Percentage
Ethernet 1	5 Mbps	10%
Ethernet 2	2 Mbps	7%
Cable	220 kbps	62%
DSL	120 kbps	21%

In addition to virtually eliminating replicas and collisions, there is an additional benefit to distinctive RFMD: It makes it easier for the receiving peer to locate a replacement peer. In a P2P VoD system, a receiving peer (or the "system" on the behalf of the receiving peer) can track back-up supplying peers that can be quickly summoned when substreams are lost. Suppose a receiving peer tracks B back-up supplying peers. Without applying distinctive RFMD coding, we would naturally attempt to assign the B back-up supplying peers evenly to M substreams, in which case there would only be B/M back-up supplying peers for each substream. (For layered coding, we would assign more back-up supplying peers to more important layers.) In contrast, with RFMD, any of the back-up supplying peers can be used as replacements for any substream. Therefore, with RFMD coding, a receiving peer needs to track fewer back-up supplying peers. Henceforth, we will take RFMD to designate distinctive RFMD.

V. Simulation Study

In this section, we perform extensive simulations to study the performance of the P2P VoD systems for different coding schemes.

A. Simulation setup

In our simulation, we use a pool of 3000 heterogeneous peers. At any given instant, some of these peers are active (viewing the video) and the remainder are inactive. We assume the end-to-end bandwidth bottleneck is at the access links and not in the Internet core. 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 peers is the supplying peer's upload rate. Table I shows the bandwidth distribution in our simulation. The distribution is based on the findings reported in [22], but we do not include the dial-up users.

We have $J = 30$ 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. Denote the probability that the jth video is requested by $\lambda_j = j^{-(1-\rho)}/I$, where I is the normalization factor 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 [23]. The new requests are modeled as a Poisson process with constant rate λ; we change

the rate to get different average numbers of active peers. The length of each video is T. We assume that user's watching time for a video is uniformly distributed in $[0, T]$: thus the average number of active peers n in the system roughly equals $\lambda T/2$. We assume a peer only contributes its uplink bandwidth to serve other peers when it is viewing a video; when a peer finishes viewing, it leaves the system.

For scalable coding, we code the "Foreman" video sequence in CIF (352x288) resolution with a frame rate of 30 frame/sec into a FGS bit stream using the most advanced SVC codec [24], at a base layer rate of 70 kbps. Each GOP has a duration of 4 seconds. The output bits from each GOP are converted to M substreams for layered coding, MD-FEC and RFMD. To make a fair comparison, each substream has the same transmission rate R. For the single layer coding scheme, we code this "Foreman" sequence into a single layer bit stream with bit rate RM using the H.264 codec JM9.6 [25]. Then we divide each GOP into M blocks and generate M substreams. We pre-compute the operational rate-distortion function for "Foreman" based on SVC and H.264 respectively, and assume that all the 30 videos have the same characteristics as the "Foreman" sequence.

In our simulations, we compare MD-FEC, layered coding, single layer, and RFMD. For MD-FEC, the optimal rate partition [21] is applied to adapt to the node availability (description loss rate) by varying the RS code rate for different segments of the video stream; for layered coding, we give more protection to the more important layers by storing more copies of important layers [18]; for single layer coding, we assume high rate erasure coding is applied to generate a large number of distinctive substreams, so that if a peer gathers any M substreams, it can recover the entire bit stream [14]. The number of parity substreams we assign to each video is proportional to the video popularities. For simplicity, we call this scheme SLRS in our simulation.

For RFMD, each peer only stores one description. Recall that for RFMD, if the transmission rate for one substream is R, the storage for one substream is $RT\sum_{k=1}^{M} 1/k$. For all the other schemes, if the transmission rate for one substream is R, the storage consumed is equal to RT. To make the comparison fair, for MD-FEC, layered coding and single layer, we place $\lceil \sum_{k=1}^{M} 1/k \rceil$ different substreams of one video on each peer.

