applied sciences

MDPI

Review

Survey of Network Coding Based P2P File Sharing in Large Scale Networks

Anas A. AbuDaqa *, Ashraf Mahmoud, Marwan Abu-Amara and Tarek Sheltami

Department of Computer Engineering, King Fahd University of Petroleum Minerals, Dhahran 31261, Saudi Arabia; ashraf@kfupm.edu.sa (A.M.); marwan@kfupm.edu.sa (M.A.-A.); tarek@kfupm.edu.sa (T.S.)

* Correspondence: g201202060@kfupm.edu.sa

Received: 18 February 2020; Accepted: 17 March 2020; Published: 25 March 2020

Abstract: Peer-to-peer (P2P) content distribution and file sharing systems aim to facilitate the dissemination of large files over unreliable networks. Network coding is a transmission technique that has captured the interest of researchers because of its ability to increase throughput and robustness of the network, and decrease the download time. In this survey paper, we extensively summarize, assess, compare, and classify the most recently used techniques to improve P2P content distribution systems performance using network coding. To the best of our knowledge, this survey is the first comprehensive survey that specifically focuses on the performance of network coding based P2P file sharing systems.

Keywords: content distribution networks; peer-to-peer computing; random linear network coding; file sharing; rarest-piece issue; information thoery

1. Introduction

The Peer-to-Peer (P2P) architecture has triggered a technical revolution of large content distribution service on the Internet. The P2P model is the antithesis of the classic client–server architecture, in which each node can behave either as a server or a client. In the client–server architecture, data stored on the server is sent to clients on a request basis. However, for a large number of clients, the workload on the server may be too heavy, leading to extremely low download rates. Moreover, for this architecture, the server represents a single point of failure. On the contrary, in the P2P architecture, each node is called a servent [1,2] which means that a node may behave as a server as well as a client at the same time. This leads to higher bandwidth utilization, more availability of content, and reduced download times [3]. The BitTorrent protocol [4] is considered to be one of the most successful P2P file sharing protocols. BitTorrent-like systems work in two phases, the first phase utilizes a discovery protocol, in which peers discover and establish connections to each other. Detailed discussions on the discovery protocol can be found in [5–10]. The second phase utilizes file dissemination protocol, in which peers start to download and upload the chunks of a file until they get the complete file.

In the file dissemination protocol, the algorithm used for the selection of the pieces to download is highly important to achieve an acceptable level of system performance. In fact, wrong selection choices may lead to a situation where pieces of a file owned by a peer are not required by any other peer. Conversely, a piece which is needed by many peers is either very rare or not available within the network. Consequently, the peer requires a specific policy for downloading the pieces of the file. The original specification of the BitTorrent protocol [4] requires that clients are able to download pieces in a purely random way. Subsequently, new selection techniques were introduced to improve performance [11,12]. In general, an involved peer in any P2P file sharing network must

answer the following questions when requesting a file: (1) which pieces should be downloaded, and (2) from which peers? This is known as piece scheduling [13], piece selection [14], or the coupon collector's problem [15]. The other important challenge, especially in large-scale content distribution networks, is that the departure, or the failure of a peer is naturally dynamic as the peer may depart or fail without notice, which is known as dynamics peer participation or churn [16]. Subsequently, pieces that were owned by the departing peers may become rare or unavailable.

With the assistance of network coding [17], the aforementioned issues can be partially resolved. Rather than distributing the original data pieces, a peer shares encoded pieces, where each encoded piece is a linear combination of the original pieces. In this manner, there is no need for the requester peer to ask for special data pieces since all encoded pieces are almost equally likely to be beneficial to the peer. A peer in the network can blindly download encoded pieces from other peers. When a sufficient number of encoded pieces are owned, the original file can be reconstructed. Using this scheme, the rarest piece problem is avoided, and the download time of the file is potentially reduced [18]. Another feature of network coding is the robustness against sudden peer departure. In the original BitTorrent, if a peer who is the single owner of a piece, leaves the swarm, the complete original file cannot be collected by the remaining peers. However, with network coding, the concern regarding the absence of certain data pieces is no longer an issue. Since the pieces are linearly combined together, each of them is available in a large number of encoded pieces in the network.

The main contributions of this paper are as follows. First, we provide a detailed tutorial of network coding implementation for P2P file sharing. Second, we extensively survey the most important and recent network coding based P2P file sharing protocols and provide a classification for those protocols. Third, this paper, to the best of our knowledge, is the first paper that is specific to network coding based P2P content distribution systems.

The rest of the paper is organized as follows: Section 2 introduces a brief background along with a network coding tutorial, while Section 3 outlines the literature review. In Section 4, different network coding based P2P file sharing systems are summarized. A discussion and comparison between these systems is conducted in Section 5 and the paper is concluded in Section 6.

2. BitTorrent-Like Systems and Network Coding Background

2.1. How BitTorrent-Like Systems Work

Initially, a seeder, who owns the full content of a file, divides the file into equal size segments called pieces, chunks, or blocks. For the rest of the paper, the piece terminology is used. As new peers join the swarm, i.e., P2P network, the seeder randomly selects and distributes pieces to leechers which are nodes that need some pieces or all the pieces to complete the file downloading. When a peer finishes downloading a piece, it can upload this piece to other peers who request it. As shown in Figure 1a, the peer maintains a structure called a bitmap which indicates whether the peer has a certain piece or not. The original specification [4] of the protocol requires that clients download pieces in a purely random way, but subsequently new techniques were introduced to improve the performance [11,19]. An example of an improved technique is the local rarest first, which is called the rarest first piece selection algorithm as well. In such technique, a piece that is the least common among all the neighbor peers is selected to be distributed with high priority. The target is to ensure a uniform distribution of the pieces. In practical systems, the distribution of the rare pieces is still a big issue—since the local rarest piece among neighbors is not necessary, the same global rarest piece among the entire network [14,18].

In the network coding BitTorrent as shown in Figure 1b, a peer first encodes the pieces that it has by multiplying each piece with a coefficient (c), which is randomly chosen from a finite field, known as Galois Field (GF). Then, the pieces are linearly combined as one encoded piece. The encoded piece is distributed to other downstream peers. At the receiver peer, the decoding process is performed when a sufficient number of encoded pieces are received.

(**a**) (**b**)

Figure 1. Illustration of piece distribution in baseline and network coding BitTorrent. (**a**) Baseline BitTorrent; (**b**) Network coding BitTorrent.

2.2. Network Coding Overview

Network coding is a recent technique that changes the traditional idea of store-and-forward routing by allowing a network node to linearly combine incoming packets and then send the combined packet to the next hub node. The idea was introduced by Ahlswede et al. [20], showing that the network coding can increase network throughput and improve robustness. The basic idea is relatively simple: the intermediate nodes combine the received packets, then send a linear combination of these packets, creating a new representation of the original packets. In order to decode and rediscover the original packets, the recipient must have a sufficient number of independent linear combinations of these packets. The benefits of network coding in computer networks have been demonstrated by the study of different aspects such as throughput and end-to-end delay [20], wireless resources [21–23], and problems of security and resilience [24–29]. The following subsection demonstrates the benefit of using network coding.

2.3. The Butterfly Example

In this subsection, we show a simple multicast network, with a single source node that sends messages to multiple recipient nodes. The butterfly network is the conventional example where throughput gain using network coding can be demonstrated.

In the directed graph shown in Figure 2, the source S sends two messages to each of the recipients R_1 and R_2. With traditional routing, R_1 will receive the first message through the path A-R_1, and the second message through the path C-B-D-R_1. Similarly, R_2 will receive the first message through the path C-R_2, and the second message through the path A-B-D-R_2. Assuming that each node can send only one multicast message per time slot, then using the traditional store-and-forward routing scheme, the link BD becomes a bottleneck. Furthermore, R_1 will receive message <a> two times, and R_2 will receive message <b> two times, leading to redundancy and wastage of the link bandwidth.

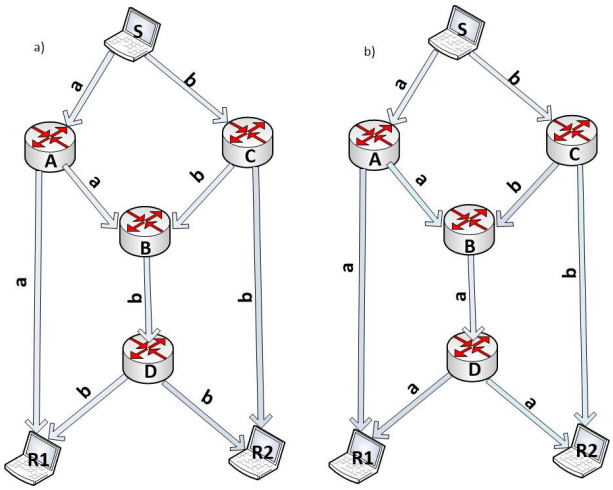

Figure 2. Butterfly network with traditional routing. (**a**) node B sends packet b; Then, (**b**) node B sends packet a [20].

Applying network coding as shown in Figure 3, the two messages <a> and <b> can be XORed at router B, a new message <c> is created such that <$c = a \oplus b$>. Then, this message is forwarded to router D which will send message <c> as a multicast to both R_1 and R_2. R_1 can retrieve message <b> by Xoring message <a> with message <c>, as <$b = a \oplus (a \oplus b)$>, while R_2 can retrieve message <a> by Xoring message <b> with message <c>, as <$a = b \oplus (a \oplus b)$>. Using this novel encoding technique, the time required to deliver the two messages to the recipients is decreased by one time slot. Furthermore, neither redundant packets nor wastage of bandwidth are incurred.

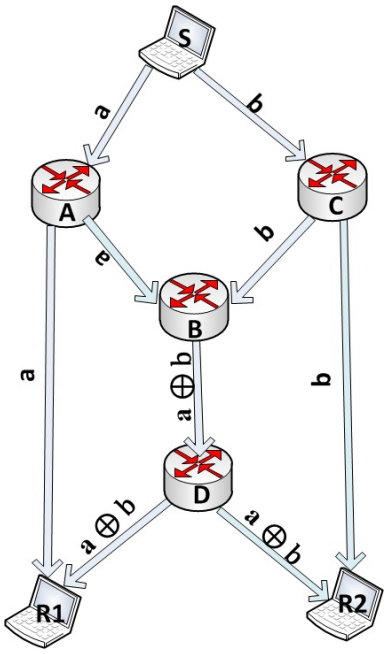

Figure 3. Butterfly network using network coding [20].

In more complex scenarios, we need more complex coding operations such as random linear network coding (RNLC) [30,31], or vector network coding [32–34]. With RNLC, the outgoing packets are linear combinations of original packets, so each encoded packet contains data plus coefficients. RLNC can be applied in a distributed fashion, where each node can independently and randomly select coefficients from the Galois field (GF). GF is very useful for practical purposes because each element of the GF can be represented with the same number of bits that guarantees that the result is representable with the same number of bits as the original packets.

2.4. Detailed Network Coding Tutorial for P2P File Sharing Systems

Since most of the published papers provide only high level information about implementation of network coding, we provide a thorough tutorial of network coding implementation.

The use of network coding requires the use of Galois field (GF) and its operations as well as basics of linear algebra. A GF is a field with finite number of elements. For instance, $GF(2^3)$ is a field whose elements are eight and each element is presented by 3-bits. The $GF(2^3)$ elements are generated using the following polynomial:

$$A(x) = a_2x^2 + a_1x + a_0 \tag{1}$$

Table 1 shows the elements (polynomials) of $GF(2^3)$ generated by eqrefeq:1 and their corresponding binary presentation.

Table 1. $GF(2^3)$ elements presentation.

Binary	$GF(2^3)$
000	0
001	1
010	x
011	$x+1$
100	x^2
101	x^2+1
110	x^2+x
111	x^2+x+1

The following four operations over any $GF(2^m)$ field are defined:

- Addition and Subtraction: The sum of two elements is computed according to (2):

$$C(x) = A(x) + B(x) = \sum_{i=0}^{m-1} c_i x^i, c_i \equiv a_i + b_i \; mod \; 2. \tag{2}$$

Addition and subtraction modulo 2 are the same. Moreover, addition modulo 2 is simply equivalent to bitwise XOR.

