This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE P2P 2010 proceedings.

Multi-Source Scheduling in Streaming Erasure-Coded Video over P2P Networks

M. L. Ma and Jack Y. B. Lee, *Senior Member, IEEE*

***Abstract*—The efficient scheduling of streaming data delivery in a peer-to-peer (P2P) network is a hard problem due to the Internet's lack of support for resource allocation and performance guarantees. In particular, the bandwidth resources available to a peer is constantly in flux and the future bandwidth availability is very difficult, if not impossible, to predict accurately. This work proposes to tackle this problem from a different angle. We investigate the use of erasure codes to encode the media data and then schedule multiple peers to stream the encoded data simultaneously to a receiver. By exploiting the order-invariant property of erasure codes this approach enables the sending peers to fully utilize their available bandwidth resources and yet does not need to estimate or predict their bandwidth availability. Moreover, we develop distributed scheduling algorithms to juxtapose the data transmissions from multiple peers so that the coding and storage complexities can be kept at practical level in scaling up the system. This paper describes the motivation, architecture, and design of the proposed coding/scheduling algorithms; develops a performance model to characterize the algorithms' performance bounds; and evaluates them through simulation as well as experiments.**

***Index Terms*—Erasure codes, media streaming, multi-source, peer-to-peer, scheduling**

I. Introduction

THE rapid growth of multimedia contents in the Internet has put much strain on the content providers. In particular, the conventional centralized server architecture has increasingly becoming the bottleneck in larger-scale content distribution due to the immense bandwidth required. The recent developments in distributed content delivery systems such as peer-to-peer (P2P) and hybrid P2P open up new ways to tackling this bandwidth-scaling problem.

Common to these new architectures are the use of peers, distributed across the Internet, individually capable of serving whole or part of the content, and together can provide sufficient bandwidth to serve a huge number of concurrent users. This distributed approach can achieve far better scalability by recruiting peers to serve as servers.

As a result, P2P have grown tremendously over the past decade, enabling the distribution of large files efficiently across the Internet. Beyond file download, the next frontier is in streaming contents such as audio and video. Unlike file download, streaming media is often decoded and played back while the data are being transferred. Thus, hiccups in data distribution may lead to playback interruption, often in the form of sudden playback pauses or in some cases, audio-visual degradation of the decoded media.

Fundamentally, if data delivery failure is a result of insufficient bandwidth (across the whole system), then the only solution would be to adapt the contents to fit the amount of bandwidth available using techniques such as scalable video codec [1], [2] or real-time transcoders [3], [4]. However, even if there is sufficient bandwidth available across the system the required media data may still fail to be delivered to the client in time for playback. Specifically, given a number of peers available for streaming media data, the client will need to determine which peers to employ and what part of the media stream each should transmit and at what time. Given that the Internet does not guarantee bandwidth availability nor is there a reliable way to predict future bandwidth availability, this task is far from trivial.

This work tackles this resource allocation problem from a different angle. Instead of attempting to predict bandwidth availability and then developing algorithms to adaptive to it, we propose the use of erasure codes to encode the media data and then transmit them from multiple peers simultaneously to the client. By exploiting the order-invariant property of erasure codes this approach completely eliminates the need of estimating or predicting the bandwidth availability in the data transfer process. The peers simply transmit data at the maximum rate allowed by the transport protocol, the network path, and any other arbitrary constraints, and the client can simply wait until sufficient amount of encoded media data are received, irrespectively of which peers they came from, and then decode the data to obtain the original media data for playback. This approach also solves the problem of peer churn common in P2P systems as the bandwidth void created from peers leaving the system will automatically be filled up by the extra transmission from the remaining peers.

Nevertheless, two challenges remain when scaling up multi-source erasure-coded streaming to a large user population over P2P networks. First, the computational complexity of many erasure-correction codes such as Reed-Solomon (RS) codes [5] increases super-linearly with the size of the coding space, i.e., the size of the peers' population. For large systems with potentially hundreds, if not thousands, of source peers, the erasure encoding process will become the system bottleneck. Second, to guard against malicious peers from injecting bogus data into the data stream it is essential to have a way to verify the data fragments as they are received from the many different peers.

978-1-4244-7139-3/10/$26.00 ©2010 IEEE

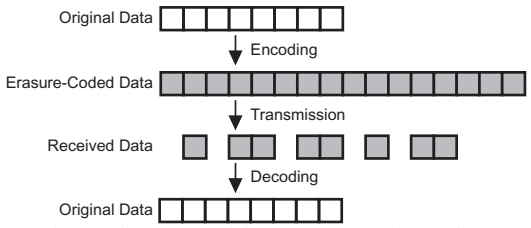

Fig. 1. The process of the erasure-correction coding.

This is most commonly done by creating a hashed key for each data fragment and distributing the hash key from a trusted server. The client can then verify the authenticity of the data fragments received by comparing its hash with the hash key from the trusted server. While this works well in existing systems such as Bit-Torrent (BT) [6], our study revealed that the hash-generation process's computational complexity and storage requirement increases exponentially when applied in the erasure-coded environment, thereby limiting the system's scalability.

This paper tackles these two challenges by introducing a novel scheduling algorithm to decouple the dimension of the coding space from the number of source peers in the system. This enables the use of small coding space which are both computationally efficient and storage efficient. The underlying principle is to allow data duplications in the encoding process, which reduces the coding space size, and then schedule the peers' data transmission in such a way to reduce the amount of collisions (i.e., transmission of duplicate data). Our initial findings confirmed that using multi-source scheduling it is possible to reduce the coding space size from the number of sources to three and yet maintain an average collision rate of less than 1%.

The rest of the paper is organized as follows: Section II reviews some previous related works; Section III presents the details of our proposed scheduling algorithm; Section IV analyzes the performance of the proposed scheduling algorithm; Section V presents the simulations and experimental results; Section VI summarizes our work and outlines some future work.

II. Background and Related Work

In a conventional media streaming system, the encoding of media data (e.g., video compression) is typically orthogonal to the way data are delivered over the network to the client. Given the media encoded data rate, the streaming protocol will need to ensure that media data can be delivered to the client in time for playback. As discussed in Section I, the Internet's inherent bandwidth fluctuations is a major challenge to this end and most of the existing solutions rely on pull-based protocols. The client react to bandwidth and peer quality fluctuations by dynamically rescheduling the data delivery process among multiple peers to compensate for any bandwidth deficiencies [7]-[10] or exploit path diversities in streaming data over multiple network paths [11]-[12].

Pull-based protocols can and do work well in practice for file sharing applications, e.g., BT. However, extending these protocols to media streaming is more challenging due to the far more stringent time constraints of streaming media. As pull-based protocols work by reacting to network resource fluctuations, it is essential to first detect such fluctuation and then react to it by reconfiguring the system accordingly. However, despite more than a decade of research, the challenge of bandwidth estimation along a network path is still not yet fully resolved. The nature of the Internet and the dynamics of competing traffics severely limit the accuracy of bandwidth measurement tools, especially over a short time scale.

In this paper, we explore an alternative approach to the problem by combining erasure codes with scheduling. The principle is to substitute bandwidth estimation and prediction by erasure codes and multi-source media streaming. This enables the system to utilize the available network resources fully without *a priori* knowledge of their availability.

