Round-Robin Overlapping Generations Coding for Fast Content Download

Gauri Joshi
EECS Dept., MIT
Cambridge, MA 02139, USA
Email: gauri@mit.edu

Emina Soljanin
Bell Labs, Alcatel-Lucent
Murray Hill NJ 07974, USA
Email: emina@alcatel-lucent.com

***Abstract*—We analyze the download time of a large file, divided into chunks called generations, and transmitted over an erasure channel without feedback. We compare the round-robin and random scheduling policies of coding over generations, and show that round-robin scheduling gives significantly faster file download. We then analyze coding with overlapping generations and show that the optimal overlap size is small compared to the number of generations, which implies that the download time can be reduced with only a small increase in computational complexity. Further, for a given overlap size, we propose overlap structures lower complexity and easy to implement, but still give file download as fast as the best previously proposed structures.**

I. Introduction

A. Motivation

Recently, there has been a rapid increase in audio/video traffic on the Internet. A survey of Internet traffic composition [1] shows that real-time entertainment including Netflix, YouTube, and BitTorrent contributes to a staggering 58% of the fixed access traffic in North America. The reason for this dominance is that these applications require the transfer of inherently large files. In addition, they demand fast content delivery to the user. Thus, there is a need to design transmission schemes that guarantee fast download with limited available bandwidth.

B. System Model

We consider point-to-point transmission of a large file of M packets. In each time slot, the source transmits a coded packet which is a linear combination of some subset of the M packets in the file. Assume that the coefficients are chosen from a field of large enough alphabet size such that all combinations are linearly independent. Each transmitted packet is received in error, or erased with probability ϵ, and received successfully otherwise. This model is suitable for the Internet since it uses checksum tests to detect errors with high probability. We assume that the source receives an instantaneous and error-free acknowledgment on the successful decoding of all M packets; there is no intermediate feedback. Our objective is to minimize the download time – the time from start of transmission until the receipt of the acknowledgment.

C. Previous Work

The best throughput-delay trade-off is achieved when each coded packet is an independent linear combination of all M packets, such that the file can be decoded from any M unerased combinations. However, this is impractical for large M since decoding involves the computationally expensive inversion of a size M matrix. Sparse codes proposed in [2], [3] can be used to reduce complexity, but they lose their sparsity properties over a general network. Batched sparse codes [4] preserve the sparsity properties by performing network coding at intermediate nodes in the network.

Another way to reduce the computational complexity proposed in [5], [6] is to divide the file into disjoint sets of packets called generations (or chunks), and form each coded packet by a linear combination of packets in one of the generations. This idea is particularly useful in a peer-to-peer network where the generations can be stored and downloaded from different nodes in the network.

During the file transmission, the user may receive more coded packets for some generations than others. This asymmetry arises for two reasons – 1) the scheduling policy at the source may favor some generations while transmitting coded packets, and 2) the coded packets of some generations may experience more erasures. This motivates the idea of overlapping generations, which has received a lot of attention in recent work, see e.g. [7]–[9] and references therein. In [7], the authors propose a random annex code, where the overlap packets are randomly chosen across the file. A better overlap structure based on expander graphs is presented in [8]. Both these papers have considered random scheduling over generations, and for a lossless channel. Round-robin scheduling is considered only in [9], which analyzes the complexity of coding over disjoint generations, and shows that small generation sizes are best for practical implementation.

D. Our Contributions

We present a novel theoretical analysis round-robin scheduling, comparing it with the random scheduling policy considered in most previous work [6]–[8]. We show that the expected download time is proportional to $n \log n$ for both policies, but round-robin scheduling gives a smaller proportionality constant for all values of erasure probability ϵ.

For coding with overlapping generations, we derive an upper bound on the optimal size of overlap, and show that it scales $O(\log n)$ with the number of generations n. We also propose *deterministic* structures of overlap that are low-complexity, and easier to implement than random overlap structures presented in previous work [7], [8].

II. Scheduling Over Disjoint Generations

A. Dividing the File into Generations

As shown in previous work, the computational complexity of coding can be reduced by dividing the file into disjoint sets of packets called generations, defined as follows.

