Stored Media Streaming in BitTorrent-like P2P Networks

Kyung Wook Hwang*, Vishal Misra†, and Dan Rubenstein*
*Department of Electrical Engineering †Department of Computer Science
Columbia University
New York, NY, USA
Email: {kwhwang, danr}@ee.columbia.edu, misra@cs.columbia.edu

Abstract

Peer-to-peer (P2P) networks exist on the Internet today as a popular means of data distribution. However, conventional uses of P2P networking involve distributing stored files for use after the entire file has been downloaded. In this work, we investigate whether P2P networking can be used to provide real-time playback capabilities for stored media. For real-time playback, users should be able to start playback immediately, or almost immediately, after requesting the media and have uninterrupted playback during the download. To achieve this goal, it is critical to efficiently schedule the order in which pieces of the desired media are downloaded. Simply downloading pieces in sequential (earliest-first) order is prone to bottlenecks. Consequently we propose a hybrid of earliest-first and rarest-first scheduling - ensuring high piece diversity while at the same time prioritizing pieces needed to maintain uninterrupted playback. We consider an approach to peer-assisted streaming that is based on BitTorrent. In particular, we show that dynamic adjustment of the probabilities of earliest-first and rarest-first strategies along with utilization of coding techniques promoting higher data diversity, can offer noticeable improvements for real-time playback.

1. Introduction

In a peer-to-peer (P2P) network, there is no clear distinction between client and server nodes. In fact, every node (or peer) on the network that requires resources from the network also contributes resources to the network. This cooperative arrangement means that as more peers enter the network, more resources such as CPU, storage, and bandwidth become available for other peers. Using this scheme, P2P networks provide superior resilience and scalability when compared to classic client-server networks. In recent years, existing P2P networks have been typically used to distribute stored media. "Stored media" refers to content that has been previously recorded and encoded. Apple's iTunes Music Store (ITMS) [15] serves as an example of a centralized stored media system. In this example, content is available in its entirety and is stored on a central server on the network. However, existing P2P networks such as Kazaa [1], eDonkey/eMule [11], and BitTorrent [9] are not capable of allowing recipients to playback the stored media until the entire file has been downloaded.

With real-time playback, users can start playback immediately or almost immediately after requesting the media content. Pieces which form the media content file are constantly being downloaded as the media content is being played. If the playback reaches a point where the piece has not been downloaded yet, it must stop and wait for the piece to be downloaded. For media streaming which provides real-time playback, there are generally two different kinds: *live media streaming* and *stored media streaming* [25]. A network such as CoolStreaming and Pplive described in [28], [2] offers live media streaming. It is well suited for distributing or broadcasting live content such as live sporting events or live news reports. In this scenario, all participating users are interested in the same (most recent available) portion of the media content because they are all watching the same point of the live content. Therefore, the download scheduling of the content file pieces is simple for live media streaming. The network is generally distributing the most current piece to all the users. In this situation, most peers should be trading the same pieces.

On the other hand, stored media streaming is previously recorded media that is immediately available for real-time playback. With stored media streaming each user can be watching different parts of the stream at any arbitrary moment unlike live media streaming. Also, since the media is prerecorded, users can get a later part of the stream into their buffers in advance before it is needed so that their stream will not stop in the middle. Thus, the specific pieces of the media content a given user will currently have, or need, at any given point in time will likely be different than that of any other user. During playback, remaining portions of the

content are located on the network and downloaded. The resulting user experience is similar to TiVo [3], where stored media is available for playback immediately upon request.

In this work, we explore through simulation whether P2P networking technology can be used to provide real-time playback capabilities for stored media. Specifically, we adopt a currently very popular and prototypical P2P network called BitTorrent [9] as P2P system environment for our entire simulations. Therefore, the overall P2P network mechanism and all the behaviors of each peer in our simulator are basically the same as those in the current BitTorrent system except for the download scheduling strategy.

In a BitTorrent network, a peer uses a combination of Local Rarest First (LRF) policy and a random scheme [6], [20] for the download scheduling strategy to both ensure a uniform dissemination of file pieces and prevent peers from waiting too long to find the last missing pieces. However, this is adequate only for distributing a complete file first because all parts of a file need to be obtained before it can be played. The download scheduling strategy of BitTorrent aims only to minimize the amount of time it takes for peers to get the complete file, and this strategy fails to emphasize the need to have the earlier pieces before the later pieces. As a result, the network may use its resources to download later pieces when the playback is interrupted because an earlier piece is not available. On the other hand, in order to support real-time or uninterrupted playback, peers may choose to use earliest-first, which favors downloading the earlier parts of a file first. However, this strategy may neglect exceptionally rare pieces on the network and result in bottlenecks later in the download process.

To solve these piece selection or download scheduling problems, this paper makes the following contributions:

1. We propose a probabilistic hybrid strategy of earliest-first and rarest-first. Instead of using earliest-first or rarest-first by itself, whenever a peer selects the next piece to download, with probability p it chooses a piece using earliest-first strategy. Otherwise (with $1 - p$), it chooses a piece using rarest-first strategy. This hybrid strategy enables us to take advantage of both of each piece download strategy. It combines the real-time playback capabilities of earliest-first with the advantages of using a BitTorrent-like network (rarest-first). We show in Section 4.1 and 4.2 that by effectively maintaining the balance between earliest-first and rarest-first our proposed hybrid strategy outperforms pure earliest-first or rarest-first strategy in terms of *playback continuity*.

2. We investigate dynamic adjustment of the probabilities of earliest-first and rarest-first by implementing a *contiguous index* which is the number of continuous pieces downloaded and buffered by a peer starting from the peer's current playback position in the playback sequence (Section 4.3). If the contiguous index decreases, then we will increase the probability of earliest-first downloads (p). This allows the peer to aggressively increase the pieces that are needed sooner. However, if the contiguous index increases, then we can increase the percentage of rarest-first downloads. This helps rare pieces propagate through the network therefore increasing the overall performance of the network. We show that instead of fixing p, by having each peer adjust its own p dynamically according to the contiguous index, we can improve the streaming performance.

3. In Section 5, we attempt more in-depth investigations such as coding techniques which offer more content data diversity to a network causing high utilization of the network by allowing all the peers in the network to encode and transmit data pieces [13], [14], [12], [8]. Those coding schemes make the probability of duplication of transmitted data to the network extremely low. With coding, each peer XOR's all or subset of the data pieces it has and exchanges the resulting codeword with its neighbors. We show that coding provides higher content data diversity than rarest-first strategy does, contributing to the higher network utilization.