A. Erasure-Correction Coding

Fig. 1 illustrates the process of erasure-correction coding. We first divide a segment of data into n data blocks. These n data blocks are then fed into an erasure-code encoder (e.g., RS codes encoder) which produces as output n/r encoded data blocks. The parameter r ($r<1$) is called the code rate and it controls the amount of redundancy in the encoded data. The key idea is that the client can recover the original n data blocks whenever any n blocks out of the n/r encoded data blocks are successfully received.

This last property can be exploited to solve the network resource estimation problem. Specifically, a segment of data is first distributed to k different peers. Upon receiving a request from the client, each peer encodes its data into n/r encoded data blocks and begins transmitting the encoded data blocks to the client. Thus, all k peers transmit data to the client simultaneously at a rate determined by their respective network resource availability. In other words, peers with more resources (i.e., fast peers) will transmit more data at a given time compared to slow peers. Moreover, the client can simply perform erasure correction to obtain the original n data blocks once it receives exactly n encoded data blocks from any combination of the peers. The exact number of encoded data blocks received from each of the peers no longer matters, thus eliminating the need to estimate network resource availability and to schedule data transmission among the peers.

This advantage does come with a price – the erasure encoding/decoding process is computation expensive. Worst still, the computation complexity of the traditional erasure codes such as RS codes increases exponentially with coding space size, which is proportional to the data length. For example, using Cauchy-based Reed-Solomon Codes [13] to encode a 1-MB data, for 1-KB packet size, the encode rate and decode rate achieved by a Pentium-class 2 GHz CPU are only 0.91 MBps and 0.30 MBps respectively.

More recently, researchers proposed a new class of erasure codes called rateless codes such as LT Codes [14], Online Codes [15], and Raptor Codes [16] to address the computational efficiency issue. Compare to traditional erasure codes, rateless codes are simple and efficient in encoding and decoding using only XOR operations. More importantly,

rateless-code encoder does not require a predetermined coding space size. It can dynamically generate coded data of any size on the fly, by performing XOR operations on a subset of the original data.

However, rateless codes' performance superiority can only be realized when the data length is large (e.g, hundreds of MBs or more). While this works well in file download [17] where the whole file can be encoded as a single unit, streaming applications require the division of the data stream into small data units (e.g., KBs) such that playback can begin shortly after a few units of data are received. For example, if we employ 1 MB data unit size at an aggregate streaming bit-rate of 256 kbps, then the initial startup delay will be at least 32 seconds. Using the Online codes decoder from [18] with number of data blocks equal to 10,000 in a Pentium-class 2 GHz CPU, the decode rate is only 0.54 MBps. Moreover, at this configuration the client will also need to receive an additional 5.09% of data, called decoding overhead, before it can recover the original data, thus further reducing its efficiency.

Note that while it is possible to reduce decoding overhead by increasing the number of data blocks in the encoding process, e.g., increasing it from 10,000 to 40,000 will reduce the decoding overhead from 5.09% to 4.09%., the decode rate will deteriorate further, dropping from 0.76 MBps to only 0.08 MBps – which is even lower than the traditional RS codes. On the other hand, decreasing the number of data blocks will improve decode rate at the expense of decoding overhead. See Table I for more examples of these tradeoffs. Therefore while rateless codes have several attractive properties and have been proposed for peer-to-peer video streaming [19]-[21], their computational complexity and decoding overhead still present significant challenges in the application to media streaming.

Another alternative codec is the class of Low-Density Parity-Check (LDPC) codes [22]-[24]. These LDPC codes have substantially better decoding performance than rateless codes. For example, the HLDPC codes [22] provide similar erasure recovery capabilities as the rateless codes but at a significantly lower computation complexity in decoding. The study in [24] shows that the decoding overheads of the LDPC-triangle codes for various code rate are less than 1%, even with a small number of blocks (i.e., 1,000), with decoding time always an order of magnitude faster than RS codes. Nevertheless, unlike rateless codes, the encoded data blocks in the LDPC codes are interdependent so that its memory requirement is proportional to the size of the coding space. Thus for large coding space the memory requirement will quickly become the bottleneck.

More recently, network coding [25] has also been applied to solve the data distribution problem in P2P streaming [26], [27]. Nevertheless network coding still suffer from the same encoding and decoding complexity issue as conventional erasure codes and the authors are currently investigating the application of the scheduling algorithms presented in this work to mitigate the problem in network-coding-based P2P systems.

TABLE I
PERFORMANCE OF ONLINE CODES OF ENCODING AND DECODING 1MB DATA

Number of blocks	Block size	Overheads	Encoding rate	Decoding rate
2000	500 B	15.38 %	17.22 MBps	6.70 MBps
4000	250 B	8.62 %	10.20 MBps	2.82 MBps
10000	100 B	5.09 %	4.44 MBps	0.76 MBps
20000	50 B	4.74 %	2.34 MBps	0.29 MBps
40000	25 B	4.09 %	1.22 MBps	0.08 MBps

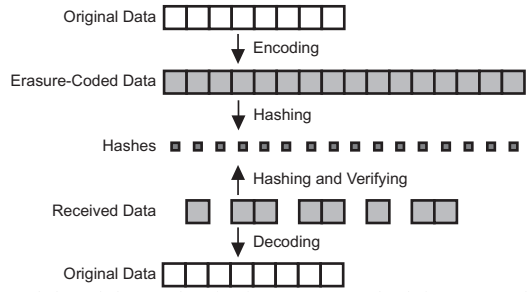

Fig. 2. Hash-based data authentication process. A hash is computed for each erasure-coded data block. The hashes are then distributed through a trusted channel.

B. *Integrity and Security in Content Distribution*

Apart from transmission efficiency, integrity and security are the fundamental problems in distributed systems. In particular systems such as P2P primarily relies on untrusted user hosts which could be compromised to inject malicious data into the system to disrupt normal service or worst – to intrude other hosts, e.g., by exploiting bugs in the media decoder. Currently this problem is solved by the use of data authentication as depicted in Fig. 2.

Specifically, the original source of the media data will compute a security hash for each media block (e.g., SHA1 [28]). These hashes are then distributed by a trusted server, typically together with a list of peers or trackers (e.g. BT) for new clients to begin the data distribution process. Armed with the hashes a client can then verify the received media data blocks and simply discard those that failed the test while requesting the missing media data from another peer.

This hash-based data authentication process works well in a pull-based data distribution protocol such as BT. Applying the same process to multi-source erasure-correction coded data distribution raises two challenges however. First, hash generation is computation expensive. For example, with 10 MB data and 100 peers (i.e., code rate is r=0.01), generating hashes using SHA1 for the 1 GB erasure-coded data on a Pentium-class processor running at 2 GHz can achieve an effective encode rate of only 0.13 MBps, which is even lower than the erasure-correction encode rate. Second, assuming a 20-byte SHA1 hash is generated for every 16 KB of data block (similar to BT), a total of 1,250 KB hashes will be generated for the 10 MB data, representing storage and bandwidth overheads of 12.2%.