- Multiplication: two elements of $GF(2^m)$ are multiplied using the standard polynomial multiplication rule. However, if the product polynomial has a degree higher than $m-1$, then it has to be reduced. Irreducible polynomials, which are roughly comparable to prime numbers in such that their only factors are 1 and the polynomial itself, are used for the modulo reduction. Let $P(x)$ be an irreducible polynomial over $GF(2^m)$; then, the multiplication of two elements is done as in (3):

$$C(x) = A(x) \cdot B(x) \; mod \; P(x). \tag{3}$$

To find the product of $6(x^2+x)$ and $7(x^2+x+1)$ in $GF(2^3)$, we begin by doing normal polynomial multiplication: $C(x) = (x^2+x) * (x^2+x+1) = x^4+x$. Since $x^4+x \notin GF(2^3)$, the irreducible polynomial $P(x) = x^3+x+1$ is needed and thus we calculate $C(x) = (x^4+x) \; mod \; (x^3+x+1) = x^2 \in GF(2^3)$.

- Inversion: Every Element (except 0) in $GF(2^m)$ has an inverse. The inverse A^{-1} of a nonzero element $A \in GF(2^m)$ is defined as in (4):

$$A^{-1}(x) \cdot A(x) \equiv 1 \; mod \; P(x) \tag{4}$$

For example, in $GF(2^3)$ and $P(x) = x^3+x+1$, (x) is the inverse of (x^2+1) since $x * (x^2+1) \; mod \; (x^3+x+1) = 1$.

To use network coding for sharing a file within a P2P network, it should be started by decomposing the whole file into equal pieces. For example, each piece size is 16 KB, 32 KB, or 64 KB. Furthermore, each piece is divided into equal smaller units called blocks. For example, each block size is 4-bits or 8-bits as shown in Figure 4. Subsequently, network coding is applied per block, but a piece is still the transmission unit. Since $GF(2^8)$ is always recommended for real implementation scenarios, an 8-bits block is widely used.

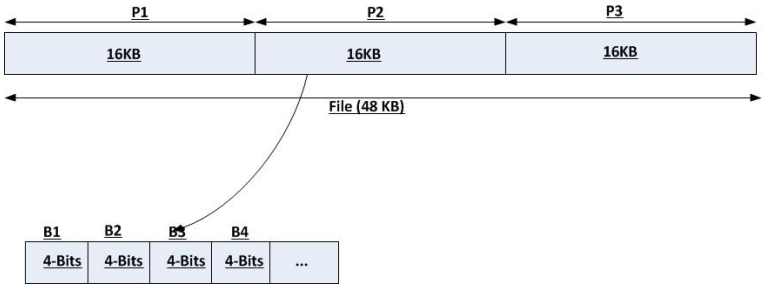

Figure 4. File segmentation phases.

The following example demonstrates how this process works using $GF(2^3)$. Assume, without loss of generality, that we want to share a 12-bits file using BitTorrent. Assume that the 12-bits file binary representation is (100011110110):

- 1st Step: The seeder decomposes the file into two 6-bit pieces (100011, 110110).
- 2nd Step: For encoding over $GF(2^3)$, we should decompose each piece into two 3-bits blocks: P_1(100, 011), and P_2(110, 110).
- 3rd Step: Interpret each block as $GF(2^3)$ element (using (1)); then, we have $P_1(x^2, x+1)$, and $P_2(x^2+x, x^2+x)$.
- 4th Step: Since we have two plain pieces, we need two coded pieces. Randomly draw coefficients from $GF(2^3)$ and multiply them by the blocks. Assuming we first draw $c_1 = x$, $c_2 = 1$, then the first block of the first encoded piece (P_{c1}) is $B_{c10} = x*(x^2)+1*(x^2+x) = x+1$ (011), and the second block is $B_{c11} = x*(x+1)+1*(x^2+x) = 0$ (000). Now, $P_{c1}(x+1, 0)$ is completely encoded and ready to be shared. Next, to get the second encoded piece, we randomly draw two additional coefficients, say $c_1 = x+1$, $c_2 = x$ and generate $B_{c20} = x+1*(x^2)+x*(x^2+x) = 0(000)$, and $B_{c21} = x+1*(x+1)+x*(x^2+x) = x^2$ (100). Thus, $P_{c2}(0, x^2)$ is completely encoded and ready to be shared.
- 5th Step: The sender shares the encoded pieces along with their coefficients $(P_{c1}, x, 1)$, and $(P_{c2}, x+1, x)$. This is algebraically represented as:

$$\begin{array}{ccccc} A & * & B & = & C \\ \left(\begin{array}{cc} x & 1 \\ x+1 & x \end{array}\right) & & \left[\begin{array}{c} b_{10} \\ b_{11} \end{array}\right] & = & \left[\begin{array}{c} x+1 \\ 0 \end{array}\right] \\ \left(\begin{array}{cc} x & 1 \\ x+1 & x \end{array}\right) & & \left[\begin{array}{c} b_{20} \\ b_{21} \end{array}\right] & = & \left[\begin{array}{c} 0 \\ x^2 \end{array}\right] \end{array} \tag{5}$$

  where A is the coefficients matrix, B is the original blocks vector, and C is the encoded blocks vector.

- 6th Step: Upon receiving the encoded pieces with their coefficients, the receiver should solve the previous matrices system to recover the original blocks and thus the original pieces. This is achieved by using the formula: $B = A^{-1} * C$. First, find the determinant of the coefficients matrix as shown in (8):

$$det(A) = \left|\left(\begin{array}{cc} x & 1 \\ x+1 & x \end{array}\right)\right| = x^2+x+1. \tag{6}$$

  Since $det(A) \neq 0$, the system is solvable:

$$\begin{aligned} A^{-1} &= \frac{1}{det(A)}\left(\begin{array}{cc} x & 1 \\ x+1 & x \end{array}\right) \\ &= \left(\begin{array}{cc} x^2+x+1 & x^2+1 \\ x & x^2+x+1 \end{array}\right), \end{aligned}$$

$$b_1 = A^{-1} * \begin{bmatrix} x+1 & 0 \end{bmatrix} = \begin{bmatrix} x^2 & x^2+x \end{bmatrix},$$

$$b_2 = A^{-1} * \begin{bmatrix} 0 & x^2 \end{bmatrix} = \begin{bmatrix} x+1 & x^2+x \end{bmatrix}.$$

This leads to reconstructing the blocks and thus the pieces: $P_1(x^2, x+1)$ and $P_2(x^2+x, x^2+x)$.

- 7th Step: Simply interpret the pieces' blocks as binary presentation and link the pieces together to reconstruct the original file.

The previous steps can be generalized for any number of pieces and for any finite field $GF(2^m)$. However, as the number of pieces increases, solving such linear system becomes far expensive.

Another network coding implementation guidance can be found in [35] and a ready Java and C implementations can be found in [36].

2.5. Network Coding Challenges

The use of linear algebra in encoding and decoding incurs additional computation overhead that is not present in the baseline BitTorrent. In BitTorrent, the file is split into pieces, and the peers are only concerned with completing the file by exchanging pieces with each other. A piece is considered useful if it is received by a peer for the first time. With network coding, the coded pieces are distributed rather than the original pieces. As the coefficients are received with a piece, they must be tested against all previous sets of coefficients that the peer has received to examine if the new piece contains any previously unknown information. The received encoded piece is said to be innovative if it contains beneficial information; otherwise, it will be discarded. Additionally, decoding the file after receiving enough pieces is more computationally intensive using network coding. 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 done until all the needed pieces are received. Consequently, decoding the file requires solving many linear equations. As the size of the file increases, the cost of solving these equations increases [32]. For example, for downloading a 3 GB file that is segmented into 1000 pieces, the required disk read operations are 3000 GB on each receiving node. Meanwhile, the BitTorrent system only needs to write 3 GB of data [33].

3. Related Works

Two major survey papers of network coding based P2P applications are found in the literature. First, Matsuda et al. [37] provide a comprehensive survey on network coding applications including techniques for throughput/capacity improvement, robustness improvement, network tomography, and security. Secondly, Li et al. [38] focus in their paper on both file sharing and media streaming. However, the latter survey is not comprehensive and its concentration appears to be on network coding theory rather than its applications. A very brief survey on network coding based P2P file sharing for delay tolerant networks (DTN), namely wireless channel networks, can be found in [39]. Additional theoretical network coding comprehensive survey is found in [40] and two other shorter surveys are found in [15,41].

Since some recent research related to network coding based P2P file sharing systems have been appearing, this survey aims to cover all network coding based P2P sharing systems focusing on the more recent techniques and solutions. In addition, this survey paper is specific in that it focuses only on P2P file sharing systems. Finally, this paper attempts to, in addition to surveying, provide classification, discussion, and assessment of the respective applications.

4. Network Coding Based P2P File Sharing Systems

Practically, five network coding schemes are applied to P2P File sharing Systems: (1) full coding approach in which the encoded piece is a linear combination of all the original pieces, (2) sparse coding approach in which only subset of pieces are selected for encoding, and (3) generation-based approach in which the file is segmented into groups; then, network coding is applied per group separately, (4) combined coding scheme which uses a mix between generation-based and sparse coding, and (5) an overlapped coding scheme that is fundamentally a generation-based approach but allows overlapping between groups. In the following subsections, we extensively survey the studies that are related to each of these approaches.

4.1. Full Network Coding

Gkantsidis et al. [42] proposed Avalanche, which is the first P2P file sharing system that applies the concept of network coding for P2P content distribution networks. The main three contributions of that work are: (1) Apply a practical network coding based method which alleviates the piece selection problem across a large scale distributed system and makes better use of the available network bandwidth. (2) Provide experimental evaluation which shows that the network coding outperforms both the encoding at the source approach [43] and no coding at all, by a factor of two and three, respectively. (3) 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 is left after uploading exactly one copy of the file to the network. For the evaluation, the authors implement a real system simulator "Avalanche" that utilizes network coding. They evaluate the performance for different network topologies, a heterogeneous number of peers, and dynamic peers' arrivals and departures. They calculate the time it takes for one peer to download the file, measuring both the average download time and the maximum. Another performance metric used is the network throughput as the total number of pieces transferred in a unit of time. In general, the results show that network coding performs well when the peers have different link rates, when there is a churn, and when incentive mechanisms [44,45] such as Tit-for-Tat are in place to reduce free-riders and encourage cooperation.

A theoretical analysis of a simplified version of Avalanche can be found in [46]. The study assumes one downloader and multiple servers in which each server contains a small part of the file. Moreover, each server does not have any knowledge of what the others are storing (i.e., stored pieces). The analysis and results show that the encoding vectors generated by network coding, for $GF(2^3)$ and a 1 GB file size, require an additional space of around 10^{-4}% at the sources, whereas the erasure coding [47] ($\beta = 8$) requires 9 GB storage space at the file server. Furthermore, results show that RLNC is highly applicable for uncoordinated P2P networks by making the system behave as if the different server peers, which the downloader connects to, coordinate effectively for distributing the different pieces of the file.

A similar work to [46] was proposed by Deb et al. [48]. The study applied RLNC to gossip-based protocols [49], where nodes act based on their own state only without any knowledge of the contents of other nodes. The model is very similar to the rumor mongering model [50], where each node chooses a partner in a randomly uniform manner and only one piece can be transmitted per round. RLNC is compared with Random Piece Selection (RPS) and modified RPS (mRPS). In the RPS approach, the piece to be sent is selected among other pieces with equal probability, whereas the mRPS approach allows negotiation between the two connected nodes before a piece is sent. The theoretical analysis shows that the computational overhead of RNLC to recover the file is rather reasonable for $n \leq 1000$, where n is the number of pieces. Simulation results show that RLNC always outperforms RPS and mRPS for dissemination pieces over the networks. None of the aforementioned studies [42,46,48] simulate the computation overhead incurred within the decoding process, which is considered an important issue of the full network coding since it is accomplished by inverting an n x n matrix which requires $O(n^3)$ operations.