In our simulation, the video length T is set to 60 minutes, representing TV shows and movies. The test time is 3 hours. For the performance metric, we compare both playback continuity and decoded video quality. For playback continuity, we define the discontinuity ratio, denoted by α, as the percentage of undecodable GOPs for all video sessions:

$$\alpha = \frac{\text{number of undecodable GOPs}}{\text{Total number of GOPs}} \tag{3}$$

For the schemes using scalable coding (MD-FEC, layered coding, RFMD), if the base layer is received, we consider the GOP as decodable and playable. To represent decoded video quality, we use average PSNR averaged over all video sessions. For average PSNR calculation, we assume PSNR=0 for GOPs that are not decodable. We define average PSNR as

$$\text{average PSNR} = \sum_{m=1}^{M} p(m) PSNR(m), \tag{4}$$

where $p(m)$ is the probability of receiving m substreams; $PSNR(m)$ is the average PSNR over a GOP when m substreams are received. Note that for single layer coding, when $m < M$, $PSNR(m) = 0$. Typically an average PSNR gain of 1dB is visually distinguishable.

B. Simulation results

In the simulations, we compare the performance of different coding schemes in several different scenarios. In the first scenario, we suppose that whenever there is an available supplying peer with sufficient uplink bandwidth, the receiving peer can always find it and start streaming before the playback deadline. In this simulation, we set $M = 8$ and $R = 70$ kbps. For RFMD, N is set to 255. For single layer coding with high-rate erasure coding, we assume all substreams are distinctive. It is important to investigate the system performance for different system scales. We vary the request rate λ to change the average number of active peers n in the system.

Fig. 7 (a) shows the discontinuity ratio α versus average number of active peers n. As expected, for all the schemes, as n increases, α decreases, which means fewer discontinuities occur. This indicates that for the upload distributions in Table I, P2P VoD is scalable, with more active peers giving better system performance and individual video quality. Note that RFMD outperforms the other schemes, especially when the average number of active peers is small. The reason is that for RFMD (i) a substream at any available peer can be used to recover the base layer; and (ii) there are no wasted transmission bits (as with MD-FEC). In the case of layered coding, even though we make more copies for the lowest layer, there is no guarantee that the base layer is available to all peers. For single layer coding, all M blocks are required for decoding; when the total available uplink bandwidth of a video is not sufficient to support all the active video sessions, some video sessions will experience severe video quality degradation.

Fig. 7 (b) compares the average PSNR. We see that RFMD always outperforms the other schemes when n is not large. When n is small, video quality for single layer coding is very poor. However, when n is large, single layer achieves the best PSNR performance. This is because single layer coding always has a higher coding efficiency than scalable coding; given that a receiving peer can always gather all data blocks, the decoded video quality should be better.

Thus for this scenario, in terms of both video continuity and decoded video quality, RFMD outperforms MD-FEC and layered coding. While single layer coding has higher average PSNR when n is large, its discontinuity ratio is still higher than the other three schemes. We argue that when PSNR reaches a high level (e.g. above 30 dB), continuous playback becomes more important.

