Received August 29, 2020, accepted October 14, 2020, date of publication October 27, 2020, date of current version November 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3034191

Super Generation Network Coding for Peer-to-Peer Content Distribution Networks

ANAS AHMAD ABUDAQA, (Member, IEEE), ASHRAF MAHMOUD, (Member, IEEE), MARWAN ABU-AMARA, (Associate Member, IEEE), AND TAREK R. SHELTAMI
Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

Corresponding author: Anas Ahmad Abudaqa (g201202060@kfupm.edu.sa)

This work was supported in part by the King Fahd University of Petroleum and Minerals (KFUPM), and in part by the Department of Computer Engineering.

ABSTRACT In peer-to-peer (P2P) content distribution systems, network coding is known as a helpful method for increasing the content availability, accelerating the download process, and robustness against churn. Originally, dense network coding (DNC) has been proposed and theoretically considered as an optimal solution. However, due to its huge computational overhead, it is not viable for real-world systems. Subsequently, sparse, generations, and overlapped generations network coding schemes are proposed as possible alternatives but at reduced performance compared to that provided by the DNC. Further in this article, an improved applicable network coding scheme for P2P content distribution systems referred to therein as Super Generation Network Coding (SGNC) is proposed. SGNC maximizes the generation size so that it is as close as possible to the optimal size without adding computational overhead. Theoretical analysis and experimental work show that SGNC outperforms classical and all previous coding based schemes for P2P content distribution systems in terms of content availability, download time, overhead, and decodability for all piece scheduling policies.

INDEX TERMS Content distribution networks, peer-to-peer computing, network coding, file sharing, bittorrent, piece scheduling policies, rarest piece syndrome.

I. INTRODUCTION

P2P content distribution systems, also known as file sharing, have become a popular and effective alternative to the classical server-client file sharing approach. The basic idea in P2P content distribution protocols is quite simple; a single server, also known as the seeder, distributes a large file, typically in the order of gigabytes, to a large number of interested peers over an intranet or the Internet. Rather than uploading the file to every single peer, the seeder first fragments the file into data packets called blocks or pieces, and then distributes these pieces in a smart manner such that participating peers may exchange them with one another. The apparent advantage of P2P content distribution is to dramatically minimize the download time for each peer and alleviate the pressure on the single server. Since the participating peers utilize their own upload bandwidth to serve each other, the overall bandwidth in the network is significantly improved, yielding a much faster file downloading process.

The associate editor coordinating the review of this manuscript and approving it for publication was Cesar Vargas-Rosales.

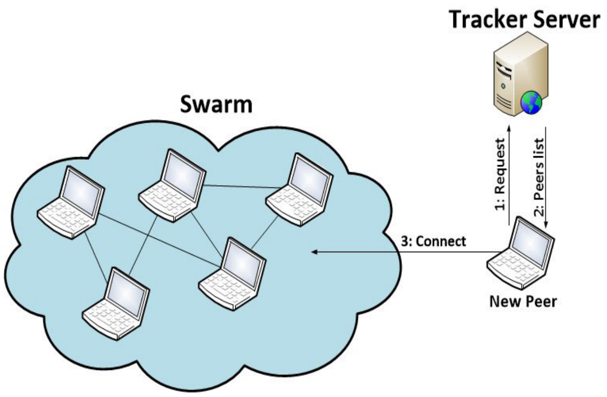


FIGURE 1. Process of new peer joining BitTorrent system.

BitTorrent [1] is known as one of the most famous P2P content distribution systems. Details of BitTorrent components are found in [2]. In BitTorrent, once a peer joins the system, it contacts a central server called the tracker to obtain a list of some online peers that are currently downloading the file as seen in Fig. 1. The tracker usually gives the specifics of 50 peers, randomly chosen from among all online peers. The peer then tries to establish connections to a minimum of 20 peers and a maximum of 40 peers. These connected peers form a swarm and the peers inside the swarm are called neighbors. All swarms form a dynamic logical network, known as the P2P overlay network, as shown in Fig. 2.

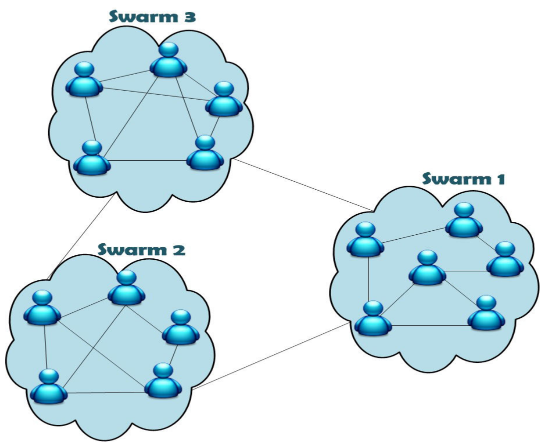


FIGURE 2. **P2P overlay network.**

In spite of the promising advantages of BitTorrent, two problems that significantly degrade the performance of BitTorrent have been recognized. First, the piece scheduling policy problem that specifies how a peer shares pieces of a file. The original BitTorrent protocol [1] suggests that clients can download the pieces in a pure random way. However, this can lead to a situation in which some pieces owned by a peer are not anymore significant to the other peers, or pieces that are needed by many peers are either very rare or not available within the network. Subsequently, rarest-first (RF) scheduling techniques are proposed to improve the performance [3]–[5]. RF techniques partially solve the scheduling problem as they are usually applied locally within a swarm and do not take into account the pieces within other swarms. The second important problem is that the nature of P2P content distribution networks is dynamic, and hence peers can depart suddenly which is known as dynamic peers participation or churn [6]. This in turn may affect the download time or or even prevent the completion of the download.

Network coding [7] is theoretically well-known in its benefits for large networks where no topology information are available and links among the nodes are unreliable. Practically, random linear network coding (RLNC) was proposed in [8] for robust, distributed transmission, and compression of information in networks. Subsequently, the idea found a place in Avalanche, a P2P content distribution system from Microsoft [9]. In Avalanche, a file of n plain pieces $p_1, p_2, \ldots, p_n$ is represented as set of n coded pieces $P_{c1}, P_{c2}, \ldots, P_{cn}$ such that each coded piece P_c is linear combination of the file's plain pieces multiplied by a vector of coefficients $e_{c,1}, e_{c,2}, \ldots e_{c,n}$. Usually the vector of coefficients, whose elements belong to a Galois Field $GF(q)$ with q elements, is either selected randomly or is deterministic [10]. Equation (1) represents mathematically an encoded piece P_c, while (2) shows the overall encoded pieces at the sender side.

$$P_c = \sum_{i=1}^{n} e_{c,i} * p_i \tag{1}$$

$$\begin{bmatrix} P_{c1} \\ P_{c2} \\ \vdots \\ P_{cn} \end{bmatrix} = \begin{bmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,n} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ e_{n,1} & e_{n,2} & \cdots & e_{n,n} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} \tag{2}$$

The introduction of network coding in P2P content distribution systems brings about the issue of computational complexity. The use of linear algebra in encoding and decoding incurs an additional computation overhead that is not present in the original BitTorrent. Consider sharing a file consisting of n pieces, the file decoding within BitTorrent is very trivial and simple. Each piece is checked and verified separately, then it is directly stored on the hard disk. Conversely, the decoding process in network coding cannot be accomplished until all the encoded pieces are received, and requires solving a system similar to that shown in (3). Typically, the Gaussian elimination method is employed which requires $O(n^3)$ operations. Therefore, as the size of the file increases, the cost of solving these equations becomes increasingly prohibitive.

$$\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,n} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ e_{n,1} & e_{n,2} & \cdots & e_{n,n} \end{bmatrix}^{-1} \begin{bmatrix} P_{c1} \\ P_{c2} \\ \vdots \\ P_{cn} \end{bmatrix} \tag{3}$$

In an effort to minimize the computational complexity, information pieces are partitioned into mutually exclusive subsets referred to as groups or generations, and coding is done only within a generation. This approach scales down the encoding and decoding problem from the whole file size to the generation size. The concept of generations in network coding was first proposed by Chou *et al.* in [11].

A. RELATED WORK

C. Gkantsidis *et al.* [9] propose Avalanche which is the first P2P content distribution system that utilizes the concept of network coding for P2P content distribution networks. The authors show that by using network coding, the network is very robust to churns such that peers can finish downloading even if the original seeder leaves after uploading exactly one copy of the file to the network. Simulation results show that network coding outperforms both the encoding at the source approach [42] and no coding at all, by factors of two and three, respectively. However, these results are based on simulation only and do not account for the computational complexity of decoding nor that of encoding.

Chou *et al.* [11] propose the concept of generation network coding which splits a file into generations such that each generation contains a mutually exclusive subset of the file's pieces. Network coding is applied on each generation separately. The authors suggest that the scheduling among the generations can be done sequentially, i.e. generation by generation. Control messages among nodes are exchanged to request a generation and then to inform that the generation is fully received.

Maymounkov *et al.* [12] suggest that in order to avoid the overhead of control messages, a generation can be selected randomly. The authors argue that performance improved without the need for feedback. Nonetheless, redundant pieces could still be exchanged, and thus yielding even more overhead.