Definition 1 (Disjoint Generations)**.** *A file with M packets is divided into n disjoint sets B_i, $1 \leq i \leq n$ called generations, consisting of $h = M/n$ packets each. Each coded packet is a linear combination of the packets in generation B_i for some $1 \leq i \leq n$.*

A generation is said to be decoded when h unerased coded packets are received by the user. Since we are interested in the transmission of files of large size, assume that the number of generations $n \gg h$, the size of each generation.

In every slot the source selects a generation B_i, and transmits a linear combination of the h packets in it. Since there is no intermediate feedback, the scheduling policy that dictates the choice of a generation B_i in every slot must be blind to the status of decoding at the user. We consider such a class of policies called symmetric scheduling policies formally defined as follows.

Definition 2 (Symmetric Scheduling Policy)**.** *A scheduling policy is symmetric over the n generations if the number of time slots T_i, from an interval of T slots, in which it transmits a coded packet from generation B_i satisfies*

$$\lim_{T \to \infty} \frac{T_i}{T} = \frac{1}{n}$$

We analyze the expected download time of the file with two such symmetric scheduling policies that operate without intermediate feedback - namely random and round-robin scheduling. Random scheduling is asymptotically symmetric as $T \to \infty$, whereas round-robin scheduling is symmetric for any interval of $T = n$ slots.

B. Random Scheduling

In the random scheduling policy, the source randomly picks one of the n generations in every slot and transmits a coded packet from that generation. Random scheduling is useful in a peer-to-peer network because it does not require any coordination between the nodes.

Theorem 1 (Expected Download Time with Random Scheduling)**.** *The expected download time of a file divided into n generations of h packets each for large n is*

$$\mathbb{E}[T_{n,h}^{(ra)}] = \frac{n}{1-\epsilon}\left(\log n + (h-1)\log\log n + o(\log\log n)\right), \tag{1}$$

where $\log$ *is with respect to the natural base.*

Proof: The download time $T_{n,h}^{(ra)}$ of the file is the maximum of the decoding times of its n generations. Its expected value $\mathbb{E}[T_{n,h}^{(ra)}]$ for the lossless channel with $\epsilon = 0$ is derived in [7]. Since each transmission is statistically identical, independent channel erasures with probability ϵ simply increase the expected download time by the factor $1/(1-\epsilon)$. ■

We observe that in (1), every extra packet added to each generation, that is an increase in h contributes to an additional $\frac{n}{1-\epsilon}\log\log n$ term to the growth of $\mathbb{E}[T_{n,h}^{(ra)}]$.

C. Round-Robin Scheduling

In this scheduling policy, the source transmits combinations from each of the n generations in a round-robin fashion. Unlike random scheduling, this policy if applied to a peer-to-peer network, requires a mechanism to poll the nodes in a round-robin fashion.

Theorem 2 (Expected Download Time with Round-Robin Scheduling)**.** *The expected download time of a file divided into n generations of h packets each for large n is*

$$\mathbb{E}[T_{n,h}^{(rr)}] = n(\log_{\frac{1}{\epsilon}} n + (h-1)\log_{\frac{1}{\epsilon}}\log_{\frac{1}{\epsilon}} n + o(\log_{\frac{1}{\epsilon}}\log_{\frac{1}{\epsilon}} n)) \tag{2}$$

where ϵ is the erasure probability of the channel.

Proof: In the round-robin scheduling policy, each round consists of n slots where in the k^{th} slot, the source transmits a combination of packets in the k^{th} generation. Define X_k as the number of rounds required to decode the k^{th} generation. The random variables X_k, $1 \leq k \leq n$ are i.i.d. according to the negative binomial PMF given by

$$\Pr(X = m) = \binom{m-1}{h-1}\epsilon^{m-h}(1-\epsilon)^h. \tag{3}$$

When $h = 1$, that is each generation consists of a single packet, the negative binomial PMF in (3) reduces to a geometric distribution.