4. We investigate improvements on the playback continuity performance in Section 6 by varying the values of key parameters of our simulator such as the number of seeds, aggregate seed bandwidth, seed leaving rate, the number of neighbors, and so on. We also confirm that our proposed download strategies perform the best in those diversified scenarios.

In this paper, we show that it is feasible to provide a real-time streaming service of stored media by using the currently available P2P systems such as BitTorrent with modifications to the download scheduling mechanisms.

2 Related Work

During the last years, there have been a number of studies in the area of P2P multimedia streaming based on BitTorrent-like content distribution systems [28], [26], [29], [24], [10], [27]. Among them BiToS [26] and Zhou et al. [29] are closely related to our work in that they suggest modifications to piece selection strategies of the original BitTorrent systems (rarest-first) to support streaming capabilities. In [26], the buffer for downloading pieces is divided into two sets, high priority set and remaining pieces set (lower priority set), and in the piece selection process the peers choose to download a piece from the high priority set

with probability p and download a piece from the low priority set with $1 - p$. Similarly, [29] suggests Greedy Strategy where peers select a piece which is closest to its playback deadline, Rarest First Strategy, and Mixed Strategy, which is a combination of Rarest First and Greedy.

However, both [26] and [29] deal with live media streaming while we explore stored media streaming. In a live media streaming scenario, only a few pieces are available for sharing in the few seconds of the playback buffer, and future pieces are not available. On the other hand, for stored media streaming, all the pieces of the content file being streamed are available. Therefore, the piece selection strategies in [26], [29] need to focus limited sets of the pieces using a small sliding time window from which peers select to download a piece whereas our strategy needs to consider the entire file.

Furthermore, all the peers have a synchronized playback deadline in live media streaming, and a newly arriving peer with an empty buffer therefore also starts to play back the same piece as existing peers do. In this scenario, pieces played back in the previous time are usually removed as they are useless. However, since peers do not need to synchronize their viewing times in our work, peers keep the pieces even after their scheduled playback deadline so that they can share those pieces for other peers including newly joining peers.

Recent work to provide stored media or Video-on-Demand streaming service is BASS [10] and RedCarpet [21], [4]. BASS presents a hybrid system approach of a streaming server and a BitTorrent-based P2P system. In BASS, all the peers are connected to the dedicated server downloading pieces from it in the meantime they share the pieces using the BitTorrent network to help each other and supplement the server-based streaming. BASS shows that the reductions in the server-side load can be achieved with the BitTorrent approach. However, in our work, peers use only P2P network to download the pieces without the server/client relationship. For our work, it is more required to investigate the effectiveness of various piece propagation or download strategies among the peers.

In RedCarpet, the stored media file is divided into segments, each segment comprising consecutive pieces. The authors apply random, local-rarest, and sequential download policies to choosing a piece from the target segment (each segment is picked sequentially). On the other hand, we propose download strategies in which any of the pieces of the file needed by a peer may be downloaded any time they are available. RedCarpet also adopts network coding to improve the network throughput and playback rate. However, their coding schemes differ from ours. In their work, the initial server linearly combines all the pieces of the file with random coefficients to create an encoded codeword, and they modify the scheme restricting coding to only a given segment of the file since otherwise, a peer has to wait until it downloads the whole file before it can start decoding. In our approach we consider all the pieces of the file but still ensure fast encoding and decoding time at each peer by introducing a degree distribution that indicates the probability distribution over the number of the original data pieces encoded in a codeword. Given degree d, only d randomly chosen pieces form a codeword, and peers can immediately recover data from low degree codewords.

3 Implementation

We use a simulation-based approach for measuring and understanding stored media streaming performance. Such an approach provides the flexibility of carefully controlling the configuration parameters of the various streaming mechanisms. We modify a time-based BitTorrent simulator previously developed at Carnegie Mellon University called OctoSim [6], [5].

3.1 Download Scheduling Strategies

We are interested in three different scheduling strategies. BitTorrent uses a combination of rarest-first and random selection to achieve a low download time. Live media streaming applications such as CoolStreaming download the piece that is most current. In order to minimize interruptions in real-time playback, we will explore different combinations of the following three strategies.

- ***Earliest-first scheduling***: Earliest-first prioritizes the download of pieces by their position in a media file. A piece earlier in the playback sequence will have a higher priority than a piece that appears later in the playback sequence. Earliest-first appears to be a natural strategy for real-time feedback because earlier pieces are needed before later pieces. However, there may be some problems with strategy that are not immediately obvious. In this scheme, the network may not propagate an exceptionally rare piece. Therefore, every peer that needs the rare piece will encounter a bottleneck because they will struggle to find a peer that has it.

- ***Rarest-first scheduling***: Rarest-first prioritizes the download of pieces by their rarity in the network. In the BitTorrent implementation, local rarest-first is used. Peers in the network are grouped into a smaller, logical neighborhoods. When a peer requests a piece from its neighbor, the neighbor will offer the rarest piece in the network that it has. We follow this implementation of rarest-first. However, this does not work well in streaming applications because a user

needs immediate access to the beginning of the media file. The obvious problem with this strategy is that resources may be allocated to download later pieces when the playback is waiting for an earlier piece.

- ***Random-selection scheduling***: Random-selection is a download strategy that can be used when there is no available information to use for alternative strategies. For instance, BitTorrent peers initially start with random-selection because information about the rarest pieces may not be available. We will investigate the performance of random-selection in combination with the other download strategies.

3.2 Simulation Details

OctoSim is a discrete-event simulator. It models peer activity (joins, leaves, neighbor selection, piece exchanges, etc.) as well as many of the associated BitTorrent mechanisms (local rarest-first, tit-for-tat, etc.) in detail [6]. However, this simulator was developed to simulate the distribution of non-streaming content. Therefore, we modify the simulator to control the scheduling and download progress of the streaming file, such as which piece of the file should be downloaded at what point in time. This allows us to determine the most effective heuristic for obtaining the necessary pieces of a file that allows uninterrupted playback using combinations of earliest-first, rarest-first, and random-selection schemes.