Xu *et al.* propose I-Swifter [13], in an attempt to improve generation-based network coding by reducing the control messages overhead and eliminating the distribution of encoding vectors. To achieve a high scheduling efficiency, the work utilizes the local rarest-first scheduling policy at the generations level. To minimize requesting messages, the receiver peer will send a request for the rarest generation and once the request is received by the sending side, the sender peer continuously shares pieces with the receiver in a push-based manner. This is continued until a control message from the receiver arrives at the sender informing it that the local rarest generation has changed.

BRONCO was proposed by Hundeboll *et al.* [14]. BRONCO considers an initial server that distributes a file's pieces to multiple peers, then the peers exchange their pieces. The authors state that there are three important parameters that affect the performance of the system and thus should be selected carefully: (1) number of generations (w) for a given file, (2) size of the finite field (q), and (3) size of each piece in a generation. The selection of these parameters is a trade-off process between the computational complexity of the network coding, and the probability of creating linear dependent vectors. Increasing w or q for a constant file size reduces the expected number of the linear dependent vectors, and subsequently, the amount of valid vectors increases. However, the complexity of encoding and decoding pieces also increases. The authors argue that if Avalanche consumes 20% to 40% of CPU utility, BRONCO needs only 5% of CPU utility to share the same file, but with a redundant packets overhead of 9%. For evaluation purposes, BRONCO is compared to HTTP, a standard server-client approach, and BitTorrent. As expected, BRONCO far outperforms HTTP, but performs almost as well as BitTorrent.

Niu and Li [15], [16] determine when it is advantageous to use network coding while incurring acceptable computational coding complexity. Mathematical tools such as Markov chain processes and differential equations, and simulations are both used to evaluate a large scale dynamic P2P system. The authors find out that a generation with around 20 to 30 pieces is sufficient to benefit from network coding features with reasonable coding complexity.

Leu *et al.* [17] perform extensive simulations including both with and without network coding schemes and deduce that applying generation network coding for a P2P content distribution network adds only a relatively small overhead and could perform much better than trivial routing schemes as long as the following conditions are met. First, deterministic linear network coding (DRLNC) [18] is used to avoid dependency between encoding vectors for different pieces. Second, the number of coded pieces per generation should be selected carefully such that the coding speed is not slower than the transmission speed. Third, Gauss-Jordan elimination is applied for instant decoding.

Braun *et al.* [19] propose Network Coding Messaging Extension (NCME) as an extended network coding feature for BitTorrent and backward compatibility without affecting the baseline BitTorrent. The peer must decide which optimal communication paradigm to use choosing either baseline BiTorrent or NCME. After receiving the coded pieces and a generation becomes fully downloaded and decoded, BitTorrent pieces can be restored and shared with a non-NCME compatible BitTorrent. For evaluation purposes, a comparison between NCME and standard BitTorrent is conducted. Results show that NCME distributes the file among the peers 20% faster. Moreover, to maintain NCME's superior performance over the baseline BitTorrent, it is suggested to use a generation size of 43 pieces. However, the work does not specify the optimal piece size, nor does it provide an experimental support for the piece size selection.

Su *et al.* [20] propose PCLNC, a Push-based Combined coding strategy with an adaptive encoding window size and Low-cost computational Network Coding operations. The sender composes a coded piece based on the requested piece such that it contains the requested piece with probability 1 and other pieces with probability 0.5, and then the coded piece is pushed to the receiver. To accelerate the decoding process, an upper triangle matrix is introduced. To evaluate the performance of PCLNC, it is compared with native BitTorrent and sparse network coding scheme proposed by [21]. The study evaluated the average download time and the average time it takes a peer to start sharing with other peers, referred to therein as the start-up time. Results show that PCLNC download time is 3.17% and 21.0% faster when compared to [21] and BitTorrent, respectively. In addition, a PCLNC peer can start sharing a piece faster than BitTorrent by 36.8%, and almost as fast as [21].

Further, overlapped generations network coding studies such as [22]–[27] are proposed to boost the performance of the original generation network coding. In this scheme, the generation's pieces are not mutually exclusive, but generations are allowed to overlap. By this scheme, generations that are decoded quickly can assist in decoding other generations through back substitutions when some pieces of a generation are rare or missing. This in turn improves the network throughput. These studies are summarized in [28].

Fulcrum network coding (FNC) proposed by Lucani *et al.* [29], Nguyen *et al.* [30] [31] is another variation of generation network coding that aims to alleviate the decoding complexity for heterogeneous small devices with limited power. This is accomplished by reducing the coding coefficients overhead, and allowing recoding at the intermediate nodes and decoding at the end nodes using $GF(2)$. For powerful devices, FNC decodes over $GF(2^l)$ for $l > 1$, and behaves exactly as RLNC but with a significantly lower coefficients overhead. Since our work is dedicated to sharing large file sizes over reliable links utilizing powerful end-devices, such as PCs, and assuming generation sizes range from 1280 to 2560 packets, we compare our work to FNC with $GF(2^8)$ as a minimum field size.

Although generation network coding and its variants can reduce the computational complexity to a large extent, robustness degrades as the number of generations increases. Controlling the number of the generations is typically not under user control for fixed generation sizes and as files sizes increase. Moreover, all the former studies do not show the best physical generation size, in mega bytes (MB) for practical operations.

B. MAIN CONTRIBUTIONS AND ORGANIZATION

In this article we address the aforementioned problems and provide the following contributions:

- The problem of P2P content distribution is modeled as the deck of cards probability problem [32]. Based on this modeling, both baseline and coding-based P2P content distribution systems are analysed.
- The best generation size is measured based on a real implementation.
- We propose a novel network coding scheme, referred to herein as super generation network coding (SGNC), that maintains the computational cost at reasonable level and boosts the robustness of the network by improving pieces diversity and availability.
- The paper compares SGNC with normal generation network coding (NGNC), overlapped generation network coding (OGNC), Fulcrum network coding (FNC) and the baseline BitTorrent.
- Experiments are run based on three different scheduling policies namely random, local rarest-first, and global rarest-first. The evaluations consider four performance metrics which are robustness to churn, overhead rate, download time, and decodability rate.

The rest of the paper is organized as follow. Section II presents important preliminaries and the theoretical analysis. The novel SGNC is detailed in Section III, while the experimental work and results are presented in section IV. Section V provides the conclusions and outlines the future work.

II. PRELIMINARIES AND THEORETICAL ANALYSIS

Consider a P2P content distribution communication network where peers are forming swarms and each peer is connected to a subgroup of the other peers in order to download a designated file which is chunked into pieces of equal sizes. In what follows we model this problem as the well-known the deck of cards probability problem [32] such that pieces are modeled as the cards and each round of the sharing process is modeled as selecting a subgroup of cards from the overall cards.

A. BASELINE BitTorrent ANALYSIS

A file F is chunked into n pieces; these pieces are initially owned by an initial seeder or server who shares them among other peers. The peers then in turn start sharing after they get some or all the pieces to help the initial seeder. The sharing process is accomplished in a rounds manner such that for each round r, k pieces out of n are selected to be shared, where k is much less than n. The number k is decided based on the limits of the communication links. Thus, after n/k rounds a flow that is equal in size to the flow of the file is shared. This assumes that the seeder shares the pieces randomly and with replacement. The sharing with replacement model avails itself to model network churn and failure of links.

We are interested in finding how likely that the content is available after the initial seeder randomly and with replacement shares exactly a flow equal to the flow of the File F. Let X be a random variable representing the content availability.

Theorem 1: For the baseline BitTorrent, the probability that the content is completely available after the seeder randomly shares exactly a flow equal to the content flow in a churned P2P network is obtained by:

$$P(X = 100) = \frac{((n-k)!)^{\frac{n}{k}}}{n!^{\frac{n}{k}-1}}. \tag{4}$$

Proof: The number of ways that the seeder shares these pieces randomly and with replacement may be computed as follows. For the first round r_1, the seeder shares k out of n pieces in $\binom{n}{k}$ ways. For the second round r_2, the seeder shares k out of n pieces in $\binom{n}{k}$ ways, and so on until last round r_t, where $t = \frac{n}{k}$. Mathematically, this is written as

$$\binom{n}{k}_{r_1} * \binom{n}{k}_{r_2} * \ldots * \binom{n}{k}_{r_t} = \left(\frac{n!}{k!(n-k)!}\right)^{\frac{n}{k}} \tag{5}$$

Equation(5) represents the universal set, i.e. all the possibilities.

