

Coding Based Peer-to-Peer Content Distribution Network

BY

Anas Ahmad Abu-Daqa

A Dissertation Presented to the
FACULTY OF THE COLLEGE OF GRADUATE STUDIES
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

In

Computer Engineering (Networking)

April 2021

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This dissertation, written by **ANAS AHMAD ABU-DAQA** under the direction of his dissertation adviser and approved by his dissertation committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY IN COMPUTER ENGINEERING (NETWORKING)**.

Dissertation Committee



Dr. Ahsraf Mahmoud (Adviser)

Marwan Abu-Amara 20/04/2021

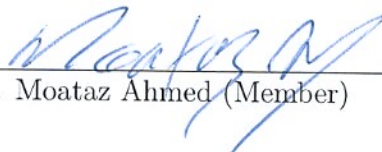
Dr. Marwan Abu-Amara (Co-Adviser)



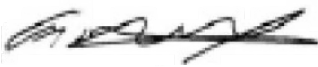
Dr. Tarek Rahil Sheltami (Member)



Dr. Aiman El-Maleh (Member)



Dr. Moataz Ahmed (Member)



Dr. Aiman El-Maleh
Department Chairman



Dr. Suliman S. Al-Homidan
Dean of Graduate Studies

28/4/2021

Date



©Anas Ahmad Abu-Daqa
2021

Dedication To my father, the soul of my mother, brothers, and sisters

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Ashraf Mahmoud, for his patience, comments and help throughout the work of my PHD Thesis. I would also like to thank my co-advisor, Dr. Marwan Abu Amara for his support and help. Besides, I would like to thank the rest of my thesis committee; Dr. Tarek Sheltami, Dr. Aiman Almalih, and Dr. Moataz Ahmed. My gratitude goes to for the support extended towards my research and for granting me the opportunity to pursue graduate studies, special thanks are given to the Department of Computer Engineering.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	vi
LIST OF TABLES	x
LIST OF FIGURES	xi
LIST OF ABBREVIATIONS	xiv
ABSTRACT (ENGLISH)	xvi
ABSTRACT (ARABIC)	xviii
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement	3
1.2 Dissertation Contribution	5
1.3 Dissertation Organization	6
1.4 List of Journal Publication	6
CHAPTER 2 BACKGROUND	8
2.1 BitTorrent Components	8
2.2 How BitTorrent-Like Systems Work	9
2.3 Network Coding Overview	11
2.4 The Butterfly Example	12
2.5 Detailed Network Coding Tutorial for Peer-to-Peer (P2P) File Shar- ing Systems	15

CHAPTER 3	RELATED WORK	21
3.1	Full Network Coding	21
3.2	Sparse Network Coding	27
3.3	Generation Based Network Coding	33
3.4	Combined Network Coding	40
3.5	Multi-Generation Mixing(MGM) and Overlapping Network Coding . .	43
CHAPTER 4	SUPER GENERATION NETWORK CODING	51
4.1	Problem Statement	51
4.2	Main Contributions	52
4.3	Preliminaries and Theoretical Analysis	53
4.3.1	Baseline BitTorrent Analysis	54
4.3.2	Analysis for Network Coding Based BitTorrent	56
4.3.3	Comparative Analysis	60
4.3.4	Best Generation Size	60
4.4	Super Generation Network Coding (SGNC)	61
4.4.1	SGNC Overhead	65
4.5	Experimental Work and Results	67
4.5.1	Content Availability (Robustness to Churn)	69
4.5.2	Overhead Due to Duplicate Pieces	69
4.5.3	Download Time	71
4.5.4	Decodability	73
4.5.5	Network coding coefficients overhead	77
4.6	Chapter Summary	79
CHAPTER 5	ONE ITEM NETWORK CODING VECTORS	81
5.1	Problem Statement	81
5.2	Methods in Literature	83
5.3	Chapter Contributions	86
5.4	Preliminaries and Design Model	87
5.4.1	Primarily step : generating the primes	89

5.4.2	Encoding	90
5.4.3	Decoding	91
5.4.4	Recoding	92
5.5	Theoretical Analysis and Proof of Correctness	94
5.5.1	Analysis of network coding coefficients overhead	94
5.5.2	Analysis of selecting different values from Allowed Selection Range (ASR)	98
5.5.3	Analysis of generating two identical coding vectors	101
5.5.4	Analysis of encoding cost	104
5.6	Experimental Work	107
5.6.1	Wireless Sensors Network Case Study	107
5.6.2	Ethernet Protocol Network	115
5.6.3	Large Scale P2P Content distribution network	123
5.7	Chapter Summary	124
CHAPTER 6 CONCLUSIONS AND FUTURE WORK		126
6.1	Conclusion	126
6.2	Limitations and Future Work	128
REFERENCES		129
VITAE		146

LIST OF TABLES

2.1	$GF(2^3)$ elements presentation.	16
3.1	Full network coding approaches and their characteristics.	26
3.2	Sparse network coding approaches and their characteristics.	32
3.3	Generation network coding approaches and their characteristics.	39
3.4	Combined network coding approaches and their characteristics.	43
3.5	Overlapping network coding approaches and their characteristics.	50
4.1	File decoding time based on dense network coding	58
5.1	Examples of GMSs	89
5.2	Encoding cost (secs) of a 128 bytes packet	105
5.3	Encoding cost (secs) of a 1500 bytes packet	106
5.4	Specifications for different Network Coding (NC) methods	109
5.5	New Specifications of One item network coding vectors (OINCV) and $OINCV_{opt}$	114
5.6	Specifications for different NC methods ($n = 256$)	117
5.7	Specifications for different NC methods ($n = 512$)	120
5.8	Coefficients overhead (MB) in P2P content distribution network.	124

LIST OF FIGURES

2.1	Process of new peer joining BitTorrent system.	9
2.2	P2P overlay network.	10
2.3	Illustration of piece distribution in baseline and network coding Bit-Torrent.	10
2.4	Butterfly network with traditional routing. (a) node B sends packet b; Then, (b) node B sends packet a [1].	13
2.5	Butterfly network using network coding [1].	14
2.6	File segmentation phases.	18
3.1	(a) File X with $n = 8$, (b) Tree coding of file X	31
3.2	Multiple-Generation-Mixing (MGM) coding illustration, E_x denotes encoded piece, and c_i denotes a coefficient.	44
3.3	(a) square grid overlapping structure, (b) diagonal grid overlapping structure.	46
3.4	a d -regular graph ($d = 3$) with 5-vertices (generations).	48
4.1	File decoding time based on dense network coding and 32KB piece size.	58
4.2	Theoretical analysis of the content availability ($N=100$, $w=10$, 20, and 50).	61
4.3	Best generation size for different piece sizes	62
4.4	Decodable piece's forms: (a) Lower triangular, (b) Diagonal, and (c) Random.	63
4.5	Normal generation vs. super generation	65

4.6	Content availability based on different scheduling policies:(a) Random, (b) Local rarest-first, and (c) Global rarest-first.	70
4.7	The redundancy overhead percentage based on different scheduling policies:(a) Random, (b) Local rarest-first, and (c) Global rarest-first. .	72
4.8	BitTorrent overhead based on different scheduling policies.	73
4.9	Download time of 320MB file	73
4.10	Download time of 640MB file	74
4.11	Download time of 1.2GB file	74
4.12	Download time of 5.1GB file	75
4.13	Decodability based on random scheduling policy	75
4.14	Decodability based on local rarest-first scheduling policy	76
4.15	Decodability based on global rarest-first scheduling policy	76
4.16	Network coding coefficients overhead percentage of 320MB file	77
4.17	Network coding coefficients overhead percentage of 640MB file	78
4.18	Network coding coefficients overhead percentage of 1.2GB file	78
4.19	Network coding coefficients overhead percentage of 5.1GB file	79
4.20	The overall overhead based on global-rarest first scheduling policy . . .	79
5.1	Two network coding packets: (a) RLNC's packet, and (b) OINCV's packet	91
5.2	Decoding process at the receiver side	92
5.3	(a) Recoding process at an intermediate node and (b) Decoding of a recoded packet at the receiver.	93
5.4	Log scale plot of network coding coefficients overhead without recoding	96
5.5	Log scale plot of network coding coefficients overhead with recoding . .	97
5.6	Probability that traversed Coefficients are distinguishable; theoretical vs simulation	99
5.7	Encoding cost comparison between Random linear network coding (RLNC) and OINCV	105
5.8	Performance metrics of 0-hops Wireless Sensors Network (WSN) for different NC methods	110

5.9	Performance metrics of 0-hops WSN; OINCV vs. $OINCV_{opt}$	112
5.10	Performance metrics of 0-hops WSN; OINCV vs. $OINCV_{opt}$ vs. $OINCV_{optPlus}$	113
5.11	2-hops WSN with two servers and one receiver.	114
5.12	Performance metrics of WSN for different network coding vectors meth- ods with recoding.	116
5.13	Network coding coefficients overhead with recoding; 0 to 12 hops and $\epsilon = 0$	117
5.14	Effect of sensitivity analysis on OINCV performance metrics.	118
5.15	Performance evaluation of Ethernet\IP network for different NC meth- ods, $n = 256$	119
5.16	Performance evaluation of Ethernet network for different NC methods, $n = 512$	121
5.17	Performance evaluation of Ethernet\IP network for multi OINCV vari- ations.	122
5.18	log scale of coefficients overhead in P2P content distribution network. .	124

LIST OF ABBREVIATIONS

AP	access point
ASR	Allowed Selection Range
CPU	central processing unit
CRT	Chinese remainder theorem
DNC	dense network coding
GB	giga bytes
GF	Galois Field
GMS	Generation Moduli Set
MB	mega bytes
MTU	maximum transmission unit
NC	Network Coding
NGNC	normal generation network coding
OGNC	overlapped generation network coding
OINCV	One item network coding vectors
P2P	Peer-to-Peer
RF	rarest first

RLNC	Random linear network coding
SGNC	Super generation network coding
WSN	Wireless Sensors Network

DISSERTATION ABSTRACT

NAME: Anas Ahmad Abu-Daqa
TITLE OF STUDY: Coding-Based Peer-to-Peer Content Distribution Network
MAJOR FIELD: Computer Engineering (Networking)
DATE OF DEGREE: December 2020

Peer-to-Peer (P2P) model has gained popularity as a promising technology for large-scale file sharing, a.k.a content distribution networks. Network Coding (NC) is known as a helpful method for increasing the content availability, accelerating the download process, and robustness against churn in such networks. Originally, dense network coding (DNC) has been proposed and theoretically considered as an optimal solution. However, due to its huge computational overhead, it is not viable for real-world systems. Subsequently, sparse, generations, and overlapped generations network coding schemes are proposed as possible alternatives but at reduced performance compared to that provided by the DNC.

This work is divided into two main parts. In the first part, an improved applicable network coding scheme for P2P content distribution systems referred to therein as Super generation network coding (SGNC) is proposed. SGNC maximizes the generation

size so that it is as close as possible to the optimal size without adding computational overhead. Theoretical analysis and experimental work show that SGNC outperforms classical and all previous coding based schemes for P2P content distribution systems in terms of content availability, download time, overhead, and decodability for all piece scheduling policies.