Wang et al. [51] proposed Downloader Initiated Random Linear Network Coding (DRLNC) to solve the problem of generating useless coded piece from two or more useful coded pieces or what

they called "unlucky combination". In DRLNC, each node keeps copies of the current coefficients matrices of its neighbors and updating information exchanged among these nodes once a new innovative piece is received. Based on this information, a node knows where the innovative pieces are placed and thus sends a request to the designated node for downloading. By this scheme, any unlucky linear combination is avoided. The DRLNC approach can eliminate the unlucky combination problem even with a field size of 2 unlike the other approaches such as Avalanche which state that the problem could be considered negligible if the field size is increased. However, the authors show that increasing the field size to $q = 7$ may waste $1/8$ of the network bandwidth. Simulation results show that DRLNC has less number of rounds to complete downloading than traditional RLNC approaches. However, extra control messages must be changed among peers which may incur extra overhead to the network.

Yeung et al. [52] concluded in their analysis of Avalanche that Avalanche can achieve the theoretical lower bound of the file downloading time. However, the real performance of Avalanche is affected by some factors such as computational efficiency, incentive mechanisms, and piece scheduling algorithms.

However, Chiu et al. [13] stated that Avalanche did not present enough proof to its performance claims. The authors studied network coding in P2P file sharing by modeling a simple star topology network, assuming that this topology has the important features of a P2P systems. They show that there is no advantage of network coding over routing, since the network coding can be beneficial only if it is applied on a network which supports multicast, and indeed this is not the case for P2P file sharing networks.

Soro et al. [53] proposed a network topology based on Fast Fourier Transform (FFT) graphs and adapting network coding to achieve robustness in the case of churn, and flexibility of content distribution. The authors claim that this topology can improve throughput and solve the rarest piece problem; however, neither analysis nor experimental work are presented by the authors to support their claim.

Since the Avalanche claim is based on simulation and not real implementation and to perfectly judge the feasibility of applying network coding to P2P file sharing systems, Wang et al. [54] implemented an empirical testbed. The experimental results show that the use of network coding in P2P content distribution proceeds even worse than any trivial BitTorrent system. The main two reasons stated of this inferior performance are: (1) the heavy decoding computation overhead and (2) the latencies of buffering coded blocks before the ability to generate new coded blocks.

Li et al. [55] tried to improve the neighbors selection of Avalanche by giving each neighbor a degree, and based on the neighbor degree the decision made whether to select the neighbor or not. The degree criteria determines how a particular neighbor can utilize the network bandwidth, and thereby accelerates the download time. However, the computational complexity of Avalanche is not discussed at all.

To conclude, full network coding suffers from very high computational overhead which makes it impractical in large scale P2P file sharing systems. All the studies that show the benefits of full network coding are based on theoretical analyses and/or numerical simulation, whereas the studies that show the infeasibility of full coding are based on real implementation. Table 2 summarizes the most important studies related to the full network coding.

Table 2. Full network coding approaches and their characteristics.

Reference	Approach	Main Findings	Verification Method	Network Topology	Performance Metrics
Gkantsidis et al. [42]	Utilizing RNLC on P2P file sharing networks.	- RLNC alleviates the pieces selection problem and churns, and accelerates the downloading process.	Numerical Simulation	Single overlay, and Multi-clusters overlay.	- Download time. - Throughput.
Acedanski et al. [46]	Providing extensively theoretical analysis for P2P RNLC-based method.	- RLNC overhead is about $10^{(-4)}$ % of the file size. - RLNC is highly applicable for uncoordinated P2P networks.	Numerical Simulation	Star.	- Probability of download completion. - Probability of contents availability.
Deb et al. [48]	proposing RLNC-based approach for P2P gossip protocols.	- Computational overhead of RNLC to reconstruct the file is rather reasonable for $n \leq 1000$.	Numerical Simulation.	Complete Mesh.	- Download time.
Wang et al. [51]	Proposing DRLNC to mitigate unlucky combination problem.	- Unlucky combination problem could be eliminated even with $GF(2)$.	Numerical simulation.	Mesh.	- Rounds to complete download.
Yeung [52]	Analysis Avalanche using graph theory.	- Avalanche can achieve the theoretical lower bound of the file downloading time.	No experimental work.	Mesh.	—
Chiu et al. [13]	Avalanche is studied by modeling a simple star topology network.	- No advantage of network coding over traditional routing.	No experimental work.	Star.	—
Wang et al. [54]	Justifying the feasibility of RLNC on P2P systems by realistic application layer implementation.	- RNLC performs worse than any conventional store and forward P2P file sharing system.	High performance C++ implementation.	Mesh	- Average downloadtime. - Encoding/ decoding complexity.

4.2. Sparse Network Coding

In sparse coding scheme, instead of encoding all the pieces, only some pieces are selected in random to be encoded aiming to alleviate encoding and decoding computational complexities.

Guanjun et al. [56] proposed a sparse network coding scheme based on stochastic formulas; the probability of the number of pieces that will be encoded by the seeder, which has all the pieces (n) is set as in (7). For a non-source peer which has k pieces, the encoding probability is set as in (8):

$$p = (logn + d)/n \tag{7}$$

$$p = (logk + d)/n \tag{8}$$

where d is a nonnegative constant $(d = 6)$. The study also focuses on how to choose coefficients such that no linear dependency appears. The Chord overlay network [1] that is used as a network topology and the encoding interval, which is a window to limit selecting pieces for encoding according to their expected innovation, is used to minimize the likelihood of transmitting dependent pieces. The authors argue that using the aforementioned formulas, (7) and (8), for encoding probability ensures that all of the encoded pieces generated by the same peer are linear independent with high probability (99.991%), taking into account that the used finite field is $GF(2^8)$. Moreover, a dependency test at the receiver nodes is conducted to prevent receiving more dependent pieces which works as follows: let H and G be two connected peers via a link, if no less than 3% of the pieces from H to G are dependent, H will stop requesting G for pieces for 30 s. If the dependency rate is greater than 5%, the link between the two peers will be torn down. We believe that the dependency rate is not increasable since the coefficients are randomly selected and the finite field size is large enough. Empirical results indicate that the sparse network coding outperforms the full coding in terms of encoding/decoding rate by 10% and slightly outperforms the baseline BitTorrent with a rarest-first scheduling policy in terms of download and distribution times. The definition of total download time does not include the decoding time. We believe that it would be fairer if the robustness to churn test was conducted between the sparse and the full network coding.

Ortolf et al. [57] proposed a "Paircoding" scheme. In Paircoding, each coded piece, $P_{(i,j)}$, is a linear combination of only two original pieces (X_i, X_j) such that the decoding overhead remains quite reasonable. Using this approach, the coded piece given by (9) has half of the information about the original pieces X_i and X_j that it is generated from. If the coefficients are linear independent, two different coded pieces generated from the same original piece contain the original data of the original piece:

$$P_{(i,j)} = C_i X_i + C_j X_j \tag{9}$$

The authors claim that Paircoding relatively decodes pieces as good as BitTorrent, and, for some scenarios, it shares pieces as good as full network coding. For the sake of the decoding speed comparison between Paircoding and BitTorrent, a file of n pieces ($n = 30$) is distributed by a seeder to one downloader. For BitTorrent, the seeder selects at random one piece for sharing, while, in Paircoding, the seeder randomly selects two pieces and then shares a linear combination of them. Experimental results show that, through a file sharing process, BitTorrent can decode (i.e., collects non-redundant pieces) better than Paircoding in the first quarter of the process, but, as time progresses, Paircoding outperforms BitTorrent. Paircoding can start decoding in the best case after receiving two encoded pieces. For the content availability metric, Paircoding falls in the middle between Avalanche and BitTorrent. Nonetheless, in some scenarios, Paircoding provides content availability as Avalanche but with very low probability.

Fixed-Paircoding [58,59] is a variation of Paircoding in which each encoded piece as shown in (10) is a combination of two adjacent original pieces. Fixing the choice of the two original pieces yields a faster decoding. However, its influence on the content availability is worse than Paircoding

$$P_{(i,j)} = C_i X_i + C_j X_j \tag{10}$$

where i is an odd number, $1 \leq i \leq n-1$ and $j = i+1$.

Cai et al. [18] conducted an in-depth analysis of Avalanche and Paircoding, and proposed a new fixed coding approach that is very similar to fixed-Paircoding. Furthermore, the rarest-first scheduling policy is also considered. The proposed approach depends on computing the importance of the encoded pieces according to their underlying original pieces: those encoded pieces that consist of the original pieces that are less distributed among neighbors will be assigned with a higher priority to be requested. In contrast to [60], the authors claim that fixed Paircoding could promote the availability and diversity of the pieces. To evaluate the performance, the proposed approach is tested and compared against: (1) BitTorrent, (2) adaptive neighbor selection BitTorrent [19], and (3) BitTorrent using FEC [43]. The results show that the proposed approach outperforms the other implementations with regard to throughput with a slight increase in the control overhead. In addition, it has been shown that the duration of the period during which no seeder is available while content is available, is greatly prolonged and only a few peers are sufficient to render the content available.

Other variations of Paircoding are Treecoding and Tree network coding [58–60]. In these schemes, the encoded pieces are defined by a complete binary tree, where the leaves contain the original pieces such that each leaf contains an original piece multiplied by a coefficient. For the upper levels, each piece is generated by XORing its children pieces. If the coefficients' vector is known, then any encoded piece in the tree can be decoded simply by XORing its two encoded children pieces, or from its encoded parent piece and its sibling. The main difference between Tree coding and Tree network coding is that the former only allows encoding at the seeders, whereas the latter allows recoding within the other peers. The main feature of Treecoding is that an encoded piece may contain information up to all n original pieces when compared with Paircoding whose encoded piece covers only two original pieces at most. However, the static encoding, namely allowing encoding only at the seeder, yields a reduction in the availability of pieces. Figure 5 shows an example of Treecoding.

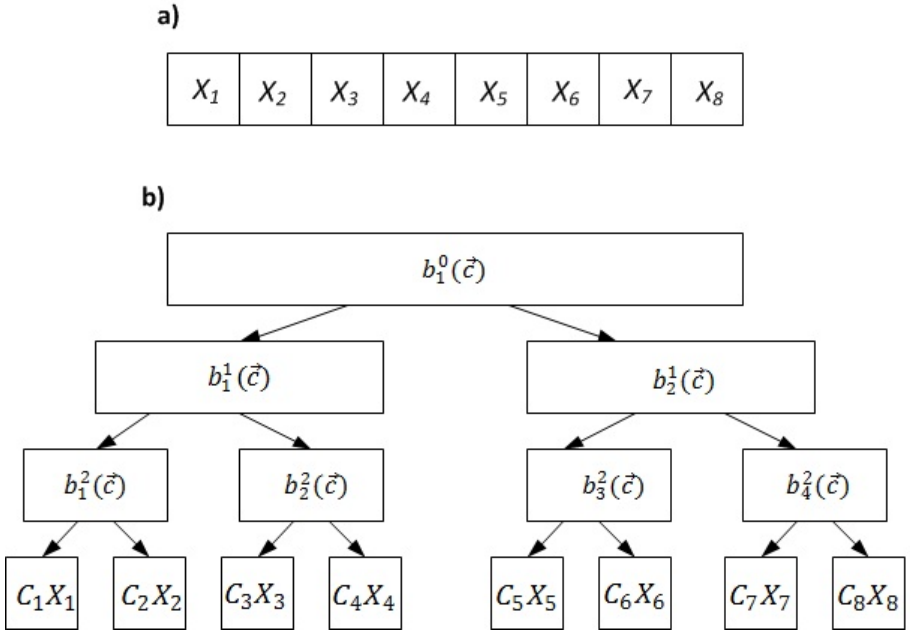

Figure 5. (**a**) File X with $n = 8$, (**b**) Tree coding of file X.