Out of the universal set, the number of ways that the exact content is available can be modeled as the number of ways that the seeder shares the pieces without replacement. These ways may be computed as follows. For the first round r_1, the seeder shares k out of n pieces in $\binom{n}{k}$ ways. For the second round r_2, the seeder shares k out of $n-k$ pieces in $\binom{n-k}{k}$ ways. Similarly for the third round r_3, the seeder shares k out $n-2k$ pieces in $\binom{n-2k}{k}$ ways, and so on until the last round when the seeder shares k out of k pieces in $\binom{k}{k} = 1$ ways. Mathematically, this is written as (6)

$$\begin{aligned}\binom{n}{k}_{r_1}&\binom{n-k}{k}_{r_2}\binom{n-2k}{k}_{r_3}\cdots\binom{k}{k}_{r_t} \\ &= \frac{n!}{k!(n-k)!} * \frac{(n-k)!}{k!(n-2k)!} * \frac{(n-2k)!}{k!(n-3k)!} * \ldots * 1 \\ &= \frac{n!}{k!k!\ldots k!} = \frac{n!}{k!^{n/k}}\end{aligned} \tag{6}$$

To compute the required probability, we need to divide (5) by (6) which yields

$$\frac{n!}{k!^{n/k}} \div \left(\frac{n!}{k!(n-k)!}\right)^{n/k} = \frac{n!}{k!^{n/k}} * \frac{k!^{n/k}(n-k)!^{n/k}}{n!^{n/k}} = \frac{(n-k)!^{n/k}}{n!^{\frac{n}{k}-1}}$$

□

B. ANALYSIS FOR NETWORK CODING BASED BitTorrent

In this scheme, a file F is chunked into n pieces. Furthermore, these pieces are either grouped and coded into one generation, as in dense network coding, or are grouped and coded into mutually exclusive groups or generations, as in generation-based network coding. Each generation contains s pieces out of n and the total number of generations is given by $w = \frac{n}{s}$. Similar to the case of baseline BitTorrent, the seeder shares the pieces in a rounds manner, but the seeder selects k coded pieces out of w rather than n because pieces within the same generation are indistinguishable. Since each encoded piece contains information of s pieces, the probability of the content being available is increased.

Lemma 1: Dense network coding has the following properties:

a) *Theoretically, the probability of the complete content being available after the seeder randomly shares exactly a flow equal to the content flow in churned P2P network approaches 1.*
b) *Practically, dense network coding is infeasible due to its prohibitive decoding computational requirements.*

Proof: a) Since dense network coding means that all pieces are grouped in one generation and thus all have the same information, this can be modeled as a selection from one group of indistinguishable objects. Whether we select k pieces from such group with replacement or without replacement it is always the same for all the rounds r_t. Mathematically, this can be written as the follows:

$$\frac{\binom{n}{k}\binom{n-k}{k}\binom{n-2k}{k}\cdots\binom{k}{k}}{\binom{n}{k}\binom{n-k}{k}\binom{n-2k}{k}\cdots\binom{k}{k}} = 1$$

TABLE 1. File decoding time based on dense network coding.

File Size(MB)	5	10	20	40	80	160
Decoding Time (mins)	0.13	0.482	1.8327	7.48	31.63	146.13

b) This can be proved experimentally based on the fact that the decoding requires solving n linear equations using the Gaussian elimination method where number of required operations asymptotically grows up at cubic rate $O(n^3)$. Table.1 and Fig. 3 show the results of a dense network coding experiment for file sizes ranging from 5MB to 160MB, utilizing coefficients from $GF(256)$. The experiment each time doubles the file size and computes the decoding time.

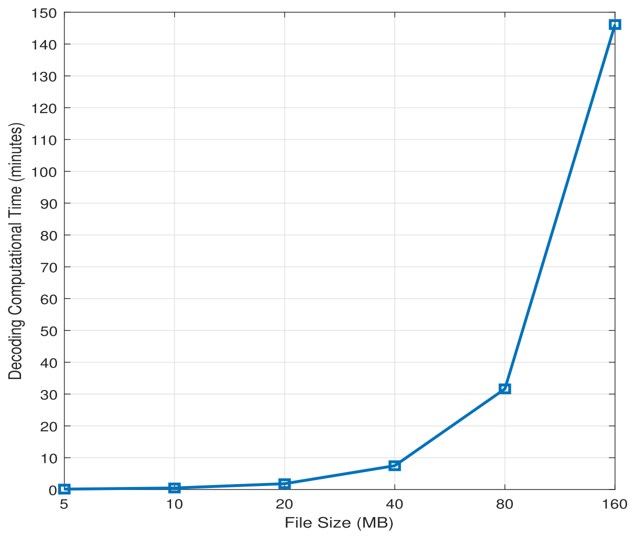


FIGURE 3. File decoding time based on dense network coding and 32KB piece size.

It can be noticed that as the file size increases, the coding time grows up rapidly. For instance, the 160MB file needs about two hours and half to be decoded although this file is considered very small for P2P content distribution networks which usually consider files in the gigabytes. This implies that dense network coding is impractical. □

Theorem 2: For generation network coding with w generations, the probability that the content is completely available after the seeder randomly shares exactly a flow equal to the content flow in a churned P2P network is obtained by:

$$P(X = 100) = \frac{(w-k)!^{w/k}}{w!^{w/k}} \tag{7}$$

Proof: Since coded pieces within the same generation are indistinguishable, then the problem is minimized to selecting k pieces from w generations. Similar to the proof of theorem 1, the universal set is given by

$$\binom{w}{k}_{r_1} * \binom{w}{k}_{r_2} * \cdots * \binom{w}{k}_{r_t} = \left(\frac{w!}{k!(w-k)!}\right)^{\frac{w}{k}} \tag{8}$$

Out of the universal set, the number of ways that the exact content is available can be written as in (9)

$$\binom{w}{k}_{r_1}\binom{w-k}{k}_{r_2}\binom{w-2k}{k}_{r_3}\cdots\binom{k}{k}_{r_t}$$
$$= \frac{w!}{k!(w-k)!} * \frac{(w-k)!}{k!(w-2k)!} * \frac{(w-2k)!}{k!(w-3k)!} * \cdots * 1$$
$$= \frac{w!}{k!k!\ldots k!} = \frac{w!}{k!^{w/k}} \tag{9}$$

Thus to get the final probability, we need to divide (8) by (9)

$$\frac{w!}{k!^{w/k}} \div \left(\frac{w!}{k!(w-k)!}\right)^{w/k} = \frac{w!}{k!^{w/k}} * \frac{k!^{w/k}(w-k)!^{w/k}}{w!^{w/k}} = \frac{(w-k)!^{w/k}}{w!^{\frac{w}{k}-1}}$$

□

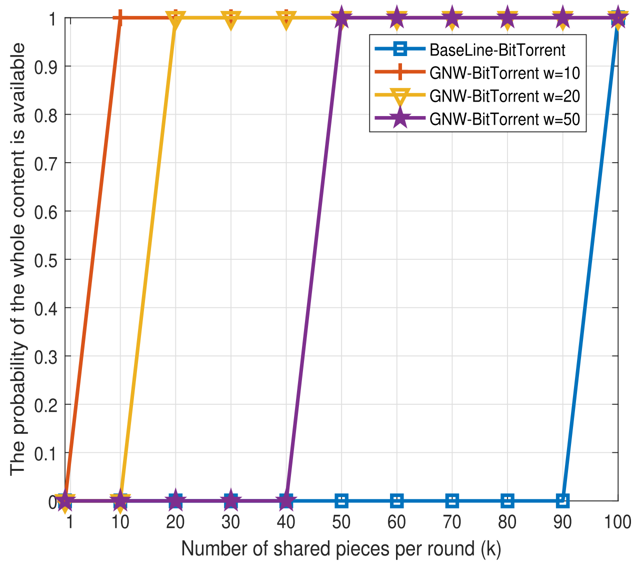


FIGURE 4. **Theoretical analysis of the content availability (n=100, w=10, 20, and 50).**

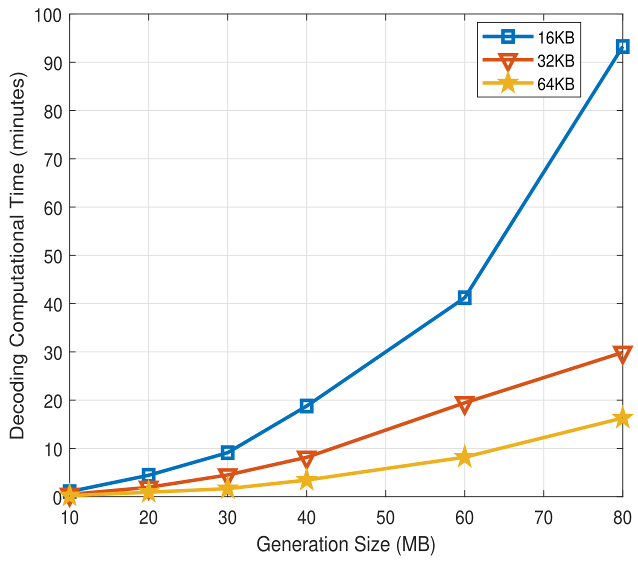


FIGURE 5. **Best generation size for different piece sizes.**

C. COMPARATIVE ANALYSIS

Fig. 4 compares (4) and (7). It is apparent that network coding increases the probability of content availability. Indeed, this probability approaches one when $w = k$. However, for large file size w and k are not equal or even close to each other. k is always constant regardless of the file size as it is limited by the data rate of the peers, whereas w increases as the file size increases because each generation size is limited by the available CPU computation and memory space. Moreover, the figure shows that as w decreases, the content becomes completely available sooner. Thus, for perfect generation-based network coding, we need to minimize w as much as possible and this can be done by increasing the generation size to the fullest extent, namely, utilizing the maximum CPU and memory resources. Let this be denoted as the best generation size which will be discussed in the next subsection.