In the second part, the problem of network coding coefficients overhead is revisited. A novel approach based on modular arithmetic and prime numbers is proposed to reduce the coefficients overhead by augmenting only one item coefficient of size two or maximum four bytes within the payload packet. The proposed approach successfully addresses all the limitations of the previous methods including the limitations on the generation size and density of the packets within the generation, rocoding at the intermediate nodes, and creating innovative coding vectors. Probabilistic theoretical analysis addresses questions about the proposed method and proves its viability and reliability. The experimental work is conducted for three different networks classes, i.e., Wireless Sensors Network (WSN) , Ethernet-IP network with with 1500 packet size, and large scale P2P file sharing network. The results show that the proposed method far outperforms all the previous methods in terms of coding coefficients overhead, download time, throughput, number of dropped packets at the receiver side.

ملخص الرسالة

الاسم: أنس أحمد أبو دقة

عنوان الدراسة: تطبيق التجميع الهوائي المشفر على شبكات مشاركة الملفات بطريقة النظر للنظر

التخصص: قسم الحاسب الآلي

تاريخ الدرجة العلمية: ربيع الثاني 1442

تعتبر شبكات مشاركة الملفات باستخدام طريقة النظر للنظر هي الأكثر شهرة وانتشارا حول العالم. تُعرف طريقة التجميع الهوائي المشفر على أنها طريقة مساعدة في زيادة توفر بيانات الملفات وتسريع عملية التحميل وأخيرا الثبات والاستمرارية عند حدوث انقطاع في الاتصال ومغادرة مفاجئة لبعض النظراء. إلا أن طريقة التجميع الهوائي المشفر مع مميزاتها تعاني من ثقلها على معالج الحاسوب المركزي خاصة في عملية فك التشفير. لذلك تم اقتراح بعض الآليات التي تخفف من هذا الثقل ولكن على حساب جودة الكفاءة المتوقعة من عملية التجميع الهوائي المشفر. في هذا البحث قمنا بتطوير آلية تعتمد على طريقة التجميع الهوائي المشفر بحيث تحافظ على كفاءة عالية مع عدم زيادة الضغط على المعالج المركزي. التحليل النظري الرياضي والتجارب العملية أثبتت أن الآلية المقترحة تتفوق على جميع الآليات السابقة من حيث توفر بيانات الملفات، وسرعة التحميل، وسرعة فك التشفير. وأيضا قمنا بتطوير آلية تخفف من الأحمال الزائدة على شبكة الاتصال التي يسببها التجميع الهوائي المشفر. الآلية تعتمد على الحساب المعياري والأرقام الأولية. استطاعت الآلية المقترحة بدلا من إرسال صف كامل من المعاملات الرياضية مع كل حزمة بيانات، من إرسال معامل واحد مما يقلل الحمل الزائدة على الشبكة بشكل كبير. الآلية المقترحة وخلافا للآليات السابقة قابلة للتطبيق على مستوى واسع من الشبكات كشبكات الاستشعار اللاسلكي الصغيرة والشبكات السلكية ذات حزم البيانات الصغيرة وذات حزم البيانات الكبيرة جدا. نتائج التجارب العملية تشهد لآلية المقترحة بالتفوق الكبير في تقليل الأحمال الزائدة وتسريع التحميل وزيادة إنتاجية الشبكة ، والتقليل من إرسال الحزم المكررة.

CHAPTER 1

INTRODUCTION

The 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* [2], [3] which means that a node may behave as a server as well as a client at the same time and all content recipients are becoming content suppliers who participate cooperatively in the delivery of content. Since their roles are equal, in contrast to the client-server relationship, they are called *peers*. This leads to higher bandwidth utilization, more availability of content, and reduced download times [4]. The BitTorrent protocol [5] 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 [6], [7], [8], [9], [10] and [11]. 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 [5] requires that clients are able to download pieces in a purely random way. Subsequently, new selection techniques were introduced to improve performance [12], [13]. 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 [14], piece selection [15], or the coupon collector's problem [16]. 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 [17]. Subsequently, pieces that were owned by the departing peers may become rare or unavailable.

With the assistance of NC [18], 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 [19]. 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.

1.1 Problem Statement

Two main problems when applying network coding are appeared. One related to the computation and the other is related to the communication. First, the network coding usage 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 process 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 [20]. Indeed, the complexity of encoding is $O(n)$, where n is the number of pieces of the file. On the other side, the decoding is accomplished by inverting an $n \times n$ matrix which requires $O(n^3)$ operations.

The second problem is about the communication overhead. Since the encoded packet is a linear combination of n plain packets such that the coded packet contains the original data plus coefficients header, a.k.a coding vector. The coefficients can be selected independently and randomly from a Galois Field (GF) with q elements, $GF(q)$, and the total coefficients overhead is $n \log_2 q$ and $n^2 \log_2 q$ per packet and file, respectively. Although the coefficients overhead can be considered small compared to the large data packets, there are some cases in which this overhead can be significant and consumes the network bandwidth and the devices power. For instance, in wireless sensor networks where packets have short sizes and devices have limited power, the coefficients overhead can very fast become prohibitive[21]. Even in wired networks if only standard maximum transmission units (MTUs), e.g., 1500 bytes, are permitted, the header fraction could be too high [22]. Moreover, in P2P content distribution networks where huge size files are shared, e.g., 10 gigabytes or larger, the overall coefficients overhead incurred is in hundreds of megabytes which negatively affects

the goodput.

1.2 Dissertation Contribution

This Dissertation aims to make the following contributions:

- In contrast to the most published works which provide only high level information about the implementation of network coding, a detailed tutorial of network coding implementation for P2P file sharing is provided.
- We extensively survey the most important and recent network coding based P2P file sharing protocols. The survey aims to cover all network coding based P2P sharing systems focusing on the more recent techniques and solutions and provides classification, discussion, and assessment of the respective applications.
- The problem of P2P content distribution is modeled as the deck of cards probability problem [23] and both baseline and coding-based P2P content distribution systems are analysed. Based on the model and the analysis, we propose a novel network coding scheme, referred to herein by SGNC, that maintains the computational cost at reasonable level and boosts the robustness of the network by improving pieces diversity and availability.
- For the network coding coefficients overhead problem, We propose a new method, based on modular arithmetic that perfectly compresses the coefficients overhead to a minute value. The proposed method is dense, recodable, and reliable in the sense that it can be used over a wide range of applications and

devices; from wireless sensors networks to high performance computers.

1.3 Dissertation Organization

The rest of this Dissertation is organized as follow:

Chapter 2 presents a background about BitTorrent components and how it works, network coding overview, examples, and challenges, and detailed tutorial network coding of network coding implementation for P2P file sharing.

Chapter 3 surveys, classifies, and assesses the most important and recent network coding based P2P file sharing protocols.

Chapter 4 proposes Super generation network coding (SGNC), a novel approach to improve the pieces diversity and availability and accelerate the download time and decodability of network coding based P2P file sharing.

Chapter 5 introduces One item network coding vectors (OINCV), a novel method that perfectly reduces the network coding coefficients overhead to a very small value.

Finally in **Chapter 6** we conclude and outline the future work.

1.4 List of Journal Publication

[J1]: A A. AbuDaqa, A S. Mahmoud, M. Abu-Amara, T. Sheltami, "Survey of Network Coding Based P2P File Sharing in Large Scale Networks". In MDPI Applied Sciences. Vol. 10, March 2020.

[J2]: A A. AbuDaqa, A S. Mahmoud, M. Abu-Amara, T. Sheltami, "Super Generation Network Coding For Peer-to-Peer Content Distribution Networks". In IEEE

Access. Vol. 8, pages. 195240-195252, Oct. 2020.

[J3]: A S. Mahmoud, A A. AbuDaqa, "One Item Network Coding Vectors", (Under Progress) to be submitted to IEEE Transactions on Communications.

CHAPTER 2

BACKGROUND

2.1 BitTorrent Components

Current BitTorrent consists of the five main elements [5], [24] to work properly:

1. ***Tracker server***: a central server which arranges and manages the communication between BitTorrent peers.
2. ***Metafile (.torrent file)***: contains a metadata or information about the shared file/content. Details about the structure and the content of this file can be found in [25].
3. ***BitTorrent Client***: it is an application that implements the BitTorrent protocol. There are a lot available versions of them such as BitTorrent, BitComet, uTorrent, etc.
4. ***Initial peer (seeder)***: it is the original user who has the complete file to be shared. The existence of this component is essential since it is the source of the data, and any other peer that get the complete file becomes a seeder

5. **Downloader (leecher)**: it is a peer that has some pieces or no pieces at all.

Once it owns some pieces, it can start uploading thereof while it is downloading the rest.

2.2 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 dissertation, the piece terminology is used. before a peer joins the system, it contacts the tracker to obtain a list of some online peers that are currently downloading the file, the peer then tries to establish connections to some of the online peers as shown Fig. 2.1. These connected peers form a swarm and the peers inside the swarm are called neighbors. All the swarms form a dynamic logical network, namely a P2P overlay network, as shown in Fig. 2.2.

As new peers have joined the swarm, i.e., P2P network, The seeder randomly selects and distributes pieces to leechers. When a peer finishes downloading a piece,

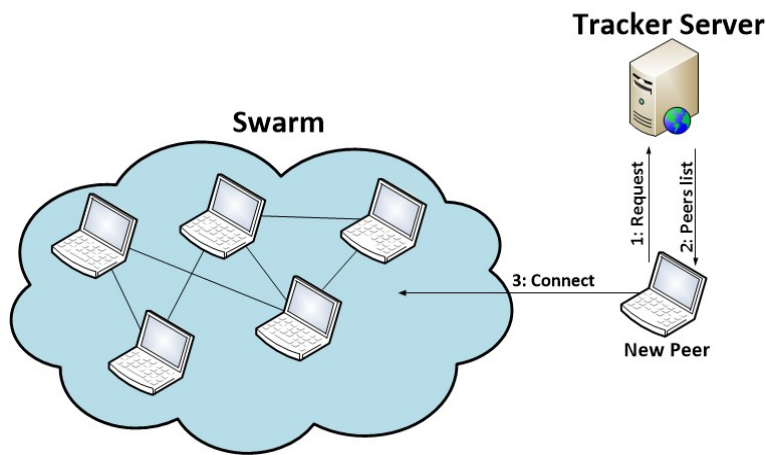


Figure 2.1: Process of new peer joining BitTorrent system.

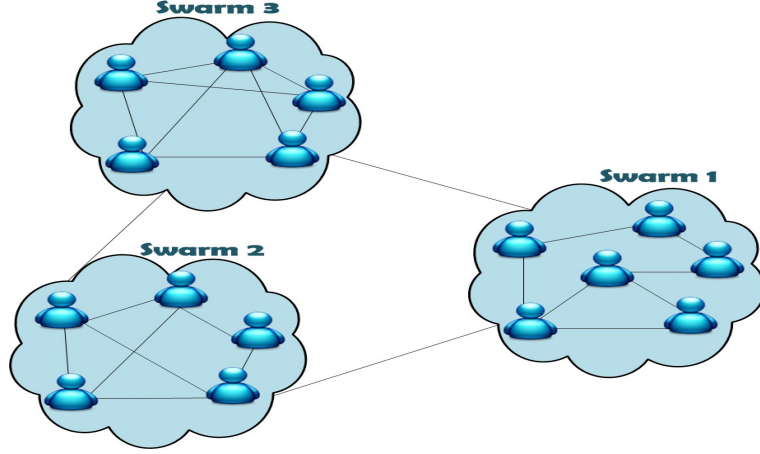


Figure 2.2: P2P overlay network.

it can upload this piece to other peers who request it. On the other hand, in network coding BitTorrent, a peer first encodes the pieces that it has by multiplying each piece with a coefficient e , which is randomly chosen from GF. Then, the pieces are linearly combined as one encoded piece. The encoded piece is distributed to other leechers. At the receiver peer, the decoding process is performed when a sufficient number of encoded pieces are received. Figure. 2.3 illustrates the piece distribution in both baseline and network coding BitTorrent.