In conclusion, sparse network coding could be considered practical and valid for implementation for real world P2P file sharing systems. We re-evaluate the decoding rate of Paircoding and, as depicted in Figure 6, the results show that Paircoding is outperformed only for the initial rounds, then it significantly outperforms BitTorrent. Table 3 summarizes the most important studies related to sparse network coding.

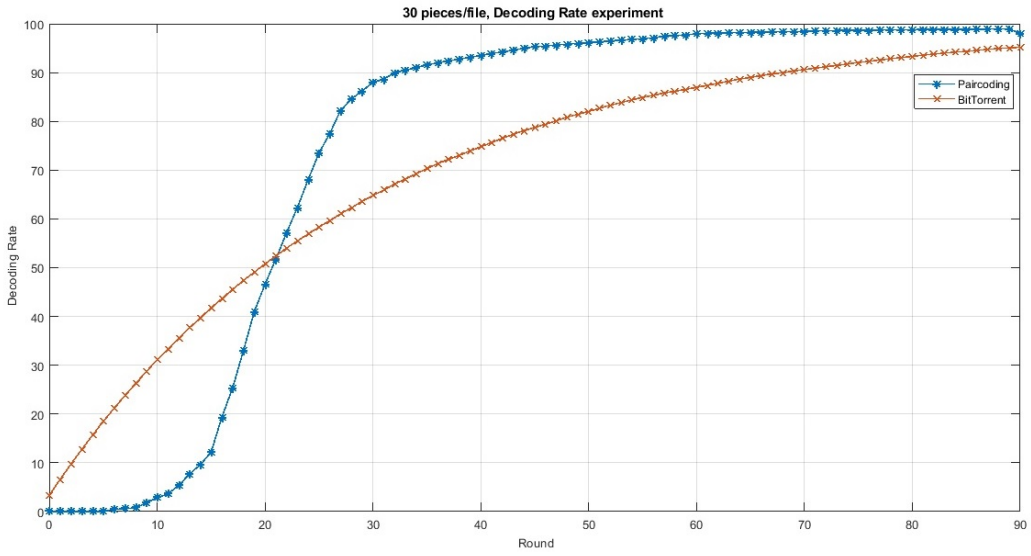

Figure 6. Paircoding vs. BitTorrent decoding Rates.

Table 3. Sparse network coding approaches and their charactersitics.

Reference	Approach	Main Findings	Verification Method	Network Topology	Performance Metrics
M. Guanjun et al. [56]	Proposing sparse network coding based on stochastic formulas.	- Sparse coding encodes/decodes faster than full coding and slightly downloads faster than baseline BitTorrent. - Encoding interval and dependency test can minimize the drop rate of dependent coded pieces.	Implementation	Chord overlay	- Encoding/decoding rate. - Download time.
C. Ortolf et al. [57]	Proposing sparse coding such that each encoded piece is a combination of only two randomized original pieces.	- Paircoding relatively decodes pieces as good as BitTorrent, and for some scenarios, it achieves the piece diversity of full coding.	Numerical Simulation	Mesh	- Decoding rate. - Content availability.
C. Ortolf et al. [60]	Proposing sparse coding such that each encoded piece is a combination of two adjacent original pieces.	- Fixing the choice of the two original pieces yields to faster decoding. yet affects the piece diversity.	No experimental work	—	—
Q. Cai et al. [18]	Analyze Avalanche and Paircoding, and propose a Fixed-Paircoding with considering the rarest first scheduling policy.	- Achieves both fast decoding opposed to Avalanche and wide piece diversity opposed to Paircoding. - Increases throughput with slightly control overhead.	Numerical Simulation	Mesh	- Content availability. - Control overhead. - Download time.
C. Ortolf et al. [55]	Proposing sparse coding by modeling encoded pieces as full binary tree	- allowing encoding only in the seeder, yields to worse pieces availability. On the other hand, dynamic encoding yields to much complexity.	No experimental work	—	—

4.3. Generation Based Coding

After the appearance of Avalanche and the associated issues, generation-based network coding [61], also known as grouped [23], chunked [62], segmented [63], or clustered [64], is proposed. In such a scheme, the file is segmented into groups where each group contains a mutually exclusive subset of the overall pieces; afterwards, network coding is applied per group only. The two major benefits of this scheme are: (1) reducing the required computations and disk operations, and (2) minimizing the size of encoding vectors. For the rest of the paper, we use the terminology 'generation based coding' or 'generation coding' for simplicity.

Chou et al. [65] propose a so-called practical network coding which splits a file into generations with each generation containing a subset of the file's pieces. Subsequently, network coding is applied on each generation separately. The authors suggest that the transmission of encoded pieces 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 completed and thus decoded.

Maymounkov et al. [66] suggest that, in order to avoid the overhead of the control messages, a generation can be selected randomly and then the generation based encoded piece is sent. The authors argue that this method achieves a good level of the performance without the need for feedback. However, redundant and unrequired pieces could be pushed to the network, and thus yielding even more overhead. The former two studies are general and not intended for P2P content distribution networks.

To address the overhead problems of the aforementioned studies [65,66], Xu et al. propose Swifter [67], a scheme that mixes local-rarest-first scheduling policy with generation coding algorithms so as to achieve a high scheduling efficiency. The authors start by studying the trade-off between the network coding overhead and the scheduling overhead. To make sure that the pieces are shared

in a high diversity, the local-rarest-first scheduling algorithm is applied at the generation level as follows: each peer presents a list of the number of pieces in each generation within every neighbor. Periodically, this information is exchanged among the peers and updated. Among the generations which still need pieces to be decoded, the requester can choose the rarest generation based on that list and then pull a piece in that generation from the sender. If there is more than one rarest generation, a generation is randomly selected among them. After collecting all the pieces in one generation, the receiver can reconstruct the original pieces of the generations. The authors argue that this scheme improves the encoding/decoding speed and there is no need to use sparse network coding that is used within Paircoding. However, this claim is not supported. To evaluate the performance of Swifter, the authors compare Swifter with [66] denoted by "R-push" and another scheme that is a modified version of Swifter yet with random scheduling denoted by "R-Swifter". The results show that Swifter can enhance the average download time by at most 40% compared to R-push and by at most 6% compared to R-Swifter.

Since Swifter is a pull-based scheme, it still incurs moderate control messages overhead. Xu et al. [62] propose an improved Swifter, denoted by I-Swifter, in an attempt to improve generation-based network coding by reducing the control messages overhead and eliminating the distribution of encoding vectors. The architecture of I-Swifter is based on the architecture of Swifter, and hence it inherits Swifter components and adds two new elements: the requests reducer that lowers the requesting overhead, and the coefficients' vectors reducer that stops the distribution of encoding vectors. Extra requests from a receiver peer appear when continuous requests are sent to the sender peer for the same generation. To minimize these request messages, the receiver peer will send a generation request once and upon the request received by the sending side, the sender peer continuously send pieces to the receiver in a push-based manner until a control message, from the receiver, arrive at the sender saying that the local-rarest generation is changed. Moreover, the distribution of encoding vectors is eliminated by the encoding vectors reducer which allows the sender to send only random seed rather than the coefficients' vectors. The receiver peer can generate the encoding vectors by using the same pseudo random number generator. Experimental results show that I-Swifter can enhance the average download time by as much as 4% when compared to Swifter. In both Swifter and I-Swifter, the optimal number of generations is not specified.

Hundeboll et al. [68] propose BRONCO. BRONCO considers a scenario where there is one server that initially distributes pieces of a file to multiple peers with a total download rate greater than the upload rate of the server and then the peers exchange their pieces. The authors state that there are three important parameters that affect the performance of the system and thus these parameters should be selected carefully: (1) number of generations (g) for a given file, (2) size of the finite field (q), and (3) size of each piece in a generation (b). The selection of these parameters is a trade-off process between the computation complexity of the network coding, and the probability of creating linear dependent vectors. Increasing g or q while keeping the size of the file constant reduces the expected number of the linear dependent vectors, and, subsequently, the amount of valid vectors is increased. However, the complexity of encoding and decoding pieces will increase as well. 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, yet with a redundant packets' overhead of 9%. For evaluation purposes, BRONCO is compared to HTTP (as a standard method to transfer a file), and BitTorrent. As expected, BRONCO far outperforms HTTP, and almost performs as well as BitTorrent.

Niu et al. [69,70] specify when it is advantageous to use network coding while getting acceptable computational coding complexity. Theoretical analysis based on Markov 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, considering peers collaboration system (no free-riders), although it is prone to a high churn rate.

Zhang et al. [71,72] present the impact of network coding by using a game theory framework. They consider a non-collaborative system, in which egoistic free riders comprise the dominant part of the system. Thus, the network is modeled as a market, and peers are modeled as agents who purchase and resell pieces. The prices of the pieces are strategically set according to their availability. Mathematical analysis results show that full network coding is the most robust to churn, but with high cost of encoding and decoding complexities. When generation network coding is used with a generation size (g) that is smaller than a threshold that depends on the churn rate, robustness could be achieved with feasible cost of encoding and decoding rates. This conclusion is obtained by using $g = 20$ and $g = 5$. In addition, the results show that BitTorrent without coding is the most affected by churn even when the churn rate is very small (close to zero).

Leu et al. [73] execute a wide range of P2P file sharing simulations including both with and without network coding schemes for the sake of comparison and analysis. Results show that utilizing generation network coding for a P2P file sharing network incurs only a relatively small overhead and could perform much better than trivial routing approaches as long as the following conditions are met—first, using DRLNC [51] to avoid unlucky combinations; second, the selection of the size and the number of coding pieces should be done carefully such that the coding speed is no longer slower than the transmission speed; and, third, applying Gauss–Jordan elimination for early decoding to avoid long delay in the decoding process.

Yang et al. [74] propose P2P FilE sharing based on nEtwork coDing (PPFEED). PPFEED studies network coding by utilizing a special network topology called a combination network [75,76], which can be modeled by a graph that contains three types of nodes: one source, relays, and receiver nodes. The authors claim that such a topology has a good level of performance in terms of network coding gain. In contrast to RLNC, a deterministic network coding scheme is proposed based on [77] such that the encoded pieces are always innovative. For n pieces, generation size g, and $GF(q)$ with q different symbols such that $q \geq n$, pieces of a generation are encoded n times and for each row vector only one coefficient, c_i, is randomly selected from $GF(q)$ and then the encoding is done as follows: the 1st piece's value is multiplied by 1, the 2nd piece's value is multiplied by the selected coefficient, the 3rd piece's value is multiplied by the same coefficient squared, and so on until the kth piece is multiplied by the same coefficient raised to a power of $g - 1$. For instance, for $g = 3$, coefficients vectors are selected as follows: $(1, c_1, c_1^2 mod q)$, $(1, c_2, c_2^2 mod q)$, ..., $(1, c_n, c_n^2 mod q)$. By this scheme, any g pieces of these n pieces can be used to decode the original g pieces. Moreover, relay and receiver nodes no longer need to re-code pieces, and just need to forward them properly such that a peer doesn't receive the same coefficient vector two times. To evaluate the performance of this scheme, simulation comparisons are conducted against Narada [78], a P2P multicast system, and Avalanche. The results show that the overall download time of PPFEED is shorter than Narada and Avalanche by 15%–20% and 8%–10%, respectively.

Braun et al. [79] propose Network Coding Messaging Extension (NCME). NCME is not an alternative for BitTorrent; it only extends the features of BitTorrent and backward compatibility while the baseline BitTorrent is kept. The peer must decide which optimal communication paradigm to use, either pure BiTorrent or NCME. After encoded pieces are received 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 BitTorrent, it is suggested to keep a generation size of 43 pieces. However, neither is the best piece size mentioned nor an experimental work provided to support this suggestion.

From the aforementioned studies, we can conclude that generation network coding can be applied practically. However, to get the best gain of generation network coding, parameters such that generation size and piece/generation selection policy should be perfectly selected. Table 4 summarizes the most important studies related to generation Network coding.

Table 4. Generation network coding approaches and their characterstics.