D. BEST GENERATION SIZE

To get the most benefits of network coding, the generation size should be the maximum possible allowed by the computational resources. To determine the best generation size, an experiment is conducted using java as programming language, and based on Strassen algorithm [33], [34] as an implementation for matrix multiplication and inversion. The PC used for simulation has Core i5-2430M CPU with speed 2.40 GHz and 16GB RAM. We assume that 10 minutes is an acceptable decoding time given the maximum delay tolerated by the end user. Other thresholds are equally valid. Fig. 5 shows the decoding computational time of different generation sizes and piece sizes of 16KB, 32KB, and 64KB as these sizes are typical in P2P content distribution systems. From the figure, we notice that for 16KB pieces, the generation size should not exceed 30MB. This can be increased to 40MB for the case of 32KB pieces, and even increased further to 60MB when 64KB pieces are used.

III. SUPER GENERATION NETWORK CODING (SGNC)

Based on the earlier analysis that shows that dense network coding is impractical, and the limitation on the generation size for generation network coding, this section proposes a novel method referred to herein as the Super Generation Network Coding (SGNC).

The idea of SGNC is to extend the generation size even bigger than the best generation size of 40 MB while maintaining almost the same decoding computation time and complexity. Precisely, the generation size in SGNC is chosen to be *double* the generation size in classical generation network coding. In addition, each SGNC's generation contains three types of coded pieces which are unity piece, decodable piece, and rich piece. Assume the pieces of the file are designated as $A, B, \ldots, Z$, the following defines the three types of pieces in SGNC.

Definition 1 (Unity Piece): The piece is encoded by multiplying each plain piece of the generation by 1. The form of this piece is $(A + B + C + \ldots + Z)$. It is unique and basic for every generation.

Definition 2 (Decodable Piece): This piece is encoded by multiplying a plain piece of the generation by a coefficient e drawn from $GF(q)$, while the remaining pieces of the generation each is multiplied by 1. The decodable piece $(e.A + B + C + \ldots + Z)$ is always instantaneously decoded by XORing with the unity piece.

It is not necessary that the coefficient is placed such that consecutive decodable pieces are arranged diagonally as they can be arranged randomly. Another allowed form of consecutive decodable pieces is lower-triangular provided that a decodable piece is always decoded as early as possible. These potential forms of decodable pieces are shown in Fig. 6.

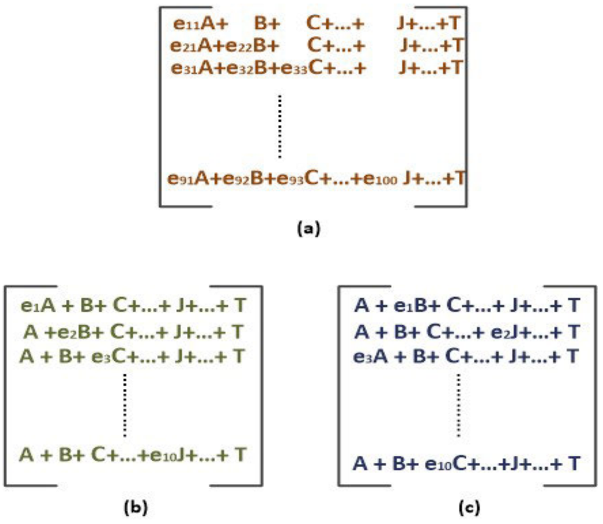


FIGURE 6. **Decodable piece's forms: (a) Lower triangular, (b) Diagonal, and (c) Random.**

Definition 3 (Rich Piece): The rich piece is full of information and is encoded by multiplying all the pieces in the generation by coefficients drawn randomly from $GF(q)$ such that the encoded piece is given by $(e_1A + e_2B + e_3C + \ldots + e_sZ)$.

Since decodable pieces are always instantly and self decodable, they may be back substituted to expedite the decoding of rich pieces.

Furthermore, a sharing method is proposed such that each time a sender is requested by a receiver, the sender should send two pieces: the decodable piece which can be decoded immediately, and the rich piece which should be buffered. Therefore, after receiving all the generation pieces, already 50% thereof are decoded and back substituted. Consequently, rather than solving a $2s$ by $2s$ matrix, we solve only a s by s matrix using Gaussian elimination. Using this scheme, we maintain the computational complexity of decoding almost as in the conventional generation network coding, and increase the diversity of the pieces such that it is as close as possible to the case of the dense coding.

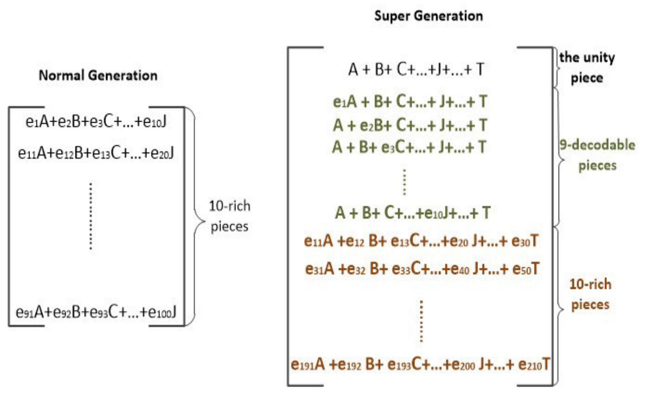


FIGURE 7. **Normal generation vs. super generation.**

Fig. 7 shows a normal generation with 10 pieces and a super generation with 20 pieces. Although the size of the super generation is doubled, the decoding time is almost the same. The figure also shows that SGNC incurs additional coefficients overhead. The SGNC overhead in terms of coefficients is discussed in the next section.

A. SGNC OVERHEAD

Recall that w is the generation size and the coefficients are drawn from $GF(2^8)$ as in most cases, then the coefficients overhead per generation for normal network coding is w^2 bytes. Since the generation size of SGNC is twice that for the normal generation, then the coefficients overhead for SGNC is supposed to be four times that of the normal generation, notably, $4w^2$. However, because the coefficients values for the majority of decodable pieces are equal to 1, then the coefficients overhead of SGNC can be minimized by using the method inspired by Li *et al.* [35].

The method simply avoids sending a byte coefficient for a coefficient whose value is equal to one. Instead it sends a vector of 0 and 1 bits, where a 0 bit indicates a coefficient value of one, while a 1 bit indicates the coefficient value is greater than one. In addition, coefficients whose values are greater than one are sent as bytes coefficients, as in the conventional case. Therefore, the coefficients overhead for SGNC caused by the rich pieces is equal to $2w^2$ bytes, whereas the overhead needed by the unity and decodable pieces is equal to ($w - 1$ bytes + w^2 bits). Hence, the overall SGNC coefficients overhead per generation in bytes can be written as:

$$2w^2 + \frac{w^2}{8} + w - 1 \tag{10}$$

It's clear from (10) that the overhead is upper bounded by $\mathcal{O}(w^2)$. Also, since the number of generations in SGNC is half the number of generations in normal network coding, the overall coefficients overhead of SGNC is only slightly larger than the normal network coding overhead as will be shown experimentally in section IV.

Another source of overhead in SGNC is sharing the decodable pieces which are prone to be highly duplicated when they are received from multi-servers especially when no coordination is enforced. Since heavy coordination between the servers eradicates the network coding key features, we address this issue by proposing a transmission method that requires minimal coordination among the peers.

The proposed transmission method works as follows. Once a receiver peer establishes connections to m seeders, the receiver sends to each seeder a unique number, i, where $0 \leq i < m$. As the number i is received at the seeder side, the seeder firstly sends the ($i \mod m$) piece, then for the subsequent transmissions, and without any coordination or feedback, the seeder sends pieces that are congruent to ($i \mod m$), namely, $i + m \equiv i + 2m \equiv i + 3m \equiv \ldots \equiv (i \mod m)$.

Example: Assuming 4 seeders are to share 16 decodable pieces to a receiver peer. Then the corresponding pieces for each seeder are

$$M_1 = \{0, 4, 8, 12\}$$
$$M_2 = \{1, 5, 9, 13\}$$
$$M_3 = \{2, 6, 10, 14\}$$
$$M_4 = \{3, 7, 11, 15\}$$

The receiver only updates and sends feedback when a seeder peer leaves, changes, or joins the P2P network.

IV. EXPERIMENTAL WORK AND RESULTS

We consider a P2P network which is divided into swarms as shown in Fig. 2 with each swarm containing a maximum of 8 nodes. Each node can connect to at most 4 other nodes selected at random from the same swarm or nearby swarms. In addition, we consider that all the peers have same data rate and therefore the download time can now be measured in terms of number of rounds. SGNC is compared with the optimal theoretical solution, normal generations network coding (NGNC), overlapped generations network coding (OGNC), Fulcrum network coding (FNC), and the baseline BitTorrent system. For some experiments, only NGNC is considered, while OGNC and/or FNC are omitted since they behave mostly as NGNC.