Downloading peers, or *leechers* join the network at the arrival rate λ (peers/second) according to a Poisson distribution. They persist in the lifetime of the streaming but depart as soon as they finish downloading. On the other hand, there initially exists one *seed* defined to be a peer that has already received a complete file and has been willing to serve it to leechers. The initial seed stays in the network throughout the duration of the simulation. The default streaming rate is 400Kbps and the size of each piece of a media file is 2000Kbits (=250KBytes). Therefore, each piece contains five seconds of the stream. Unless other values for the parameters are mentioned, we use the following default settings in our simulations:

- Number of initial seeds: 1
- Leecher arrival rate (λ): 0.1 peers/second
- Streaming rate of the media: 400Kbps
- File size: 90MBytes (360 pieces of 250KB (=2000Kbits) each, which corresponds to 30 minute length playback)
- Seed uplink bandwidth: 800Kbps
- Leecher downlink/uplink bandwidth: 1600/800Kbps
- Number of neighbors of each peer (degree, d): 10
- Maximum number of concurrent upload transfers of each peer: 5
- Each leecher departs the network as soon as it completes downloading.
- Leechers are independent and homogeneous following the same download scheduling strategy.
- All the behaviors of each peer (including piece information collection and neighbor selection for piece exchanges) are the same as those used in the current BitTorrent system [9] except for the download scheduling strategy.
- Number of peers that we collect and average data from: 200 peers
- All simulations average the results of 30 runs.

With the above default settings we want to first look at the streaming performance under quite poor conditions for P2P real-time streaming services. We assume that there exists only one seed throughout the entire simulation period and all the leechers leave our simulation network right after they get the complete media file. The seed's uplink bandwidth is also relatively small compared with each leecher's downlink bandwidth. Thus, it is hard to provide uninterrupted streaming services for quickly arriving leechers with the default settings. Once we observe results of the streaming performance under this situation in Section 4, we also diversify the simulation scenarios in Section 6 by changing the values of the above configuration parameters to find key parameters and corresponding values which can offer uninterrupted real-time playback.

3.3 Playback Continuity

We quantify the effectiveness of the stored media streaming in terms of *playback continuity*. Maintaining continuous playback is a primary objective for streaming applications to ensure a satisfactory user experience. The number of pieces missing playback deadlines should be kept to a minimum.

To measure our playback continuity (PC), we first obtain a given peer i's actual playback time (T) taken to complete the entire playback. We start to measure T as soon as peer i joins the network and requests the media. Note that T differs from the time taken to complete downloading and cannot be smaller than it. Second, we consider the minimum start-up time (S_{min}) required for peer i to be able to start playback after it joins the network. Each piece can be played back when it is completely downloaded, and peer i can start to play back the media only after it completely

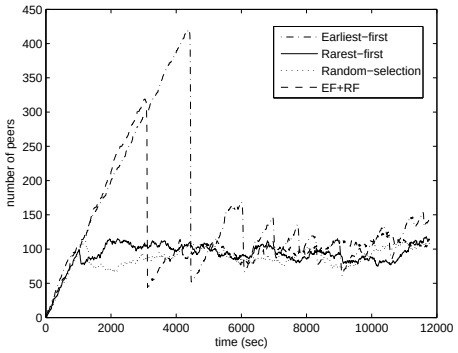


Figure 1. Transition of the number of peers in the network with different download scheduling strategies as a function of simulation time. (The curve named 'EF+RF' indicates the hybrid strategy of earliest-first and rarest-first, which will be explained in Section 4.1.)

downloads the first piece of the media file. Given the original time length of the media content (L), we define playback continuity as $PC = (T - S_{min})/L$. Since the minimum start-up time is achieved when peer i requests none other than the first piece of the file and downloads it from a neighbor with the highest uplink bandwidth (although there exists peer i's downlink bandwidth constraint), we define S_{min} as follows:

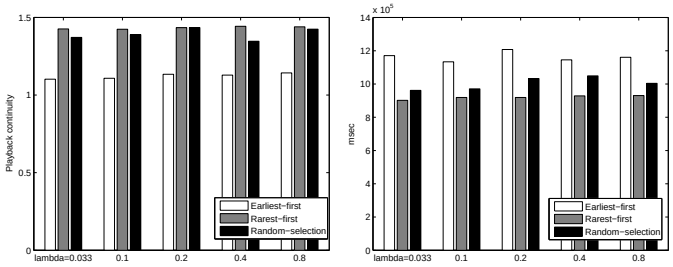
$$S_{min} = \frac{v}{\min\{\gamma_i, \max(U)\}}$$

where v is the size of each piece, γ_i is the downlink bandwidth of peer i, and U is a set of all distinct uplink bandwidths of neighboring seeds and leechers. For example, in Section 4.1, we treat homogeneous peer bandwidth (1600/800Kbps), and since the uplink bandwidth is smaller than the downlink bandwidth and each piece size is 250KB(=2000Kbits), we constantly has $S_{min} = 2000Kbits/800Kbps = 2.5seconds$. In case of the heterogeneous environment in Section 4.2, S_{min} varies depending on peer i's downlink bandwidth itself as well as its neighbors' uplink bandwidths.

Since $T - S_{min}$ cannot be smaller than original length of the media (L), playback continuity is always greater than or equal to 1. If playback continuity equals 1, it means that users can play back the media without any interruption. A playback continuity of 2 indicates that a user spends twice as much time as L finishing playback due to frequent interruption.

4 Probabilistic Hybrid Strategies

We first investigate streaming performance applying the three download scheduling strategies introduced in Section



(a) Playback continuity (b) Download time

Figure 2. Average playback continuity and download time of the peers when each of the three different scheduling schemes works alone. 'lambda' indicates each peer's arrival rate (per second) to the P2P network.

3.1. Figure 1 graphs the transition of the number of peers in the network as a function of simulation time - when each of the three different scheduling schemes is applied in isolation. We can see that in the beginning of each simulation the number of peers in the network keeps growing. This is because no peer has downloaded the entire file yet (and therefore are unable to depart from the network) but new peers are continually arriving at rate $\lambda = 0.1$. As simulation time goes by, peers begin to complete their downloads and leave the network. As a result, the size of the network stops growing and then decreases drastically until most of those initially arriving peers have left the network. After this point, regardless of download scheduling strategies, the number of peers gradually stabilizes to an point even though there are still small fluctuations on the curves. We observed, through several experiments with different values of λ, that the stabilized point depends on λ and the point almost linearly goes up as λ increases.

Based upon these findings, we first investigate the performance of peers that join the network after the size of the network has stabilized. (We will subsequently examine the behavior of the early arriving peers in Section 4.1 and 4.2.) In figure 2(a) we plot the average playback continuity of 200 homogeneous (with respect to bandwidth) peers across an increasing value for λ. We select those 200 peers sequentially after a sufficiently large number of peers (2500 peers) in the P2P network have obtained the complete file. In other words we collect from the 2501st to 2700th peers to finish downloading.

In the figure, there are three sets of bars that represent the earliest-first, rarest-first, and random-selection download strategies. We can see that earliest-first offers the greatest degree of continuous playback. This fits our expectations that giving earlier pieces higher priority will minimize breaks in continuity. We note that both rarest-first and random-selection offer equally poor performance with respect to this measure. However, as can be seen from figure

2(b), in terms of the average download time it takes for a peer to download the entire media, rarest-first performs the best by a significant margin, followed by random-selection, with earliest-first exhibiting the worst performance. It is not surprising that rarest-first offers the best download times, as it is the primary strategy implemented by BitTorrent.