The original specification [5] of the protocol requires that clients download pieces in a purely random way, but subsequently new techniques were introduced to improve the performance [12], [26]. An example of an improved technique is the local

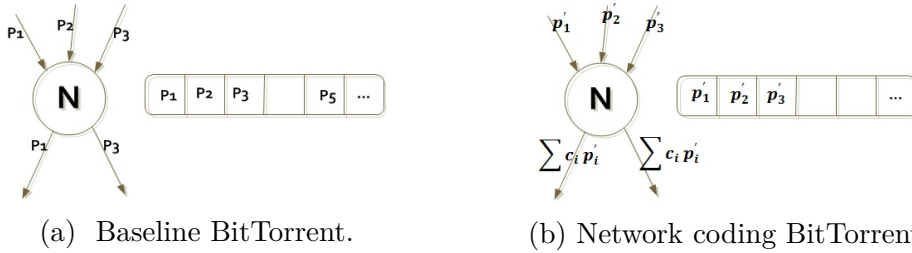


Figure 2.3: Illustration of piece distribution in baseline and network coding BitTorrent.

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 overlay network [15], [19].

2.3 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. [1], showing theoretically that the network coding can increase network throughput and improve robustness. Practically, RLNC was proposed in [27] for robust, distributed transmission, and compression of information in networks. 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 [1], wireless resources [28], [29], [30], and problems of security and resilience [31], [32], [33], [34], [35], [36].

Subsequently, the idea found a place in Avalanche, a P2P content distribution system from Microsoft [37]. In avalanche, a file of n plain pieces $p_1, p_2, \dots, p_n$ is represented as set of n coded pieces $P_{c1}, P_{c2}, \dots, P_{cn}$ such that each coded piece P_c is linear combination of the file's plain pieces multiplied by a vector of coefficients $e_{c,1}, e_{c,2}, \dots, e_{c,n}$ usually selected randomly or deterministically [38] from a GF with q elements, $GF(q)$. (2.1) represents an encoded piece P_c mathematically, and (2.2) shows the overall encoded pieces at the sender side.

$$P_c = \sum_{i=1}^n e_{c,i} * p_i \quad (2.1)$$

$$\begin{bmatrix} P_{c1} \\ P_{c2} \\ \vdots \\ P_{cn} \end{bmatrix} = \begin{bmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,n} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ e_{n,1} & e_{n,2} & \cdots & e_{n,n} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} \quad (2.2)$$

The following section demonstrates the benefit of using network coding.

2.4 The Butterfly Example

In this section, 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 Fig. 2.4, 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

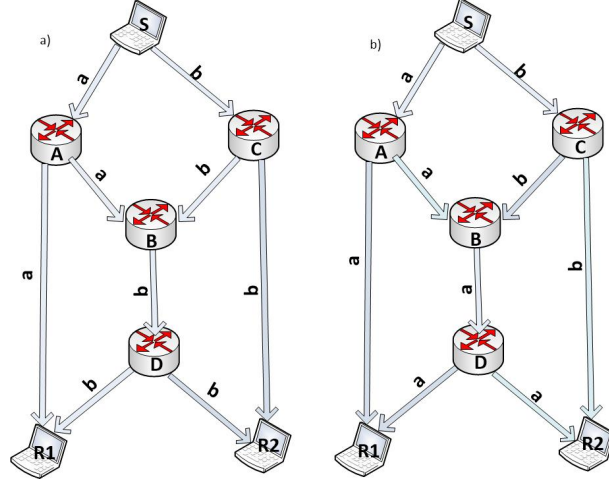


Figure 2.4: Butterfly network with traditional routing. (a) node B sends packet b; Then, (b) node B sends packet a [1].

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 $\langle a \rangle$ two times, and R_2 will receive message $\langle b \rangle$ two times, leading to redundancy and wastage of the link bandwidth.

Applying network coding as shown in Fig. 2.5, the two messages $\langle a \rangle$ and $\langle b \rangle$ can be XORed at router B , a new message $\langle c \rangle$ is created such that $\langle c = a \oplus b \rangle$. Then, this message is forwarded to router D which will send message $\langle c \rangle$ as a multicast to both R_1 and R_2 . R_1 can retrieve message $\langle b \rangle$ by Xoring message $\langle a \rangle$ with message $\langle c \rangle$, as $\langle b = a \oplus (a \oplus b) \rangle$, while R_2 can retrieve message $\langle a \rangle$ by Xoring message $\langle b \rangle$ with message $\langle c \rangle$, as $\langle a = b \oplus (a \oplus b) \rangle$. Using this novel encoding technique, the time required to deliver the two messages to the recipients is decreased by one time

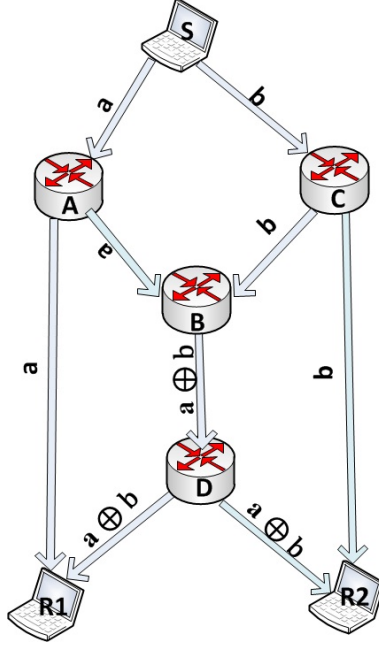


Figure 2.5: Butterfly network using network coding [1].

slot. Furthermore, neither redundant packets nor wastage of bandwidth are incurred.

In more complex scenarios, we need more complex coding operations such as RLNC [39], [27], or vector network coding [20], [40], [41]. With RLNC, 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 a 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 presentable with the same number of bits as the original packets.

2.5 Detailed Network Coding Tutorial for P2P File Sharing Systems

Since most of the published materials 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 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:

$$A(x) = a_2x^2 + a_1x + a_0 \tag{2.3}$$

Table 2.1 shows the elements (polynomials) of $GF(2^3)$ generated by (2.3) and their corresponding binary presentation.

Table 2.1: $GF(2^3)$ 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 (2.4):

$$C(x) = A(x) + B(x) = \sum_{i=0}^{m-1} c_i x^i, c_i \equiv a_i + b_i \text{ mod } 2. \quad (2.4)$$

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 (2.5):

$$C(x) = A(x) \cdot B(x) \text{ mod } P(x). \quad (2.5)$$

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 $C(x) = (x^4 + x) \text{ mod } (x^3 + x + 1) = x^2 \in GF(2^3)$.

- 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 (4.5):

$$A^{-1}(x) \cdot A(x) \equiv 1 \text{ mod } P(x) \quad (2.6)$$

[35] For example, in $GF(2^3)$ and $P(x) = x^3 + x + 1$, (x) is the inverse of $(x^2 + 1)$ since $x * (x^2 + 1) \text{ mod } (x^3 + x + 1) = 1$.

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 2.6.

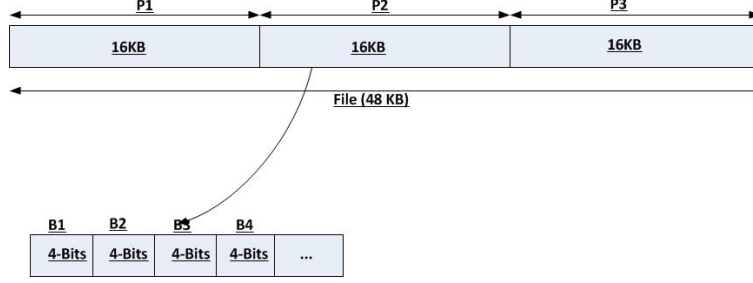


Figure 2.6: File segmentation phases.

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.

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 (4.2)); 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 $e_1 = x$, $e_2 = 1$, 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 $e_1 = x + 1$, $e_2 = x$ 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:

$$\begin{aligned}
 A \quad \quad \quad * \quad \quad B \quad \quad &= \quad \quad C \\
 \left(\begin{array}{cc} x & 1 \\ x + 1 & x \end{array} \right) \begin{bmatrix} b_{10} \\ b_{11} \end{bmatrix} &= \begin{bmatrix} x + 1 \\ 0 \end{bmatrix} \\
 \left(\begin{array}{cc} x & 1 \\ x + 1 & x \end{array} \right) \begin{bmatrix} b_{20} \\ b_{21} \end{bmatrix} &= \begin{bmatrix} 0 \\ x^2 \end{bmatrix}
 \end{aligned} \tag{2.7}$$

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 (2.8):

$$\det(A) = \left| \begin{pmatrix} x & 1 \\ x + 1 & x \end{pmatrix} \right| = x^2 + x + 1. \tag{2.8}$$

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

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

$$\begin{aligned} b_1 &= A^{-1} * \begin{bmatrix} x+1 & 0 \end{bmatrix} = \begin{bmatrix} x^2 & x^2+x \end{bmatrix}, \\ b_2 &= A^{-1} * \begin{bmatrix} 0 & x^2 \end{bmatrix} = \begin{bmatrix} x+1 & x^2+x \end{bmatrix}. \end{aligned}$$

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 [42] and a ready Java and C implementations can be found in [43].

CHAPTER 3

RELATED WORK

In the literature there are five network coding schemes that have been 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, (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 mix between generation-based and sparse coding, and (5) overlapped coding scheme which 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.

3.1 Full Network Coding

C. Gkantsidis et al. [37] 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 [44] 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 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 [45], [46] 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 [47]. 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 [48] ($\beta = 8$) requires 9 GB storage space at the file server. Further, 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 [47] was proposed by S. Deb et al. [49]. The study applied RLNC to gossip-based protocols [50], 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 [51] 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 RLNC 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 [37], [47], [49] 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 $n \times n$ matrix which requires $O(n^3)$ operations.

N. Wang et al. [52] 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.

R. Yeung et al. [53] 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.

On the other hand, D. Chiu et al. [14] 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.

A. Soro et al. [54] 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, M. Wang et al. [55] 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.

T. Li et al [56] 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. On the other hand, the studies that show the infeasibility of full coding are based on real implementation. Table 3.1 summarizes the most important studies related to the full network coding.

Table 3.1: Full network coding approaches and their characteristics.

Reference	Approach	Main Findings	Verification Method	Network topology	Performance Metrics
C. Gkantsidis et al. [37]	Utilizing RLNC 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.
S. Acedanski et al. [47]	Providing extensively theoretical analysis for P2P RLNC-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.
S. Deb et al. [49]	proposing RLNC-based approach for P2P gossip protocols.	- Computational overhead of RLNC to reconstruct the file is rather reasonable for $n \leq 1000$.	Numerical Simulation.	Complete Mesh.	- Download time.
N. Wang et al. [52]	Proposing DRLNC to mitigate unlucky combination problem.	- Unlucky combination problem could be eliminated even with $GF(2)$.	Numerical simulation.	Mesh.	- Rounds to complete download.
R. W. Yeung [53]	Analysis Avalanche using graph theory.	- Avalanche can achieve the theoretical lower bound of the file downloading time.	No experimental work.	Mesh.	—
D.Chiu et al. [14]	Avalanche is studied by modeling a simple star topology network.	- No advantage of network coding over traditional routing.	No experimental work.	Star.	—
M. Wang et al. [55]	Justifying the feasibility of RLNC on P2P systems by realistic application layer implementation.	- RLNC performs worse than any conventional store and forward P2P file sharing system.	High performance C++ implementation.	Mesh	- Average download time. - Encoding/ decoding complexity.