Files of sizes equal to 320MB, 640MB, 1.2GB, and 5.1GB are shared. Each file is divided into generations based on (11) and (12) for NGNC and SGNC, respectively.

$$\#Generations = \frac{FileSize}{BestGenerationSize} \tag{11}$$
$$\#Generations = \frac{FileSize}{BestGenerationSize * 2} \tag{12}$$

Based on our analysis in section II we consider the practical best generation size to be 40MB. Therefore, the 320MB file is divided into 8 generations for NGNC and 4 for SGNC and so on for the other file sizes. Also, we consider that coefficients are drawn randomly from $GF(256)$ and we assume that all pieces are innovative; that is there is no linear dependency among the coefficients of encoded pieces.

The evaluation considers the following performance metrics: content availability as a measure to robustness to churn, overhead rate caused by duplicated pieces, download time, decodability rate, and the network coding coefficients overhead percentage. For each of the above metrics, the following three scheduling policies are considered:

- ***Random***: assumes zero coordination among all the peers.
- ***Local rarest-first***: coordination among the peers is restricted to the peers within the same swarm.
- ***Global rarest-first***: global coordination among the peers in the entire network is assumed.

For network coding schemes, the RF-policies are applied at generations level rather than at the pieces level. The following subsections detail the results obtained for each of the considered metrics.

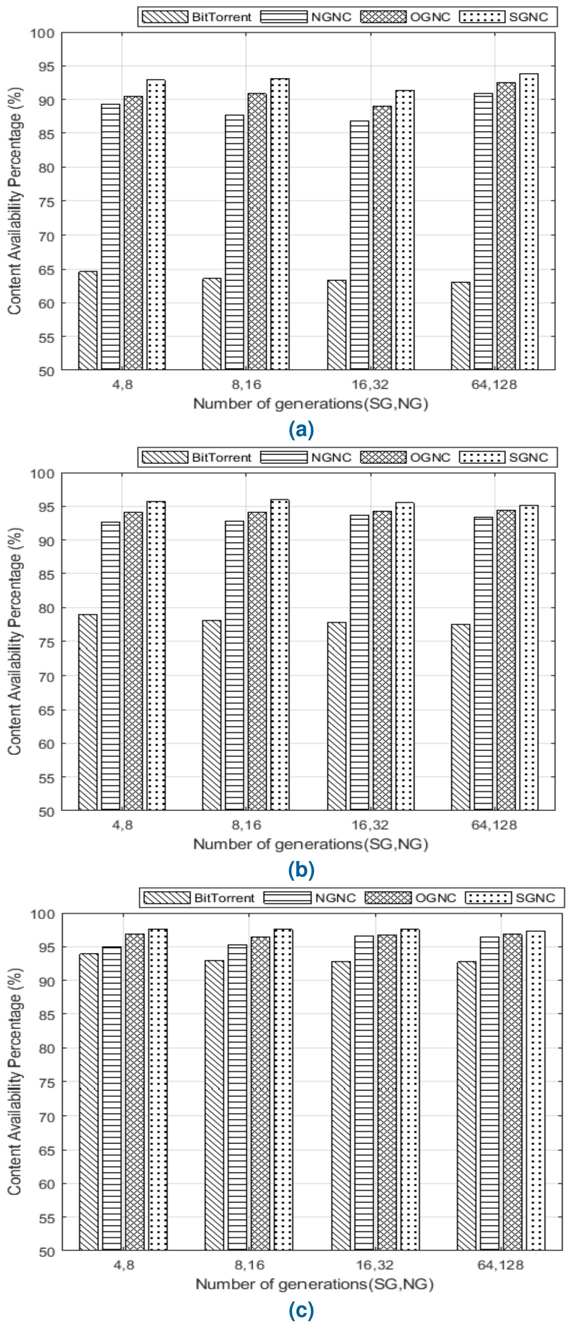
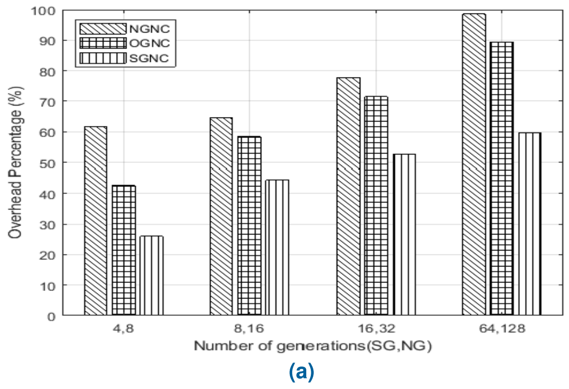
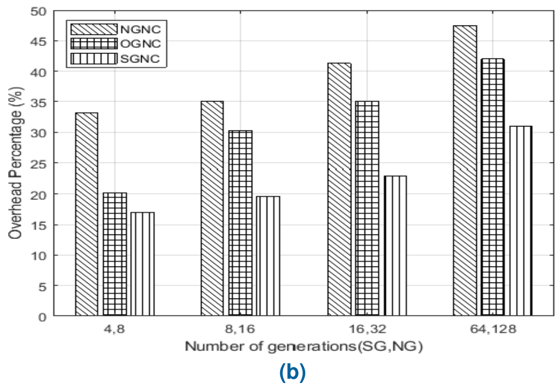


FIGURE 8. Content availability based on different scheduling policies:(a) Random, (b) Local rarest-first, and (c) Global rarest-first.

A. CONTENT AVAILABILITY (Robustness TO Churn)

This experiment assumes that the seeder or seeders leave the network after they share a number of pieces that are equal to the file pieces. The remaining peers cooperate to complete the file download from each other. Fig. 8 shows results based on random, local rarest-first, and global rarest-first scheduling policies, respectively. The results show improvement in content availability for SGNC over NGNC and OGNC by at least 2% and 1%, respectively. For baseline BitTorrent, the content availability improves significantly when some RF-policy is applied, while the improvement provided by the RF-policy is modest when network coding is employed. This is a direct result of the fact that network coding does not require global topology information and can handle high churn in the network.





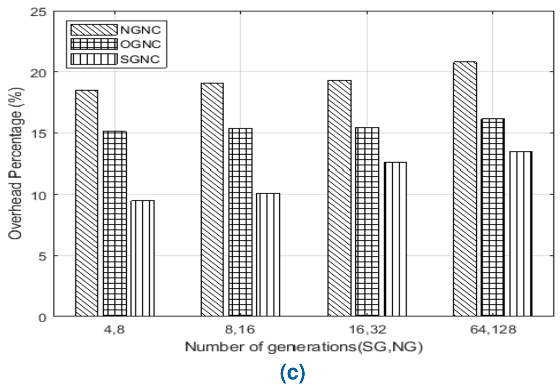


FIGURE 9. **The redundancy overhead percentage based on different scheduling policies:(a) Random, (b) Local rarest-first, and (c) Global rarest-first.**

B. OVERHEAD DUE TO DUPLICATE PIECES

In this experiment seeders continue to send pieces, based on the designated scheduling policy, to a peer until the peer completes the download. As a result, duplicated pieces may be received. For network coding schemes, the duplicate piece is considered an overhead if it is received after all the corresponding generation pieces are complete. We also assume that when the RF-policy is applied, the coordination is done only among the seeders and the receiver peer does not send any feedback. Fig. 9a, Fig. 9b and Fig. 9c show results of the overhead based on the random, local, and global rarest-first scheduling policies, respectively. The figure shows clearly that the random scheduling policy must be avoided as it incurs the largest overhead percentage relative to the other two scheduling policies. It can also be noticed that SGNC always incurs the lowest overhead for all the scheduling policies relative to the other schemes. The results, across the considered number of generations, also show that SGNC has about half or quarter of the overhead relative to NGNC for the local and global RF policies, respectively. We omit the comparative overhead results for baseline BitTorrent from Fig.9 since it is very high and exceeds 100% for all file sizes relative to NGNC, OGNC, and SGNC. Fig.10 shows that distributing a file to the baseline BitTorrent incurs extreme overhead even when using local and global scheduling policies.