These results show that while the system utilization of pure rarest-first is high, the playback continuity is quite low, due to out-of-order delivery of the pieces of the file. On the other hand, we believe that a pure earliest-first strategy is too aggressive in seeking the earliest pieces, to meet the needs of the individual peers. Earliest-first neglects the needs of the overall network and fails to propagate latter pieces that are needed by neighbor peers. We additionally observe that both playback continuity and download times remain almost unchanged, despite increases in λ. We examine the relationship between λ and playback continuity in Section 4.1 and 4.2. The following analysis is based upon those findings.

4.1 Homogeneous Setting

Instead of using each download scheduling strategy alone, we propose to use all the three download schemes at the same time but change the percentage or probability of each scheme and see which combination is optimal in terms of the playback continuity. After several experiments we concluded that the combination of earliest-first and rarest-first schemes offers the best performance in terms of the playback continuity. The inclusion of random-selection always decreased the performance of a hybrid scheduling strategy.

Based on the previous experiment, we considered a homogeneous setting in which all leechers have the same bandwidth (1600/800Kbps as noted in Section 3.2). We plot the playback continuity and the system utilization (uplink bandwidth utilization of the entire system) using a probabilistic hybrid strategy of earliest-first and rarest-first (see figure 3). With the probabilistic hybrid strategy, when a peer chooses the next piece to download, it chooses a piece using earliest-first strategy with probability p and chooses a piece using rarest-first strategy with $1 - p$. In other words, whenever a peer has a chance to download a piece from one of its neighbors, with probability p it requests the earliest piece the neighbor has, and with probability $1-p$ it requests the rarest piece in its neighborhood the neighbor has.

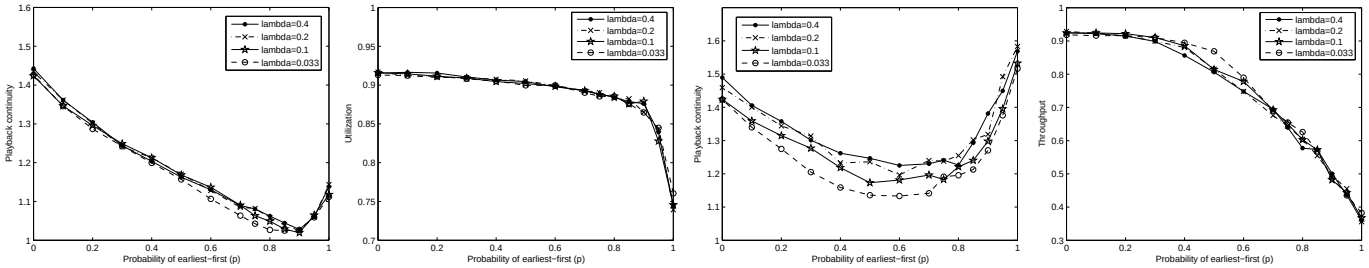
Based on the transition curves of the number of peers in the network (figure 1), we first define S_l as the set of 200 peers that finish downloading from 2501st to 2700th and S_e as the set of the first 200 peers that finish downloading. We expect that the two groups of peers will show different streaming performance results. Thus, in figures 3(a) and 3(b), like in figure 2, we collect and average results from peers in S_l. On the other hand, figures 3(c) and 4(b) are plotted from peers in S_e.

In figure 3(a), we can see that as the probability of earliest-first (p) increases, the playback continuity improves. However, we notice that there is a large rise at the right side of the curves (where the ratio of rarest-first to earliest-first is zero). This tells us that we cannot use only earliest-first because even small percentage of rarest-first contributes to much better playback continuity. We note that the hybrid strategy enables nearly uninterrupted playback where p is around $0.8 \sim 0.9$.

This result can be explained from figure 3(b). The utilization indicates the amount of uplink bandwidth actually used to transmit data during the whole simulation period over the total aggregate uplink bandwidth in the system. The utilization can be also considered as bandwidth efficiency. The figure shows that utilization drops drastically when the probability of rarest-first is zero, which implies that leechers are not contributing to the uplink bandwidth. This happens because the main disadvantage of earliest-first scheme is that the leechers have very similar pieces of the content file and a leecher has fewer chances to provide its pieces to its neighbors. However, we can see that the small percentage of rarest-first ($0.1 \sim 0.2$) even contributes to the fairly high utilization of the system.

This shows the trade-off between uninterrupted playback and the system utilization. General P2P systems achieve high utilization when the peers can exchange content with each other and this happens only if they have non-overlapping pieces of the content file [21]. Therefore, general P2P systems usually direct peers to download rarest pieces first because it enables high piece diversity among the peers. However, in order to provide the uninterrupted real-time playback, the peers would choose to use earliest-first. However, If they did so, the peers would not propagate rare pieces and would have very similar pieces, causing the possibilities for exchanging content to become low. As a result, the utilization of the P2P system would be low as well.

Figures 3(c) and 3(d) show the similar trend that the hybrid strategy greatly outperforms pure earliest-first or rarest-first even though the ratios of the combination for the best playback performance are different from those in figure 3(a). However, we notice that playback continuity itself of S_e in figure 3(c) is worse than S_l in figure 3(a). In figure 3(c), we observe the first 200 peers that receive the complete file, and in our simulations they are mostly the first 200 peers that arrive in the network (ignoring the cases of much higher peer arrival rates than we use for our simulations). Therefore, since there exist not many peers in the network that can provide pieces except an initial seed when those early peers first arrive, we note that they suffer low upload utilization in figure 3(d) compared to that in figure



(a) Playback continuity of S_l (b) Utilization of S_l (c) Playback continuity of S_e (d) Utilization of S_e

Figure 3. Playback continuity and utilization results in the homogeneous environment when earliest-first and rarest-first are used together. (a) and (b) plot 200 peers that finish downloading from 2501st to 2700th (S_l) while (c) and (d) plot the first 200 finished peers (S_e). The probability of rarest-first is $1-$ prob. of earliest-first.

3(b), resulting in the poor playback performance.

Another observation we make is that playback continuity in figure 3(c) improves as λ decreases while it remains almost constant in figure 3(a). We believe that the improvement on the performance is a result of the fact that since peers join at a slower rate, the new peer will encounter neighbors that have been on the network for a longer time. These neighbors will likely have many more pieces that are sought by the new peer. On the other hand, when λ is large, especially earliest-first shows very poor performance. This is obvious because with large λ, many peers newly arrive within a short period and request very similar pieces from the limited number of earlier peers which have those pieces.