3.2 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.

M. Guanjun et al. [57] 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 (4.6). For a non-source peer which has k pieces, the encoding probability is set as in (4.7).

$$p = (\log n + d)/n \quad (3.1)$$

$$p = (\log k + d)/n \quad (3.2)$$

where d is a nonnegative constant ($d = 6$). The study also focuses on how to choose coefficients such that no linear dependency appears. Chord overlay network [2] 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, (4.6) and (4.7), 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 follow: 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 seconds. 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 robustness to churn test conducted between the sparse and the full network coding.

C. Ortolfo et al. [58] 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 (4.8) has half of the information about the original pieces X_i and X_j 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.

$$P(i, j) = C_i X_i + C_j X_j \quad (3.3)$$

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 on the other hand, the seeder in Paircoding 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 the time progresses, Paircoding outperforms BitTorrent. Paircoding can start decoding in the best case after receiving 2 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 [59], [60] is a variation of Paircoding in which each encoded piece as shown in (4.9) 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

$$P(i, j) = C_i X_i + C_j X_j. \quad (3.4)$$

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

Q. Cai et al. [19] conducted an in-depth analysis of Avalanche and Paircoding, and proposed a new fixed coding approach which 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 [61], 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 [26], and (3) BitTorrent using FEC [44]. The results show that the proposed approach outperforms the other implementations with regard to throughput with a slight increase in the control overhead. Also, 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 [61], [59], [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 2 original pieces at most. However, the static encoding, namely allowing encoding only at the seeder, yields a reduction in the availability of pieces. Fig. 3.1 shows an example of Treecoding.

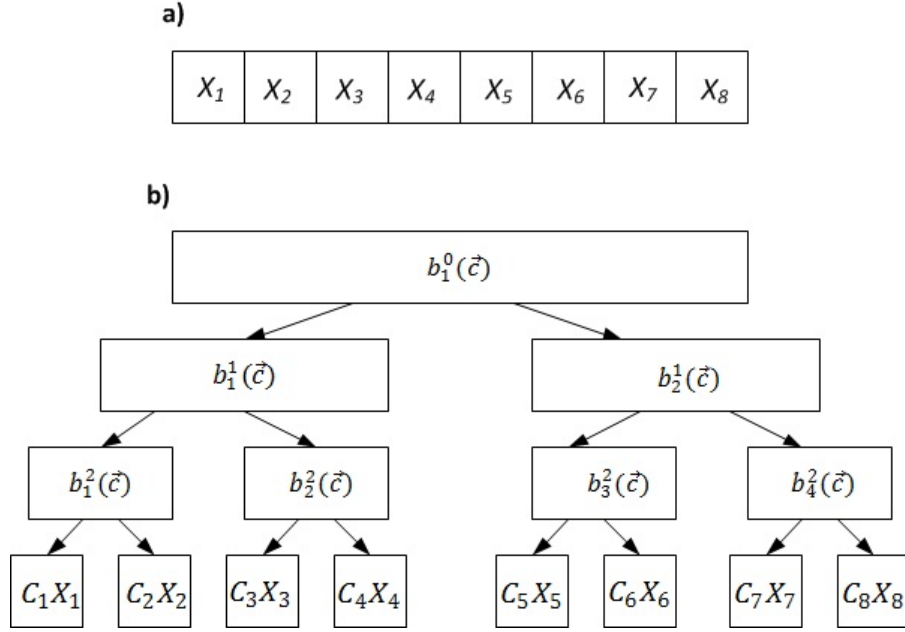


Figure 3.1: (a) File X with $n = 8$, (b) Tree coding of file X .

In conclusion, sparse network coding could be considered practical and valid for implementation for real world P2P file sharing systems but with less pieces diversity. Table 3.2 summarizes the most important studies related to sparse network coding.

Table 3.2: Sparse network coding approaches and their characteristics.

Reference	Approach	Main Findings	Verification Method	Network topology	Performance Metrics
M. Guanjun et al. [57]	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. [58]	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. [61]	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. [19]	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	—	—

3.3 Generation Based Network Coding

After the appearance of Avalanche and the associated issues, generation-based network coding [62], also known as grouped [30], chunked [63], segmented [64], or clustered [65], 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.

P. A. Chou et al. [66] 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.

P. Maymounkov et al. [67] 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 [66], [67], J. Xu

et al. propose Swifter [68], 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 tradeoff 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 [67] denoted by “R-push” and another scheme which 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. J. Xu et al. [63] 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 com-

ponents 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 telling 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.

M. Hundeboll et al. [69] 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.

D. Niu et al. [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.

X. Zhang et al. [71] 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. On the other hand, 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 $g = 5$. Besides, the results show that BitTorrent without coding is the most affected by churn even when the churn rate is very small (close to zero).

J.-S. Leu et al. [72] 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 [52] 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. Third, applying Gauss-Jordan elimination for early decoding to avoid long delay in the decoding process.

M. Yang et al. [38] propose P2P File sharing based on nEtwork coDing (PPFEED). PPFEED studies network coding by utilizing a special network topology called a combination network [73], [74], 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 [75] 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 k th 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: $(1, c_1, c_1^2 \bmod q)$, $(1, c_2, c_2^2 \bmod q)$, ..., $(1, c_n, c_n^2 \bmod q)$. 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 [76], 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.

P. Braun et al. [77] 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 BitTorrent 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. But to get the best gain of generation network coding, parameters such that generation size and piece/generation selection policy should be perfectly selected. Table 3.3 summarizes the most important studies related to generation Network coding

Table 3.3: Generation network coding approaches and their characteristics.

Reference	Approach	Main Findings	Verification Method	Network topology	Performance Metrics
J. Xu et al. [68]	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.
J. Xu et al. [63]	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
M. Hundeboll et al. [69]	Implementing generation coding system to study the parameters of generation coding: generation size, GF size, and piece size.	- BRONCO far outperforms HTTP meanwhile 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
D. Niu et al. [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.
X. Zhang et al. [71]	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 encoding/decoding cost. - network coding can improve the peers' incentive. Only mathematical and analytical model.	Only mathematical and analytical model	—	- Robustness to churn.
J.-S. Leu et al. [72]	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 [78]	GIA [79] overlay	- Encoding/decoding rates. - Download time. - Network overhead.
M. Yang et al. [38]	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.
P. Braun et al. [77]	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.

3.4 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.

H. Zeng et al. [80] 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. On the other hand, 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.

L. Yong et al. [64] use the same parameters proposed in [80] (generation size=10,

and encoding size=2) and propose a coding system based on CoolStreaming architecture [81] 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 BitTorrent system.

O. Kaiqian et al. [82] propose Dasher system which 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 download slightly faster than Dasher, however, Dasher can decode faster than Sparser by 10%.

J.Su et al. [83] 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 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 [57] . 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 seconds that 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 [57]. Metrics selected for evaluation are: (1) Average download time: PCLNC download time is shortened by 3.17% and 21.0% compared to [57] 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 [57], 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 [57].

These studies are limited and don't show deep analysis of the ability of the com-

bined coding to provide wide sparsity and diversity of the content (file’s pieces). Table 3.4 summarizes the most important studies related to combined network coding.

Table 3.4: Combined network coding approaches and their characteristics.

Reference	Approach	Main Findings	Verification Method	Network topology	Performance Metrics
H.Zeng et al. [80]	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.
L.Yong et al. [64]	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
O.Kaqian et al. [82]	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 [84] and LAN testbeds.	Mesh	- time. Download - Decoding speed.
J.Su et al. [83]	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 [85]	Mesh	- Download time - Start-up time. - coding degree.

3.5 Multi-Generation Mixing(MGM) and Overlapping Network Coding

In MGM [86], 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. Fig. 3.2 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 in decoding generations of lower position indices in the same mixing set resulting in robustness to churn and a wide piece diversity.

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

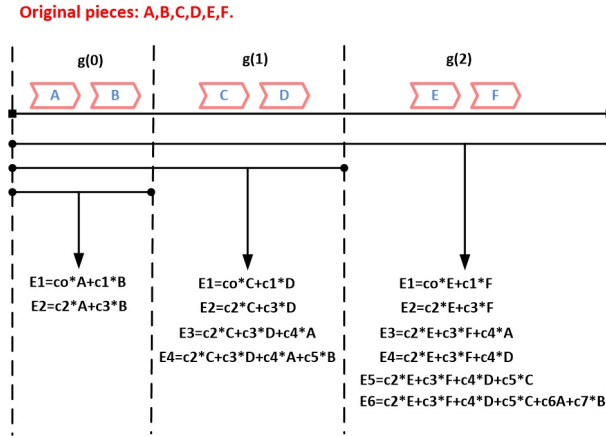


Figure 3.2: Multiple-Generation-Mixing (MGM) coding illustration, E_x denotes encoded piece, and c_i denotes a coefficient.

throughput of the network.

X. Wei et al. [87] 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.

D. Silva et al. [88] propose the first overlapping coding scheme by introducing both square and diagonal grid structures. In the square grid structure, shown in Fig. 3.3a, 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 Fig. 3.3b, supporter generations are diagonally stacked with an angle θ . The authors aim to mix the benefits of network coding and fountain codes [89], [90] such as Luby Transform (LT) code [91] and Raptor [92]. 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

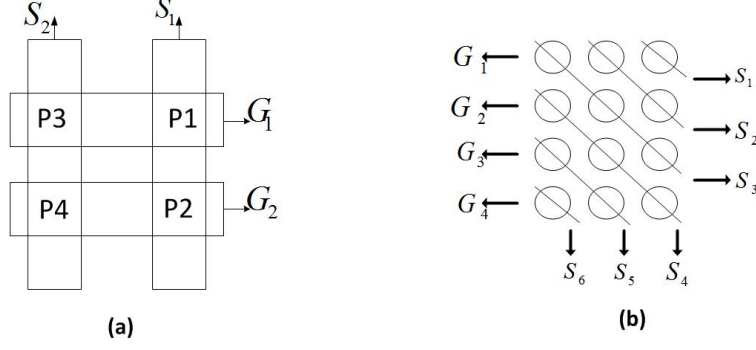


Figure 3.3: (a) square grid overlapping structure, (b) diagonal grid overlapping structure.

same time enjoy a fountain-like decoder? A practical simulation shows improved performance over independent generation coding in terms of decoding complexity overhead tradeoff. 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.

A. Heidarzadeh et al. [93] 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 [88] 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 generations 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 length 1 and 2, respectively. Also, they suggest using

overlapped chunked codes for multimedia transmission as shown in [93].

Y.Li et al. [94] propose random annex codes which allows 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.

B. Tang et al. [95] 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. Fig. 3.4 shows an example of EOC graph with $d = 3$ and 5 generations, each of 7 pieces (4 distinct and 3 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.