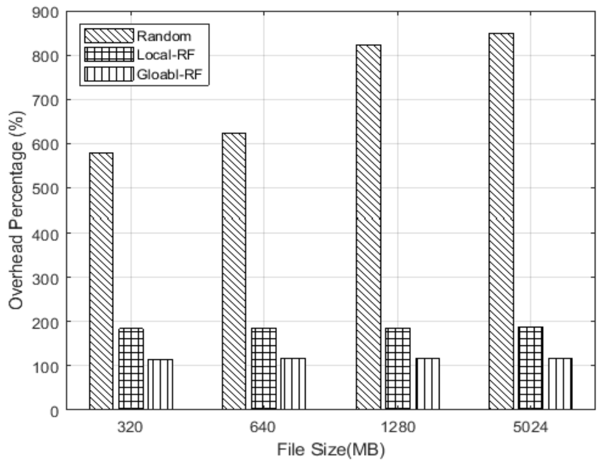


FIGURE 10. **BitTorrent overhead based on different scheduling policies.**

C. DOWNLOAD TIME

Definition 4 (Optimal Download Time): Assuming the number of pieces shared during each round k is the same, then the optimal or theoretical download time is given by

$$OptimalDownloadTime = \frac{FileSize}{k} \tag{13}$$

Accordingly, the download time experiment assumes a peer receives pieces of the file from other peers, such that each peer sends 1MB per round, until the peer completes the download. Results, depicted in Fig.11, Fig.12, Fig.13, and Fig.14, are for file sizes 320MB, 640MB, 1.2GB, and 5.1GB, respectively. The results show that for all scheduling policies and all the considered file sizes, SGNC always achieves the closest download time to the optimal case. Indeed, SGNC can download the file faster than NGNC by at least 10%. As the number of generations increases, the improvement also increases.

D. DECODABILITY

Definition 5 (Decodability): Refers to the cumulative number of already decoded pieces from the instant of receiving the first piece to a certain point of time. Higher

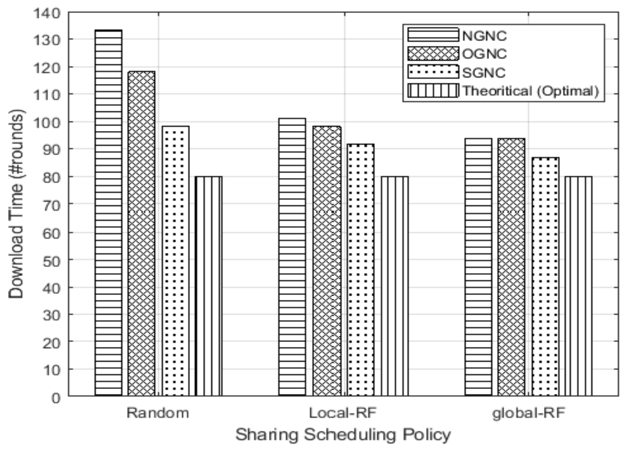


FIGURE 11. Download time of 320MB file.

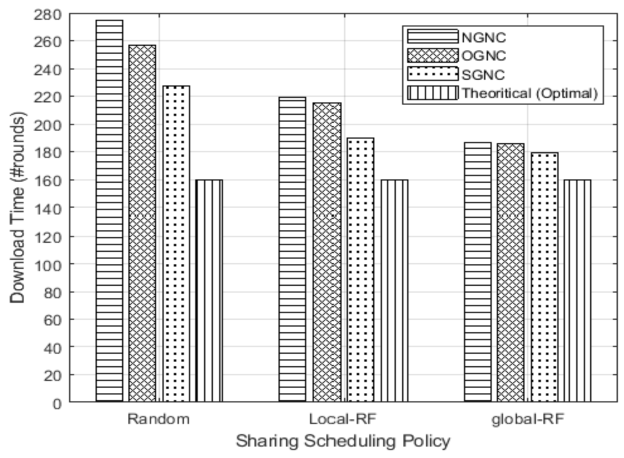


FIGURE 12. Download time of 640MB file.

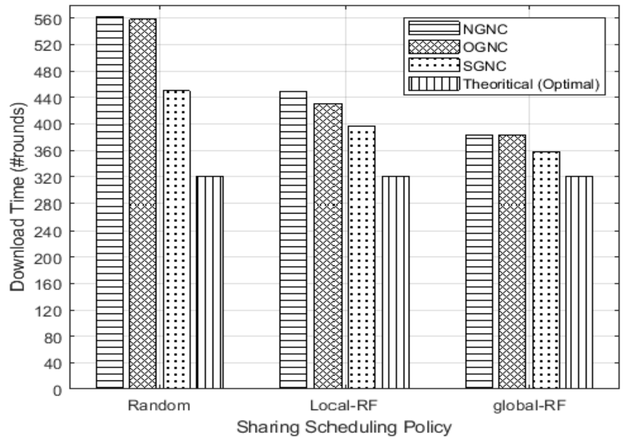


FIGURE 13. Download time of 1.2GB file.

decodability number leads to higher diversity in pieces and less CPU load.

As such, the decodability of a file with 80 pieces is measured from round 0 to round 100 in this experiment. In general, network coding suffers from low decodability especially

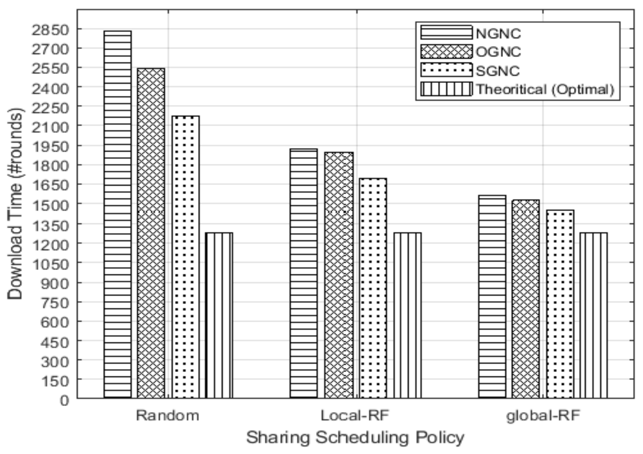


FIGURE 14. Download time of 5.1GB file.

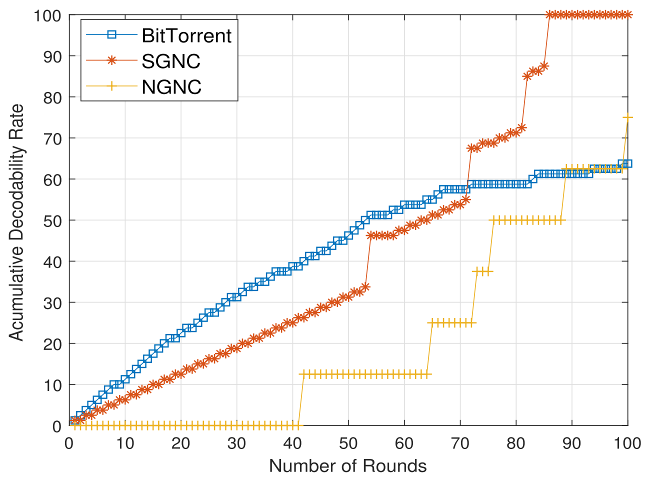


FIGURE 15. Decodability of BitTorrent and different network coding schemes based on random scheduling policy.

at the initial rounds. Fig.15 shows that normal generations network coding cannot decode any piece until round 42. On the other hand, baseline BitTorrent can decode almost one piece every round. The figure also shows that SGNC can start decoding from very early rounds at least at the rate of one piece for every two rounds. This superior performance of SGNC is the result of the self-decodable pieces. This supports pieces' diversity and availability, and balances the CPU load by distributing the required load smoothly over all rounds. Fig.16 and Fig.17 consider the same experiment but for the other two scheduling policies; the local and global rarest-first, respectively.

E. NETWORK CODING COEFFICIENTS OVERHEAD

Since the coefficients for all the experiments in this article are taken from $GF(2^8)$, then the coefficients overhead for a generation of size w is w^2 bytes for NGNC, $w^2 + w$ bits for FNC, and $2w^2 + \frac{w^2}{8} + w - 1$ for SGNC as specified by (10).

In this experiment after a peer completes the download of a file, the network coding coefficients overhead caused by the downloaded file and the redundant pieces is calculated.

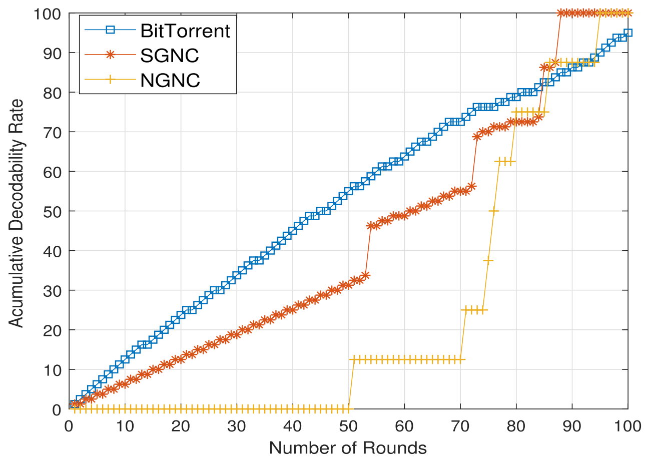


FIGURE 16. Decodability of BitTorrent and different network coding schemes based on local rarest-first scheduling policy.