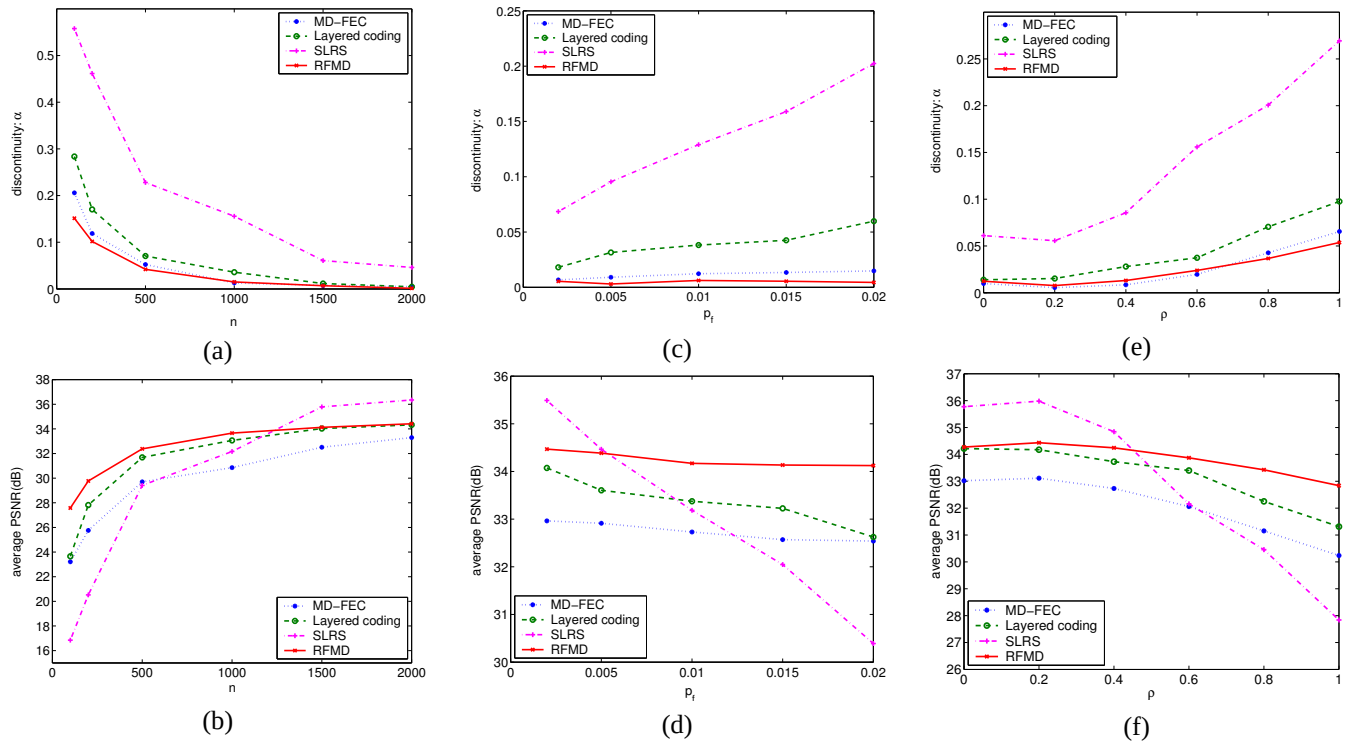

Fig. 7: Comparison of different schemes. (a-b) performance vs. number of active users, assuming $t_r = p_f = 0$; (c-d) performance vs. p_f, assuming $t_r = 4$ seconds,$n = 1500$; (e-f) impact of mismatch between actual video popularity (ρ) and expected popularity (ρ_p), assuming $t_r = p_f = 0, \rho_p = 0.27, n = 1500$.

In the second scenario, we investigate the robustness of the schemes when, with some probability, a receiver can no longer receive the substream that is being supplied by one of its supplying peers (for example, due to congestion, network outage, and so on). We introduce a parameter p_f to represent the supplying peer failure probability. A higher p_f means a receiving peer needs to look for replacement peers more frequently. In this scenario, we also assume a receiving peer cannot find a replacement peer immediately. We introduce another parameter, t_r, to represent the additional time after the playback deadline required to locate a replacement supplying peer and establish the connection. t_r depends on the buffer lengths as well as the technique employed to locate a replacement supplying peer. In this scenario, we set $t_r = 4$ seconds and vary p_f.