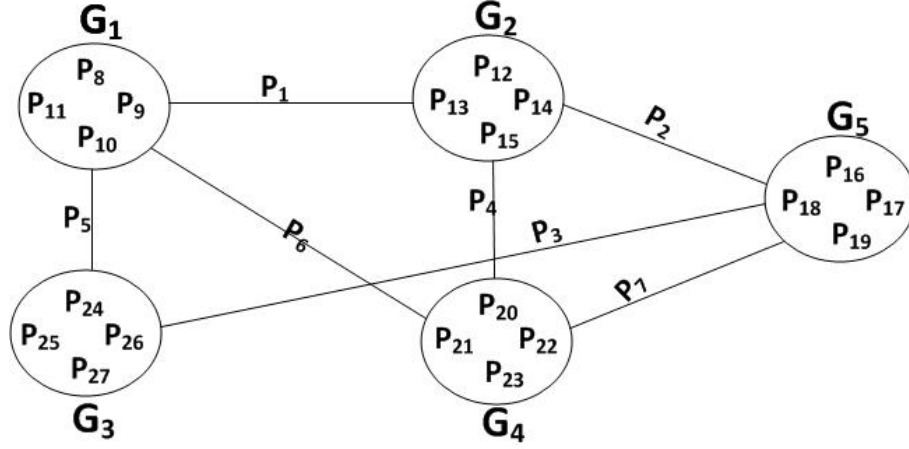


Figure 3.4: a d -regular graph ($d = 3$) with 5-vertices (generations).

G. Joshi et al. [96] 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 show that round-robin scheduling can accelerate the download time significantly for small generations sizes (≤ 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, Y. Li et al. [97] 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 which 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 achieve the minimum overhead (10%) comparing to disjointed generations (60%), head-to-toe (40%), and random annex (20%) codes. Although, the disjointed generations code achieves the minimum decoding complexity. The overall better overhead-complexity tradeoff 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 [88], [96] consider P2P networks. Table 3.5 summarizes the most important studies related to generations overlapping network coding.

Table 3.5: Overlapping network coding approaches and their characteristics.

Reference	Approach	Main Findings	Verification Method	Network topology	Performance Metrics
Silva et al. [88]	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 tradeoff.
Heidarzadeh et al. [93]	Proposing head-to-toe overlapping scheme over line network topology	- Overlapping can reduce network overhead for line networks. - Overlapping is appropriate for multi-media streaming.	Numerical Simulation	Line networks overlay	- Decoding probability.
Li et al. [94]	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. [95]	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. [96]	Proposing deterministic structures of overlapping and consider round-robin scheduling	- The upper bound limit of the optimal overlapping is $O(\log n)$. - 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. [97]	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 tradeoff is achieved by the proposed scheme.	Numerical Simulation	Point-to-point	- Decoding rate. - Overhead.

CHAPTER 4

SUPER GENERATION

NETWORK CODING

4.1 Problem Statement

The introduction of network coding in P2P content distribution systems brings about the issue of computational complexity. The use of linear algebra in encoding and decoding incurs an additional computation overhead that is not present in the original BitTorrent. Consider sharing a file consisting of N pieces, the file decoding within BitTorrent is very trivial and simple. Each piece is checked and verified separately, then it is directly stored on the hard disk. Conversely, the decoding process in network coding cannot be accomplished until all the encoded pieces are received, and requires solving a system similar to that shown in (4.1). Typically, the Gaussian elimination method is employed which requires $O(n^3)$ operations. Therefore, as the size of the file

increases, the cost of solving these equations becomes increasingly prohibitive.

$$\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,n} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ e_{n,1} & e_{n,2} & \cdots & e_{n,n} \end{bmatrix}^{-1} \begin{bmatrix} P_{c1} \\ P_{c2} \\ \vdots \\ P_{cn} \end{bmatrix} \quad (4.1)$$

In an effort to minimize the computational complexity, information pieces are partitioned into mutually exclusive subsets referred to as groups or generations, and coding is done only within a generation. This approach scales down the encoding and decoding problem from the whole file size to the generation size.

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

4.2 Main Contributions

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

- The problem of P2P content distribution is modeled as the deck of cards probability problem [23]. Based on this modeling, both baseline and coding-based

P2P content distribution systems are analysed.

- The best generation size is measured based on a real implementation.
- We propose a novel network coding scheme, referred to herein as Super generation network coding (SGNC), that maintains the computational cost at reasonable level and boosts the robustness of the network by improving pieces diversity and availability.
- We compare SGNC with normal generation network coding (NGNC), overlapped generation network coding (OGNC), and the baseline BitTorrent.
- Experiments are run based on three different scheduling policies namely random, local rarest-first, and global rarest-first. The evaluations consider four performance metrics which are robustness to churn, overhead rate, download time, and decodability rate.

4.3 Preliminaries and Theoretical Analysis

Consider a P2P content distribution communication network where peers are forming swarms and each peer is connected to a subgroup of the other peers in order to download a designated file which is chunked into pieces of equal sizes. In what follows we model this problem as the well-known the deck of cards probability problem [23] such that pieces are modeled as the cards and each round of the sharing process is modeled as selecting a subgroup of cards from the overall cards.

4.3.1 Baseline BitTorrent Analysis

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

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

Theorem 4.1 *For the baseline BitTorrent, the probability that the content is completely available after the seeder randomly shares exactly a flow equal to the content flow in a churned P2P network is obtained by:*

$$P(X = 100) = \frac{((N - k)!)^{\frac{N}{k}}}{N!^{\frac{N}{k} - 1}}. \quad (4.2)$$

Proof. The number of ways that the seeder shares these pieces randomly and with replacement may be computed as follows. For the first round r_1 , the seeder shares k out of N pieces in $\binom{N}{k}$ ways. For the second round r_2 , the seeder shares k out of N

pieces in $\binom{N}{k}$ ways, and so on until last round r_t , where $t = \frac{N}{k}$. Mathematically, this is written as

$$\binom{N}{k}_{r_1} * \binom{N}{k}_{r_2} * \dots * \binom{N}{k}_{r_t} = \left(\frac{N!}{k!(N-k)!} \right)^{\frac{N}{k}} \quad (4.3)$$

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

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

(4.4)

$$\begin{aligned} \binom{N}{k}_{r_1} \binom{N-k}{k}_{r_2} \binom{N-2k}{k}_{r_3} \dots \binom{k}{k}_{r_t} = \\ \frac{N!}{k!(N-k)!} * \frac{(N-k)!}{k!(N-2k)!} * \frac{(N-2k)!}{k!(N-3k)!} * \dots * 1 = \end{aligned} \quad (4.4)$$

$$\frac{N!}{k!k!\dots k!} = \frac{N!}{k!^{N/k}}$$

To compute the required probability, we need to divide (4.3) by (4.4) which yields

$$\begin{aligned}
\frac{N!}{k!^{N/k}} \div \left(\frac{N!}{k!(N-k)!} \right)^{N/k} &= \frac{N!}{k!^{N/k}} * \frac{k!^{N/k}(N-k)!^{N/k}}{N!^{N/k}} \\
&= \frac{(N-k)!^{N/k}}{N!^{\frac{N}{k}-1}}
\end{aligned}$$

I

4.3.2 Analysis for Network Coding Based BitTorrent

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

Lemma 4.1 *Dense network coding (DNC) has the following properties:*

- a. *Theoretically, the probability of the complete content being available after the seeder randomly shares exactly a flow equal to the content flow in churned P2P*

network approaches 1.

- b. *Practically, DNC is infeasible due to its prohibitive decoding computational requirements [39].*

Proof.

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

$$\frac{\binom{N}{k} \binom{N-k}{k} \binom{N-2k}{k} \dots \binom{k}{k}}{\binom{N}{k} \binom{N-k}{k} \binom{N-2k}{k} \dots \binom{k}{k}} = 1$$

- b. This can be proved experimentally based on the fact that the decoding requires solving N linear equations using the Gaussian elimination method where number of required operations asymptotically grows up at cubic rate $O(N^3)$. Table.4.1 and Fig. 4.1 show the results of a dense network coding experiment for file sizes ranging from 5MB to 160MB, utilizing coefficients from $GF(256)$. The experiment each time doubles the file size and computes the decoding time. It can be noticed that as the file size increases, the decoding time grows up rapidly. For instance, the 160MB file needs about two hours and half to be decoded although this file is considered very small for P2P content distribution networks which usually consider files in the giga bytes (GB). This implies that

dense network coding is impractical.

I

Table 4.1: File decoding time based on dense network coding

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

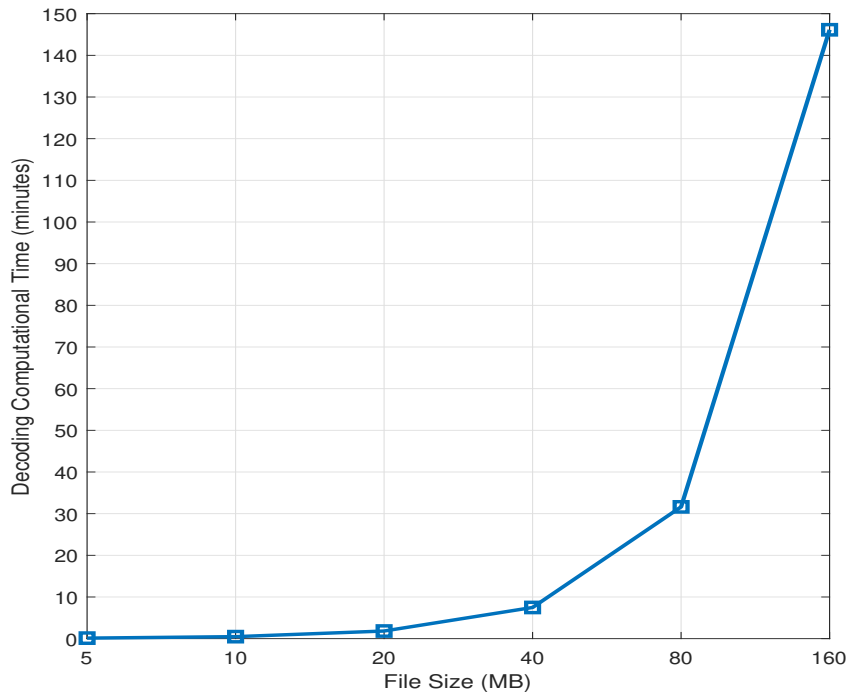


Figure 4.1: File decoding time based on dense network coding and 32KB piece size.

Theorem 4.2 *For generation network coding with w generations, the probability that the content is completely available after the seeder randomly shares exactly a flow equal to the content flow in a churned P2P network is obtained by:*

$$P(X = 100) = \frac{(w - k)!^{w/k}}{w!^{w/k}} \quad (4.5)$$

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

$$\binom{w}{k}_{r_1} * \binom{w}{k}_{r_2} * \dots * \binom{w}{k}_{r_t} = \left(\frac{w!}{k!(w-k)!} \right)^{\frac{w}{k}} \quad (4.6)$$

Out of the universal set, the number of ways that the exact content is available can be written as in (4.7)

$$\begin{aligned} \binom{w}{k}_{r_1} \binom{w-k}{k}_{r_2} \binom{w-2k}{k}_{r_3} \dots \binom{k}{k}_{r_t} = \\ \frac{w!}{k!(w-k)!} * \frac{(w-k)!}{k!(w-2k)!} * \frac{(w-2k)!}{k!(w-3k)!} * \dots * 1 = \end{aligned} \quad (4.7)$$

$$\frac{w!}{k!k!\dots k!} = \frac{w!}{k!^{w/k}}$$

Thus to get the final probability, we need to divide (4.6) by (4.7)

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

4.3.3 Comparative Analysis

Fig. 4.2 compares (4.2) and (4.5). It is apparent that network coding increases the probability of content availability. Indeed, this probability approaches one when $w = k$. However, for large file size w and k are not equal or even close to each other. k is always constant regardless of the file size as it is limited by the data rate of the peers, whereas w increases as the file size increases because each generation size is limited by the available CPU computation and memory space. Moreover, the figure shows that as w decreases, the content becomes completely available sooner. Thus, for perfect generation-based network coding, we need to minimize w as much as possible and this can be done by increasing the generation size to the fullest extent, namely, utilizing the maximum CPU and memory resources. Let this be denoted as the best generation size which will be discussed in the next subsection.