The download time $T_{n,h}^{(rr)}$ of the file is equal to n times the number of rounds required to decode all generations, and its expected value is given by

$$\mathbb{E}[T_{n,h}^{(rr)}] = n \cdot \mathbb{E}[\max(X_1, X_2, ..X_n)]. \tag{4}$$

We obtain (2) by substituting the expression for the asymptotic scaling of $\mathbb{E}[\max(X_1, X_2, ..X_n)]$ derived in [10]. ■

D. Comparative Analysis

The dominant terms in the expected download times of the two policies derived in (1) and (2), are $n\log_{\frac{1}{\epsilon}} n$ and $\frac{n}{1-\epsilon}\log n$ respectively. Although both are proportional to $n\log n$, the proportionality constant $1/\log{1/\epsilon}$ with round-robin scheduling is lower than $1/(1-\epsilon)$ with random scheduling for all values of erasure probability ϵ.

Even for $\epsilon = 0$, the expected download time for random scheduling is $O(n\log n)$, similar to the coupon-collector's problem. Round-robin scheduling eliminates the $\log n$ growth term and gives a download time of $M = nh$ slots.

When $h = M$, we do not divide the file into generations and hence $\mathbb{E}[T_{n,h}^{(rr)}]$ and $\mathbb{E}[T_{n,h}^{(ra)}]$ are equal to $M/(1-\epsilon)$, which is the lowest achievable expected download time. Thus, we use $M/(1-\epsilon)$ as the normalization constant when comparing the

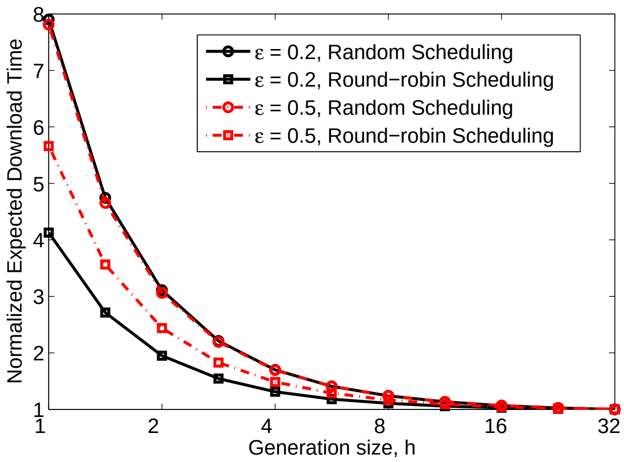

Fig. 1. Simulation plots of the expected download times $\mathbb{E}[T_{n,h}^{(ra)}]$ and $\mathbb{E}[T_{n,h}^{(rr)}]$ with the random and round-robin scheduling normalized by their lower bound $M/1-\epsilon$, plotted against the generation size h. For small generation sizes, the round-robin policy gives significantly faster download.

download time for the two policies in Fig. 1. It is a simulation plot of $\mathbb{E}[T_{n,h}^{(rr)}]$ and $\mathbb{E}[T_{n,h}^{(ra)}]$ normalized by $M/(1-\epsilon)$ for file size $M = 1024$ packets, erasure probability $\epsilon = 0.2$, and $\epsilon = 0.5$. We observe that the download time with round-robin scheduling outperforms random scheduling for small h, and as h increases to M, both converge to $M/(1-\epsilon)$.

Fig. 1 shows that the normalized expected download time for round-robin scheduling increases with ϵ, but it remains the same for random scheduling because of the $1/1-\epsilon$ scaling factor in (1). This shows that the inefficiency of random scheduling is not because of channel erasures, but is in the policy itself.

III. Coding with Overlapping Generations

In this section we discuss the motivation behind coding with overlapping generations and analyze how the size and structure of overlap between generations affects the download time.

A. Motivation behind Overlap

The download time of the file with n disjoint generations is the maximum of the time taken by each generation to collect h combinations and decode, and it is $O(n \log n)$ as derived in Section II. For both random and round-robin scheduling, it can be shown [11] that the expected minimum of the decoding times of the n generations is $O(hn)$, which is significantly smaller than $O(n \log n)$, for h and n in our range of interest. Since there is no intermediate feedback, the source continues to transmit coded packets from all generations, irrespective of their decoding. Hence, by the time the last generation(s) collect h combinations, the earlier decoded generations may collect extra combinations.