Figures 7 (c) and (d) compare the four schemes when $t_r = 4$ seconds under different supplying peer failure probabilities. For this scenario, the average number of active peers is fixed at 1500. Under this scenario, RFMD significantly improves the system performance. It is almost constant when varying p_f. This indicates that the RFMD scheme is robust to supplying peer failures. In terms of discontinuity, the performance of MD-FEC is similar to RFMD; however, its average PSNR is much lower. The reason is that unlike MD-FEC, the RFMD does not transmit any redundant bits. Therefore, for similar transmission rates, RFMD achieves better video quality. In contrast, the performance of layered coding and single layer coding becomes much worse when p_f increases. The reason is that for both layered coding and single layer coding, substreams are highly dependent. When one substream is lost and a replacement is being sought, the decoding of other substreams will be affected and more substreams become undecodable, thus reducing the video quality dramatically.

In all our simulations so far, we have assumed the system knows the popularity of each video precisely and creates the copies (or distinct parity substreams) for the substreams accordingly. However, the popularities of videos are always changing, and the system cannot adapt the number of copies of videos to their popularities immediately. In this third scenario, we want to investigate the robustness of the schemes to the mismatch, where the number of copies of a video does not truly reflect their popularities. In this simulation, we assume the number of copies of each video is Zipf distributed with $\rho_p = 0.27$ but the request rate of each video is varying with different ρ values.

Figures 7 (e) and (f) compare the four schemes when there is a mismatch between video placement and video request popularity. The average number of active peers is 1500. Here we fix $\rho_p = 0.27$ and vary ρ. When ρ is larger, it means the request rates for different videos are more even. One observation is when ρ is close to ρ_p, all four schemes can achieve their best performance, both in the sense of video continuity and average PSNR. However, when video placement does not match the current video request popularity well, such as when

TABLE II: Comparisons between various schemes

	SLRS	Layered coding	MD-FEC	RFMD
Bandwidth adaptation		√	√	√
No collisons	√			√
Redundancy free transmission	√	√		√
Equal importance	N/A		√	√

ρ is increasing, the different schemes have different behaviors. We see that RFMD is the most robust to the mismatch, so that the performance drops smoothly both with respect to α and to average PSNR. In contrast, the single layer scheme is most sensitive to the mismatch, with performance dropping dramatically. The layered coding scheme is better than the single layer scheme, but is still not as good as RFMD. The reason is that when the mismatch occurs, for some particular videos with more requests than expected, the receiving peers cannot find sufficient upload bandwidth. The adaptive coding schemes can adapt and stream some substreams to handle this situation. The reason why RFMD outperforms layered coding is that for RFMD every substream can be used for decoding, so that the available bandwidth for the requested video can be fully utilized; but for layered coding, when the bandwidth for the lower layers is not sufficient in the system, the available upload bandwidth of higher layers cannot be utilized.

We summarize the properties of the four coding schemes in Tab. II. As shown, only RFMD provides adaptation to available bandwidth, no collisions, no redundancy, and equal importance substreams.

VI. Conclusion

In a P2P VoD system, the rate at which peers receive a video fluctuates due to peer churn. Because of this rate fluctuation and because of unexpected supplier peer disconnects, it is natural to consider multi-stream video coding. In this paper we proposed a new multi-stream coding and adaptive transmission scheme, RFMD, that has been specifically designed for P2P VoD systems. RFMD has the following three beneficial features:

- Unlike layered video, all substreams have equal importance. Thus, video quality gracefully degrades as substreams are lost, independently of which particular substreams are lost.
- Only the source bits are collectively transmitted by the supplying peers, so that there is no transmission redundancy. This allows more substreams than what can be supported by other multiple-description schemes.
- When combined with high-rate erasure coding, any combination of M or fewer substreams stored in the system can be used in reconstructing video.

In our simulations, we compared MD-FEC, layered coding, single layer with high-rate erasures, and distinctive RFMD. The single-layer scheme performs poorly whenever there is significant fluctuation in available upload bandwidth or when the number of parity substreams per video is not properly matched to video popularity. Among the scalable schemes, we found that RFMD offers significant advantages in all the representative scenarios.

