Efficient Substream Encoding and Transmission 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 adapt to long-term rate variations, existing scalable video schemes have not been tailored for P2P systems for which substreams emanate from churning 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 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. Thus, all transmitted bits contribute to improve video quality. Finally, RFMD can be used to create any number of descriptions. We conduct an extensive simulation study, comparing single layer coding with high-rate erasure codes, layered coding, multiple description coding (MD-FEC) and RFMD. The simulations show that RFMD performs best in a variety of representative scenarios.**

I. Introduction

Video-on-demand (VoD) has become a killer application in the Internet in recent years. Currently, VoD is distributed to users by server farms and /or Content Distribution Networks (CDNs). In such approaches, dedicated servers are deployed for storing and distributing content to clients. When a user requests a video, the system redirects the client to one or more of its dedicated servers, which stream the stored video to the client. Such solutions have significant infrastructure costs, not only for the dedicated servers they deploy, but also for the access bandwidth they consume.

Peer-driven video streaming is an alternative architecture for VoD service. In a peer-driven VoD system, all peers are Internet-connected hosts, which store and stream the video to the requesting clients. The cost of these peers and Internet access would be borne by the users rather than by infrastructure companies providing the service. Because there is an abundant supply of potential supplying peers with excess unused bandwidth and storage, peer-driven architectures should have costs that are significantly less than the traditional client-server and CDN solutions.

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).

However, in peer-driven VoD systems, the rate at which a peer can receive a video will fluctuate because of peer churn [1]. Indeed, every active peer in the system is both a consumer and a supplier of upload bandwidth, with different peers (residential peers, institutional peers, and so on) contributing different amounts of upload bandwidth. As peers come and go, the ratio of the upload-bandwidth supply to the upload-bandwidth demand for a given video fluctuates on both long-term and short-term time scales. As with traditional client-server streaming, buffering 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 video at a rate less than the compressed video rate during the bandwidth deficient period.

Therefore, to deal with long-term variations in available bandwidth, scalable multi-substream coding is a natural candidate for P2P VoD system. Such multi-substream coding techniques include layered coding and multiple description coding (MDC). Since the receiving peers receive video from multiple "unreliable" peers, the design of multi-stream coding and delivery schemes for P2P VoD system brings forth many new challenges. In particular, layered coding is vulnerable to peer disconnects due to the recursive dependency of layers. MDC, while giving equal importance to each substream, typically introduces significant redundancy across streams.

In this paper, we propose a new multi-stream coding and transmission scheme that has been designed for P2P VoD systems. We refer to this scheme as Redundancy-Free Multiple Description Coding and Transmission, or more simply as RFMD. After describing RFMD, we carry out an extensive simulation study that compares single layer coding with high-rate erasure codes, layered coding, a well-known MD coding technique known as MD-FEC [2] and RFMD under a variety of representative scenarios. Careful attention is placed in making the comparisons as fair as possible. We find that RFMD provides the best overall performance as compared to other coding schemes.

This paper is organized as follows. In Section II, we present a short overview of the P2P VoD context, which forms the basis of our study. Section III describes the

traditional multi-stream coding and transmission schemes. Section IV describes the proposed RFMD scheme. 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 designs for P2P VoD, and for each design the distribution, replication, and search for video substreams can be done differently. Here, we describe a broad P2P VoD context which focuses on 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. Since a peer has limited storage, it may not store every substream of every video. The video substreams could be scattered over the peers in an on-demand fashion, or could be assigned by a placement algorithm. The placement of the substreams onto the peers is orthogonal to the video delivery problem, and is not be considered further.

When a peer wants to see a video (the receiving peer), the "system" provides the peer with a list of "active" peers containing one or more substreams (the supplying peers). The receiving peer then requests and receives substreams from one or more of the supplying 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. After a small playback delay, the receiving peer assembles, decodes, 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 supplying peer that stores a missing substream. It may lose a substream because a supplying peer may stop delivering the substream. Generally, as the ratio of the available upload capacity of the supplying 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 will be forced to drop substreams.

When a receiving peer loses a substream, it will naturally attempt to find a replacement supplying peer that can provide the lost substream. Even if such a supplying peer is present, there will be a delay until the receiving 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 and transmission scheme for P2P VoD, we therefore have the following objectives:

- The 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 supplying peer suddenly stops providing a substream, and the receiving peer does not have a sufficient prefetched reservoir of bits of that substream buffered locally, quality degradation should be minimal until a substitute supplying peer with the same substream begins to deliver the substream.
- It should carry little or no redundancy so that all transmitted bits contribute towards improved video quality.