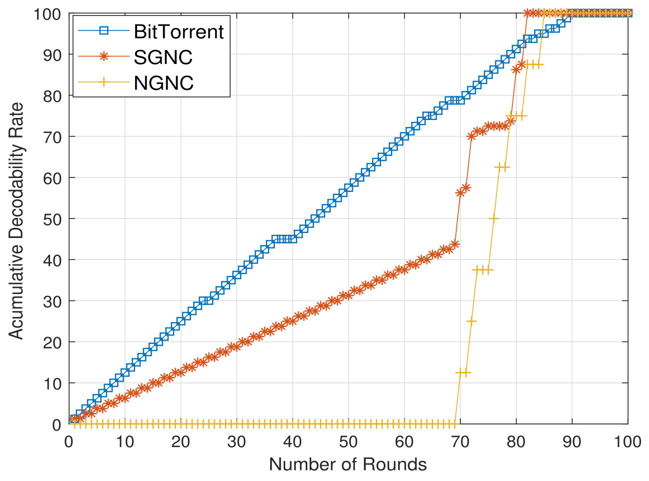


FIGURE 17. Decodability of BitTorrent and different network coding schemes based on global rarest-first scheduling policy.

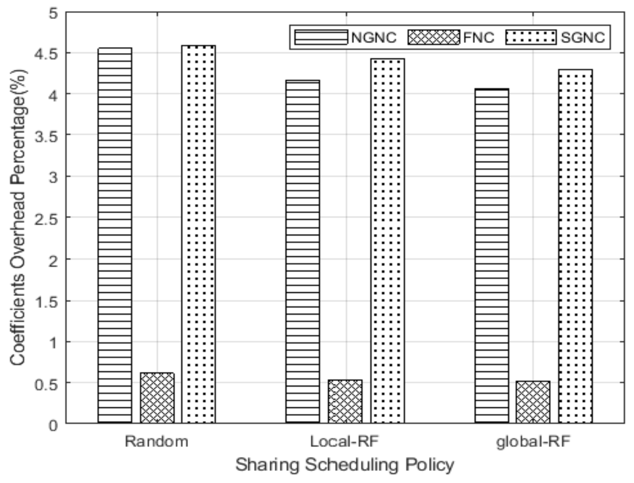


FIGURE 18. Network coding coefficients overhead percentage of 320MB file.

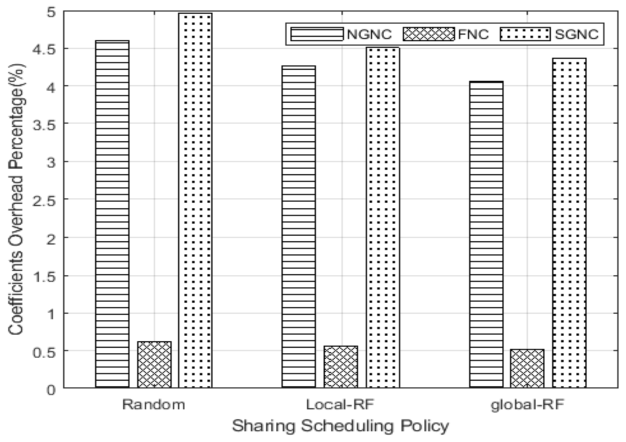


FIGURE 19. Network coding coefficients overhead percentage of 640MB file.

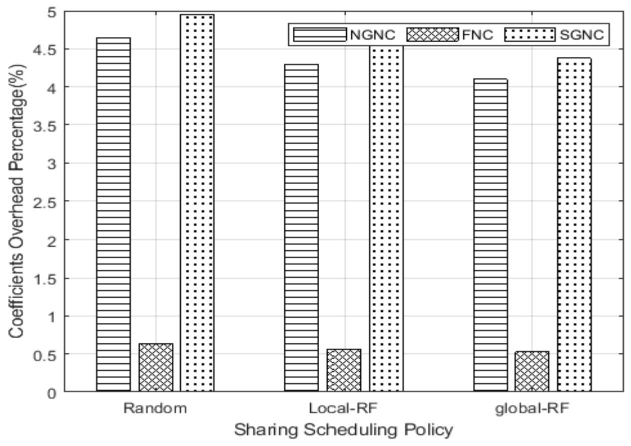


FIGURE 20. Network coding coefficients overhead percentage of 1.2GB file.

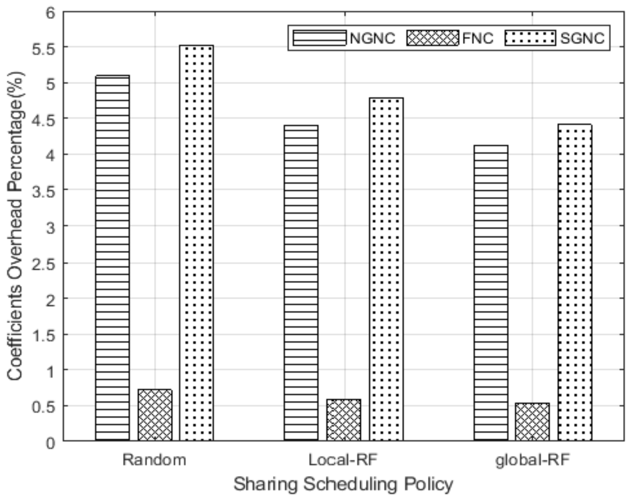


FIGURE 21. Network coding coefficients overhead percentage of 5.1GB file.

Results corresponding to the file sizes of 320MB, 640MB, 1.2GB, and 5.1GB are depicted in Fig.18, Fig.19, Fig.20, and Fig.21, respectively. The results show the following: 1) the network coding coefficients overhead does not exceed 6% for all the network coding schemes, file sizes, and sharing scheduling policies, 2) FNC incurs the minimum coefficients overhead with at most 0.5%, and 3) coefficients overhead incurred by SGNC is roughly equal to that of NGNC.

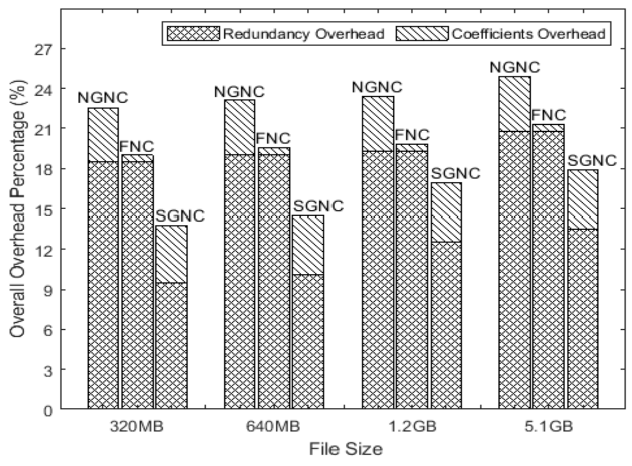


FIGURE 22. The overall overhead based on global-rarest first scheduling policy.

While SGNC has the maximum coefficients overhead, however, if we consider the overall overhead which is the cumulative overhead caused by network coding coefficients and the duplicate pieces at the receiver side, then SGNC incurs the minimum overall overhead. Fig.22 shows the results of the overall overhead for NGN, FNC, and SGNC for different file sizes based on global rarest-first scheduling policy. It is clear that SGNC has the lowest overhead among all network coding schemes.

V. CONCLUSION AND FUTURE WORK

In this article two contributions are presented. First, we model the problem of P2P content distribution as the deck of cards probability problem and we analyze it based on this model for both baseline and network coding P2P content distribution systems. Second, we propose a novel procedure referred to herein as super generation network coding (SGNC) which is a network coding scheme that boosts the robustness against churn in P2P networks, while maintaining roughly the computational cost relative to generation network coding. In addition, we evaluate the proposed method and compare it with normal generation network coding, overlapped generation network coding, and baseline BitTorrent. The experiments consider file sizes ranging from 320MB to 5GB, and utilize three piece scheduling policies namely: random, local RF, and global RF. The aforementioned schemes are evaluated in terms of availability, overhead, download time, and decodability. Simulation results show that SGNC can improve the robustness of the network, minimize the download time, incurs low overhead, and speeds up the decodability which in turns alleviates the CPU load.

Future work includes considering the effect of local rarest-first and global rarest-first policies over wide area networks with large diameters. Another potential research direction is to study and analyze the overhead of network coding coefficients for very large files, i.e. more than 10GB's files, and study the feasibility of adapting FNC to SGNC to utilize its strength in terms of minimum coefficients overhead.

REFERENCES

[1] B. Cohen, "The bittorrent protocol specification," Tech. Rep., 2008.

[2] A. Srinivasan and H. Aldharrab, "XTRA—eXtended bit-torrent pRotocol for authenticated covert peer communication," *Peer-Peer Netw. Appl.*, vol. 12, no. 1, pp. 143–157, Jan. 2019.

[3] J.-L. Chiang, Y.-Y. Tseng, and W.-T. Chen, "Interest-intended piece selection in BitTorrent-like peer-to-peer file sharing systems," *J. Parallel Distrib. Comput.*, vol. 71, no. 6, pp. 879–888, Jun. 2011.

[4] J. Luo, B. Xiao, K. Bu, and S. Zhou, "Understanding and improving piece-related algorithms in the BitTorrent protocol," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 12, pp. 2526–2537, Dec. 2013.