Reference	Approach	Main Findings	Verification Method	Network Topology	Performance Metrics
Xu et al. [67]	Proposing pull-based generation coding system and mixing rarest-first selection policy with generation coding to alleviate the overhead of control messages.	- Swifter can reduce the average download time by 40% compared to push-based generation system with random selection policy.	Implementation over LAN with 30 nodes.	Partially Meshed	- Download time.
Xu et al. [62]	Proposing push-based generation coding system promising to reduce the requests overhead.	- I-swifter can reduce the average download time by 4% compared to Swifter.	Implementation over LAN with 30 nodes.	Partially Meshed	- Download time
Hundeboll et al. [68]	Implementing generation coding system to study the parameters of generation coding: generation size, GF size, and piece size.	- BRONCO far outperforms HTTP while it performs almost as good as BitTorrent. - BRONCO consumes almost quarter the CPU utility Avalanche consumes but with extra 9% redundant pieces.	C++ Implementation	Partially Meshed	- Download time
Niu et al. [69,70]	Modeling generation coding system by Markov process and differential equations and study the optimal generation size.	- The optimal generation size to enjoy network coding is 20–30 pieces.	Numerical simulation	Mesh	- Decoding rate. - Download time.
Zhang et al. [71,72]	Proposing a game theory framework to study generation network coding considering a system with many free-riders.	- network coding can enhance the market's flexibility for urgent peers, but with the high encodingdecoding cost. - network coding can improve the peers' incentive. Only mathematical and analytical model.	Only mathematical and analytical model	—	- Robustness to churn.
Leu et al. [73]	Proposing a framework based on simulations to deeply analyze and understand generation network coding.	- Network coding outperforms trivial approaches when (1) DRLNC is used, (2) appropriate coding size is selected and, (3) Gauss-Jordan elimination is applied for early decoding.	C++ P2P simulator [80]	GIA [81] overlay	- Encoding/ decoding rates. - Download time. - Network overhead.
Yang et al. [74]	Proposing deterministic network coding and utilizing a special network topology "combination network".	- The overall download time of PPFEED is shorter than Narada and Avalanche by 15-20% and 8-10% respectively.	Simulation	Combination network overlay	- Throughput. - Reliability (to churn). - Link stress (redundancy). - Download time.
Braun et al. [79]	Proposing generation coding P2P file sharing system with backward compatibility with standard BitTorrent.	- In some scenarios, NCME can share a file to the network 20% faster than BitTorrent. - The suggested generation size for good level of performance is 43.	Java Implementation	Partially Meshed	- Download time. - Generation size.

4.4. Combined Network Coding

Since, in the worst case, sparse coding decodes almost as poorly as full coding, and generation coding limits the diversity of pieces, combined network coding is proposed. In this scheme, generation network coding is mixed with sparse network coding aiming to minimize the decoding/encoding computational complexity, and at the same time keep the diversity of the pieces widespread.

Zeng et al. [82] propose a redesigned model for network coding based P2P file sharing systems, by adding two new parameters: generation size and encoding size. The model is a generation based network coding model, but the main distinguishing feature of this model is that the encoding size may be less than or equal to the generation size while the decoding size is always kept equal to the generation size. The authors claim that the appropriate optimal choice of the two new parameters can show the benefit of using network coding. If the swarm size is small (20 to 40), determining the appropriate encoding size is hard and the system performance is unstable. However, if the swarm size is relatively large (90 to 100), the best encoding size is 4 when the generation size is fixed at 10. The average download time is used as a performance metric, and the proposed system is compared with the original BitTorrent and another combined network coding system whose parameters are not selected properly (generation size = 12, and encoding size = 2). Simulation results show that the

optimized network coding guarantees improved performance by at most 20% to 30% over BitTorrent, and 10% over coding system with poorly selected parameters.

Yong et al. [63] use the same parameters proposed in [82] (generation size = 10, and encoding size = 2) and propose a coding system based on CoolStreaming architecture [83] by adding a network coding encoder/decoder and a scheduler. Empirical results show that the best encoding size of the original seeder is 6, and the average download time is 10% and 20%–30% less than that for Avalanche and BitTorrent, respectively. Moreover, an improvement is shown in resisting peer churn by 12.5% compared with the BitTorrent system.

Kaiqian et al. [84] propose a Dasher system that is a combined network coding scheme. In addition, the local rarest-first policy is adopted to deal with generation scheduling. The main feature of Dasher over the previous combined network coding schemes is that the encoding size of Dasher varies based on a probability formula. For the sake of comparison, Dasher and three other systems are implemented: Sparser with sparse coding only, Chunker which uses generation coding only, and a BitTorrent-like system. Dasher benefits from this combined scheme such that it can nearly decode as fast as Chunker, and at the same time it can download almost as fast as Sparser. Experimental results show that Dasher can improve the download time by up to 43% and 14% against Chunker and BitTorrent, respectively, when the generation size is 256. However, when the generation sizes are small (16 to 64), Chunker and Dasher download nearly at the same speed and with an improvement of 12% relative to BitTorrent. The results also show that Sparser always downloads slightly faster than Dasher; however, Dasher can decode faster than Sparser by 10%.

Su et al. [85] propose PCLNC, a Push-based Combined coding strategy with adaptive encoding window size and Low-cost computational Network Coding operations. The sender prepares a coded piece based on the requested piece such that the requested piece encoded with probability 1 and the other pieces are encoded with probability 0.5; then, the encoded piece is pushed to the requester. To speed up the decoding process, an upper triangle matrix is introduced. To minimize the encoding/decoding cost, coefficients are selected from $GF(2)$ and thereby XOR operations are used rather than addition and multiplication. To reduce the high probability of linear dependency of selecting coefficients from $GF(2)$, two schemes are proposed: (1) Postponement scheme which is very similar to the scheme inspired by [56] . However, in this scheme, after a requesting peer receives a linear dependent piece from a requested peer, then, the requesting peer will stop requesting the requested peer for two seconds, if the requesting peer receives consecutive linearly dependent pieces, the waiting time will be twice the number of these dependent pieces. (2) Loop self-checking scheme: When a peer is receiving linearly dependent pieces from any neighboring peer for more than 180 s, this means that there is a high probability that these two peers have the same pieces, thus they tear their connection and try to find other neighbors. To evaluate the performance of PCLNC, it is compared with native BitTorrent and sparse network coding scheme proposed by [56]. Metrics selected for evaluation are: (1) Average download time: PCLNC download time is shortened by 3.17% and 21.0% compared to [56] and BitTorrent, respectively. (2) Average start-up time: PCLNC peer can start sharing a piece faster than BitTorrent by 36.8% , and almost as fast as [56], and (3) Coding degree (number of innovative pieces): for PCLNC as the number of peers increases, the coding degree decreases and is always less than that for [56].

These studies are limited and don't show deep analysis of the ability of the combined coding to provide wide sparsity and diversity of the content (file's pieces). Table 5 summarizes the most important studies related to combined network coding.

Table 5. Combined network coding approaches and their characteristics.

Reference	Approach	Main Findings	Verification Method	Network Topology	Performance Metrics
Zeng et al. [82]	Introducing and adjusting new parameters (generation size and encoding size).	- If parameters are well tuned, network coding outperform other traditional P2P schemes. Otherwise, network coding performs worse.	Simulation on NS-2 platform	Partially Meshed	- Download time. - Robustness to churn.
Yong et al. [63]	Studying the effect of the generation and encoding sizes on download time and churn.	- The download time is shortened compared with Avalance and BitTorrent by 10% and 20%–30%. - Churn resist can be improved by 12.5%.	Simulation based on CoolStreaming overlay network.	Partially Meshed	- Download time. - Effect of churn.
Kaqian et al. [84]	Proposing combined coding with adoption of local rarest first policy for generation scheduling.	- Dasher can download faster than Chunker and BitTorrent as well as decode faster than Sparser.	Implementation. Tested both on Planet-Lab [86] and LAN testbeds.	Mesh	- Download time. - Decoding speed.
Su et al. [85]	Proposing (1) adaptive encoding window size and upper triangle matrix to speed-up encoding/decoding and (2) postponement and loop self-checking schemes to minimize linear dependency.	- PCLNC is shortened the download time by 3.17% and 21.0% compared to sparse coding and BitTorrent, respectively. - PCLNC peer can start sharing a piece faster than BitTorrent by 36.8% , and almost as fast as sparse coding.	Simulation based on peerSim [87]	Mesh	- Download time. - Start-up time. - coding degree.

4.5. Multi-Generation Mixing (MGM) and Overlapping Network Coding

In MGM [88,89], a file is chunked into g generations. Each of these generations contains p pieces. Additionally, among all the generations, m generations are grouped in a mixing set where $m \leq g$. In such a scheme, in addition to the trivial coding per generation, pieces are encoded/decoded per the mixing set as well. Precisely, the first generation's pieces in the mixing set are just encoded as they are encoded in the classical generation coding, but the second generation's pieces in the same mixing set are encoded separately in their separate generation k times and then combined and encoded with either some or all the pieces of the first generation at least once, continually, the third generation is combined with the first and the second generations and so on. By this scheme, each earlier generation could be considered as a subgroup of the subsequent generations. Figure 7 shows a simple example which illustrates how MGM encoding is done where E_x denotes an encoded piece and c_x denotes a coefficient. The main advantage of this scheme is that an extra encoded piece received with a generation of a higher position index in the mixing set may assist with decoding generations of lower position indices in the same mixing set resulting in robustness to churn and a wide piece diversity.

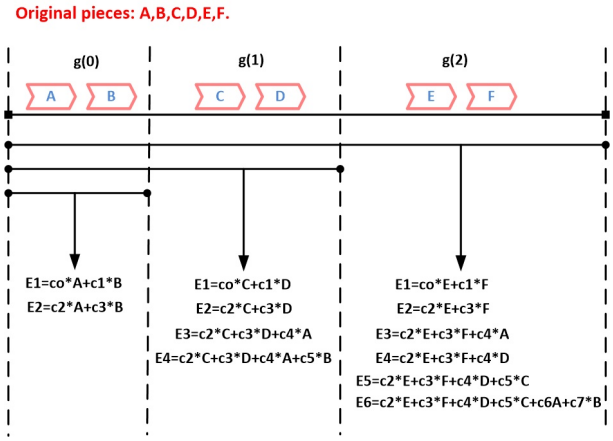

Figure 7. Multiple-Generation-Mixing (MGM) coding illustration, E_x denotes encoded piece, and c_i denotes a coefficient.

Similarly, in the overlapping coding scheme, generations are prepared in such a way that the same piece/s are available in more than one generation. Two generations g_0 and g_1 overlap if they have a non-empty intersection. For instance, $g_0 \cap g_1 = p_i$. By this scheme, generations that are fast decoded can assist other generations in decoding when some generation's pieces are rare or missed by allowing back substitution. As a result, the number of innovative coded pieces is reduced, which in turn enhances the throughput of the network.

Wei et al. [90] propose a P2P Content-propagation Mechanism that utilizes Network Coding (PCMNC) based on MGM. PCMNC is similar to PPFEED, whence it uses network combination topology and selects the coefficients deterministically. On the other hand, it differs from PPFEED by applying MGM coding rather than generation coding. The authors argue that PCMNC can achieve the maximum network coding gain by suggesting optimal size of the mixing set, $m = 3$. Additionally, PCMNC is superior in performance to PPFEED by allowing more piece diversity and thus lower chances of decoding failure. Empirical evaluation shows that PCMNC can shorten the download time by 26% and 13% when compared to BitTorrent and P2P randomized-generation coding, respectively. In literature, this is the only study that has been found that applies MGM to P2P content distribution networks.

Silva et al. [91] propose the first overlapping coding scheme by introducing both square and diagonal grid structures. In the square grid structure, shown in Figure 8a, base generations are horizontally arranged while overlapping (supportors) generations are vertically stacked in such a way that each overlapping generation overlaps with all base generations by exactly one piece per generation. In the diagonal grid structure, shown in Figure 8b, supporter generations are diagonally stacked with an angle θ. The authors aim to mix the benefits of network coding and fountain codes [92,93] such as Luby Transform (LT) code [94] and Raptor [95]. Indeed, a fountain decoder is a propagative decoder that has the feature of achieving a small overhead with a low decoding complexity by allowing back substitution. The authors mainly try to answer the following question: is it possible to use true network coding and at the same time enjoy a fountain-like decoder? A practical simulation shows improved performance over independent generation coding in terms of decoding complexity overhead trade-off. Exactly, the results show that for fixed expected complexity, the overlapped generations codes can reduce the overhead by up to 70%. However, the theoretical analysis is done only for short lengths equal to four groups.