The fundamental problem here is that in order to cater for the potentially large number of peers in the system, the content publisher must encode the data using a very small code rate, e.g., using 0.01 for 100 peers. This increases the computation complexity of both erasure-correction encoding and hash generation, and the storage overhead incurred by the generated hashes. By contrast, if the content publisher limits the coding

space to a small number, e.g., 10 peers, then it may not be able to utilize all the available peers in the system.

C. Comparisons

In summary, the existing erasure-correction codes all have limitations when applied to media streaming: RS codes are limited by their high computational complexity; rateless codes are limited by their computational complexity and/or decoding overhead; and LDPC codes variants limited by their memory requirement and/or decoding overhead. In addition, the hash-based data authentication process is also limited by the computational complexity and hash overhead. The root cause of these limitations is the need to match the code rate to the potentially large number of peers in the system so that the encoded data are all *orthogonal* across all peers.

This work proposed a radically different approach to solving this problem – we decouple the code rate from the size of the system, and then develop transmission scheduling algorithms to reduce the likelihood of duplicated data transmission (called collisions) from different peers. This approach enables the use of coding space significantly smaller than the number of peers in the system, thus substantially reducing the computational complexity and memory requirement for all types of erasure-correction codes and hash-generation processes. Although the encoded data, now being only partially orthogonal, may result in duplicated data transmissions, our preliminary results show that even including the effects of collisions our approach can still achieve lower transmission overhead than rateless codes (c.f. Section V). We present in the next section the principles of the proposed multi-source transmission scheduling algorithm.

III. PARALLEL TRANSMISSION SCHEDULING

In a P2P media streaming system, the original media data such as a video data stream is first sequentially divided into numerous segments and distributed to the peers. While a new client join the system, it collects the information of available peers from dedicated servers or trackers (e.g. BT) and downloads the media data segments successively from the available peers. If more than one peer carries the same media segment, then the segment is further subdivided into n data blocks, which are then encoded using an erasure-correction code to n/r encoded data blocks.

Now let k be the number of source peers in the system. If we set $r=1/k$, then a total of nk encoded data blocks will be produced for each media data segment. These nk blocks are then distributed equally among the k source peers so that there will be no duplication of any of the encoded data blocks. In actual streaming, the client simply connects to as many peers as they are available and requests them to transmit the encoded data simultaneously. Once n encoded data blocks are received, irrespective of which peers they originate from, the client can decode the original n data blocks to form the media data segment for playback. This process repeats for subsequent media data segments until playback completes.

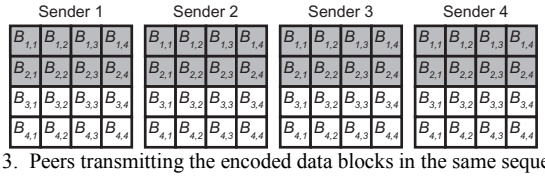

Fig. 3. Peers transmitting the encoded data blocks in the same sequence.

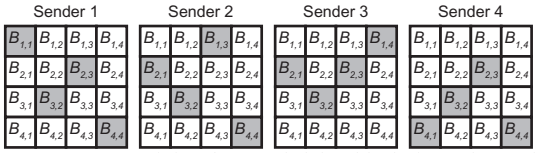

Fig. 4. Peers transmitting the encoded data blocks in random order.

A. Data Collisions

The number of peers in a distributed system such as P2P network can be very large, e.g., hundreds to tens of thousands. Thus the trivial approach of setting $r=1/k$ will run into scalability limitations as discussed earlier. Alternatively, if we decouple r from k, then by keeping r large we will be able to avoid these scalability limitations. However, this creates a new problem – data collisions in the transmission process.

Fig. 3 illustrates this data collision problem with $n=8$, $r=1/2$, and $k=4$. The 8 data blocks are encoded into 16 unique encoded data blocks, of which any eight of them will be sufficient for decoding the original eight data blocks. Assuming all four peers have a copy of the eight data blocks. Then upon receiving a request from the client, they will begin encoding the data blocks and transmit them as fast as network resources allow. Fig. 3 illustrates the situation with four peers all sending at the same rate and each sends out four encoded data blocks, which are indicated in grey colour. Immediately we can observe the collision problem – if the peers transmit the encoded data blocks in the same sequence (e.g., $B_{1,1}$, $B_{1,2}$, $B_{1,3}$, $B_{1,4}$, $B_{2,1}$, $B_{2,2}$, etc.) then the client will be receiving many duplicate encoded data blocks – data collisions. Duplicate encoded data blocks do not contribute to the decoding process and thus are simply discarded, thereby wasting network resources in transmitting them. Clearly better scheduling algorithms can reduce data collisions, which are the focus of the following sections.

B. Randomized Scheduler

The reason for the extensive collisions resulted from the transmission schedule in Fig. 3 is simple – the peers all have the same transmission schedule. Therefore, the first improved scheduling algorithm we consider is the Randomized Scheduler. Specifically, each peer will send out encoded data blocks in a random order so that the likelihood of duplicated transmission is reduced. Fig. 4 illustrates the Randomized Scheduler with four peers all sending at the same rate. After they send out 16 encoded data blocks as depicted in Fig. 4, the client receives eight distinct encoded data blocks while there are eight duplicated encoded data blocks.

The primary advantage of the Randomized Scheduler is its simplicity. Specifically, individual peer can form its own transmission schedule without the need to coordinate with other peers. This is a significant advantage in practice as coordination in a large-scale distributed system is non-trivial in its own right and will certainly incur additional control overheads. We will

analyze the performance of the Randomized Scheduler in Section IV, and experimentally in Section V.

C. Disjoint-Prefix Randomized Scheduler

Reconsidering the transmission collision problem one observation is that it is possible to avoid collision in the first place if we can guarantee the peers to transmit disjoint sets of the encoded data blocks. Specifically, we first divide the n/r encoded data blocks into k disjoint subsets, e.g., $\{B_{i,1}, B_{i,2},\ldots, B_{i,n/rk}\}$ for subset i=1,2,…,k. Peer i is assigned a schedule to always transmit its own disjoint subset i first, and then transmit the rest of the encoded data blocks using the Randomized Scheduler. We call this the Disjoint-Prefix Randomized Scheduler (DPRS).

Fig. 5 illustrates the DPRS with four peers all sending at the same rate. Each peer only sent out two encoded data blocks, the client could receive eight distinct encoded data blocks without any collisions. Compare to the Randomized Scheduler, DPRS guarantees that the first n/rk encoded data blocks transmitted are always unique. In the best-case scenario, all k peers send at the same data rate, resulting in zero collisions in transmitting the n/r encoded data blocks. Nevertheless, collisions are still possible if the peers' transmission rates are different, thus beyond the disjoint subset the remaining data blocks are transmitted in random order to avoid synchronized collisions.

More generally, we can vary the size of the disjoint subset in the DPRS from zero up to n/rk. With the size of the disjoint subset equal to zero the DPRS algorithm then simple reduces to the Randomized Scheduler. We analyze the DPRS algorithm in the next section and show that optimal performance is achieved when the disjoint subset size is equal to n/rk.

IV. PERFORMANCE MODELING

In this section, we investigate the worst-case and the average-case performances of the Disjoint-Prefix Randomized Scheduler (DPRS). Specifically, we derive the worst-case and average number of transmission collisions in terms of other system parameters including the number of original data blocks, the code rate, and the number of source peers. For simplicity, we assume the erasure codec has zero decoding overhead (e.g. RS codes). The results can be extended to other erasure correction codes with non-zero decoding overheads in a straightforward manner. Table II summarizes the notations used in the following sections.