A. Related work

In recent years, a significant amount of research efforts have been directed toward peer-driven video streaming, e.g., [3]–[9] and the references therein. However, to date, only a few groups have considered applying multi-stream coding for improving system performance. CoopNet is designed for live streaming and uses MD-FEC to code the source into several sub-streams [5]. 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 to combat peer failure, but they do not focus on the design of an efficient MD coding scheme for P2P video streaming. In PeerStreaming [4], the author proposes to use high-rate erasure coding to generate parity substreams instead of replicating directly. However, the original media stream and corresponding coded stream is non-scalable, so that receiving less than M blocks will make the video undecodable. In [10], the authors compare the performance of multiple description streaming in P2P networks and CDNs. However, they only consider using two descriptions which limits the achievable performance. In our previous work [11], [12], we compared the performance of MD-FEC and layered coding for a P2P VoD system. In this paper, we develop a multi-stream video coding and transmission scheme based on scalable video for P2P VoD.

III. Traditional Substream Generation

To motivate our proposed RFMD scheme, we first review several existing approaches for generating substreams and discuss the problems with these schemes.

A. Single layer coding with high-rate erasure codes

Single layer coding with high-rate erasure coding (for brevity, SLRS) [4] is based on a non-scalable video stream. The media stream is broken up into "data units," so that each "data unit" is independently decodable. For example, a data unit can be a group of pictures (GOP).

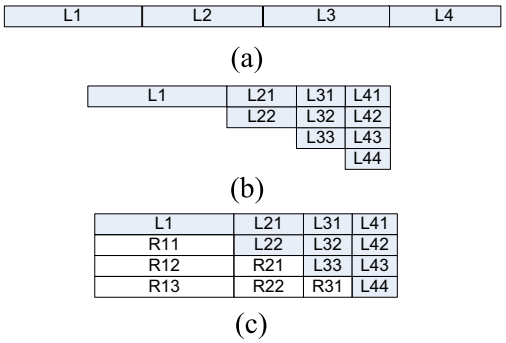


Fig. 1. MD-FEC encoding procedure.

A data unit is then divided into M blocks and high-rate Reed-Solomon (RS) coding is employed to generate $N - M$ parity blocks, resulting a total of N blocks for each data unit, with $N \gg M$. Each block is assigned a unique block number. The blocks with the same block number from different data units can be grouped to generate a substream. These N substreams are distributed to up to N supplying peers. The receiving peer can recover the original video stream from any M substreams. However, a data unit is non-decodable if less than M blocks are received for this unit.

B. Layered coding

Layered coding generates multiple layers for each data unit with recursive dependency. Specifically, layer $k+1$ can only be decoded if layers 1 through k are available. MPEG-4 Fine Grain Scalable (FGS) [13] 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 [14], also has a base-layer and a successively 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, and that the enhancement layer can be coded efficiently. Layered coding is generally less efficient than single-layer coder, in that, at the same total rate r, the distortion achievable is larger than that by a single layer coding. This is true for both MPEG-4 FGS vs. MPEG-4 and SVC vs. H.264.

C. Multiple Description FEC (MD-FEC)

MD-FEC is a popular scheme for multiple description encoding with many descriptions. We now briefly explain MD-FEC coding [2], as it forms the basis of our RFMD scheme, which is described in the next section. As shown in Fig. 1, the original scalable bitstream from each data unit (e.g., a GOP) is partitioned into M layers. We define the bitrate of layer k as R_k. For $k = 1, \ldots, M$, 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 per layer. 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 for a data unit, 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 $\{R_1, R_2, \ldots, R_M\}$. Given the description loss distribution, an optimal layer partition can be performed to minimize the expected video distortion [2].

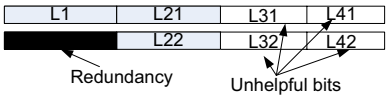


Fig. 3. The redundancy and unhelpful bits in MD-FEC when only two descriptions are available.

D. Inadequacies of existing schemes

Figures 2 (a), (b) and (c) illustrate how the SLRS, layered coding, and MD-FEC 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 for a data unit. The black rectangles indicate the substreams that are not available. The white rectangles indicate the substreams that are received but cannot be decoded. The shaded area in the rectangle indicates the portion of data that can be decoded and contributes to video quality.

For the SLRS scheme, as shown in Fig. 2 (a), if a receiving peer can only locate three supplying peers for a data unit, the received data from the three supplying peers are undecodable. For layered coding, as shown in Fig. 2 (b), although the receiving peers can find three peers, since layer 2 is unavailable, thus make layer 3 and layer 4 useless, the decoder can only make use of layer 1 from supplying peer 1. As illustrated in Fig. 2 (c), 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.

Figure 2 (d) shows the ideal coding scheme. If the receiving peer can locate *any* m ($m < M$) peers, each holding a different substream, it can decode the first m layers. Moreover, all of the bits transmitted by the supplying peers are useful; there is no wasted bandwidth. In the next section, we design such an ideal coding and transmission scheme for a P2P VoD system.