4.3.4 Best Generation Size

To get the most benefits of network coding, the generation size should be the maximum possible allowed by the computational resources. To determine the best generation size, an experiment is conducted using java as programming language, and based on Strassen algorithm [98], [99] as an implementation for matrix multiplication and inversion. The PC used for simulation has Core i5-2430M CPU with speed 2.40 GHz and 16GB RAM. We assume that 10 minutes is an acceptable decoding time given

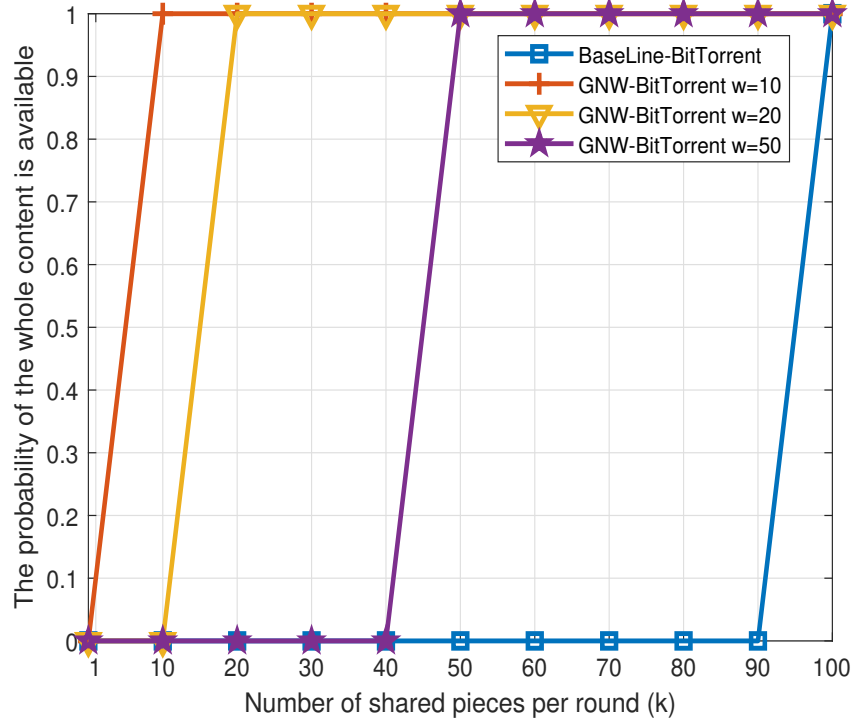


Figure 4.2: Theoretical analysis of the content availability ($N=100$, $w=10$, 20 , and 50).

the maximum delay tolerated by the end user. Other thresholds are equally valid. Fig. 4.3 shows the decoding computational time of different generation sizes and piece sizes of 16KB, 32KB, and 64KB as these sizes are typical in P2P content distribution systems. From the figure, we notice that for 16KB pieces, the generation size should not exceed 30MB. This can be increased to 40MB for the case of 32KB pieces, and even increased further to 60MB when 64KB pieces are used.

4.4 Super Generation Network Coding (SGNC)

Based on the earlier analysis that shows that dense network coding is impractical, and the limitation on the generation size for generation network coding, this section

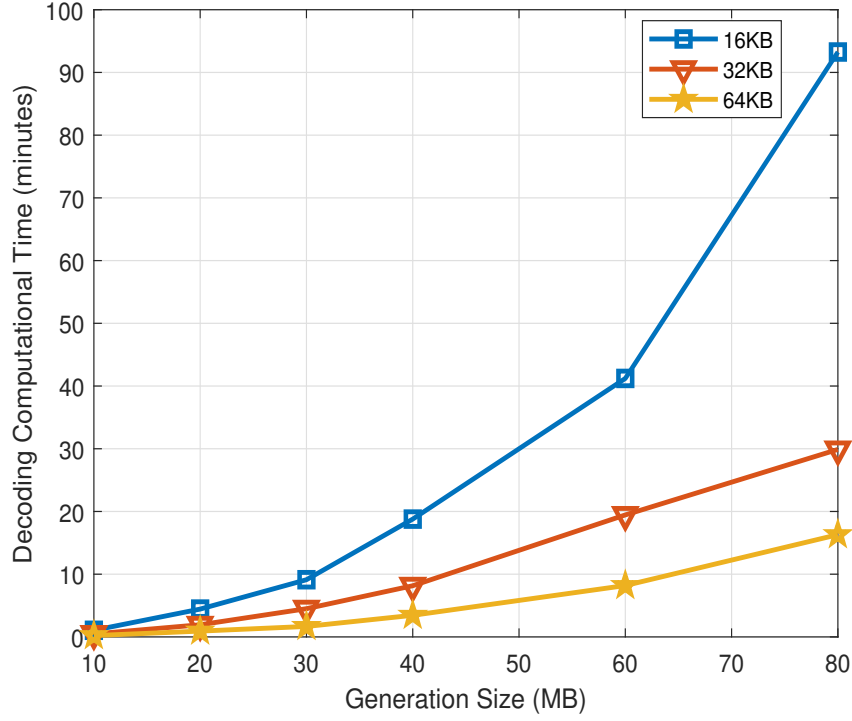


Figure 4.3: Best generation size for different piece sizes

proposes a novel method referred to herein as the Super Generation Network Coding (SGNC).

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

Definition 4.1 (Unity Piece) *The piece is encoded by multiplying each plain piece of the generation by 1. The form of this piece is $(A + B + C + \dots + Z)$. It is unique*

and basic for every generation.

Definition 4.2 (Decodable Piece) *This piece is encoded by multiplying a plain piece of the generation by a coefficient e drawn from $GF(q)$, while the remaining pieces of the generation each is multiplied by 1. The decodable piece $(e.A + B + C + \dots + Z)$ is always instantaneously decoded by XORing with the unity piece.*

It is not necessary that the coefficient is placed such that consecutive decodable pieces are arranged diagonally as they can be arranged randomly. Another allowed form of consecutive decodable pieces is lower-triangular provided that a decodable piece is always decoded as early as possible. These potential forms of decodable pieces are shown in Fig. 4.4.

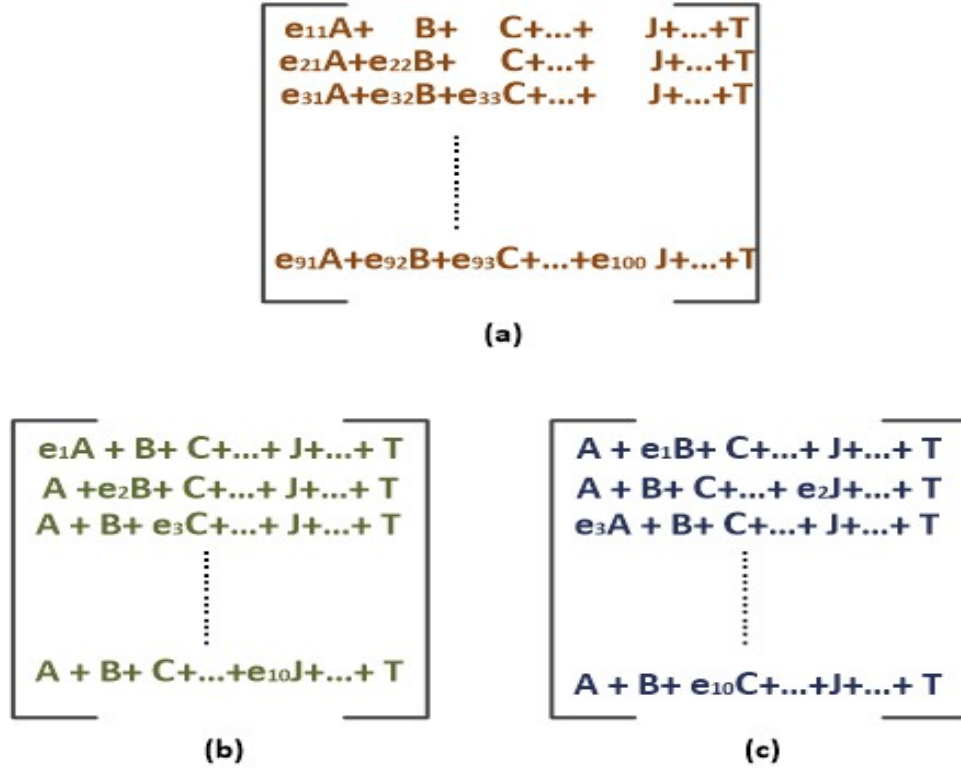


Figure 4.4: Decodable piece's forms: (a) Lower triangular, (b) Diagonal, and (c) Random.

Definition 4.3 (Rich Piece) *The rich piece is full of information and is encoded by multiplying all the pieces in the generation by coefficients drawn randomly from $GF(q)$ such that the encoded piece is given by $(e_1A + e_2B + e_3C + \dots + e_sZ)$.*

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

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

Fig. 4.5 shows a normal generation with 10 pieces and a super generation with 20 pieces. Although the size of the super generation is doubled, the decoding time is almost the same. The figure also shows that SGNC incurs additional coefficients overhead. The SGNC overhead in terms of coefficients is discussed in the next section.

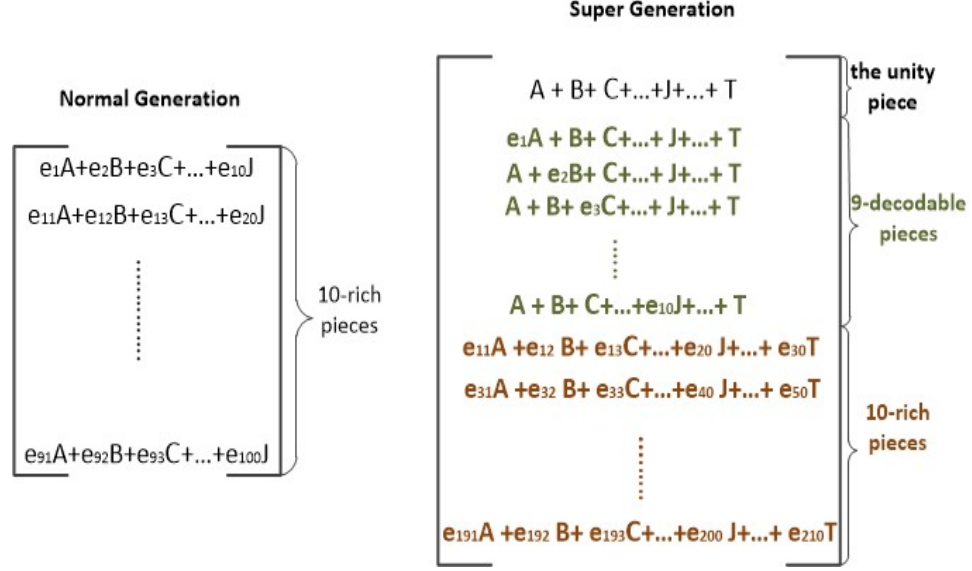


Figure 4.5: Normal generation vs. super generation

4.4.1 SGNC Overhead

Recall that n is the generation size and the coefficients are drawn from $GF(2^8)$ as in most cases, then the coefficients overhead per generation for normal network coding is n^2 bytes. Since the generation size of SGNC is twice that for the normal generation, then the coefficients overhead for SGNC is supposed to be four times that of the normal generation, notably, $4n^2$. However, because the coefficients values for the majority of decodable pieces are equal to 1, then the coefficients overhead of SGNC can be minimized by using the method inspired by Y. Li et al. [100].