A. System Model

Let n, r, k be the number of original data blocks, the code rate, and the number of source peers (numbered from zero to k-1) respectively. After erasure encoding we have n/r encoded data blocks. Suppose n/r is a natural number, we can define the set of encoded data blocks D to be a set of n/r natural numbers from zero to n/r–1 where each number represents an encoded data block, i.e.,

$$D = \{0,1,\ldots,n/r-1\} \tag{1}$$

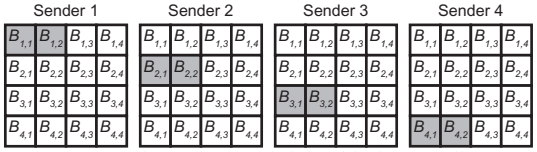

Fig. 5. Peers transmitting the encoded data blocks according to DPRS.

TABLE II
SUMMARY OF NOTATIONS USED IN THE DPRS MODEL

Symbol	Description
n	Number of original data blocks
r	Erasure code rate
k	Number of peers
s	Size of disjoint subset
D	Set of encoded data blocks
p_i	Set of blocks in the disjoint subset transmitted by peer i
q_i	Set of blocks in the randomized subset transmitted by peer i
w_i	Transmission rate of peer i
t	Transmission time
C	Collisions, number of duplicated blocks transmitted

Let s be the size of the disjoint subset such. Then peer i will first transmit its own disjoint subset i and then transmit the remaining encoded blocks randomly. Note that if we set s to zero then the DPRS algorithm reduces to the Randomized Scheduler.

We first make an observation in Theorem 1 for the case where there is no collision.

Theorem 1. There is no collision if all peers each transmit less than s blocks.

Proof. Under DPRS the first s blocks are guaranteed to be unique across all peers. Thus, in this case there will be no collision among transmissions from peers. □

A corollary of Theorem 1 implies that collision occurs only if at least one peer transmitted more than s blocks, and the collisions are thus from the randomized subset. Let p_i, q_i be the set of blocks sent by peer i in its own disjoint subset and randomized subset respectively at the time the client receives n unique encoded data blocks:

$$p_i \subseteq \begin{cases} \varnothing & s = 0 \\ \left\{\frac{in}{rk}, \frac{in}{rk}+1,\ldots,\frac{in}{rk}+s-1\right\} & 0 < s < \frac{n}{rk} \end{cases} \tag{2}$$

$$q_i \subseteq \begin{cases} D & s = 0 \\ D \setminus \left\{\frac{in}{rk}, \frac{in}{rk}+1,\ldots,\frac{in}{rk}+s-1\right\} & 0 < s < \frac{n}{rk} \end{cases} \tag{3}$$

where

$$|p_i| < s \Rightarrow q_i = \varnothing \tag{4}$$

Since each peer must send the disjoint subset before the randomized subset, q_i remains empty until the peer has transmitted the entire disjoint subset (i.e., $|p_i| = s$).

Let w_i be the transmission rate of peer i (in blocks/s), and t be the transmission time for the client to receive n unique encoded data blocks. Thus at time t, the number of blocks sent by peer i will be equal to $w_i t$:

$$w_i t = |p_i \cup q_i| = |p_i| + |q_i| \tag{5}$$

Let C be the number of collisions, which can be computed from

$$C = t \cdot \sum_{i=0}^{k-1} w_i - n \tag{6}$$

We now reconsider the sets p_i and q_i. As the client received exactly n unique encoded data blocks at time t, the cardinality

of the union of all p_i and q_i, must be equal to n:

$$\left|\bigcup_{i=0}^{k-1}(p_i \cup q_i)\right| = n \tag{7}$$

Noting that the p_i's are unique, we can rewrite the L.H.S of (7) as

$$\left|\bigcup_{i=0}^{k-1}(p_i \cup q_i)\right| = \left|\bigcup_{i=0}^{k-1} p_i\right| + E\left[\left|\bigcup_{i=0}^{k-1} q_i \setminus \bigcup_{i=0}^{k-1} p_i\right|\right] \tag{8}$$

where the second term counts the average number of unique encoded data blocks the client received from the randomized subset q_i's. As the p_i's are unique, the first term in the R.H.S. of (8) simply equals to the sum of the individual p_i's cardinality:

$$\left|\bigcup_{i=0}^{k-1} p_i\right| = \sum_{i=0}^{k-1}|p_i| \tag{9}$$

Now the second term in (8) can be expressed as follows:

$$E\left[\left|\bigcup_{i=0}^{k-1} q_i \setminus \bigcup_{i=0}^{k-1} p_i\right|\right] = \left|D \setminus \bigcup_{i=0}^{k-1} p_i\right| \cdot P\left\{b \in \bigcup_{i=0}^{k-1} q_i \,\middle|\, b \in D \setminus \bigcup_{i=0}^{k-1} p_i\right\} \tag{10}$$

which is the product of the total number of encoded data blocks not duplicated in all p_i's and the probability of such a data block to come from the randomized subsets q_i's. Next we rewrite (10) as

$$E\left[\left|\bigcup_{i=0}^{k-1} q_i \setminus \bigcup_{i=0}^{k-1} p_i\right|\right] = \left|D \setminus \bigcup_{i=0}^{k-1} p_i\right| \cdot \left(1 - P\left\{b \notin \bigcup_{i=0}^{k-1} q_i \,\middle|\, b \in D \setminus \bigcup_{i=0}^{k-1} p_i\right\}\right) \tag{11}$$

and then further express the probability term in terms of the individual q_j:

$$E\left[\left|\bigcup_{i=0}^{k-1} q_i \setminus \bigcup_{i=0}^{k-1} p_i\right|\right] = \left|D \setminus \bigcup_{i=0}^{k-1} p_i\right| \cdot \left(1 - \prod_{j=0}^{k-1}\left(1 - P\left\{b \in q_j \,\middle|\, b \in D \setminus \bigcup_{i=0}^{k-1} p_i\right\}\right)\right) \tag{12}$$

or simply:

$$E\left[\left|\bigcup_{i=0}^{k-1} q_i \setminus \bigcup_{i=0}^{k-1} p_i\right|\right] = \left(\frac{n}{r} - \sum_{i=0}^{k-1}|p_i|\right) \cdot \left(1 - \prod_{i=0}^{k-1}\left(1 - \frac{|q_i|}{n/r - s}\right)\right) \tag{13}$$

Equation (13) together with (2) to (6) relates the number of collisions with the sets p_i's and q_i's, and also the transmission rates of individual peers w_i's. With this model, we derive the worst-case and average-case results in the next two sections.

B. Worst-case Analysis

We observe that the number of collisions depends on the compositions of the sets p_i's and q_i's, which in turn depends on the transmission rates w_i's. Thus to derive the worst-case number of collisions we need to find the set of w_i's such that (6) is maximized. Table III summarizes the notations used in this section.

Theorem 1 states that there are collisions only if at least one peer transmitted more than s blocks. Let x be the number of *fast peers* which each sent more than s blocks. The remaining k-x peers are regarded as *slow peers*.