IV. Redundancy-Free MD-FEC Scheme

Our RFMD scheme is based on four observations on MD-FEC.

- In several proposed P2P video streaming systems, MD-FEC is adopted to handle supplying peer disconnect rather than to combat packet loss due to congestion [5], [10], [11]. Indeed, in the most popular P2P video streaming systems [4], [6], [15]–[17], the video bits are transmitted over TCP connections. Usually, a peer maintains a buffer of pre-fetched video bits to combat network jitter and short-term rate variations.

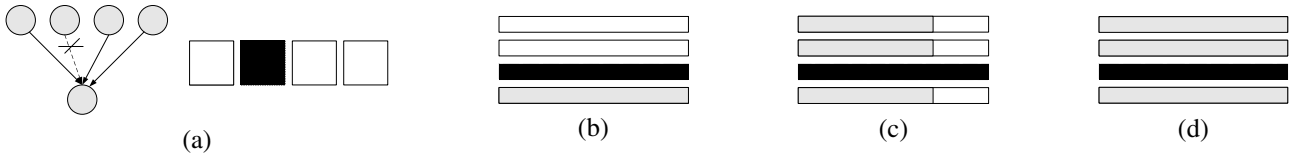


Fig. 2. Comparison of different coding schemes for P2P VoD system (a)SLRS; (b) Layered coding; (c)MD-FEC; (d) The desired scheme.

When a packet is lost, it can be retransmitted before the playback deadline. Therefore, it is not unreasonable to assume that packet loss can be handled by TCP and buffering.

- For MD-FEC, to make each description have equal importance, *redundant bits* are transmitted. Furthermore, some *unhelpful bits* are also 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.
- The unhelpful bits are not useful for handling sudden peer disconnects. When a supplying peer disconnects, more bits in each transmitted substream become unhelpful.
- Although the redundant bits are useful when a supplying peer disconnects during the streaming session and substitute streams can not be quickly found, they do not have to be always transmitted. VoD does not have as rigid a delay constraint 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 transmission pattern. With this approach, it is a receiving peer that pulls the required data from the supplying peers. Then the supplying peers push the video data based on the new transmission pattern. We note that the push-pull approach introduces additional feedback for signaling compared with a pure push approach. However, we believe that such feedback is feasible in P2P VoD application. Indeed, this push-pull approach has been successfully used in P2P live streaming [18], [19]. Compared with the most popular pure mesh-pull approach [6], [15]–[17], the overhead of feedback can be suppressed by using the push-pull.

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

We now describe our basic Redundancy-Free MD (RFMD) coding and transmission scheme. This scheme generates M descriptions using the same method as MD-FEC from a scalable bit stream. But instead of sending all the bits in a description, the supplying peer transmits only a portion of the description, chosen based on feedback from the receiving peer regarding the total number of available supplying peers. The coding and transmission procedure is as follows:

- We adopt MD-FEC coding to generate M descriptions.

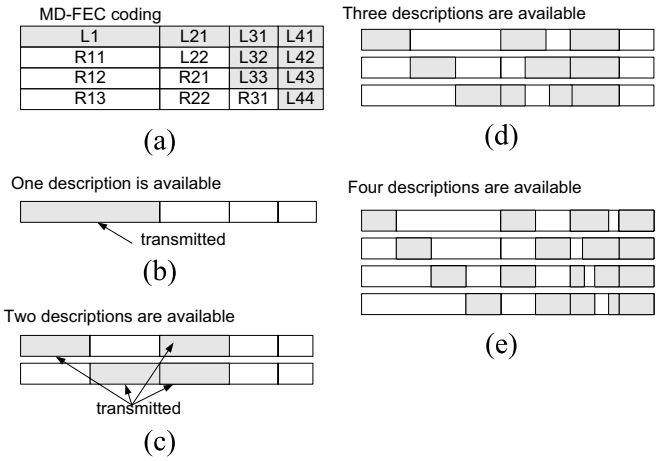


Fig. 4. Redundancy-free transmission of MD-FEC data (M=4): (a) shows the stored data in each description, with the shaded area representing the source bits and the blank area the redundancy bits; (b-e) shows the portion of the data (shaded area) delivered by each supplying node.

- If m $(1 \leq m \leq M)$ supplying peers are available, then each supplying peer only transmits a fraction k/m of the data for layer k, where $k = 1, \ldots, m$; each supplying peer transmits different portions of the layer k data.
- 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.

Figure 4 shows an example when $M = 4$. The shaded areas indicate the portion of data that will be transmitted. For example, if three supplying peers 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 of the 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. Having defined the basic RFMD, we now describe several critical refinements.