In Fig. **??** we plot a histogram of the number of combinations collected by each generation till the download is complete, that is when all generations decode. The round-robin scheduling policy is used to pick a generation in every slot. The simulation parameters are $M = 512$, $n = 32$, $h = 16$ and $\epsilon = 0.2$. We observe that by the time the last generation(s) crosses the threshold of $h = 16$ combinations and decodes, some generations have collected as many as 23 combinations.

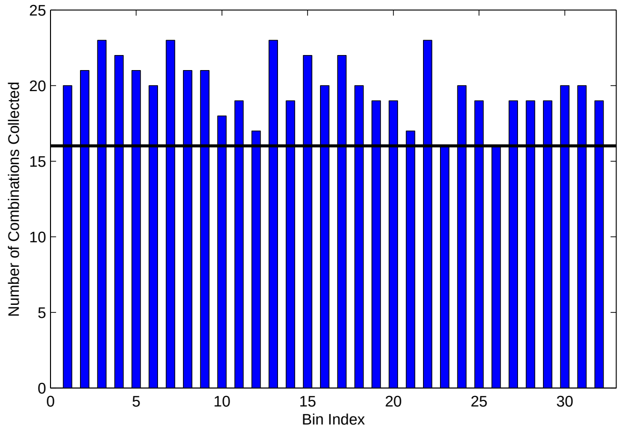

Fig. 2. Histogram of the number of combinations received by each generation, at the download completion time, when all generations cross the threshold of $h = 16$ denoted by the horizontal black line.

This motivates introduction of overlap between generations. If a generation has l packets overlapping with other generations, it needs $h + l$ combinations for decoding, but the l overlap packets can be back-substituted to help in faster decoding of other generations. Thus, we are increasing the time to decode the first generation, in order to decrease the total download time. Theoretical analysis of this decoding process will be presented in [11].

Definition 3 (Overlapping Generations)**.** *A file consisting of M packets is divided into n disjoint sets B_i, $1 \leq i \leq n$ called base generations, with $h = M/n$ packets each. Define sets R_i of l overlap packets each such that $B_i \cap R_i = \emptyset$ for all $1 \leq i \leq n$. Each coded packet is a linear combination of the $g = h + l$ packets in the extended generation $G_i = B_i \cup R_i$, for $1 \leq i \leq n$.*

We now analyze the two key parameters that goven the download time of the file – the overlap size l, and the choice of overlap sets R_i for $1 \leq i \leq n$ for a given l, or the structure of overlap.

B. Size of Overlap

There is a trade-off in the choice of the optimal size of overlap l that minimizes the expected download time. Increasing l results in an increase in the initial time taken for a generation to collect $g = h + l$ combinations decode. But with a larger overlap size l, a decoded generation can help the other generations to decode faster. Theorem 3 gives an upper bound on l.

Theorem 3 (Upper Bound on Overlap Size)**.** *For a file divided into n disjoint generations of h packets each, the introduction of l overlap packets to each generation reduces the expected*

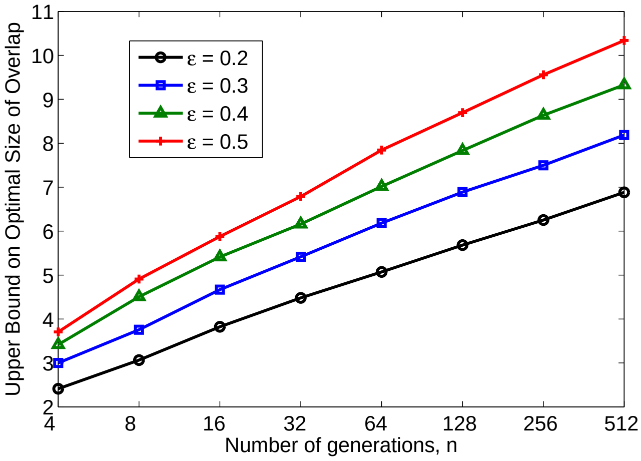

Fig. 3. Upper Bound on the optimal overlap size l, with round-robin scheduling plotted against the number of generations n. It increases with erasure probability ϵ and grows linearly with $\log n$.

download time if,

$$l \leq \max\left(\frac{(1-\epsilon)\mathbb{E}[T_{n,h}]}{n} - h, 0\right), \tag{5}$$

where $\mathbb{E}[T_{n,h}]$ is the expected download time with disjoint generations and a symmetric scheduling policy.