Theorem 2. *Irrespective of how many blocks each slow peer sent, the number of distinct blocks received is determined solely by the sum of the number of blocks sent by the slow peers.*

Proof. First substitute (9) and (13) into (8):

$$\left|\bigcup_{i=0}^{k-1}(p_i \cup q_i)\right| = \sum_{i=0}^{k-1}|p_i| + \left(\frac{n}{r} - \sum_{i=0}^{k-1}|p_i|\right) \cdot \left(1 - \prod_{i=0}^{k-1}\left(1 - \frac{|q_i|}{n/r - s}\right)\right) \tag{14}$$

Then we separate the p_i's into the sum of fast and slow peers:

$$\sum_{i=0}^{k-1}|p_i| = \sum_{i=0}^{x-1}|p_i| + \sum_{i=x}^{k-1}|p_i| = xs + \sum_{i=x}^{k-1}|p_i| \tag{15}$$

TABLE III
SUMMARY OF NOTATIONS USED IN THE WORST-CASE PERFORMANCE MODEL

Symbol	Description
x	Number of fast peers
y	Number of blocks transmitted by each fast peer
z	Number of blocks transmitted by all the slow peers
C^*	Collisions in worst-case
x^*	Corresponding value of x in worst-case
y^*	Corresponding value of y in worst-case
z^*	Corresponding value of z in worst-case

Let z be the sum of all p_i's for the slow peers:

$$z = \sum_{i=x}^{k-1}|p_i| \tag{16}$$

Finally substituting (15), (16) into (14), we have:

$$\left|\bigcup_{i=0}^{k-1}(p_i \cup q_i)\right| = xs + z + \left(\frac{n}{r} - xs + z\right) \cdot \left(1 - \prod_{i=0}^{x-1}\left(1 - \frac{|q_i|}{n/r - s}\right)\right) \tag{17}$$

Thus, the R.H.S. of (17) depends only on the sum of $|p_i|$ (i.e., z) but not the individual values. □

Next we consider the composition of the fast peers' transmission rates which maximizes collisions. We first establish in Theorem 3 a property for the worst-case scenario.

Theorem 3. *If the number of collisions is maximized, then all the fast peers must have sent the same number of blocks.*

Proof. Consider the product term in (17):

$$\prod_{i=0}^{x-1}\left(1 - \frac{|q_i|}{n/r - s}\right) \tag{18}$$

Applying the inequality of arithmetic and geometric means, we have

$$\prod_{i=0}^{x-1}\left(1 - \frac{|q_i|}{n/r - s}\right) \le \left(1 - \frac{\overline{|q_i|}}{n/r - s}\right)^x \tag{19}$$

in which the equality holds when $q_i = q_j$, for all $i, j \le x-1$, which is also the maximal for the product term. □

Theorem 3 implies that we no longer need to consider the individual fast peers' transmission rates. Instead, we let y be the number of blocks sent by each fast peer (i.e., $s<y<n$):

$$y = w_i t = |p_i| + |q_i| = s + |q_i| \quad 0 \le i \le x-1 \tag{20}$$

Then substituting (17) and (20) back into (6), we have

$$xs + z + \left(\frac{n}{r} - xs - z\right) \cdot \left(1 - \left(1 - \frac{y-s}{n/r - s}\right)^x\right) = n \tag{21}$$

Similarly, (6) can be rewritten in terms of x, y and z as

$$C = xy + z - n \tag{22}$$

In the following, we determine the values of x, y and z that maximize (22) subject to (21). First, we rearrange (21) subject to z:

$$z = \frac{n}{r} - xs - \left(\frac{n}{r} - n\right) \cdot \left(1 - \frac{y-s}{n/r - s}\right)^{-x} \tag{23}$$

Since z is non-negative, we can obtain an upper bound of y, denoted by y_{max}, in terms of x by setting $z \ge 0$ in (23):

$$y \le y_{max} = s + \left(\frac{n}{r} - s\right) \cdot \left(1 - \left(\frac{n/r - n}{n/r - xs}\right)^{\frac{1}{x}}\right) \tag{24}$$

This is the scenario where all the k-x slow peers do not send any data blocks.

Next, we substitute (23) into (22) to express the collisions in terms of x and y:

$$C = f(x,y) = xy + \frac{n}{r} - xs - \left(\frac{n}{r} - n\right)\cdot\left(1 - \frac{y-s}{n/r-s}\right)^{-x} - n \quad (25)$$

and then we can find the value of y, denoted by y_x, that maximizes C given x:

$$y_x = s + \left(\frac{n}{r} - s\right)\cdot\left(1 - \left(\frac{n/r-n}{n/r-s}\right)^{\frac{1}{x+1}}\right) \quad (26)$$

However the computed y_x may be larger than the upper bound in (24) and in that case the maximum is simply given by (24) as C is a strictly increasing function of y when $y \le y_x$. To combine the two cases we define $y_x = \min\{y_x, y_{max}\}$, which can be rewritten as

$$y_x^* = s + \left(\frac{n}{r} - s\right)\cdot\left(1 - \max\left\{\left(\frac{n/r-n}{n/r-xs}\right)^{\frac{1}{x}}, \left(\frac{n/r-n}{n/r-s}\right)^{\frac{1}{x+1}}\right\}\right) \quad (27)$$

To maximize C with respect to x, we note that x being the number of the fast peers, which is upper bounded by k. Thus we can obtain the value of x, denoted by x^*, that maximizes (25) from

$$x^* = \underset{x\in\{x\le k\}}{\operatorname{argmax}}\left(f(x, y_x^*)\right) \quad (28)$$

Finally from (23) and (27), we can obtain the corresponding values of y and z, denoted by y^* and z^* respectively, which maximizes C from

$$y^* = y_{x^*}^* \quad (29)$$

$$z^* = \frac{n}{r} - x^* s - \left(\frac{n}{r} - n\right)\cdot\left(1 - \frac{y^* - s}{n/r-s}\right)^{-x^*} \quad (30)$$

Hence, the worst-case number of collisions, denoted by C^*, can be obtained from

$$C^* = x^* y^* + z^* - n \quad (31)$$

We conjecture that the worst-case number of collisions is a non-decreasing function of k, i.e., the number of peers. Thus in the limiting case of infinite peers the upper bound is given by

$$C_{|k\to\infty}^* = x_{|k\to\infty}^* \cdot y_{|k\to\infty}^* + z_{|k\to\infty}^* - n = \frac{n}{r}\cdot\ln\left(\frac{1}{1-r}\right) - n \quad (32)$$

C. Average-case Analysis

The model in Section IV-A could also be used to find the number of collisions averaged over the peers' transmission rate distributions. Let the transmission rate w_i of peer i be a random variable with probability density function f_w and cumulative distribution function F_w on the interval [0, 1]. Then we can compute the average collisions from the expectation of (6):

$$E[C] = \int\cdots\int\left(t\cdot\sum_{i=0}^{k-1} w_i - n\right)\cdot f_w(w_0)\cdots f_w(w_{k-1})\cdot dw_0\cdots dw_{k-1} \quad (33)$$

However, (33) does not seem to be analytically tractable so we employ the systematic sampling technique [29] to compute numerical results for evaluation. Specifically, assuming the cumulative distribution function F_w is invertible (i.e., $F^{-1}{}_w$ exists), then we can generate the peers' transmission rates using the inverse transform sampled from the inverse function. Let U be a discrete random variable with uniform distribution on [0, $1/k$]. Applying the systematic sampling technique, we can express the peers' transmission rates in term of U with equal sampling interval (i.e., $1/k$):

$$w_i = F_w^{-1}(U + i/k) \quad (34)$$

Using the model in Section IV-A, we could able to find the number of collisions for the peers' transmission rate given by (34). Hence, we can obtain an approximate value of the expected collisions by taking the average for every value in U:

$$E[C] \approx \sum_U\left(\left(t\cdot\sum_{i=0}^{k-1} w_i - n\right)\cdot f_U(U)\right) \quad (35)$$

Our simulation results show that the numerical results computed from the sampling technique is very accurate.