A. Equal Layer Partition for Constant Transmission Bitrate

Because the portion of data transmitted at each supplying peer depends on how many peers are available, in general, the transmission rate at each peer is not constant. A constant bit rate can be achieved with the appropriate layer 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

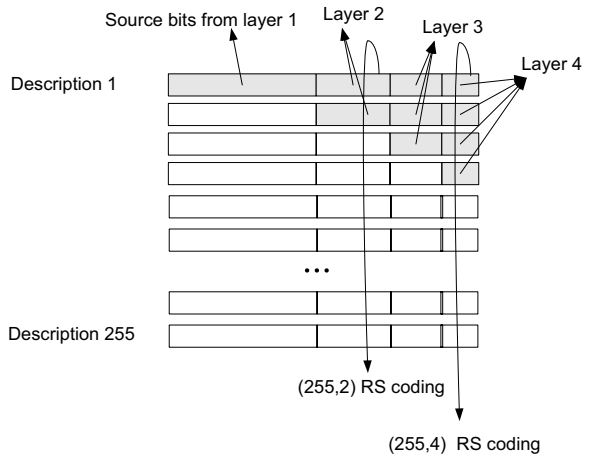


Fig. 5. Coding instead of replicating to generate substreams ($M = 4, N = 255$): collision avoidance

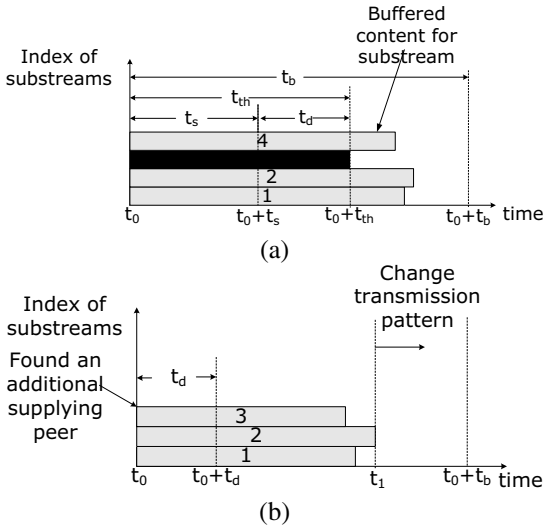


Fig. 6. Timing and buffering of RFMD. (a) supplying peer failure occurs; (b) new supplying peer is located.

rate, i.e., $R_k - R_{k-1} = R$ 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. Also, when there are only m supplying peers available, only a fraction k/m of layer k, $1 \leq k \leq m$, is transmitted in a description. 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 layer 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.

B. Distinctive RFMD

In general, with a multi-stream coding scheme (layered coding, MD-FEC, etc.), 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 supplying peers that have sufficient uplink bandwidth and have stored substreams for this video. However, not all of these supplying peers can be used, since some of them may contain identical copies of the substreams. We call this situation a "collision". In this subsection, we show RFMD can be extended so that collisions are virtually eliminated.

The idea is illustrated in Fig. 5. 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 each of the M descriptions N/M times and distribute them to N peers, we distribute the N distinctive descriptions, so that if *any* $m \leq M$ 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 will lead to higher complexity in encoding and decoding.

In addition to virtually eliminating 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 distinctive RFMD, any of the back-up supplying peers can be used as replacements for any substream. Therefore, with distinctive RFMD coding, a receiving peer needs to track fewer back-up supplying peers. Henceforth, we will consider only distinctive RFMD and simply call it RFMD.

C. Handling peer disconnection

We use buffering to deal with the supplying peer disconnection. Define the maximum prefetching buffer length to be t_b seconds. If the prefetching buffer content of a substream is less than t_{th}, $t_{th} < t_b$, then the supplying peer serving this substream is considered to be disconnected, and the receiving peer starts looking for a replacement peer. As shown in Fig. 6 (a), the playback time is at t_0 and the buffer content of substream 3 is less than t_{th}. Therefore, the receiving peer considers that the supplying peer serving substream 3 is disconnected.

Define t_s to be the time for finding a replacement peer. If within time t_s a replacement peer is found, the receiving peer notifies this replacement peer of the number of available supplying peers, and then the replacement peer can decide how to transmit the substream starting at time t_{th}. We assume the time needed to notify the replacement peer is less than $t_d = t_{th} - t_s$. The remaining supplying peers (serving substreams 1,2 and 4 as in Fig. 6 (a)) remain unchanged.