The method simply avoids sending a byte coefficient for a coefficient whose value is equal to one. Instead it sends a vector of 0 and 1 bits, where a 0 bit indicates a coefficient value of one, while a 1 bit indicates the coefficient value is greater than one. In addition, coefficients whose values are greater than one are sent as bytes coefficients, as in the conventional case. Therefore, the coefficients overhead for SGNC

caused by the rich pieces is equal to $2n^2$ bytes, whereas the overhead needed by the unity and decodable pieces is equal to $(n - 1 \text{ bytes} + n^2 \text{ bits})$. Hence, the overall SGNC coefficients overhead per generation in bytes can be written as:

$$2n^2 + \frac{n^2}{8} + n - 1 \quad (4.8)$$

It's clear from (4.8) that the overhead is upper bounded by $\mathcal{O}(n^2)$. Also, since the number of generations in SGNC is half the number of generations in normal network coding, the overall coefficients overhead of SGNC is only slightly larger than the normal network coding overhead as will be shown experimentally in section 4.5.

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

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

Example 1 *Assuming 4 seeders are to share 16 decodable pieces to a receiver peer.*

Then the corresponding pieces for each seeder are

$$M_1 = \{0, 4, 8, 12\}$$

$$M_2 = \{1, 5, 9, 13\}$$

$$M_3 = \{2, 6, 10, 14\}$$

$$M_4 = \{3, 7, 11, 15\}$$

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

4.5 Experimental Work and Results

We consider a P2P network which is divided into swarms as shown in Fig. 2.2 with each swarm containing a maximum of 8 nodes. Each node can connect to at most 4 other nodes selected at random from the same swarm or nearby swarms. In addition, we consider that all the peers have same data rate and therefore the download time can now be measured in terms of number of rounds. SGNC is compared with the optimal theoretical solution, normal generations network coding (NGNC), overlapped generations network coding (OGNC), and the baseline BitTorrent system. For some experiments, only NGNC is considered, while OGNC is omitted since it behaves mostly as NGNC.

Files of sizes equal to 320MB, 640MB, 1.2GB, and 5.1GB are shared. Each file is

divided into generations based on (4.9) and (4.10) for NGNC and SGNC, respectively.

$$\#Generations = \frac{FileSize}{BestGenerationSize} \quad (4.9)$$

$$\#Generations = \frac{FileSize}{BestGenerationSize * 2} \quad (4.10)$$

Based on our analysis in section 4.3 we consider the practical best generation size to be 40MB. Therefore, the 320MB file is divided into 8 generations for NGNC and 4 for SGNC and so on for the other file sizes. Also, we consider that coefficients are drawn randomly from $GF(256)$ and we assume that all pieces are innovative; that is there is no linear dependency among the coefficients of encoded pieces.

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

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

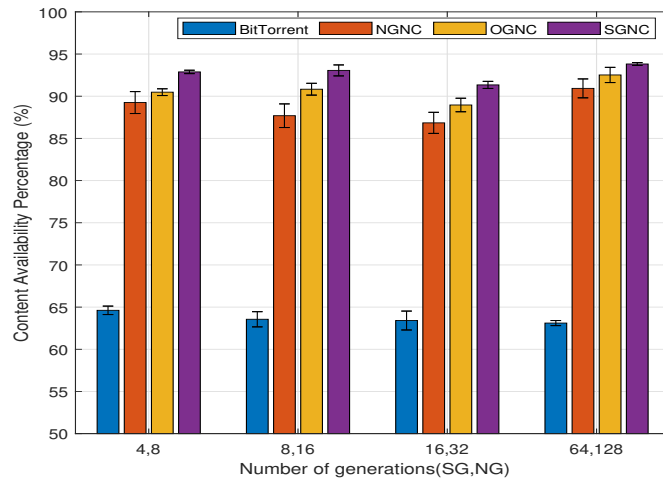
For network coding schemes, the RF-policies are applied at generations level rather than at the pieces level. The following subsections detail the results obtained for each of the considered metrics.

4.5.1 Content Availability (Robustness to Churn)

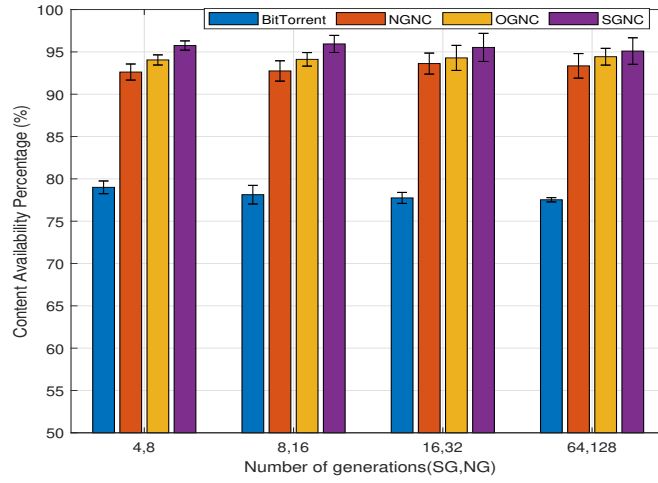
This experiment assumes that the seeder or seeders leave the network after they share a number of pieces that are equal to the file pieces. The remaining peers cooperate to complete the file download from each other. Fig. 4.6 shows results, with confidence interval=90% based on random, local rarest-first, and global rarest-first scheduling policies, respectively. The results show improvement in content availability for SGNC over NGNC and OGNC by at least 2% and 1%, respectively. For baseline BitTorrent, the content availability improves significantly when some RF-policy is applied, while the improvement provided by the RF-policy is modest when network coding is employed. This is a direct result of the fact that network coding does not require global topology information and can handle high churn in the network.

4.5.2 Overhead Due to Duplicate Pieces

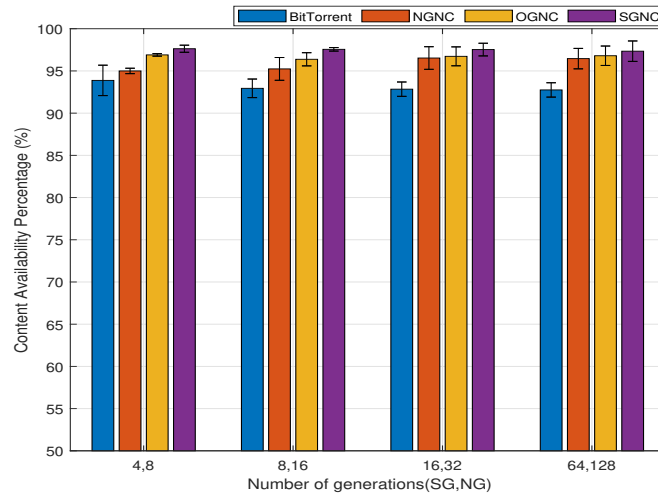
In this experiment seeders continue to send pieces, based on the designated scheduling policy, to a peer until the peer completes the download. As a result, duplicated pieces may be received. For network coding schemes, the duplicate piece is considered an overhead if it is received after all the corresponding generation pieces are complete. We also assume that when the RF-policy is applied, the coordination is done only among the seeders and the receiver peer does not send any feedback. Fig. 4.7a, Fig. 4.7b and Fig. 4.7c show results of the overhead, with confidence interval= 90%, based on the random, local, and global rarest-first scheduling policies, respectively. The figure shows clearly that the random scheduling policy must be avoided as it



(a)



(b)



(c)

Figure 4.6: Content availability based on different scheduling policies:(a) Random, (b) Local rarest-first, and (c) Global rarest-first.

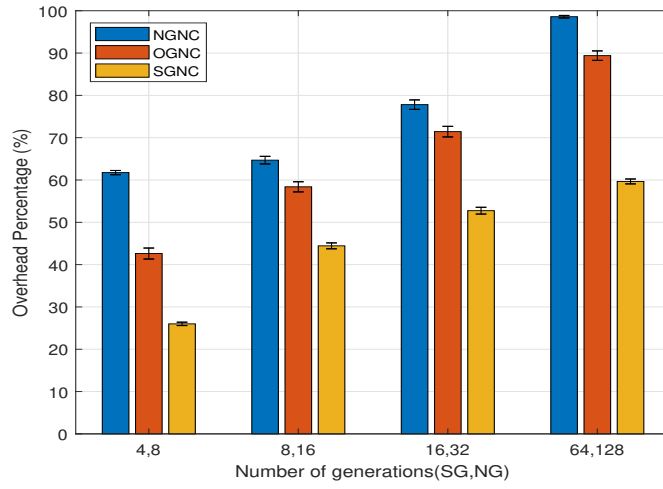
incurs the largest overhead percentage relative to the other two scheduling policies. It can also be noticed that SGNC always incurs the lowest overhead for all the scheduling policies relative to the other schemes. The results, across the considered number of generations, also show that SGNC has about half or quarter of the overhead relative to NGNC for the local and global RF policies, respectively. We omit the comparative overhead results for baseline BitTorrent from Fig.4.7 since it is very high and exceeds 100% for all file sizes relative to NGNC, OGNC, and SGNC. Fig.4.8 shows that distributing a file to the baseline BitTorrent incurs extreme overhead even when using local and global scheduling policies.

4.5.3 Download Time

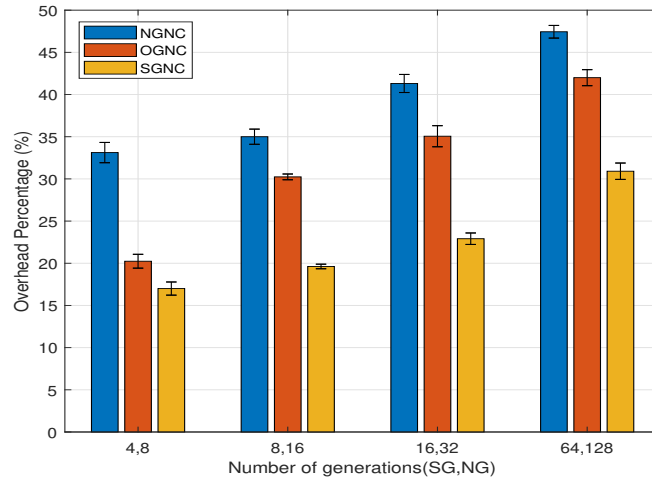
Definition 4.4 (Optimal Download Time) *Assuming the number of pieces shared during each round k is the same, then the optimal or theoretical download time is given by*

$$OptimalDownloadTime = \frac{FileSize}{k} \quad (4.11)$$

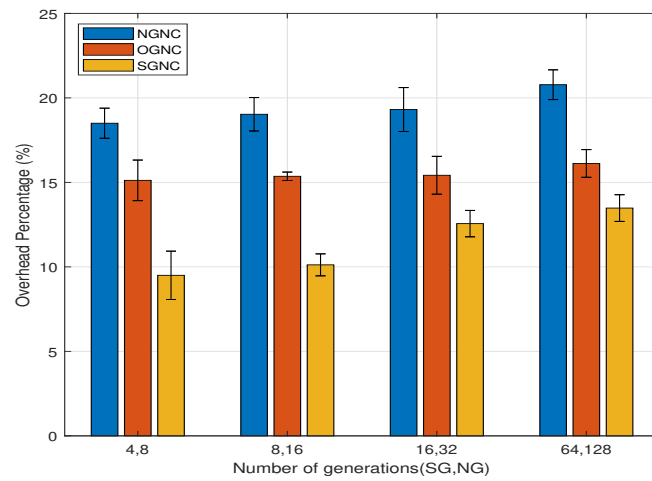
Accordingly, the download time experiment assumes a peer receives pieces of the file from other peers, such that each peer sends 1MB per round, until the peer completes the download. Results, depicted in Fig.4.9, Fig.4.10, Fig.4.11, and Fig.4.12, are for file sizes 320MB, 640MB, 1.2GB, and 5.1GB, respectively. The results show that for all scheduling policies and all the considered file sizes, SGNC always achieves the closest download time to the optimal case. Indeed, SGNC can download the file faster than NGNC by at least 10%. As the number of generations increases, the



(a)



(b)



(c)

Figure 4.7: The redundancy overhead percentage based on different scheduling policies:(a) Random, (b) Local rarest-first, and (c) Global rarest-first.