4.2 Heterogeneous Setting

Next we examine a heterogeneous environment where leecher uplink/downlink bandwidth has a wide range. Table 1 shows the distribution of leecher bandwidth, which is reported in [23].

Downlink	Uplink	Percentage
512Kbps	256Kbps	56%
3Mbps	384Kbps	21%
1.5Mbps	896Kbps	9%
20Mbps	2Mbps	3%
20Mbps	5Mbps	11%

Table 1. Bandwidth distribution of leechers for a heterogeneous setting

Figure 4 shows the average playback continuity and upload utilization of 200 heterogeneous leechers as a function of p. Except for the bandwidth of leechers, all other settings are the same as those used in the homogeneous environment discussed in the previous subsection. In this simulation, we consider both S_l and S_e. We note that the playback continuity curves move up in figures 4(a) and 4(c) when compared to the results for the homogeneous environment. This is because more than 50% of leechers have quite poor bandwidth (512Kbps/256Kbps), while the streaming rate is 400Kbps.

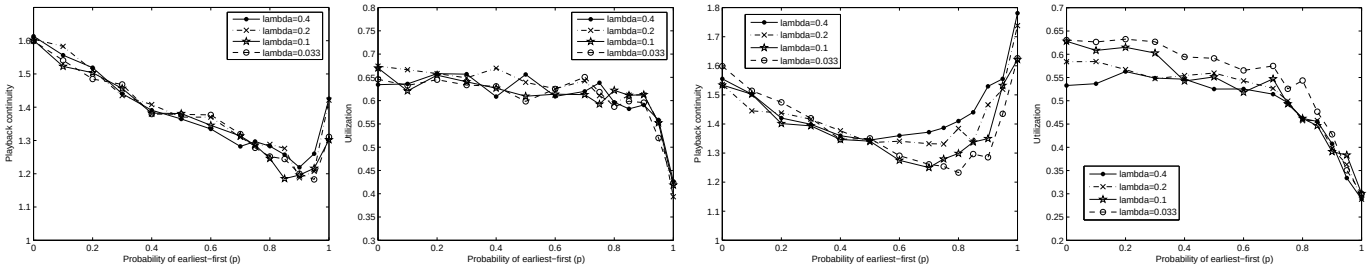
We note that our hybrid strategy still outperforms either pure earliest-first or pure rarest-first in the heterogeneous environment. Similar to the homogeneous case, the best playback continuity in Figure 4(a) is achieved where p is around 0.9. As regards upload utilization of heterogeneous leechers, we notice that there is a large drop at the right side of the curves when the fraction of rarest-first approaches 0 - just as we see in the homogeneous case. Therefore, the main disadvantage of pure earliest-first applies in the heterogeneous environment as well. In figure 4(d), we observe that heterogeneous peers in S_e still have the same low uplink bandwidth problem as noted in Section 4.1, and as a result, their playback performance is worse (see figure 4(c)) than that of peers in S_l.

In the rest of the paper, all simulations in the heterogeneous environment use the bandwidth distribution given in Table 1.

4.3 Monitoring Contiguous Buffer

Each peer has a buffer to store later pieces of the content file before they are needed. In the previous simulation in Section 4.1 and 4.2, each strategy has a fixed probability of occurring during one simulation. With heterogeneous peers in S_l and $\lambda = 0.1$ peers/sec, the performance was the best when p was fixed as 0.85 (see figure 4(a)). But now we change p dynamically during one simulation according to the number of pieces buffered by a peer contiguously from its current playback position. If this number is big enough, which denotes that the peer has many pieces contiguously from the current playback position, then the peer will not need to use earliest-first for a while. The peer may be more willing to help its neighbors by using rarest-first strategy.

Therefore, instead of fixing the probabilities of earliest-



(a) Playback continuity of S_l (b) Utilization of S_l (c) Playback continuity of S_e (d) Utilization of S_e

Figure 4. Playback continuity and utilization results in the heterogeneous environment. (a) and (b) plot peers in S_l while (c) and (d) plot peers in S_e.

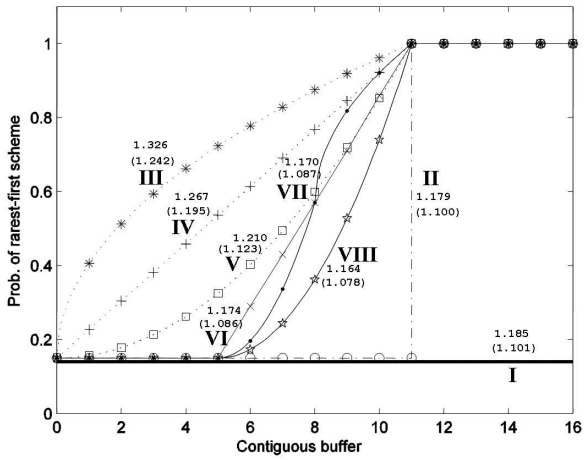


Figure 5. The dynamic adjustment of the probability of rarest-first according to the number of contiguous pieces in a buffer: The numbers next to each curve indicate playback continuity when the probabilities on the corresponding curve are used. The values in parentheses indicate Growth Codes results which will be explained in Section 5. Heterogeneous peers from S_l are simulated with $\lambda = 0.1$.

first (p) and rarest-first ($1-p$) schemes, we change p dynamically according to the number of the contiguous pieces from the playback position in order to improve the playback continuity. We define *contiguous index* c as the number of contiguous pieces from the current playback position a peer has buffered. As c increases for a given peer, then we can decrease the probability of earliest-first(p) for the peer, and as c gets small, we can increase p and make it even greater than 0.85.

We plot the results of dynamic changes of p in the heterogeneous environment in figure 5. The x-axis indicates the contiguous index c. Note that the number of contiguous pieces did not really exceed 17 or 18 during our simulation under $\lambda = 0.1$. The y-axis is the probability of rarest-first ($1-p$). The straight line (I) is our previous best result where p was fixed as 0.85 in the previous subsection. Now we attempt to change the shape of this straight line I considering the basic idea that we increase the probability of rarest-first as c (x-axis) increases. The step function curve (II) is quite an extreme case. We set probability 1 when the number of contiguous pieces is big enough, and otherwise, we maintain the original probability of rarest-first, 0.15. But even with this small modification, the playback continuity slightly improves from 1.185 to 1.179. Then we try concave(III), linear(IV), and convex(V) curves. However, the playback continuity is worse with those probability curves showing that we should not increase the probability of rarest-first too much when c is very small. Therefore, we try other various curves such as (VI)$\sim$(VIII) where the probability of rarest-first remain unchanged while c is very small ($c \leq 5$ in our simulation), and they all outperform our original best result with fixed probabilities.