If after t_s, no replacement peer is found, then the receiving peer needs to notify its remaining supplying peers of a peer disconnection (while continue its search for a replacement peer). Assume t_d to be the delay for notifying the supplying peers. Assume in Fig. 6 (a), no replacement peer is found after t_s, then the receiving peer will notify supplying peers 1, 2 and 4 that a supplying peer is disconnected and now there are three supplying peers available. After the supplying peers receive the message, they will adjust their transmissions accordingly (transmitting substreams based on the configuration of three available supplying peers from time $t_0 + t_{th}$). Given the buffer length t_b, a larger t_{th} can induce a faster detection of supplying peer failure, and leave a longer time for a receiving peer to find a replacement supplying peer and to notify the remaining supplying peers; but correspondingly a smaller $t_b - t_{th}$ may not be long enough to handle the short term bandwidth fluctuation, e.g., due to congestion, and incorrectly treat the jitter as supplying peer failure. Given t_{th}, with a larger t_s, a receiving peer has more of a chance to find a replacement supplying peer; but the time left for notifying the remaining supplying peers $t_d = t_{th} - t_s$ is shorter, such that the probability that a supplying peer does not receive the notification in time becomes higher. As long as $t_d = t_{th} - t_s$ is larger than the required round trip time and can notify the remaining supplying peers in time, RFMD can adapt to the supplying peer disconnection and there is no severe quality degradation.

If during a streaming session, an additional supplying peer is found, the receiving peer notifies its supplying peers of the number of available supplying peers and the time from which the supplying peers should change its substream transmission pattern based on the new number of available supplying peers. We explain the procedure with an example. As shown in Fig. 6 (b), the playback time is at t_0, and the buffer contents of the three substreams are shown in Fig. 6 (b). The longest prefetching buffer content is at time t_1 (substream 2). An additional supplying peer becomes available at t_0, then at this time, the receiving peer sends its supplying peers (serving substreams 1, 2, 3 and the additional supplying peer) a message $(4, t_1)$, which tells the supplying peers the number of available supplying peers and from which time instant in the video bit stream the supplying peer should change its transmission pattern (switching to four supplying peers transmission pattern). Upon receiving the message, each supplying peer can decide how to transmit the substreams as described in the previous section. Note that the reason we choose to let the supplying peers to change their transmission patterns at time t_1 is that the prefetched content in the buffer will not be wasted.

V. Simulation Study

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

A. Simulation setup

In our simulation, we use a pool of 3000 heterogeneous peers, with each peer storing one or several substreams from a single video. The videos are pre-stored on the peers in our simulated system. At any given instant, some of these peers are active (viewing the video) and the remainder are inactive. We assume a peer only contributes its uplink bandwidth to serve other peers with its stored video when it is viewing a video; when a peer finishes viewing, it leaves the system. Thus, a receiving peer for a particular video can locate several active peers storing this video and streams the substreams from the supplying peers. In the mean time, the receiving peer itself becomes a potential supplying peer. When a supplying peer disconnects, the receiving peer keeps searching for a replacement peer, until an active peer which has the desired substream and has enough uplink bandwidth is found. Receiving peers compete for the uplink bandwidth with each other and intend to stream as many substreams ($\leq M$) as possible.

The new requests for all videos are modeled as a Poisson process with constant rate λ; we change the rate to get a different average number of active peers. The length of each video is T. We assume that a 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 $\lambda T/2$. In our simulations, the video length T is set to 60 minutes, representing TV shows and movies. The simulated test time is 3 hours.

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 uplink bandwidth distribution in our simulation. The distribution is based on the findings reported in [20], 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 popularity of video j by $\beta_j = j^{-(1-\rho)}/I$, where I is the normalization factor and ρ is a control parameter. Thus, the request rate for video j is $\lambda\beta_j$. We choose the number of copies of each video to be proportional to its

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%

TABLE II

VIDEO CODING PARAMETERS

Sequence	Foreman	Mobile
Number of substreams (M)	8	8
Bitrate per substream (R)	70 kbps	120 kbps
SVC base layer rate	70 kbps	120 kbps
SVC FGS bitrate range	70 kbps - 560 kbps	120 kbps - 960 kbps
PSNR at full bitrate by SVC FGS	36.2 dB	32.4 dB
H.264 bitrate	560 kbps	960 kbps
PSNR at full bitrate by H.264	38.1 dB	33.8 dB

popularity. In our simulations, we choose $\rho = 0.27$, which is a commonly used value for video on-demand services.