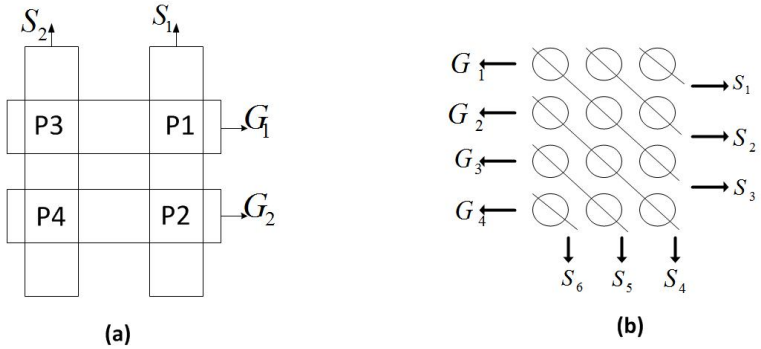

Figure 8. (**a**) square grid overlapping structure, (**b**) diagonal grid overlapping structure.

Heidarzadeh et al. [96] propose head-to-toe overlapping that allows a generation of k pieces to be overlapped with its following neighbor in at least one piece and at most $k-1$ pieces. The overlapping is allowed between neighboring generations only. More theoretical analysis than [91] is done by measuring the asymptotic performance over a line network, in which a single source is connected to a single sink via intermediate nodes such that each node is connected to exactly one sender and one receiver forming a unidirectional-line network that contains n nodes and $n-1$ links. Performance comparison of full codes, disjointed generations codes, and overlapped generation codes is conducted. Results show that the overlapping network coding can exploit the network bandwidth optimally

by reducing the network overhead by nearly 60% and 29% for a line network of lengths 1 and 2, respectively. In addition, they suggest using overlapped chunked codes for multimedia transmission as shown in [97].

Li et al. [98] propose random annex codes which allow generations to be overlapped by attaching to each generation a random annex of pieces selected randomly from other generations such that the annex size is always smaller than the base generation size. Theoretical analysis and simulations show that the optimal overlap (annex) size that increases the throughput and decreases the decoding latency is around the half of the generation size. Moreover, results show that the proposed scheme outperforms the head-to-toe scheme slightly in terms of throughput and the probability of decoding failure. Both schemes outperform the disjoint generations' coding scheme.

Tang et al. [99] propose an expander graph based overlapped generation codes (EOC) which models generations as vertices and overlapping as edges between the vertices. EOC allows each generation to overlap with another generation in one distinct piece only. The authors suggest that the number, d, of generations that overlap vary from a minimum of 3 to a maximum that does not exceed the generation size. Figure 9 shows an example of EOC graph with $d = 3$ and 5 generations, each of seven pieces (four distinct and three overlapped), i.e., generation one's (G_1) pieces are $P_1, P_5, P_6, P_8, P_9, P_10, P_11$. Simulation results show that EOC can achieve throughput rate of 93% while independent generations codes can achieve maximum rate of 73% when the generation size = 32 and GF size = 16. Additionally, the EOC receiver can decode all the pieces and reconstruct the file faster than random annex and head-to-toe schemes. The extra ratio of encoded pieces needed for decoding all the generations are $\leq$ 10%, 20%, and 40% for EOC, random annex, and head-to-toe, respectively.

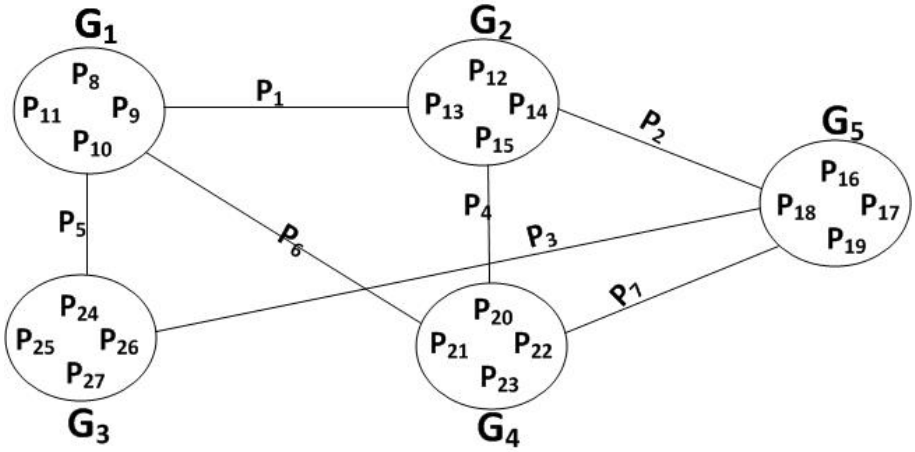

Figure 9. a d-regular graph ($d = 3$) with 5-vertices (generations).

Joshi et al. [100] proposed overlapping structure similar to EOC in which allowing overlapping between two generations in one piece only. However, in contrast to EOC, they consider deterministic overlapping structures by arranging generations circularly and allowing a generation to overlap with its successive generations in a modular arithmetic scheme. Another improved deterministic circular structure called distributed is proposed which allows overlapping based on a distance. Additionally, in opposition to the foregoing studies which consider a random scheduling, round-robin scheduling is considered and compared with the random scheduling. Analysis shows that round-robin scheduling can accelerate the download time significantly for small generation sizes ($\leq$8). Simulation results show that the proposed deterministic overlap structures minimize the expected download time compared to the random annex, and the distributed structure has almost the same download time as EOC but with less network overhead and indexing complexity of EOC.

In contrast to the aforementioned schemes, Li et al. [101] propose an overlapping network coding with unequal generation sizes. Sizes vary from a minimum, which achieves the maximum information flow of the network, to a maximum that is constrained by the buffer size or by the decoding complexity, namely, Gaussian elimination can be applied computationally. Pieces per generation are selected randomly from all the original pieces with replacement such that overlapping among generation is allowed. Sequential and random scheduling policies are merged and applied. For each generation, a fixed number of pieces are transmitted sequentially such that small generations can be decoded and subsequently help large generations for earlier decodability. Next, random scheduling is applied among generations for the remaining pieces. Experimental results show that the proposed scheme achieves the minimum overhead (10%) compared to disjointed generations (60%), head-to-toe (40%), and random annex (20%) codes. However, the disjointed generations code achieves the minimum decoding complexity. The overall better overhead-complexity trade-off is achieved by the proposed scheme.

The previous studies show clearly the benefit of overlapping coding over disjoint generation, in terms of downloading time and decoding rate, by mathematical analysis and simulation. However, only two studies [91,100] consider P2P networks. Table 6 summarizes the most important studies related to generations overlapping network coding.

Table 6. Overlapping network coding approaches and their characteristics.

Reference	Approach	Main Findings	Verification Method	Network Topology	Performance Metrics
Silva et al. [91]	Proposing first P2P overlapping coding scheme by introducing grid and diagonal structures.	- The proposed scheme can mainly reduce network overhead by up to 70%.	Simulation	Mesh	- Decoding complexity overhead trade-off.
Heidarzadeh et al. [96]	Proposing head-to-toe overlapping scheme over line network topology	- Overlapping can reduce network overhead for line networks. - Overlapping is appropriate for multimedia streaming.	Numerical Simulation	Line networks overlay	- Decoding probability.
Li et al. [98]	Proposing random overlapping by attaching an annex to the base generation.	- The optimal overlap size to achieve the highest throughput is around the half of the generation size. - Random-Annex coding outperforms both head-to-toe and disjoint generations coding.	Numerical Simulation	Point-to-point	- Probability of Decoding failure. - Throughput.
Tang et al. [99]	Proposing and modeling generations overlapping as an expander graph (EOC).	- The best number of generations to overlapped varies between 3 to the generation size. - EOC decoder decodes faster than random annex and head-to-toe.	Numerical simulation	Point-to-point	- Throughput. - Decoding rate.
Joshi et al. [100]	Proposing deterministic structures of overlapping and consider round-robin scheduling	- The upper bound limit of the optimal overlapping is $O(logn)$. - Deterministic overlapping structures avoid network overhead and index complexity of random structure (EOC). - Proposed schemes minimize the expected download time compared to random annex.	Numerical Simulation	Mesh	-Download time.
Li et al. [101]	Proposing overlapping with unequal generations' sizes based on degree distribution and merge both sequential and random scheduling	- Proposed scheme achieves the minimum overhead whereas independent generations codes achieve the minimum decoding complexity. - Among the overlapping schemes, the best overhead-complexity trade-off is achieved by the proposed scheme.	Numerical Simulation	Point-to-point	- Decoding rate. - Overhead.

5. Discussion

In this section, we discuss the approaches that are surveyed in the previous section. Avalanche is the first network coding based P2P file sharing system that improves in terms of communication but adds huge overhead in terms of computation. All subsequent studies attempt to decrease the computation overhead. In a sparse network coding scheme, the studies suggest that the network coding may be applied into some pieces rather than be applied on all the file pieces. However, whether these pieces should be selected randomly or deterministically is not clear. Furthermore, whether two original pieces or more are sufficient for encoding is not verified.

In generation based scheme, the studies agree that the network coding should be applied per generation rather than on all file data. However, the optimal size of a generation and subsequently to how many generations the file should be partitioned is not agreed upon. The arguments that generation coding outperforms sparse coding [62,67] are not supported. Moreover, there is a conflict about the scheduling policy among generations: random, sequential, round-robin, or rarest-first. In terms of network overhead, disjointed generation coding incurs additional overhead that is each encoded piece is sent with the generation sequence it belongs to. On the other hand, overlapping generations coding adds even more additional overhead, which is the sequences of the overlapping generations.

Alternatively, the combined network coding scheme attempts to mix the features of generation coding and sparse coding and get rid of their limitations. The studies for this scheme are limited and do not agree upon the optimal generation size and encoding size.

Based on the extensive survey, the discussion, and our partial simulation, Table 7 compares the coding approaches in terms of robustness to churn, decoding speed, and network overhead. More stars "⋆" means greater robustness to churn, whereas the more arrows ↑ means greater network overhead. The comparison is approximated, and we believe that, for more precise comparison, specific variations of each schemes should be simulated.

Table 7. Comparison of network coding approaches.

	Robustness to Churn	**Decoding Speed**	**Network Overhead**
Full Coding	⋆⋆⋆⋆⋆	Constant: very slow (infeasible)	↑
Sparse Coding	⋆⋆	Variant: slow at the beginning to quick at the end.	↑
Generation Coding	⋆⋆⋆	Constant: based on generation size (moderate usually).	↑↑
Combined Coding	⋆⋆⋆	Variant: moderate at the beginning to quick at the end.	↑↑
Overlapping Coding	⋆⋆⋆⋆	Variant: moderate at the beginning to very quick at the end.	↑↑↑

After this discussion, it can be stated that using network coding in P2P file sharing networks is a trade-off process, a trade-off between communication and computation, a trade-off between computation and content availability, and a trade-off between the size of the finite field and the linear dependency of the encoded packets. The studies agree in certain aspects. For example, the studies suggest the use of generation coding or sparse coding algorithms and their variations rather than the use of full coding. However, they conflict in other aspects such as determining the optimal generation size or determining the encoding size in the case of sparse coding. Furthermore, the studies show a solution for special topologies with special characteristics, while they do not propose a general framework.

6. Conclusions

In this paper, we survey network coding based P2P file sharing systems. This survey paper summarizes, assesses, and compares the most recently used techniques to improve P2P content distribution systems performance using network coding. Furthermore, we present a classification of the main network coding based P2P file sharing systems which include full coding, generation coding, sparse coding, combined coding, and overlapping coding. The extensive survey shows that applying network coding on P2P file sharing systems is a trade-off process. The challenges of network coding can be alleviated, but not completely vanished. These challenges are still a hot research topic.