These results show that dynamic adjustment of probabilities of earliest-first and rarest-first is more effective than fixed probabilities. They fit our expectation from having more rarest-first scheme(i.e., decreasing earliest-first) as contiguous index increases. However, as we see that only (II) and (VI)$\sim$(VIII) offer the improvement in the playback continuity, there are some conditions which should be satisfied to guarantee a benefit. While we simulated with several other various curves, we noticed that with very small c, when the probability of rarest-first exceeded 0.15 which is the ratio of rarest-first in our previous best result in figure 4(a), probability curves with any shapes did not offer better performance. With very small c, we should keep the probability equal to or smaller than 0.15. On the other hand, we can safely set probability 1 when c is large enough. However, we observed that when we kept setting probability 1 as c decreased, even though we set probability 0 for the rest of c, the performance became worse. There was a limit to c and it was 11 in our simulations. When $c < 11$, we should keep the probability of rarest-first below 1. With these two conditions satisfied, probability curves in general improve

the playback continuity performance in our simulations.

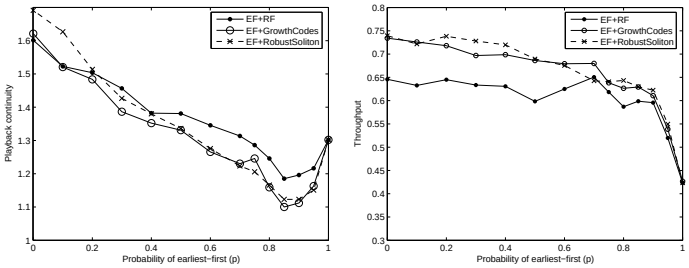
5 Effect of Coding Techniques

We now consider a more in-depth investigation by implementing and using coding techniques. Network coding has been proposed by [13], [14], [8], [12] to improve data diversity and the throughput of a network. With network coding, each node or peer of the network is able to encode and transmit pieces of information. The intermediate peer decodes the encoded data pieces and recode them into another set of pieces before it sends them out. This coding technique significantly reduces the chance of receiving the same encoded piece from the network. [17] shows that the high data diversity of coding techniques enables efficient BitTorrent's tit-for-tat exchanges and hence enhances fairness in the P2P system. Also, [12] shows that network coding improves the robustness of the system and is able to handle extreme situations where the seeds and leechers leave the system.

5.1 Coding Description

We consider low complexity erasure codes such as Growth codes [16] and LDPC codes which include Digital Fountain codes [7] (Tornado codes [19], LT codes [18], Raptor codes [22]). Those erasure codes which have low computational costs for encoding and decoding resulting in very fast encoding/decoding algorithms are practical in our real-time playback scenarios. With coding, the peers exchange *codewords* each of which is an XOR-sum of a subset of all distinct pieces in the media file. The number of the distinct pieces which are XOR'd together is referred to as the degree (D) of the codeword. Thus, degree 1 codewords consist of original data itself, and the others ($D > 1$) comprise an XOR'd combination of original data. We assume that the size of a codeword is equal to that of one piece of the media file (250KB) even though in practice some additional header space is required to recognize the set of pieces encoded in the codeword.([16] shows that this overhead has a negligible impact on the results.) The codes choose the degree of each codeword according to a pre-determined degree distribution which is a probability distribution over degree lengths.

In order to decode one original piece from a degree D codeword C, a peer should have all of the other $D-1$ original pieces encoded in the codeword. We assume that in the decoding process the peer knows which pieces are used to generate any received codeword. To decode the missing piece X_D, the peer solves $X_D = C \oplus X_1 \oplus X_2 \oplus \cdots \oplus X_{D-1}$ where $X_1, \cdots, X_{D-1}$ are the decoded pieces the peer should have. Each peer can perform the decoding process on the fly as it receives a new codeword from its neighbors, and codewords which cannot be decoded at the moment are stored for later use until more codewords arrive and additional pieces are decoded.



(a) Playback continuity (b) Utilization

Figure 6. Playback continuity and utilization results of S_l in the heterogeneous environment when coding strategies are applied. EF indicates earliest-first and RF rarest-first. The arrival rate of leechers λ is 0.2. For Robust Soliton distribution, the parameters $c = 0.2$ and $\delta = 0.05$ are used.

5.2 Simulation Results

First, we substitute coding for the rarest-first part in the hybrid strategy of the earliest-first and rarest-first schemes. Since the earliest-first part is essential and the earliest pieces are usually expected to be played back as soon as the peers receive them, we do not encode the original pieces in codewords when the earliest-first strategy part is used. Due to the time required to decode those pieces, the peers may frequently miss playback deadlines, resulting in a worse playback continuity. On the other hand, using the rarest-first strategy, peers usually exchange later pieces and those pieces have enough time until the playback deadlines. Therefore, those pieces can be encoded. With our coding strategies, instead of a peer's requesting a rarest piece to its neighbors, the peer requests a codeword in which the rarest pieces are encoded. The number of rare pieces that will be encoded in a codeword depends on the pre-determined degree distribution based on the codes. If multiple pieces have the same rarity in the neighborhood and taking all of them to generate the codeword exceeds the given degree, then the peer should pick a part of them uniformly at random. We use the degree distribution of Growth code and Robust Soliton distribution of LT codes.

Since we use coding, we can add more variety to the information exchanged between peers, leading to high throughput. Moreover, a peer has more options for an earliest piece. Without coding, a peer has to request the earliest piece which its neighbor has. However, with coding, even though the neighbors do not have the earliest piece which the peer needs, if it has a codeword in which the earliest piece is encoded, then it can get the piece indirectly

by receiving the other pieces in the codeword and decoding the desired piece from the codeword. In figure 6(a), we plot playback continuity of S_l in the heterogeneous environment versus the mixing probability p for the downloading strategies derived when using the degree distribution of Growth Codes and Robust Soliton of LT codes. The EF+RF curve replicates our previous non-coding results of the hybrid strategy in figure 4(a). With our coding strategies, instead of rarest-first, the probability of encoding the original pieces at a peer and transmitting the resulting codeword is $1-p$. We can see that the both coding strategies outperform the non-coding scheme and they can improve the best playback continuity of the non-coding strategy around $p = 0.9$. Also, figure 6(b) confirms our expectation that the coding strategies contribute to higher system throughput.