To create a scalable video, we code a 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 [21] with the SNR scalable mode. 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. For a particular video sequence, each substream has the same transmission rate R. For SLRS, we code the same sequence into single layer bit streams with bit rate RM using the H.264 codec JM9.6 [22]. Then we divide each GOP into M blocks and generate M substreams. In the case of SLRS and RFMD, we further generate $N - M$ ($N \gg M$) parity substreams from the M substreams. We make use of the operational rate-distortion function $D(r)$ for the test sequence reported in [14] based on SVC and H.264 respectively, and assume that all the 30 videos have the same characteristics as the tested sequence. In this simulation, two sequences ("Foreman" and "Mobile") are tested. These two sequences have different characteristics and require different total bitrates ($= MR$) to obtain a satisfactory quality. In this way, we can examine the impact of the video bit rate on the system performance. The detailed coding parameters are shown in Table II. We choose $M = 8$ to achieve a good balance between the overhead required to manage substreams and network utilization. With these coding parameters, the substream transmission rate R is close to the uplink bandwidth of the slowest supplying peers (those using DSL connection). These peers can at most supply one substream.

In our simulations, we compare SLRS, layered coding, MD-FEC, and RFMD. For layered coding, we give more protection to the more important layers by storing more copies of important layers [12]; for MD-FEC, the optimal layer partition [2] 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 RFMD and SLRS, we assume all substreams are distinct, which is true when N is larger than the number of peers storing a given video.

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 SLRS, layered coding and MD-FEC, we place $\lceil\sum_{k=1}^{M} 1/k\rceil$ (e.g., when $M = 8$, $\lceil\sum_{k=1}^{M} 1/k\rceil$=3) different substreams of one video on each peer. The storage used for different schemes at each peer with different video sequences is shown in Table III.

With our simulation setup, the four schemes consume similar amount of storage and transmission bandwidth. Thus, we can use the received video quality as a performance metric for comparing the different schemes. For evaluating the video quality, we use both playback discontinuity and average PSNR. The discontinuity, denoted by α, is defined 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 decodable for a GOP, we consider the GOP as decodable and playable. But for SLRS, a block is not decodable if less than M substreams are received. 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

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

where $p(m)$ is the probability of receiving m substreams,

TABLE III
STORAGE USED FOR DIFFERENT SCHEMES PER SUPPLYING PEER

	Foreman	Mobile
SLRS, Layered coding, MD-FEC	94.5 MB	162 MB
RFMD	85.6 MB	148 MB

which is determined from simulation; PSNR(m) is the average PSNR over a GOP when m substreams are received. Note that for SLRS, when $m < M$, PSNR(m) $= 0$; for layered coding and RFMD, PSNR(m) $= 10 \log \frac{255^2}{D(mR)}$; and for MD-FEC, PSNR(m) $= 10 \log \frac{255^2}{D(R_m)}$, where R_m depends on the layer partition. Note that an average PSNR gain of 1dB is typically visually distinguishable.

B. Simulation results

In the simulations, we compare the performance of different schemes in several different scenarios. Since the focus of our research is on efficient coding and transmission schemes instead of peer location, in all our simulations, we assume that after a supplying peer disconnects, whenever there is an available supplying peer with sufficient uplink bandwidth, the receiving peer can always find it and starts streaming before the playback deadline.

1) Scenario 1: It is important to investigate how the system performance varies with the number of active peers. We vary the request rate λ to change the average number of active peers n in the system. In this scenario, we use the "Foreman" sequence with a total transmission rate of 560 kbps. Based on Table I, the average available bandwidth is about 800 kbps. This scenario is intended to examine the system performance when the network is underloaded.

Figure 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 and MD-FEC have lower discontinuity than other two schemes, especially when the average number of active peers is small. The reason is that for RFMD a substream at any available peer can be used to recover the base layer. For MD-FEC, the discontinuity ratio is slightly higher than RFMD. The reason is based on the optimal layer partitions, only receiving one description may not recover the base layer. 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 SLRS, all M blocks are required for decoding; when the total available uplink bandwidth of a video is not sufficient to provide M substreams for all video sessions, some video sessions will experience severe video quality degradation.

Figure 7 (b) compares the average PSNR achievable by different schemes. We see that RFMD always outperforms the other schemes when n is not large. When n is small, video quality for SLRS is very poor. However, when n is large, SLRS achieves the best PSNR performance. This is because H.264 coding always has a higher coding efficiency than SVC[1]; given that a receiving peer can always gather all data blocks, the decoded video quality should be better. Note that although MD-FEC is similar to RFMD in terms of discontinuity, its average PSNR is significantly lower. This is because the transmitted bits with MD-FEC contain many redundant and unhelpful bits. Note that the performance of MD-FEC, in terms of PSNR, depends on M. When M is large, less redundancy is introduced (with optimal layer partition). In our previous work [12], we have shown that MD-FEC is better than the layered scheme when M is large. In the simulation presented here, we choose a relatively small M, as we think larger M may not be feasible in practical systems.