Proof: Let $T_{n,h,l}$ be the download time of a file with n generations where each generation has h base packets and l overlap packets. Introducing overlap between generations reduces the expected download time of the file if

$$\mathbb{E}[T_{n,h}] \geq \mathbb{E}[T_{n,h,l}] \geq \frac{n}{1-\epsilon}(l+h), \tag{6}$$

where we lower bound $\mathbb{E}[T_{n,h,l}]$ by the expected time for one generation to collect $g = l + h$ packets. For a symmetric scheduling policy, a coded packet is transmitted from a given generation $1/n$ fraction of times, and the probability it being received successfully is $(1-\epsilon)$. Hence, expected time for a generation to collect $g = l + h$ packets is $\frac{n}{1-\epsilon}(l+h)$. Rearranging the terms in (6) gives the result (5). ■

In Section II we showed that the expected download time $\mathbb{E}[T_{n,h}]$ scales $O(n \log n)$ with the number of generations n for both random and round-robin scheduling policies. Thus, using Theorem 3 we can infer that for these policies the best size of overlap is $O(\log n)$, which implies that the computational complexity is small even for large n.

Fig. 2 shows the upper bound on the size of overlap with the round-robin scheduling policy and generation size $h = 12$ and illustrates its linear increase with $\log n$. The overlap size also increases with erasure probability ϵ, because there is more asymmetry in the packets collected by generations of the file. For the parameters used in Fig. 5, we can evaluate the upper bound as $l \leq 5$. The plot of expected download time in Fig. 5 suggests that the optimal size of overlap is $l = 4$ or $l = 5$, showing that the bound is fairly tight.

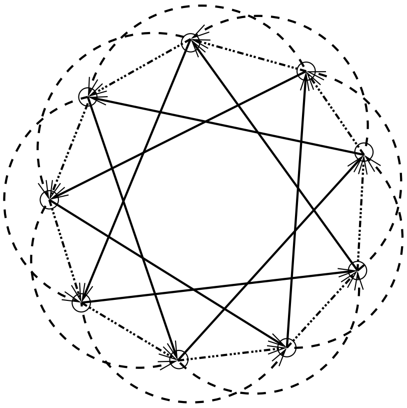

Fig. 4. Overlap graph for the circular structure for $n = 9$ generations and overlap size $l = 3$. Nodes and incoming edges represent generations and overlap packets respectively.

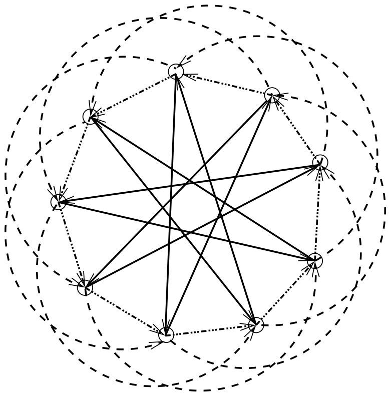

Fig. 5. Overlap graph for the distributed structure for $n = 9$ generations and overlap size $l = 3$. Nodes and incoming edges represent generations and overlap packets respectively.

C. Structure of Overlap

For a given overlap size l, we now analyze how the structure of overlap, that is the choice of overlap sets R_i for each base generation B_i, affects the expected download time of the file.

In the random annex code proposed in [7], the l packets in overlap set R_j of generation G_j are picked uniformly at random from the $(n-1)h$ packets in B_i for all $i \neq j$. However, if more than one overlap packets are chosen from the same B_i, generation G_j loses diversity of the help it receives.

The expander graph based structure proposed in [8] eliminates this loss of diversity. It chooses the l overlap packets in R_j one each from l uniformly chosen base generations B_i, $i \neq j$. This structure gives lower expected download time than the random annex code in [7].

With a random overlap structure, the source has to generate an index of the addresses of overlap packets for each file it transmits, and communicate it to the receiver. We now propose two deterministic structures of overlap which reduce the indexing complexity.