We also apply dynamic adjustment of probability of earliest-first p to our coding strategies in the same manner as in Section 4.3. Instead of fixing p, a peer increases the probability of earliest-first as contiguous index c decreases, and it increases the probability of coding as c increases. In figure 5, the y-axis, the probability of rarest-first $(1-p)$ here indicates the probability of using coding strategies, and the values in the parentheses indicates playback continuity results given by the Growth coding strategy. Similar to non-coding strategy, dynamic adjustment of the probabilities of earliest-first and coding also works more effectively than fixed probabilities.

6 Diversified Simulation Scenarios

The default settings for the previous simulations represent quite a poor circumstance where there is only one initial seed and all the leechers leave the network right after they get the complete file without serving as a seed. Also, the initial seed bandwidth (800Kbps) is relatively small and not sufficient to keep the downlink bandwidth of the leechers fully utilized, resulting in the low network utilization. In this section, we investigate improvements in the streaming performance possible by changing the values of key parameters of our simulator such as the number of seeds, aggregate bandwidth of seeds, seed leaving rate, and the number of neighbors. Especially, we focus on heterogeneous peers from set S_e since their streaming performance was the worst (figure 4(c)). At the same time, we confirm that our proposed strategies perform the best in those diversified scenarios.

6.1 Bandwidths of Seeds and Seed Leaving Rates

We examine the effect of bandwidths of seeds on the playback continuity performance considering the cases of having a single seed and having multiple independent seeds.

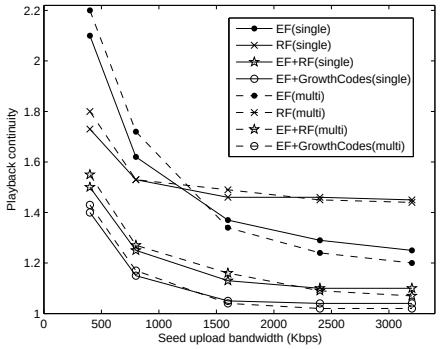


Figure 7. Comparison between a single seed and multiple seeds as aggregate seed uplink bandwidth varies.

Since all leechers leave the network right after they finish downloading without becoming a seed, here seeds indicate initially existing seeds which stay in the network throughout the entire simulation. Figure 7 shows playback continuity results of both non-coding and coding strategies as the bandwidths of seeds varies from 400Kbps to 3200Kbps. If there is a single seed, all the seed bandwidth belongs to the single seed, otherwise, the bandwidth indicates aggregate bandwidth of multiple seeds. Leechers are heterogeneous, and we focus on S_e. The probability of earliest-first (p) is fixed as 0.8 for both EF+RF and EF+GrowthCodes strategies. The results show how valuable the seed bandwidth is for the streaming performance whether there is a single seed or are multiple seeds. Also, we can see that our hybrid strategy and coding strategies still greatly outperform pure earliest-first or rarest-first as the seed bandwidth varies. EF+RF(multi) achieves the playback continuity below 1.2 when 2 seeds (1600Kbps) exist in the network while EF(multi) needs 4 seeds (3200Kbps).

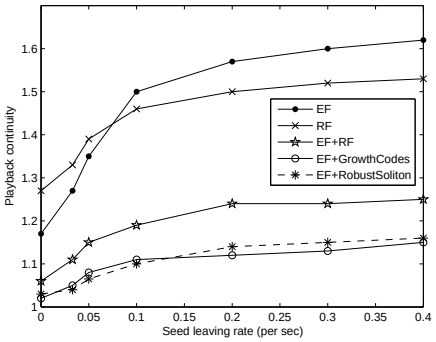


Figure 8. Impact of varying the seed leaving rate (in seeds/second). The probability of earliest-first(p) is 0.8 for EF+RF, EF+GrowthCodes, and EF+RobustSoliton.

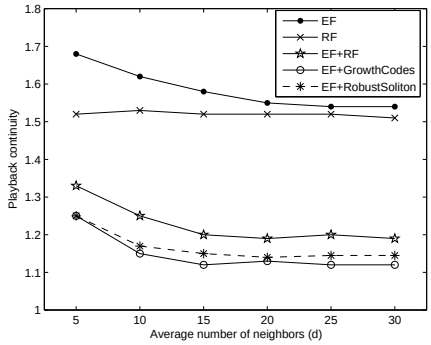
When the aggregate bandwidth across all seeds is large,

multiple seeds are more effective than a single seed due to the limitation on the number of concurrent uploads per seed (the limit is 5 in our simulation). Even if a seed has a large bandwidth available, the limit on the number of concurrent uploads can cause some of the available bandwidth to go unused. When the available aggregate bandwidth is small, this limit on the number of uploads per seed does not come into play since the load offered by the 5 concurrent upload sessions uses all the available bandwidth, even for a single seed. In this case, the playback continuity suffers more in the case of multiple seeds because the independent multiple seeds serve duplicate pieces being served by the other seeds wasting their bandwidth. We find that during the rarest-first periods almost 30% of the pieces are served duplicately by different seeds even before one full copy of the file has been served to the network.

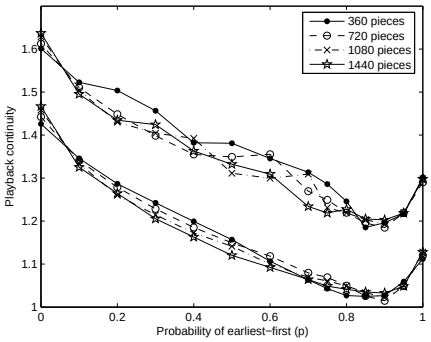
We also investigate the cases where each leecher stays for a given average time in the network serving as a seed after it completes downloading. In figure 8, we plot the playback continuity of different download strategies as the seed leaving rate is varied. Here one initial seed stays and keeps serving, and only the other non-initial seeds which used to be leechers formerly leave according to the seed leaving rate. The seed leaving rate (in seeds/second) follows the exponential distribution, and the average stay time after each leecher becomes a seed is therefore one over the seed leaving rate. In the figure, as the rate decreases, the playback continuity of all the strategies improves substantially, and we can achieve almost perfect real-time playback with the leaving rate 0 where no seeds leave the network.

6.2 Peer Degree

Next we study whether playback continuity improves with increases in peer degree, d, which denotes the average number of a peer's local neighbors. Figure 9(a) shows that playback continuity performance of peers from S_e in the heterogeneous environment generally improves modestly as the degree increases. When the degree is small, for instance, when $d = 5$, a leecher has a very restricted local view in its neighborhood [6]. Therefore, rarest-first scheduling scheme is not effective in evening out the distribution of pieces at a global level and causes poor playback continuity. However, when the degree is moderately high, rarest-first ensures greater diversity in the set of pieces held in the system [6] and it also contributes to playback continuity. Since the maximum number of concurrent uploads per peer is limited to 5, the performance is relatively insensitive to very high degree ($d > 20$).