V. Simulation And Experimental Results

In this section, we report simulation results to validate the mathematical model derived in Section IV and experimental results conducted in the Planetlab [30] to evaluate the performance of DPRS in real-world network settings. Clearly, the number of collisions is proportional to the number of blocks n. Therefore, in the following evaluations we normalize the number of collisions by n and plot collisions as the ratio C/n.

A. Model Validation and Performance Evaluation

We developed a discrete-event simulator to simulate the streaming of erasure-coded video data from multiple sources in a simplified network setting. As our goal is to validate the mathematical model in Section IV, the simulator omits some details such as network delay, peer churn, etc. A more detailed simulation study would require a more complete system design, including the control protocols, data distribution policies, competing traffics, and so on and is a subject for further study.

We model bandwidth variations across different peers using the Kumaraswamy distribution [31] which is a family of continuous probability distributions defined on the interval [0, 1] differing in the values of their two non-negative shape parameters, a and b. By varying these two parameters, we can generate a wide range of bandwidth distributions.

At the beginning of each simulation run, each peer in the system is randomly assigned a bandwidth according to the Kumaraswamy distribution. The sending peers then transmit the erasure-coded data blocks according to the DPRS algorithm at the assigned bandwidth. Once the client receives n encoded data blocks, the transmission process is terminated and the number of collisions counted.

We carry out two simulations, one for worst-case results and the other for average-case results. In the worst-case simulations, we ran a large number (thousands) of trials with different parameters $\{a, b\}$ of the Kumaraswamy distribution and recorded the maximum number of collisions. In the average-case simulations, we ran one hundred trials for each set of parameters $\{a, b\}$ and record the average number of collisions. In both cases, the size of the disjoint subset s is set to the maximum, i.e., n/rk.

Fig. 6 shows the analytical and simulated worst-case collisions against the code rate r for k=10, 20, 50, and 100 sending peers. We observe that the analytical and simulation results agree reasonably well, with the simulated collisions slightly lower than the analytical counter-part.

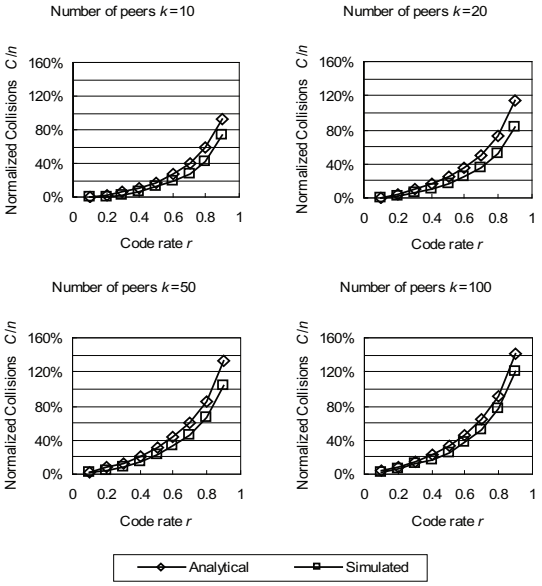

Fig. 6. Analytical and simulated worst-case collisions against the code rate r.

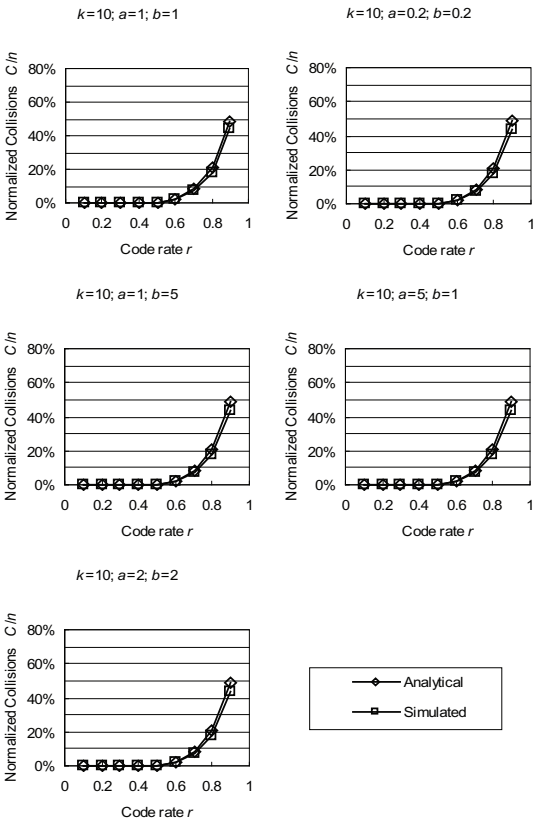

Fig. 8. Analytical and simulated average collisions against the code rate r.for number of peers k=10 with respect to different shape parameters a and b.

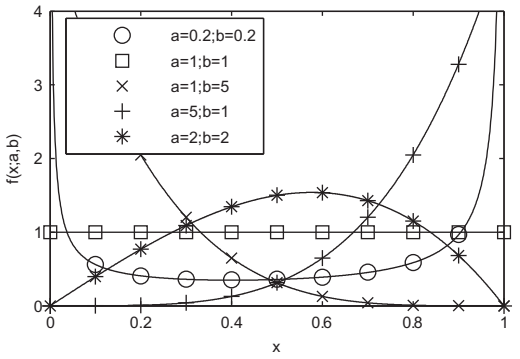

Fig. 7. Probability density function of Kumaraswamy distribution with respect to different shape parameters a and b.

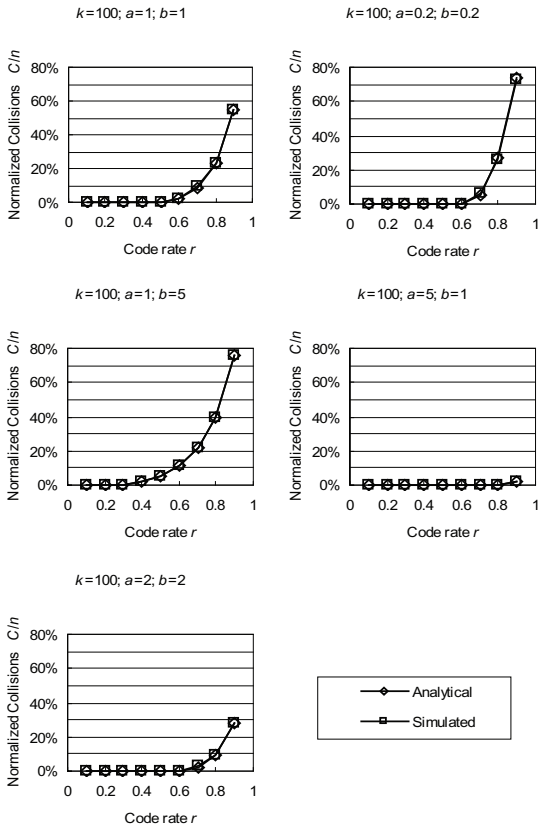

Fig. 9. Analytical and simulated average collisions against the code rate r.for number of peers k=100 with respect to different shape parameters a and b.