1) Circular overlap: For each generation G_i, the l overlap packets in R_i are chosen one each from the l base generations B_{i+k} for all $1 \leq k \leq l$, where we assume a circular arrangement of the generations such that the index for $i + k > n$

corresponds to generation $B_{(i+k) \mod n}$. Further, the overlap packets are chosen such that R_i are disjoint for all $1 \leq i \leq n$.

We define a graphical structure called the overlap graph, as shown in Fig. 3 to represent the structure of overlap.

Definition 4 (Overlap graph)**.** *The overlap graph $\text{Ov}_n(x_1, x_2, \cdots x_l)$ is a graph with n nodes, with an edge from node j and node $(j + x_k)$ for all $1 \leq j \leq n$ and $1 \leq k \leq l$. An incoming edge from node j to node i represents a unique packet in B_j which is included in the overlap set R_i.*

The overlap graph is a directed version of a class of graphs called circulant graphs extensively used in graph theory [12]. The circular overlap structure corresponds to the overlap graph $\text{Ov}_n(1,\ 2,\ \cdots l)$ as shown in Fig 3 for a file with $n = 9$ generations and overlap size $l = 3$ packets.

2) Distributed Overlap: For the circular overlap structure in Fig. 3, a decoded generation G_i helps G_{i+1}, G_{i+2} and G_{i+3}. One or more of these generations is most likely to decode next. If G_{i+1} decodes, its help to G_{i+2} and G_{i+3} could be superfluous since they were already helped by G_i. Instead, G_{i+1}'s overlap packets should be chosen from other generations that are not helped by G_i. This motivates the distributed overlap structure in which a decoded generation, helps the generation with least probability of being decoded.

We first define a notion of distance in the overlap graph and use it to construct the distributed overlap structure.

Definition 5 (Distance in Overlap Graph)**.** *Assuming every edge to be of unit length, the distance between nodes i and j is the length of the shortest undirected path between them, and is infinity if no such path exists.*

Using the notion of distance in Definition 5, we build the distributed overlap graph $\text{Ov}_n(x_1^{(d)},\ x_2^{(d)},\ \cdots x_l^{(d)})$ in a recursive manner. Starting with an empty graph of n nodes, for every $l \geq 1$, we choose the smallest index $x_l^{(d)} \geq 1$ that maximizes the distance between nodes j and $j + x_l^{(d)}$ in the overlap graph $\text{Ov}_n(x_1^{(d)},\ x_2^{(d)},\ \cdots x_{l-1}^{(d)})$ for any j.

For example if $n = 9$ and $l = 3$ we get the distributed overlap graph $\text{Ov}_9(1, 4, 2)$ as shown in Fig. 4. Note that the above recursive procedure gives a good distributed overlap graph only for odd number of generations n. If n is even, we get a cycle between nodes 1 and $n/2$. However, we can easily avoid n even by simply adding overlap packets only $n - 1$ generations and coding the n^{th} generation only over its h base packets.

3) Comparative Results: We now compare the proposed overlap structures, with the random annex [7] and expander overlap structures [8]. Fig. 5 is a simulation plot of the normalized expected download time versus the overlap size l, for round-robin scheduling of a file with $M = 1500$ packets divided into $n = 125$ generations of $h = 12$ packets each. The erasure probability $\epsilon = 0.2$, and the download time is averaged over 500 iterations. We observe that the circular and distributed overlap structures give progressively lower expected download times than random overlap.

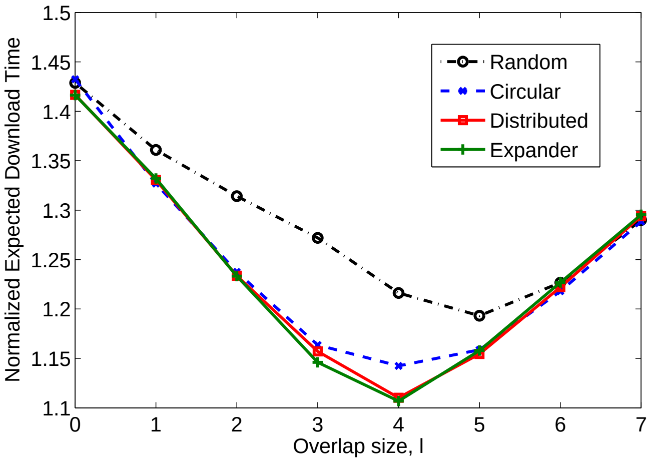

Fig. 6. Expected download time normalized by $M/(1-\epsilon)$ versus the overlap size l with $M = 1500$, $n = 125$, $h = 12$, $g = h + l$ and $\epsilon = 0.2$.