Author Contributions: Conceptualization, A.A.A. and A.M.; Funding acquisition, A.M., M.A.-A., and T.S.; Methodology A.A.A. and A.M.; Supervision A.M. and T.S.; Writing—Original draft, A.A.A.; Writing—Review and editing, A.M. and M.A.-A. All authors have read and agreed to the published version of the manuscript.

Funding: The APC was funded by King Fahd University of Petroleum and Minerals (KFUPM).

Acknowledgments: The authors would like to acknowledge the support provided by King Fahd University of Petroleum and Minerals (KFUPM) and the Department of Computer Engineering.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Stoica, I.; Morris, R.; Liben-Nowell, D.; Karger, D.R.; Kaashoek, M.F.; Dabek, F.; Balakrishnan, H. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw. (TON)* **2003**, *11*, 17–32. [CrossRef]
2. Nwebonyi, F.N.; Martins, R.; Correia, M.E. Reputation based approach for improved fairness and robustness in P2P protocols. *Peer- Netw. Appl.* **2019**, *12*, 951–968. [CrossRef]
3. Pacifici, V.; Lehrieder, F.; Dán, G. Cache bandwidth allocation for P2P file-sharing systems to minimize inter-ISP traffic. *IEEE/ACM Trans. Netw. (TON)* **2016**, *24*, 437–448. [CrossRef]
4. Cohen, B. The BitTorrent Protocol Specification. 2008. Available online: http://bittorrent.org/beps/bep_0003.html (access on 24 March 2020).
5. Lareida, A.; Bocek, T.; Waldburger, M.; Stiller, B. RB-tracker: A fully distributed, replicating, network-, and topology-aware P2P CDN. In Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), Ghent, Belgium, 27–31 May 2013; pp. 1199–1202.
6. Wu, D.; Dhungel, P.; Hei, X.; Zhang, C.; Ross, K.W. Understanding peer exchange in bittorrent systems. In Proceedings of the IEEE Tenth International Conference on Peer-to-Peer Computing (P2P), Delft, The Netherlands, 25–27 August 2010; pp. 1–8.
7. Hecht, F.V.; Bocek, T.; Stiller, B. B-tracker: Improving load balancing and efficiency in distributed p2p trackers. In Proceedings of the IEEE International Conference on Peer-to-Peer Computing, Kyoto, Japan, 31 August–2 September 2011; pp. 310–313.
8. Neglia, G.; Reina, G.; Zhang, H.; Towsley, D.; Venkataramani, A.; Danaher, J. Availability in bittorrent systems. In Proceedings of the IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications, Barcelona, Spain, 6–12 May 2007; pp. 2216–2224.
9. Fry, C.P.; Reiter, M.K. *Really Truly Trackerless BitTorrent*; School of Computer Science, Carnegie Mellon University, Tech. Rep: Pittsburgh, PA, USA, 2006; pp. 6–148.
10. Navimipour, N.J.; Milani, F.S. A comprehensive study of the resource discovery techniques in Peer-to-Peer networks. *Peer- Netw. Appl.* **2015**, *8*, 474–492. [CrossRef]
11. Luo, J.; Xiao, B.; Bu, K.; Zhou, S. Understanding and improving piece-related algorithms in the BitTorrent protocol. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *24*, 2526–2537. [CrossRef]
12. Hu, Y.; Dong, D.; Li, J.; Wu, F. Efficient and incentive-compatible resource allocation mechanism for P2P-assisted content delivery systems. *Future Gener. Comput. Syst.* **2013**, *29*, 1611–1620. [CrossRef]
13. Chiu, D.M.; Yeung, R.W.; Huang, J.; Fan, B. Can network coding help in P2P networks? In Proceedings of the 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, Boston, MA, USA, 26 February–2 March 2006; pp. 1–5.
14. Chiang, J.L.; Tseng, Y.Y.; Chen, W.T. Interest-Intended Piece Selection in BitTorrent-like peer-to-peer file sharing systems. *J. Parallel Distrib. Comput.* **2011**, *71*, 879–888. [CrossRef]
15. Fragouli, C.; Le Boudec, J.Y.; Widmer, J. Network coding: An instant primer. *ACM SIGCOMM Comput. Commun. Rev.* **2006**, *36*, 63–68. [CrossRef]
16. Terelius, H.; Johansson, K.H. Peer-to-peer gradient topologies in networks with churn. *IEEE Trans. Control Netw. Syst.* **2018**, *5*, 2085–2095. [CrossRef]
17. Feng, C.; Li, B. Network coding for content distribution and multimedia streaming in peer-to-peer networks. In *Network Coding*; Elsevier: Amsterdam, The Netherlands, 2012; pp. 61–86.
18. Cai, Q.C.; Lo, K.T. Two blocks are enough: On the feasibility of using network coding to ameliorate the content availability of bittorrent swarms. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *24*, 1682–1694. [CrossRef]
19. Huang, K.; Wang, L.; Zhang, D.; Liu, Y. Optimizing the BitTorrent performance using an adaptive peer selection strategy. *Future Gener. Comput. Syst.* **2008**, *24*, 621–630. [CrossRef]

20. Ahlswede, R.; Cai, N.; Li, S.Y.; Yeung, R.W. Network information flow. *IEEE Trans. Inf. Theory* **2000**, *46*, 1204–1216. [CrossRef]
21. Katti, S.; Rahul, H.; Hu, W.; Katabi, D.; Médard, M.; Crowcroft, J. XORs in the air: Practical wireless network coding. In *ACM SIGCOMM Computer Communication Review*; ACM: New York, NY, USA, 2006; Volume 36, pp. 243–254.
22. Chen, Y.J.; Wang, L.C.; Wang, K.; Ho, W.L. Topology-aware network coding for wireless multicast. *IEEE Syst. J.* **2018**, *12*, 3683–3692. [CrossRef]
23. Jiang, D.; Xu, Z.; Li, W.; Chen, Z. Network coding-based energy-efficient multicast routing algorithm for multi-hop wireless networks. *J. Syst. Softw.* **2015**, *104*, 152–165. [CrossRef]
24. Czap, L.; Fragouli, C.; Prabhakaran, V.M.; Diggavi, S. Secure network coding with erasures and feedback. *IEEE Trans. Inf. Theory* **2015**, *61*, 1667–1686. [CrossRef]
25. Matsumoto, R.; Hayashi, M. Universal secure multiplex network coding with dependent and non-uniform messages. *IEEE Trans. Inf. Theory* **2017**, *63*, 3773–3782. [CrossRef]
26. He, H.; Li, R.; Xu, Z.; Xiao, W. An efficient ECC-based mechanism for securing network coding-based P2P content distribution. *Peer- Netw. Appl.* **2014**, *7*, 572–589. [CrossRef]
27. He, M.; Gong, Z.; Chen, L.; Wang, H.; Dai, F.; Liu, Z. Securing network coding against pollution attacks in p2p converged ubiquitous networks. *Peer- Netw. Appl.* **2015**, *8*, 642–650. [CrossRef]
28. Xie, D.; Peng, H.; Li, L.; Yang, Y. An efficient privacy-preserving scheme for secure network coding based on compressed sensing. *AEU-Int. J. Electron. Commun.* **2017**, *79*, 33–42. [CrossRef]
29. Li, T.; Chen, W.; Tang, Y.; Yan, H. A homomorphic network coding signature scheme for multiple sources and its application in IoT. *Secur. Commun. Netw.* **2018**, *2018*, 6. [CrossRef]
30. Ho, T.; Médard, M.; Koetter, R.; Karger, D.R.; Effros, M.; Shi, J.; Leong, B. A random linear network coding approach to multicast. *IEEE Trans. Inf. Theory* **2006**, *52*, 4413–4430. [CrossRef]
31. Li, S.Y.; Yeung, R.W.; Cai, N. Linear network coding. *IEEE Trans. Inf. Theory* **2003**, *49*, 371–381. [CrossRef]
32. Etzion, T.; Wachter-Zeh, A. Vector network coding based on subspace codes outperforms scalar linear network coding. *IEEE Trans. Inf. Theory* **2018**, *64*, 2460–2473. [CrossRef]
33. Sun, Q.T.; Yang, X.; Long, K.; Yin, X.; Li, Z. On vector linear solvability of multicast networks. *IEEE Trans. Commun.* **2016**, *64*, 5096–5107. [CrossRef]
34. Ebrahimi, J.; Fragouli, C. Algebraic algorithms for vector network coding. *IEEE Trans. Inf. Theory* **2011**, *57*, 996–1007. [CrossRef]
35. Kafaie, S.; Chen, Y.P.; Dobre, O.A.; Ahmed, M.H. Network coding implementation details: A guidance document. *arXiv* **2018**, arXiv:1801.02120.
36. Keller, L. Network Coding Utilities. Available online: https://github.com/lokeller/ncutils (access on 24 March 2020).
37. Matsuda, T.; Noguchi, T.; Takine, T. Survey of network coding and its applications. *IEICE Trans. Commun.* **2011**, *94*, 698–717. [CrossRef]
38. Li, B.; Niu, D. Random network coding in peer-to-peer networks: From theory to practice. *Proc. IEEE* **2011**, *99*, 513–523.
39. Parikh, V.U.; Narmawala, Z. A survey on peer-to-peer file sharing using network coding in delay tolerant networks. *Int. J. Comput. Sci. Commun.* **2014**, *5*, 74–79.
40. Bassoli, R.; Marques, H.; Rodriguez, J.; Shum, K.W.; Tafazolli, R. Network coding theory: A survey. *IEEE Commun. Surv. Tutorials* **2013**, *15*, 1950–1978. [CrossRef]
41. Sanna, M.; Izquierdo, E. A survey of linear network coding and network error correction code constructions and algorithms. *Int. J. Digit. Multimed. Broadcast.* **2011**, *2011*, 12. [CrossRef]
42. Gkantsidis, C.; Rodriguez, P.R. Network coding for large scale content distribution. In Proceedings of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Miami, FL, USA, 13–17 March 2005; Volume 4, pp. 2235–2245.
43. Luby, M.; Vicisano, L.; Gemmell, J.; Rizzo, L.; Handley, M.; Crowcroft, J. The Use of Forward Error Correction (FEC) in Reliable Multicast; Technical Report, RFC 3453, December 2002. Available online: https://dl.acm.org/doi/book/10.17487/RFC3453 (accessed on 24 March 2020).
44. Cohen, B. Incentives build robustness in BitTorrent. In Proceedings of the Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA, 5–6 June 2003; Volume 6, pp. 68–72.