(a) Peer degree



(b) Number of pieces

Figure 9. (a) The playback continuity as a function of average number of neighbors (d). (b) The playback continuity of EF+RF strategy as the media file's size varies from 360 to 1440 pieces, which corresponds to from 0.5 to 2 hour length media. (In (a), the probability of earliest-first(p) is 0.8 for EF+RF, EF+GrowthCodes, and EF+RobustSoliton.)

6.3 Number of Pieces

The intention of this subsection is to measure how scalably our download strategies perform by changing the number of the total pieces of the media file. Figure 9(b) shows the scalability of our probabilistic hybrid strategy of earliest-first and rarest-first with media file sizes up to 2 hours in length. We plot playback results from two different groups of peers: homogeneous and heterogeneous peers. The upper four curves in the figure correspond to heterogeneous peers, and the lower four curves, homogeneous, - all peers considered belong to S_l. In the figure, the overall shapes of each curve are similar to each other showing that the hybrid strategy outperforms pure earliest-first or rarest-first scheme. The best playback continuity results are well maintained close to those obtained when the default number of pieces (360) is used. Similarly, with our cod-

ing strategies replacing rarest-first, the p-modulated hybrid strategies show the same trend (figure omitted due to space constraints).

7 Conclusions

Our work has demonstrated that enhancing the download scheduling schemes offers significant improvements in the real-time playback performance of stored media in BitTorrent-like networks. We measure the performance of uninterrupted playback through a metric called playback continuity. In our BitTorrent-like system, we combine earliest-first and rarest-first scheduling strategies and measure the playback continuity using a static ratio of those strategies. We also implement a mechanism for dynamically adjusting the ratio of the strategies based upon the number of contiguous pieces at the playback deadline that have been downloaded by a peer. We also explore the effect of coding techniques such as Growth codes and Digital Fountain codes on the playback continuity by substituting those codes for the rarest-first part of the non-coding hybrid strategy.

Motivated by our positive results so far, we intend to investigate, as future directions of research, analytical models which can express our download strategies in P2P streaming networks and study if they can explain our current results mathematically.

References

[1] Kazaa home page: http://www.kazaa.com/.

[2] Pplive home page: http://www.pplive.com/en/index.html.

[3] Tivo home page: http://www.tivo.com/.

[4] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. Rodriguez. *Is High-Quality VoD Feasible using P2P Swarming?* World Wide Web WWW'07 Conference, Canada, May 2007.

[5] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. http://www. research.microsoft.com/projects/btsim.

[6] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. *Analyzing and Improving a BitTorrent Network's Performance Mechanisms*. IEEE Infocom 2006, Barcelona, Spain, April 2006.

[7] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. *A Digital Fountain Approach to Reliable Distribution of Bulk Data*. SIGCOMM, September 1998.

[8] P. A. Chou, Y. W, and K. Jain. *Practical Network Coding*. in Allerton Conference on Communication, Control, and Computing, Oct 2003.

[9] B. Cohen. *Incentives Build Robustness in BitTorrent*. IPTPS, Feb 2003.

[10] C. Dana, D. Li, D. Harrison, and C. N. Chuah. *BASS: BitTorrent Assisted Streaming System for Video-on-Demand*. in IEEE International Workshop on Multimedia Signal Processing (MMSP), October 2005.

[11] eMule home page: http://www.emule project.net/.

[12] C. Gkantsidis, J. Miller, and P. Rodriguez. *Comprehensive view of a Live Network Coding P2P system*. ACM SIGCOMM/USENIX IMC'06, Brasil, October 2006.

[13] C. Gkantsidis and P. Rodriguez. *Network Coding for Large Scale Content Distribution*. IEEE INFOCOM'05, Miami, March 2005.

[14] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros. *The Benefits of Coding over Routing in a Randomized Setting*. IEEE International Symposium on Information Theory (ISIT), page 442, Yokohama, Japan, 2003.

[15] iTunes Music Store (ITMS): http://www.apple.com/itunes/store/.

[16] A. Kamra, J. Feldman, V. Misra, and D. Rubenstein. *Growth Codes: Maximizing Sensor Network Data Persistence*. ACM SIGCOMM, Pisa, Italy, September 2006.

[17] T. Locher, S. Schmid, and R. Wattenhofer. *Rescuing Tit-for-Tat with Source Coding*. 7th IEEE International Conference on Peer-to-Peer Computing (P2P), Galway, Ireland, September 2007.

[18] M. Luby. *LT Codes*. in Symposium on Foundations of Computer Science, 2002.

[19] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. Spielman. *Efficient Erasure Correcting Codes*. in IEEE Transactions on Information Theory, volume 47, pages 569-584, 2001.

[20] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. *A Measurement Study of the BitTorrent Peer-to-peer File-Sharing System*. Technical Report PDS-2004-003, Delft University of Technology, April 2004.

[21] P. Rodriguez, S. Annapureddy, and C. Gkantsidis. *Providing video-on-demand using peer-to-peer networks*. Internet Protocol TeleVision (IPTV) Workshop, May 2006.

[22] A. Shokrollahi. *Raptor Codes*. IEEE/ACM Transations on Information Theory, Vol. 52, No. 6, June 2006.

[23] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang. *The feasibility of supporting large-scale live streaming applications with dynamic application end-points*. Proceedings SIGCOMM 2004, Aug 2004.

[24] S. Tewari and L. Kleinrock. *Analytical Model for BitTorrent-based Live Video Streaming*. in Proceedings of IEEE NIME 2007 Workshop, Las Vegas, NV, Jan 2007.

[25] E. Veloso, V. Almeida, J. W. Meira, A. Bestavros, and S. Jin. *A Hierarchical Characterization of a Live Streaming Media Workload*. in IEEE/ACM Trans. Netw. 14, February 2006.

[26] A. Vlavianos, M. Iliofotou, , and M. Faloutsos. *Bitos: Enhancing bittorrent for supporting streaming applications*. INFOCOM 25th IEEE International Conference on Computer Communications, Proceedings, 2006.

[27] G. Wu and T. C. Chiueh. *How Efficient is BitTorrent?* in 13th Annual Multimedia Computing and Networking (MMCN'06), January 2006.

[28] X. Zhang, J. Liu, B. Li, and T. P. Yum. *CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming*. IEEE Infocom, IEEE Press, 2005.

[29] Y. Zhou, D. M. Chiu, and J. C. S. Lui. *A Simple Model for Analyzing P2P Streaming Protocols*. IEEE ICNP, Bejing, China, October 2007.