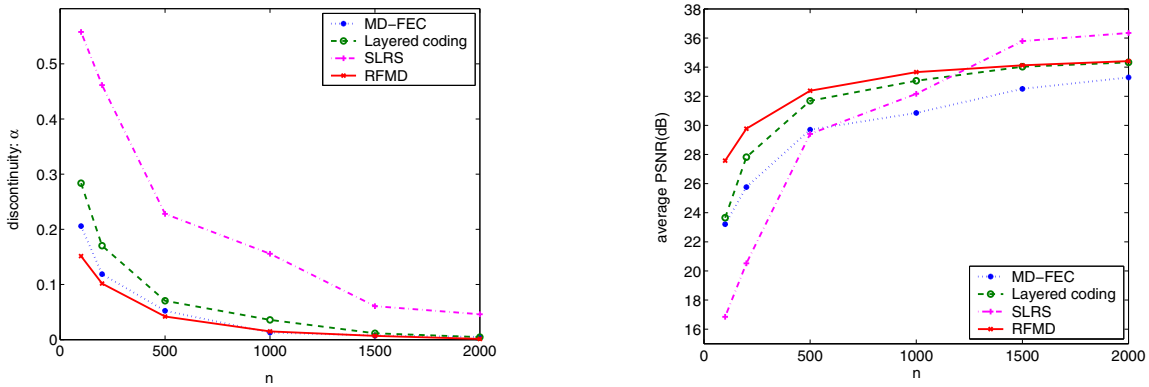
Thus for this scenario, in terms of both video continuity and decoded video quality, RFMD outperforms MD-FEC and layered coding. While SLRS has higher average PSNR when n is large, its discontinuity ratio is still higher than the other three schemes. We argue that from a human perception point of view,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

2) Scenario 2: In this scenario, we adopt a higher bitrate video stream of 960 kbps corresponding to the "Mobile" sequence. Thus, the network is overloaded. Similar to scenario 1, we also assume that the number of copies of each video matches its popularity perfectly.

Figures 8 (a) and (b) compare the four schemes in this scenario. Similar to Figures 7 (a) and (b), RFMD outperforms MD-FEC and layered coding. As before, RFMD and MD-FEC are similar in terms of discontinuity, but RFMD has much higher average PSNR. The layered scheme has worse discontinuity than MD-FEC, but better PSNR. Unlike the lower bitrate scenario, the video quality of SLRS is significantly worse than other schemes even with n is large. When the network is overloaded, the uplink bandwidth in the system is insufficient to support all video sessions, so that many viewers receive less than M substreams. With SLRS, receiving less than M substreams is often equivalent to receiving nothing and hence the PSNR is very low. The scheme based on single layer coding cannot adapt to the available uplink bandwidth, so that the video quality degrades severely.

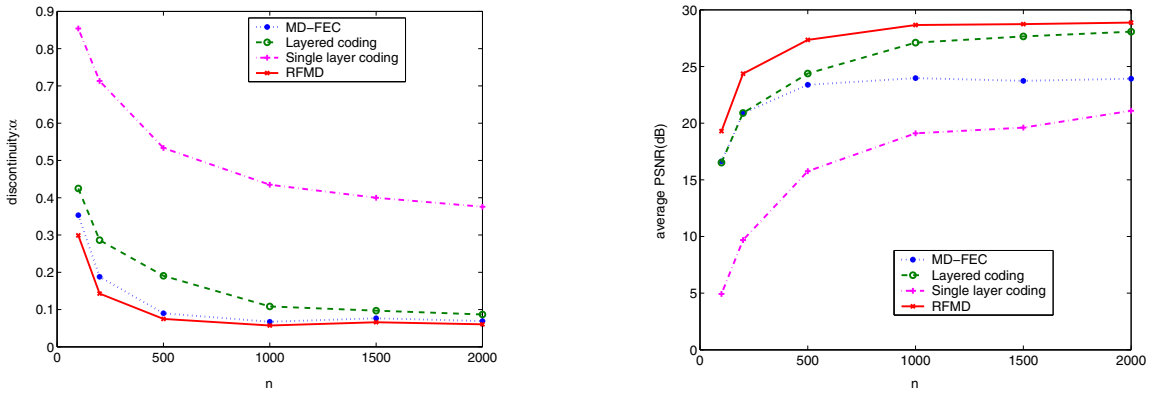
3) Scenario 3: In the previous two simulations, we have assumed that the system knows the popularity of each video precisely and creates the copies (or distinct parity substreams) for the substreams accordingly. In

[1]As shown in Table II, at the same total bit rate, SVC has a PSNR that is 1-2 dB lower than H.264



(a) (b)

Fig. 7. Performance of different schemes vs. number of active users under a low transmission rate, where the "Foreman" sequence is adopted.



(a) (b)

Fig. 8. Performance of different schemes vs. number of active users under a high transmission rate, where the "Mobile" sequence is adopted.