References

[1] "Youtube." [Online]. Available: http://www.youtube.com/
[2] "Google video." [Online]. Available: http://http://video.google.com/
[3] "Bittorrent." [Online]. Available: http://www.bittorrent.com/
[4] "edonkey." [Online]. Available: http://www.edonkey2000.com/
[5] "pplive." [Online]. Available: http://www.pplive.com/
[6] X. Hei, C. Liang, Y. Liu, and K. W. Ross, "Insights into pplive: A measurement study of a large-scale p2p iptv system," in *Workshop on Internet Protocol TV (IPTV) services over World Wide Web*, Edinburgh, Scotland, May 2006.
[7] "ppstream." [Online]. Available: http://www.ppstream.com/
[8] R. Kumar and Y. Liu and K.W. Ross, "Stochastic fluid theory for p2p streaming," *submitted*.
[9] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proceedings of NOSSDAV*, 2002.
[10] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "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.
[11] T.T.Do, K.A.Hua, and M.A.Tantaoui, "P2vod: providing fault tolerant video-on-demand streaming in peer-to-peer environment," in *Proc. of IEEE ICC*, vol. 3, June 2004, pp. 1467–1472.
[12] Y. Cui, B. Li, and K. Nahrstedt, "ostream: Asynchronous streaming multicast in application-layer overlay networks," *IEEE Journal on Selected Areas in Communication (JSAC)*, vol. 22, no. 1, pp. 91–106, January 2004.
[13] Q. Zhang and H. Chi, "Efficient search in p2p-based video-on-demand streaming service," in *IEEE International Conference on Multimedia and Expo (ICME)*, Toronto, Canada, July 2006.
[14] J. Li, "Peerstreaming: A practical receiver-driven peer-to-peer media streaming system," Microsoft Research, Tech. Rep. MSR-TR-2004-101, September 2004.
[15] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: peer-to-peer media streaming using collectcast," in *Proc. of ACM Multimedia 2003*, Berkeley, CA, November 2003, pp. 45–54.
[16] S. Khan, R. Schollmeier, and E. Steinbach, "A performance comparison of multiple description video streaming in peer-to-peer and content delivery networks," in *IEEE International Conference on Multimedia and Expo (ICME)*, Taipei, Taiwan, June 2004.
[17] X. Xu, Y. Wang, S. S. Panwar, and K. W. Ross, "A peer-to-peer video-on-demand system using multiple description coding and server diversity," in *IEEE International Conference on Image Processing (ICIP)*, Oct. 2004.
[18] Y. Shen, Z. Liu, S. S. Panwar, K. W. Ross, and Y. Wang, "Streaming layered encoded video using peers," in *IEEE International Conference on Multimedia and Expo (ICME)*, Amsterdam, The Netherlands, July 2005.
[19] W. Li, "Overview of fine granularity scalability in mpeg-4 video standard," *IEEE Trans. Circuit and System for Video Technology*, vol. 11, pp. 301–317, Mar., 2001.
[20] H. Schwarz, D. Marpe, and T. Wiegand, "Mctf and scalability extension of h.264/avc," in *Proceedings of PCS*, Francisco, USA, December 2004.
[21] R. Puri and K. Ramchandran, "Multiple description source coding through forward error correction codes," in *33rd Asilomar Conf. Signals, Systems and Computers*, Oct. 1999.
[22] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The feasibility of supporting largescale live streaming applications with dynamic application endpoints," in *Proceedings of ACM Sigcomm*, Portland, USA, August 2004.
[23] 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.
[24] H. Schwarz, D. Marpe, and T. Wiegand, "Joint scalable video model (jsvm) 2," *Joint Video Team, Doc. JVT-O202*, April 2005.
[25] A. Tourapis and K. Sühring and G. Sullivan, "Revised h.264/mpeg-4 avc reference software manual," *Joint Video Team, Doc. JVT-Q042*, October 2005.