To simulate the expander overlap structure, we generate an n-node degree-$2l$ random regular graph at every iteration. In a peer-to-peer network, where the generations are stored at different nodes, changing the overlap graph involves the exchanging of packets between the nodes. Also, the source has to send a look-up table of the indices of the overlap packets to the receiver. Fig. 5 shows that the distributed overlap structure gives almost the same expected download time as expander overlaps. But being a deterministic structure, it avoids the network overhead and indexing complexity of randomly generated structures.

IV. Concluding Remarks

In this paper, we develop codes to minimize the download time of a large file, divided into n equal-size chunks called generations, and transmitted over a packet erasure channel without feedback. We compare round-robin scheduling over the n generations with random scheduling used in most previous work. We show that the expected download time is proportional to $n \log n$ with both policies, but round-robin scheduling has a smaller proportionality constant. In particular, for small generation sizes used in practice, round-robin scheduling gives significantly faster file download.

We also present a novel analysis of the size and structure of overlap between generations. We determine a fundamental upper bound on the required overlap size in terms of n and the erasure probability ϵ. We show that it scales $O(\log n)$, which implies that download time can be reduced with only a small increase in computational complexity. We also propose deterministic overlap structures which reduce the download time, but with lower network overhead and indexing complexity than randomly generated structures proposed in previous work.

Future work includes considering intermediate feedback to the source in the design of overlap. Another interesting research direction is to use the idea of generations to reduce the complexity of streaming codes [13] that are designed to achieve fast sequential decoding and playback of packets.

References

[1] Sandvine Networks, "Global internet phenomena report," Mar. 2012.

[2] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Practical loss-resilient codes," in *ACM Symposium on Theory of Computing*, (New York, NY, USA), pp. 150–159, ACM, 1997.

[3] D. Silva, W. Zeng, and F. Kschischang, "Sparse network coding with overlapping classes," *Workshop on Network Coding, Theory, and Applications*, June 2009.

[4] S. Yang and R. Yeung, "Batched Sparse Codes," *submitted to IEEE Transactions on Information Theory*, June 2012.

[5] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," *Allerton Conference on Communication, Control and Computing*, Sept. 2003.

[6] P. Maymounkov, N. Harvey and D. Lun, "Methods for Efficient Network Coding," in *Allerton Conference on Communication, Control and Computing*, 2006.

[7] Y. Li, E. Soljanin and P. Spasojevic, "Effects of the Generation Size and Overlap on Throughput and Complexity in Randomized Linear Network Coding," *IEEE Transactions on Information Theory*, Feb. 2011.

[8] B. Tang, S. Yang, Y. Yin, B. Ye and S. Lu, "Expander Graph Based Overlapped Chunked Codes," *International Symposium on Information Theory*, July 2012.

[9] Y. Li, P. Vingelmann, M. Pedersen and E. Soljanin, "Round-robin Streaming with Generations," *International Symposium on Network Coding*, June 2012.

[10] P. Grabner and H. Prodinger, "Maximum Statistics of N Random Variables Distributed by the Negative Binomial Distribution," *Combinatorics, Probability and Computing*, vol. 6, pp. 179–183, June 1997.

[11] G. Joshi and E. Soljanin, "Round-Robin Overlapping Generations Coding for Fast Content Download," *Extended version, Available online*, Jan. 2012.

[12] J. Gross and J. Yellen, *Graph Theory and its Applications*. Chapman and Hall, 2nd ed., 2005.

[13] G. Joshi, Y. Kochman, and G. Wornell, "On Playback Delay in Streaming Communication," *International Symposium on Information Theory*, July 2012.