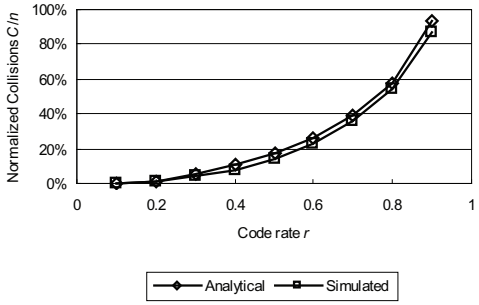

Fig. 10. Analytical and trace-driven simulated worst-case collisions against the code rate r.

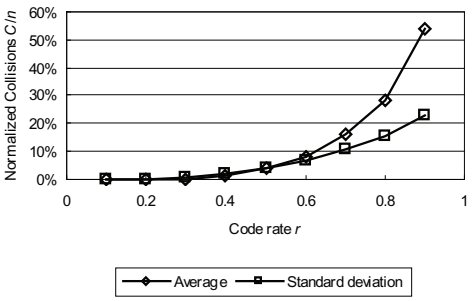

Fig. 11. Trace-driven simulated average collisions against the code rate *r*.

This is expected as the simulation does not exhaust all possible parameter combinations and so tends to underestimate the worst case collisions. We conducted a large number of additional simulation results (not shown) and found them to agree with the analytical results consistently, thereby validating the mathematical model in Section IV.

More importantly, the results reveal that we can achieve relatively low worst-case collision (e.g., 11.59% for 100 peers) even for code rate as large as r=0.3. By contrast, conventional erasure coding without multi-source scheduling for 100 peers have a code rate of r=0.01, or a coding space 30 times larger than using DPRS. This reduction in coding space will result in significantly lower computation and hashing overheads as discussed in Section II.

In the average-case simulations, we carry out five sets of simulations with different parameter sets {a, b} for the Kumaraswamy distribution. These five-parameter sets {a, b} represent a wide range of bandwidth distributions as shown in Fig. 7. Fig. 8 and Fig. 9 plots the average collisions versus code rate for k=10 and 100 respectively. We again observe that the simulation results closely match the numerical results computed from the system model in Section IV, thereby validating the system model.

In addition, the results show that the average collision depends heavily on the property of the bandwidth distribution. For example, bandwidth distribution with small variance, e.g., {a=5; b=1} and {a=2; b=2}, result in significantly lower average collisions than bandwidth distributions of large variance, e.g., {a=0.2; b=0.2} and {a=1; b=5}.

More importantly, with DPRS the average collisions can be reduced to negligible levels for code rate of just r=0.3. Compared to conventional erasure coding (with r=0.01 for k=100) the proposed DPRS algorithm can achieve comparable performance and yet can reduce the coding space size by one order of magnitude.

B. Trace-driven Simulation Results

The bandwidth distribution used in Section V.A is merely a mathematical model and thus may not resemble real-world network bandwidth distributions. To address this limitation we carried out extensive bandwidth measurements conducted in the Planetlab to capture bandwidth distributions and variations in a real network setting.

TABLE IV
COMPARISON OF DPRS AND ORDINARY PUSH-BASED APPROACHES IN DISTRIBUTING 10-MB DATA FROM 10 PEERS

	(a)	(b)	(c)	(d)
Scheduling Scheme	DPRS	DPRS	Conventional	Conventional
Coding Scheme	Online codes	RS codes	Online codes	Online codes
Code Rate	0.3	0.5	0.1	0.01
Effective Encoding Rate	9.75 MBps	0.79 MBps	3.14 MBps	0.32 MBps
Effective Hashing Rate	4.11 MBps	6.86 MBps	1.37 MBps	0.13 MBps
Decoding Rate	6.61 MBps	1.57 MBps	6.61 MBps	6.61 MBps
Hashes Overhead	0.41%	0.24%	1.22%	12.20%
Decoding Overhead	4.74%	0%	4.74%	4.74%
Average Collisions	0.28%	3.84%	0%	0%
Total Overhead	5.43%	4.08%	5.96%	16.94%

Specifically, a test program was installed to the PlanetLab nodes (total 286 nodes) for the measurements. In each measurement, N+1 nodes are randomly drawn from the pool of PlanetLab nodes, with one node acting as the receiver and the remaining N nodes acting as senders. All N senders then simultaneously transmit data to the receiver using TCP. The captured bandwidth traces are then fed into the simulator to obtain collision performances.

Fig. 10 plots the maximum collisions measured in the trace-driven simulations versus different code rate r. The results show clear agreement between the analytical results and the simulated results. Fig. 11 plots the average collisions versus code rate. Only the simulated results are shown as the curves for the analytical results overlap with the simulated ones.

We observe that in real network settings, the proposed DPRS algorithm can also reduce the average collisions to negligible levels using a code rate of just r=0.3. This result suggests that the DPRS has potential to work well in real network settings.

C. Comparisons

Table IV compares the proposed DPRS algorithm with conventional push-based transmission in distributing a 10 MB data block from 10 peers to a receiver, running in a 2 GHz Pentium-class processor. Total overhead is the sum of decoding overhead and average collisions, and represent the amount of extra network bandwidth used in delivering the data block.

Case (c) and (d) both employ conventional push-based transmission using Online codes [18] and SHA1 [28] as the hashing algorithm. The only difference is the code rate, of which case (c) is 0.1 and case (d) is 0.01. The code rate limits the number of peers that can participate in the transmission process, e.g., code rate of 0.1 and 0.01 allow up 10 and 100 peers respectively. Decreasing the code rate increases the number of sending peers at the expense of significantly lower encoding rate (e.g., 3.14 Mbps v.s. 0.32 Mbps) and hash generating rate (e.g., 1.37 Mbps v.s. 0.13 Mbps), and significantly higher total overheads (e.g., 5.96% v.s. 16.94%).

By contrast, with DPRS in case (a) we not only can achieve substantially lower total overheads (e.g., 5.43% versus 16.94%), but also at significantly higher encoding rate (e.g., 9.75 Mbps versus 0.32 Mbps) and hashing rate (e.g., 4.11 Mbps versus 0.12 Mbps). DPRS also enables the use of more computation-intensive codes such as RS codes [13] (case (b)).

The results show that DPRS is more bandwidth efficient, requires lower computation power, and thus more scalable to a large peer population.