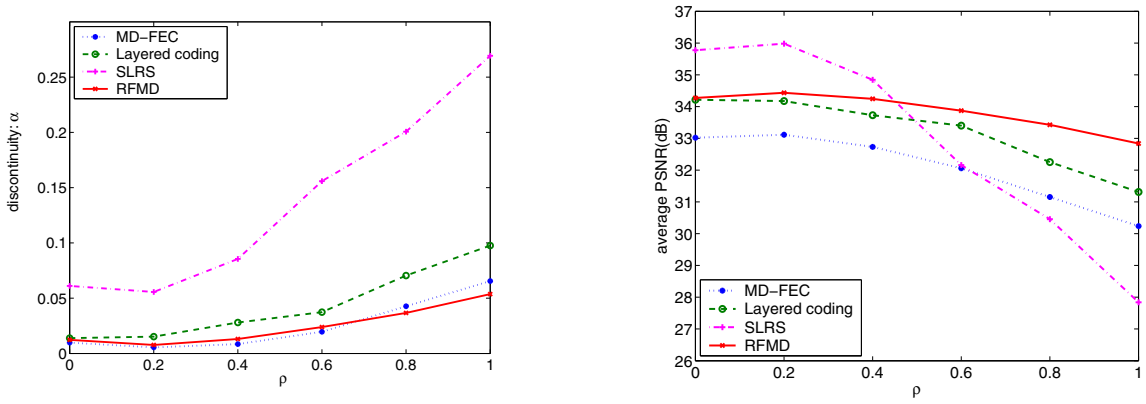
practice, the popularities of videos are time varying, and the system may not be able to adapt the number of copies of videos to their new popularities immediately. In this third scenario, we want to investigate the robustness of the schemes to this mismatch in the assumed video popularity by the system and the actual popularity. In this simulation, we assume the number of copies of each video is Zipf distributed with ρ_p, but the actual request rate of each video follows a Zipf distribution with parameter ρ.

Figures 9 (a) and (b) compare the four schemes when $\rho_p = 0.27$, and ρ varies over a large range. The average number of active peers is 1500. 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 are at or close to 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 ρ is increasing, the different schemes have different behavior. 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 SLRS scheme is most sensitive to the mismatch, with performance dropping dramatically. The layered coding scheme is better than SLRS, but is still not as good as RFMD. 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.

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 supplying 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 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. Thus, all bits transmitted are used for improving video quality.
- When combined with high-rate erasure coding, any combination of M or fewer substreams stored in the



(a) (b)

Fig. 9. Impact of mismatch between actual video popularity (ρ) and expected popularity (ρ_p). $\rho_p = 0.27, n = 1500$. The "Foreman" sequence is adopted.

system can be used in reconstructing video.

It is important to recognize that RFMD's superior performance is obtained at a higher implementation cost, requiring a feedback from the client to each supplying peer and an adaptation of the transmitted substream by the supplying peer, whenever the number of available supplying peers changes. We argue however that such feedback is feasible in typical P2P VoD applications, and the operations required for the substream adaptation is quite simple. A benefit of MD-FEC is that it provides robustness to both peer disconnect and packet losses. In the pursuit of a better utilization of transmission bandwidth, RFMD loses robustness against packet losses. However, we believe packet losses can be more efficiently handled by retransmission and buffering at the receiver. In fact, all current popular P2P streaming systems employ TCP for bit stream delivery.

REFERENCES

[1] R. Kumar and Y. Liu and K.W. Ross, "Stochastic fluid theory for p2p streaming," in *Infocom*.

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

[3] 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.

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

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] H. Schwarz, D. Marpe, and T. Wiegand, "Mctf and scalability extension of h.264/avc," in *Proceedings of PCS*, Francisco, USA, December 2004.

[15] PPLive, "http://www.pplive.com/."

[16] PPStream, "http://www.ppstream.com/."

[17] X. Hei, , C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-Scale P2P IPTV system," in *IEEE Trans. on Multimedia*, to appear.

[18] M. Zhang, J. G. Luo, L. Zhao, and S. Q. Yang, "A peer-to-peer network for live media streaming using a push-pull approach," in *Proc. of the 13th annual ACM international conference on Multimedia, 2005*.

[19] B. Li, S. Xie, and G. Y. Keung, "Inside coolstreaming: Principles and measurements," submitted.

[20] 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.

[21] H. Schwarz, D. Marpe, and T. Wiegand, "Joint scalable video model (JSVM) 2," *Joint Video Team, Doc. JVT-O202*, April 2005.

[22] A. Tourapis, K. Sühring and G. Sullivan, "Revised H.264/MPEG-4 AVC reference software manual," *Joint Video Team, Doc. JVT-Q042*, October 2005.