[5] Y. Hu, D. Dong, J. Li, and F. Wu, "Efficient and incentive-compatible resource allocation mechanism for P2P-assisted content delivery systems," *Future Gener. Comput. Syst.*, vol. 29, no. 6, pp. 1611–1620, Aug. 2013.

[6] H. Terelius and K. H. Johansson, "Peer-to-peer gradient topologies in networks with churn," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 4, pp. 2085–2095, Dec. 2018.

[7] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[8] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[9] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *Proc. IEEE 24th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 4, Mar. 2005, pp. 2235–2245.

[10] M. Yang and Y. Yang, "Applying network coding to peer-to-peer file sharing," *IEEE Trans. Comput.*, vol. 63, no. 8, pp. 1938–1950, Aug. 2014.

[11] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. Annu. Allerton Conf. Commun. Control Comput.*, 2003, vol. 41, no. 1, pp. 40–49.

[12] P. Maymounkov, N. J. Harvey, and D. S. Lun, "Methods for efficient network coding," in *Proc. 44th Annu. Allerton Conf. Commun., Control, Comput.*, 2006, pp. 482–491.

[13] J. Xu, X. Wang, J. Zhao, and A. O. Lim, "I-swifter: Improving chunked network coding for peer-to-peer content distribution," *Peer-Peer Netw. Appl.*, vol. 5, no. 1, pp. 30–39, Mar. 2012.

[14] M. Hundeboll, J. Ledet-Pedersen, G. Sluyterman, T. K. Madsen, and F. H. P. Fitzek, "Peer-assisted content distribution with random linear network coding," in *Proc. IEEE 79th Veh. Technol. Conf. (VTC Spring)*, May 2014, pp. 1–6.

[15] D. Niu and B. Li, "On the resilience-complexity tradeoff of network coding in dynamic P2P networks," in *Proc. 15th IEEE Int. Workshop Qual. Service*, Jun. 2007, pp. 38–46.

[16] D. Niu and B. Li, "Analyzing the resilience-complexity tradeoff of network coding in dynamic P2P networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 11, pp. 1842–1850, Nov. 2011.

[17] J.-S. Leu, M.-C. Yu, and H.-C. Yueh, "Improving network coding based file sharing for unstructured peer-to-peer networks," *J. Netw. Syst. Manage.*, vol. 23, no. 4, pp. 803–829, Oct. 2015.

[18] N. Wang and N. Ansari, "Downloader-initiated random linear network coding for peer-to-peer file sharing," *IEEE Syst. J.*, vol. 5, no. 1, pp. 61–69, Mar. 2011.

[19] P. J. Braun, M. Sipos, P. Ekler, and H. Charaf, "Increasing data distribution in bittorrent networks by using network coding techniques," in *Proc. Eur. Wireless; 21th Eur. Wireless Conf.*, 2015, pp. 1–6.

[20] J. Su, Q. Deng, and D. Long, "PCLNC: A low-cost intra-generation network coding strategy for P2P content distribution," *Peer-Peer Netw. Appl.*, vol. 12, no. 1, pp. 177–188, Jan. 2019.

[21] G. Ma, Y. Xu, M. Lin, and Y. Xuan, "A content distribution system based on sparse linear network coding," in *Proc. 3rd Workshop Netw. Coding (Netcod)*, 2007, pp. 1–6.

[22] D. Silva, W. Zeng, and F. R. Kschischang, "Sparse network coding with overlapping classes," in *Proc. Workshop Netw. Coding, Theory, Appl.*, Jun. 2009, pp. 74–79.

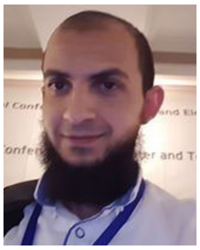
[23] A. Heidarzadeh and A. H. Banihashemi, "Overlapped chunked network coding," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Jan. 2010, pp. 1–5.

[24] Y. Li, E. Soljanin, and P. Spasojevic, "Effects of the generation size and overlap on throughput and complexity in randomized linear network coding," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 1111–1123, Feb. 2011.

[25] B. Tang, S. Yang, Y. Yin, B. Ye, and S. Lu, "Expander graph based overlapped chunked codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2012, pp. 2451–2455.

[26] Y. Li, W.-Y. Chan, and S. D. Blostein, "Network coding with unequal size overlapping generations," in *Proc. Int. Symp. Netw. Coding (NetCod)*, Jun. 2012, pp. 161–166.

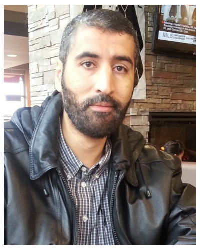
[27] G. Joshi and E. Soljanin, "Round-robin overlapping generations coding for fast content download," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2013, pp. 2740–2744.
[28] A. A. AbuDaqa, A. Mahmoud, M. Abu-Amara, and T. Sheltami, "Survey of network coding based P2P file sharing in large scale networks," *Appl. Sci.*, vol. 10, no. 7, p. 2206, Mar. 2020.
[29] D. E. Lucani, M. V. Pedersen, D. Ruano, C. W. Sorensen, F. H. P. Fitzek, J. Heide, O. Geil, V. Nguyen, and M. Reisslein, "Fulcrum: Flexible network coding for heterogeneous devices," *IEEE Access*, vol. 6, pp. 77890–77910, 2018.
[30] V. Nguyen, J. A. Cabrera, D. You, H. Salah, G. T. Nguyen, and F. H. P. Fitzek, "Advanced adaptive decoder using fulcrum network codes," *IEEE Access*, vol. 7, pp. 141648–141661, 2019.
[31] V. Nguyen, E. Tasdemir, G. T. Nguyen, D. E. Lucani, F. H. P. Fitzek, and M. Reisslein, "DSEP fulcrum: Dynamic sparsity and expansion packets for fulcrum network coding," *IEEE Access*, vol. 8, pp. 78293–78314, 2020.
[32] A. Stiglic, "Computations with a deck of cards," *Theor. Comput. Sci.*, vol. 259, nos. 1–2, pp. 671–678, May 2001.
[33] S. Huss-Lederman, E. M. Jacobson, A. Tsao, T. Turnbull, and J. R. Johnson, "Implementation of Strassen's algorithm for matrix multiplication," in *Proc. ACM/IEEE Conf. Supercomput. (CDROM)*, Jan. 1996, p. 32.
[34] G. Bilardi and L. De Stefani, "The I/O complexity of strassen's matrix multiplication with recomputation," in *Proc. Workshop Algorithms Data Struct.* Cham, Switzerland: Springer, 2017, pp. 181–192.
[35] Y. Li, S. Zhang, J. Wang, X. Ji, H. Wu, and Z. Bao, "A low-complexity coded transmission scheme over finite-buffer relay links," *IEEE Trans. Commun.*, vol. 66, no. 7, pp. 2873–2887, Jul. 2018.



ANAS AHMAD ABUDAQA (Member, IEEE) received the B.Sc. degree in computer engineering from the Islamic University of Gaza (IUG), Palestine, in 2009, and the M.S. degree in computer networks from the King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia, in 2014, where he is currently pursuing the Ph.D. degree with the Department of Computer Engineering. His research interests include computer networks optimization, coding theory, P2P file sharing, cryptography, and network security.

ASHRAF MAHMOUD (Member, IEEE) received the Ph.D. degree in electrical engineering from Carleton University, Canada, in 1997. From 1997 to 2002, he was with Nortel Networks, Canada, as a Senior Radio Systems Engineer. Since 2002, he has been with the Computer Engineering Department, KFUPM, Saudi Arabia. His research interests include performance evaluation, simulation and modeling, and software defined networking.

MARWAN ABU-AMARA (Associate Member, IEEE) received the M.S. and Ph.D. degrees in electrical and computer engineering from Texas A&M University, USA, in 1991 and 1995, respectively. From 1995 to 2003, he worked for Nortel Networks, USA, as a Senior Technical Advisor. Since 2003, he has been with the Computer Engineering Department, KFUPM, Saudi Arabia. His research interests include cloud and Internet security and resiliency, the IoT, and wireless communications.



TAREK R. SHELTAMI received the Ph.D. degree in electrical and computer engineering from the Electrical and Computer Engineering Department, Queen's University, Kingston, ON, Canada, in April 2003. He is currently a Professor with the Computer Engineering Department, King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia. He joined the department, in August 2004. He has been an Adjunct Professor with the Jodrey School of Computer Science, Acadia University, Canada, since 2010. He is currently collaborating with the Transportation Research Institute (IMOB), Hasselt University, Belgium. He has authored/coauthored more than 200 research articles. Before joining the KFUPM, he was a Research Associate Professor with the School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, ON, Canada. He worked with GamaEng Inc., as a Consultant on wireless networks, from 2002 to 2004. He worked in several joined projects with Nortel Network Corporation. He has been a member of the technical program and organizing committees of several international IEEE conferences. His research interests include ad hoc Networks, WSAN, the IoT, digitization, computer network security, quality of service in software defined networking, autonomic computing, quality of service-aware routing in software defined networking, and performance evaluation.