VI. CONCLUSION

Unlike pull-based approaches, the push-based multi-source streaming model investigated in this work eliminates the need for network resource estimation that is error-prone and difficult to implement in real networks. The proposed DPRS transmission scheduling algorithm solves the scalability problem inherent in erasure-correction codes and thus make application of the multi-source streaming model to large-scale distributed systems feasible. With the promising preliminary results, we will extend the research in two directions.

First, the two transmission schedulers described in Section III are only a starting point. In particular, the schedulers assume absolutely no knowledge of the peers' network resources, which may not be the case in practice. For example, while the peers' network resource availabilities could not be accurately predicted or estimated, they may still be bounded by practical constrains, such as the physical constraint set by the network link, or logical constraint set by the user. In this case, a partial knowledge of the peer bandwidth distribution, for example, can be exploited in the design of the transmission scheduler. More generally, the transmission scheduler may also be combined with existing bandwidth estimation tools so that partial network knowledge can be integrated into the transmission scheduler to achieve better performance.

Second, the transmission schedulers described in Section III are static in the sense that they are fixed *a priori* and remain the same for the rest of the streaming session. In real networks, the amount of network resource available often fluctuates from time to time. Thus if partial network resource information can be gathered online then it will open up the possibility to dynamically reconfigure the transmission schedules to adapt to the changing network conditions.

Finally, we intend to implement the proposed schedulers into a working P2P streaming platform so that real-world performance results can be obtained to verify the mathematical models, and to demonstrate the feasibility of multi-source streaming of erasure-coded media data in real networks.

ACKNOWLEDGMENTS

This work was partially supported by a grant from the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-02/08) and the Shun Hing Institute of Advanced Engineering of the Chinese University of Hong Kong (Project No. MMT 9/07).

REFERENCES

[1] AR Reibman, H. Jafarkhani, Y. Wang, M. Orchard, and R. Puri, "Multiple Description Coding for Video Using Motion Compensated Prediction," *Int'l Conf on Image Processing*, Kobe, Japan, vol.3, pp.837-41, Oct 1999.

[2] Y. Q. Liang and Y. P. Tan, "Methods and Needs for Transcoding MPEG-4 Fine Granularity Scalability Video," *IEEE Int'l Sym on Circuits and Systems*, Scottsdale, Arizona, vol.4, pp.719-722, May 2002.

[3] A. Vetro, C. Christopoulos and Huifang Sun, "Video Transcoding Architectures and Techniques: an Overview," *IEEE Signal Processing Magazine*, vol. 20, Is-sue 2, pp.18 – 29, March 2003.

[4] L. S. Lam, Jack Y. B. Lee, S. C. Liew, and W. Wang, "A Transparent Rate Adaptation Algorithm for Streaming Video over the Internet," *18th International Conference on Advanced Information Networking and Applications*, Fu-kuoka, Japan, March 2004

[5] I.S. Reed, G, Solomon, "Polynomial Codes Over Certain Finite Fields," *J. Soc. Indust. Appl. Math*, Vol. 8, pp. 300-304, 1960.

[6] B. Cohen, "Incentives build robustness in bittorrent," in *Workshop on Economics of Peer-to-Peer Systems*, Berkeley, USA, May 2003.

[7] P. Rodriguez, W. Biersack, "Dynamic parallel access to replicated content in the Internet," *IEEE Trans. Networking*, 10(4):455–465, August 2002.

[8] A. Zeitoun, H. Jamjoom, M. El-Gendy, "Scalable parallel-access for mirrored servers," *Proc. IASTED International Symposium on Software Engineering, Databases, and Applications*, pp.93–98, February 2002.

[9] J. Funasaka, K. Nagayasu, K. Ishida, "Improvements on Block Size Control Method for Adaptive Parallel Downloading," *ICDCSW*, 2004.

[10] Z. Xu, L. Xianliang, H. Mengshu, Z. Chuan, "A speedbased adaptive dynamic parallel downloading technique," *ACM SIGOPS Operating Systems Review*, 2005.

[11] T. Nguyen and A. Zakhor, "Path diversity with forward error correction (pdf) system for packet switched networks," in *Proc. IEEE INFOCOM*, 2003, vol. 3, pp. 663–672.

[12] B. Wang, W. Wei, Z. Guo, and D. Towsley, "Multipath live streaming via TCP: scheme, performance and benefits," in *ACM CoNEXT*. New York, NY, USA: ACM, 2007, pp. 1–12.

[13] M. Luby, open-source cauchy-based Reed-Solomon codes [Online]. Available: http://www.icsi.berkeley.edu/~luby/.

[14] M. Luby, "LT-codes," in *Proceedings of the ACM Symposium on Foundations of Computer Science (FOCS)*, 2002.

[15] P. Maymounkov, "Online Codes," *NYU Technical Report TR2002-833*, November 2002.

[16] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, 52(6):2551–2567, June 2006.

[17] P. Maymounkov and D. Mazieres, "Rateless codes and big downloads," *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems*, February 2003.

[18] M. Knezevic, V. Velichkov, B. Preneel, I. Verbauwhede, open-source Online Codes [Online]. Available: http://sourceforge.net/projects/onlinecodes.

[19] J. Wagner, J. Chakareski and P. Frossard, "Streaming of Scalable Video from Multiple Servers using Rateless Codes," in *Proceedings of IEEE International Conference on Multimedia and Expo 2006*, July 2006.

[20] C. Wu and B. Li, "rStream: Resilient and Optimal Peer-to-Peer Streaming with Rateless Codes," *IEEE Trans. Parallel and Distributed Systems*, vol. 19, no. 1, pp. 77–92, Jan. 2008.

[21] M. Knezevic, V. Velichkov, B. Preneel, and I. Verbauwhede, "On the Practical Performance of Rateless Codes," in *International Conference on Wireless Information Networks and Systems*, 4 pages, 2008.

[22] K. Nybom and J. Bj¨orkqvist, "Hldpc codes - low density, low complexity, efficient erasure correcting codes," in *Proceeding of the 13th European Wireless Conference*, Apr 2007.

[23] V. Roca, C. Neumann, and D. Furodet, "Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes," *Internet Engineering Task Force*, RFC 5170, june 2008.

[24] M. Cunche and V. Roca, "Improving the Decoding of LDPC Codes for the Packet Erasure Channel with a Hybrid Zyablov Iterative Decoding/Gaussian Elimination Scheme," *INRIA Research Report RR-6473*, Tech. Report., March 2008.

[25] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, 2000.

[26] M. Wang, and B. Li, "R^2: Random Push with Random Network Coding in Live Peer-to-Peer Streaming", in *IEEE Journal on Selected Areas in Communications*, vol.25(9), pp. 1655-1666, 2007.

[27] C. Feng, and B. Li, "On Large-Scale Peer-to-Peer Streaming Systems with Network Coding", in *Proc. of ACM Multimedia*, 2008.

[28] FIPS 180-1, Secure Hash Standard, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, VA, Apr. 1995.

[29] Ken Black, *Business Statistics: Contemporary Decision Making*, 6th Edition, Wiley, 2009, Ch 7.

[30] Planetlab [Online]. Available: http://www.planet-lab.org/.

[31] P. Kumaraswamy, "A Generalized Probability Density Function for Double-Bounded Random Processes," *J. of Hydrology*, 46:79–88, 1980.