45. Azzedin, F.; Yahaya, M. Modeling BitTorrent choking algorithm using game theory. *Future Gener. Comput. Syst.* **2016**, *55*, 255–265. [CrossRef]
46. Medard, M. How good is random linear coding based distributed networked storage? In Proceedings of the NETCOD'05, Riva del Garda, Italy, 7 April 2005.
47. Cassuto, Y.; Shokrollahi, A. Online fountain codes with low overhead. *IEEE Trans. Inf. Theory* **2015**, *61*, 3137–3149. [CrossRef]
48. Deb, S.; Médard, M.; Choute, C. Algebraic gossip: A network coding approach to optimal multiple rumor mongering. *IEEE/ACM Trans. Netw. (TON)* **2006**, *14*, 2486–2507. [CrossRef]
49. Karp, R.; Schindelhauer, C.; Shenker, S.; Vocking, B. Randomized rumor spreading. In Proceedings of the 41st Annual Symposium on Foundations of Computer Science, Redondo Beach, CA, USA, 12–14 November 2000; pp. 565–574.
50. Doerr, B.; Doerr, C.; Moran, S.; Moran, S. Simple and optimal randomized fault-tolerant rumor spreading. *Distrib. Comput.* **2016**, *29*, 89–104. [CrossRef]
51. Wang, N.; Ansari, N. Downloader-initiated random linear network coding for peer-to-peer file sharing. *IEEE Syst. J.* **2010**, *5*, 61–69. [CrossRef]
52. Yeung, R.W. Avalanche: A network coding analysis. *Commun. Inf. Syst.* **2007**, *7*, 353–358. [CrossRef]
53. Soro, A.; Lacan, J. FFT-based Network Coding For Peer-To-Peer Content Delivery. In Proceedings of the 2011 IEEE Global Telecommunications Conference-GLOBECOM 2011, Houston, TX, USA, 5–9 December 2011; pp. 1–5.
54. Wang, M.; Li, B. How practical is network coding? In Proceedings of the 14th IEEE International Workshop on Quality of Service, New Haven, CT, USA, 19–21 June 2006; pp. 274–278.
55. Li, T.; Wang, J.; You, J. Using Degree-Based Strategy for Network Coding in Content Distribution Network. In Proceedings of the International Conference on Computer and Electrical Engineering, Phuket, Thailand, 20–22 December 2008; pp. 487–491.
56. Ma, G.; Xu, Y.; Lin, M.; Xuan, Y. A content distribution system based on sparse linear network coding. In Proceedings of the Third Workshop on Network Coding (Netcod 2007), Miami, FL, USA, 29 July 2007.
57. Ortolf, C.; Schindelhauer, C.; Vater, A. Paircoding: Improving file sharing using sparse network codes. In Proceedings of the 2009 Fourth International Conference on Internet and Web Applications and Services, Venice, Italy, 24–28 May 2009; pp. 49–57.
58. Vater, A. Efficient Coding Schemes for File Sharing Networks. Ph.D. Thesis, Albert Ludwig University, Breisgau, Germany, April 2011.
59. Vater, A.; Schindelhauer, C.; Ortolf, C. Tree network coding for peer-to-peer networks. In Proceedings of the Twenty-Second Annual ACM Symposium on Parallelism in Algorithms and Architectures, Santorini, Greece, 13–15 June 2010; pp. 114–123.
60. Ortolf, C.; Schindelhauer, C.; Vater, A. Classifying peer-to-peer network coding schemes. In Proceedings of the Twenty-First, Annual Symposium on Parallelism in Algorithms and Architectures, Calgary, AB, Canada, 11–13 August 2009; pp. 310–318.
61. Shang, T.; Peng, T.; Lei, Q.; Liu, J. Homomorphic Signature for Generation-based Network Coding. In Proceedings of the IEEE International Conference on Smart Cloud (SmartCloud), New York, NY, USA, 18–20 November 2016; pp. 269–273.
62. Xu, J.; Wang, X.; Zhao, J.; Lim, A.O. I-swifter: Improving chunked network coding for peer-to-peer content distribution. *Peer- Netw. Appl.* **2012**, *5*, 30–39. [CrossRef]
63. Ren, L.Y.; He, H.Y.; Di, Z.; Lei, M. Content distribution system based on segmented network coding. In Proceedings of the International Conference on Apperceiving Computing and Intelligence Analysis, Chengdu, China, 23–25 October 2009; pp. 258–261.
64. Tao, S.; Huang, J.; Yang, Z.; Cheng, W.; Liu, W. An improved network coding-based cooperative content distribution scheme. In Proceedings of the ICC Workshops-2008 IEEE International Conference on Communications Workshops, Beijing, China, 19–23 May 2008; pp. 360–364.
65. Chou, P.A.; Wu, Y.; Jain, K. Practical network coding. In Proceedings of the Annual Allerton Conference on Communication Control and Computing, Allerton, IL, USA, 1–3 October 2003; Volume 41, pp. 40–49.
66. Maymounkov, P.; Harvey, N.J.; Lun, D.S. Methods for efficient network coding. In Proceedings of the 44th Annual Allerton Conference on Communication, Control, and Computing, Allerton, IL, USA, 27–29 September 2006; pp. 482–491.

67. Xu, J.; Zhao, J.; Wang, X.; Xue, X. Swifter: Chunked network coding for peer-to-peer content distribution. In Proceedings of the IEEE International Conference on Communications, Beijing, China, 19–23 May 2008; pp. 5603–5608.
68. Hundeboll, M.; Ledet-Pedersen, J.; Sluyterman, G.; Madsen, T.K.; Fitzek, F.H. Peer-assisted content distribution with random linear network coding. In Proceedings of the IEEE 79th Vehicular Technology Conference (VTC Spring), Seoul, Korea, 18–21 May 2014; pp. 1–6.
69. Niu, D.; Li, B. On the resilience-complexity trade-off of network coding in dynamic P2P networks. In Proceedings of the Fifteenth IEEE International Workshop on Quality of Service, Evanston, IL, USA, 21–22 June 2007; pp. 38–46.
70. Niu, D.; Li, B. Analyzing the resilience-complexity trade-off of network coding in dynamic P2P networks. *IEEE Trans. Parallel Distrib. Syst.* **2011**, *22*, 1842–1850. [CrossRef]
71. Zhang, X.; Li, B. On the market power of network coding in P2P content distribution systems. In Proceedings of the IEEE INFOCOM, Rio de Janeiro, Brazil, 19–25 April 2009; pp. 334–342.
72. Zhang, X.; Li, B. On the Market Power of Network Coding in P2P Content Distribution Systems. *IEEE Trans. Parallel Distrib. Syst.* **2011**, 2063–2070. [CrossRef]
73. Leu, J.S.; Yu, M.C.; Yueh, H.C. Improving network coding based file sharing for unstructured peer-to-peer networks. *J. Netw. Syst. Manag.* **2015**, *23*, 803–829. [CrossRef]
74. Yang, M.; Yang, Y. Applying network coding to peer-to-peer file sharing. *IEEE Trans. Comput.* **2013**, *63*, 1938–1950. [CrossRef]
75. Li, Z.; Li, B.; Jiang, D.; Lau, L.C. On Achieving Optimal End-to-End Throughput in Data Networks: Theoretical and Empirical Studies; ECE Technical Report; 2004. Available online: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.4.6912 (accessed on 24 March 2020).
76. Ngai, C.K.; Yeung, R.W. Network coding gain of combination networks. In Proceedings of the IEEE Information Theory Workshop, San Antonio, TX, USA, 24–29 October 2004; pp. 283–287.
77. Jaggi, S.; Sanders, P.; Chou, P.A.; Effros, M.; Egner, S.; Jain, K.; Tolhuizen, L.M. Polynomial time algorithms for multicast network code construction. *IEEE Trans. Inf. Theory* **2005**, *51*, 1973–1982. [CrossRef]
78. Chu, Y.h.; Rao, S.G.; Seshan, S.; Zhang, H. A case for end system multicast. *IEEE J. Sel. Areas Commun.* **2002**, *20*, 1456–1471. [CrossRef]
79. Braun, P.J.; Sipos, M.; Ekler, P.; Charaf, H. Increasing data distribution in BitTorrent networks by using network coding techniques. In Proceedings of the VDE European Wireless 2015, 21th European Wireless Conference, Budapest, Hungary, 20–22 May 2015; pp. 1–6.
80. Baumgart, I.; Heep, B.; Krause, S. OverSim: A flexible overlay network simulation framework. In Proceedings of the IEEE Global Internet Symposium, Anchorage, AK, USA, 11 May 2007; pp. 79–84.
81. Chawathe, Y.; Ratnasamy, S.; Breslau, L.; Lanham, N.; Shenker, S. Making gnutella-like p2p systems scalable. In Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Karlsruhe, Germany, 25–29 August 2003; pp. 407–418.
82. Zeng, H.; Huang, J.; Tao, S.; Cheng, W. A simulation study on network coding parameters in P2P content distribution system. In Proceedings of the Third International Conference on Communications and Networking in China, Hangzhou, China, 25–27 August 2008; pp. 197–201.
83. Zhang, X.; Liu, J.; Li, B.; Yum, Y.S. CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming. In Proceedings of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Miami, FL, USA, 13–17 March 2005; Volume 3, pp. 2102–2111.
84. Kaiqian, O.; Yinlong, X.; Guanjun, M.; Yulin, Z. Dasher: A peer-to-peer content distribution system based on combined network coding. In Proceedings of the 2nd IEEE International Conference on Broadband Network & Multimedia Technology, Beijing, China, 18–20 Ocotber 2009; pp. 687–692.
85. Su, J.; Deng, Q.; Long, D. PCLNC: A low-cost intra-generation network coding strategy for P2P content distribution. *Peer- Netw. Appl.* **2019**, *12*, 177–188. [CrossRef]
86. Chun, B.; Culler, D.; Roscoe, T.; Bavier, A.; Peterson, L.; Wawrzoniak, M.; Bowman, M. Planetlab: An overlay testbed for broad-coverage services. *ACM SIGCOMM Comput. Commun. Rev.* **2003**, *33*, 3–12. [CrossRef]
87. Montresor, A.; Jelasity, M. PeerSim: A scalable P2P simulator. In Proceedings of the IEEE Ninth International Conference on Peer-to-Peer Computing, Seattle, WA, USA, 9–11 September 2009; pp. 99–100.
88. Halloush, M.; Radha, H. Network coding with multi-generation mixing. In Proceedings of the 42nd Annual Conference on Information Sciences and Systems, Princeton, NJ, USA, 19–21 March 2008; pp. 515–520.

89. Halloush, M.; Radha, H. Network coding with multi-generation mixing: A generalized framework for practical network coding. *IEEE Trans. Wirel. Commun.* **2010**, *10*, 466–473. [CrossRef]
90. Wei, X.; Long, D.Y. P2P content-propagation mechanism tailored by network coding. In Proceedings of the International Symposium on Computer Network and Multimedia Technology, Wuhan, China, 18–20 January 2009; pp. 1–6.
91. Silva, D.; Zeng, W.; Kschischang, F.R. Sparse network coding with overlapping classes. In Proceedings of the Workshop on Network Coding, Theory, and Applications, Lausanne, Switzerland, 15–16 June 2009; pp. 74–79.
92. MacKay, D.J. Fountain codes. *IEE Proc.-Commun.* **2005**, *152*, 1062–1068. [CrossRef]
93. Byers, J.W.; Luby, M.; Mitzenmacher, M.; Rege, A. A digital fountain approach to reliable distribution of bulk data. *ACM SIGCOMM Comput. Commun. Rev.* **1998**, *28*, 56–67. [CrossRef]
94. Luby, M. LT codes. In Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, Chicago, IL, USA, 27 June–2 July 2002; pp. 271–280.
95. Shokrollahi, A. Raptor codes. *IEEE/ACM Trans. Netw. (TON)* **2006**, *14*, 2551–2567. [CrossRef]
96. Heidarzadeh, A.; Banihashemi, A.H. Overlapped chunked network coding. In Proceedings of the IEEE Information Theory Workshop on Information Theory (ITW), Cairo, Egypt, 6–8 January 2010; pp. 1–5.
97. Heidarzadeh, A.; Banihashemi, A.H. Analysis of overlapped chunked codes with small chunks over line networks. In Proceedings of the IEEE International Symposium on Information Theory Proceedings, St. Petersburg, Russia, 31 July–5 August 2011; pp. 801–805.
98. Li, Y.; Soljanin, E.; Spasojevic, P. Effects of the generation size and overlap on throughput and complexity in randomized linear network coding. *IEEE Trans. Inf. Theory* **2011**, *57*, 1111–1123. [CrossRef]
99. Tang, B.; Yang, S.; Yin, Y.; Ye, B.; Lu, S. Expander graph based overlapped chunked codes. In Proceedings of the IEEE International Symposium on Information Theory Proceedings, Cambridge, MA, USA, 1–6 July 2012; pp. 2451–2455.
100. Joshi, G.; Soljanin, E. Round-robin overlapping generations coding for fast content download. In Proceedings of the IEEE International Symposium on Information Theory, Istanbul, Turkey, 7–12 July 2013; pp. 2740–2744.
101. Li, Y.; Chan, W.Y.; Blostein, S.D. Network coding with unequal size overlapping generations. In Proceedings of the International Symposium on Network Coding (NetCod), Cambridge, MA, USA, 29–30 June 2012; pp. 161–166.