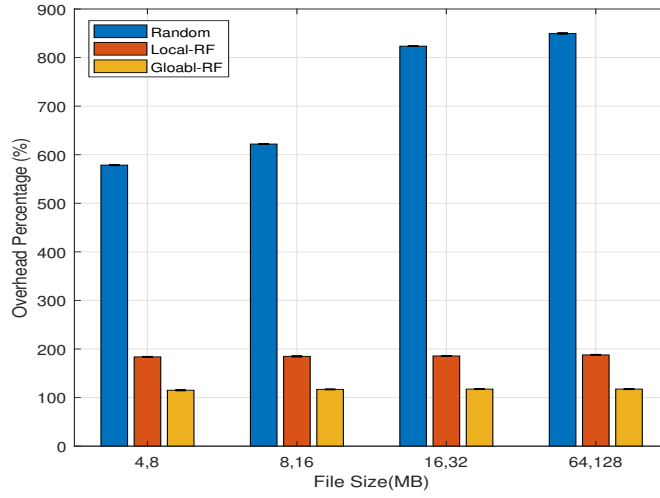


Figure 4.8: BitTorrent overhead based on different scheduling policies.

improvement also increases.

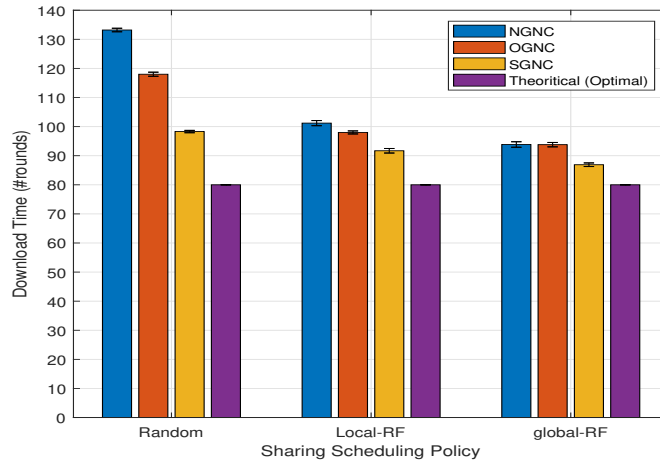


Figure 4.9: Download time of 320MB file

4.5.4 Decodability

Definition 4.5 (Decodability) *Refers to the cumulative number of already decoded pieces from the instant of receiving the first piece to a certain point of time. Higher decodability number leads to higher diversity in pieces and less CPU load.*

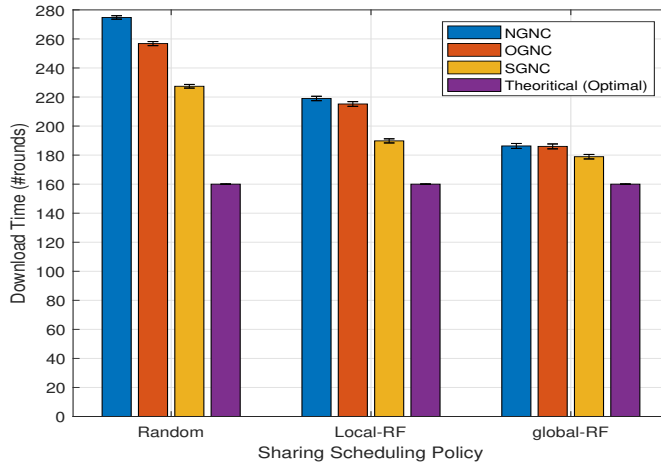


Figure 4.10: Download time of 640MB file

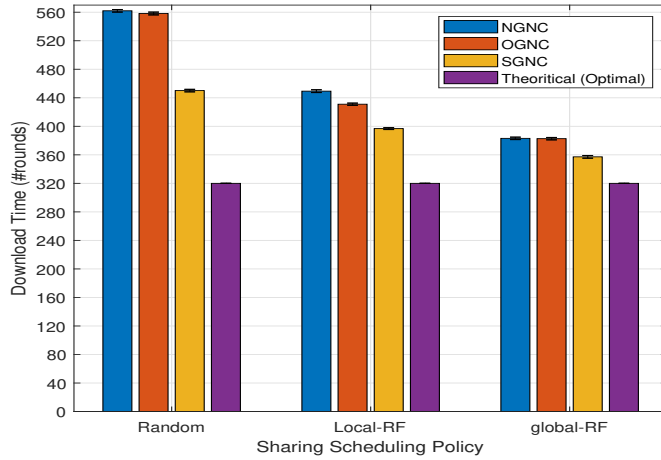


Figure 4.11: Download time of 1.2GB file

As such, the decodability of a file with 80 pieces is measured from round 0 to round 100 in this experiment. In general, network coding suffers from low decodability especially at the initial rounds. Fig.4.13 shows that normal generations network coding cannot decode any piece until round 42. On the other hand, baseline BitTorrent can decode almost one piece every round. The figure also shows that SGNC can start decoding from very early rounds at least at the rate of one piece for every two rounds. This superior performance of SGNC is the result of the self-decodable pieces. This

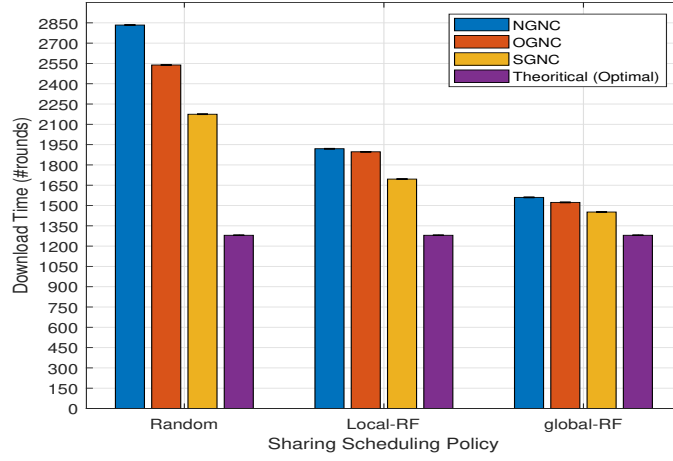


Figure 4.12: Download time of 5.1GB file

supports pieces' diversity and availability, and balances the CPU load by distributing the required load smoothly over all rounds. Fig.4.14 and Fig.4.15 consider the same experiment but for the other two scheduling policies; the local and global rarest-first, respectively.

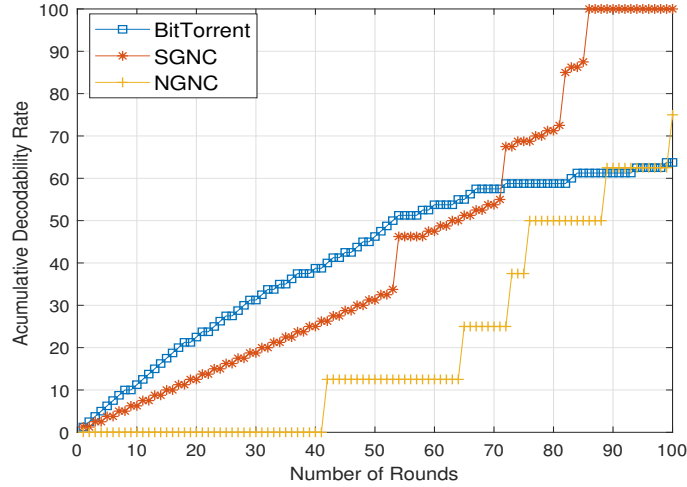


Figure 4.13: Decodability based on random scheduling policy

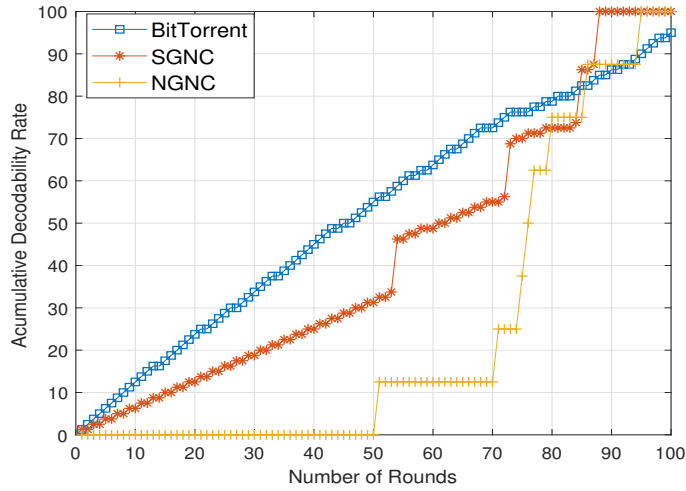


Figure 4.14: Decodability based on local rarest-first scheduling policy

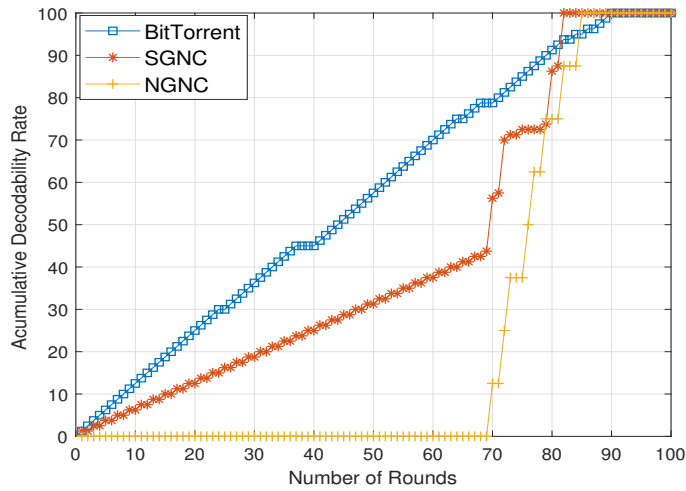


Figure 4.15: Decodability based on global rarest-first scheduling policy

4.5.5 Network coding coefficients overhead

Since the coefficients for all the experiments in this chapter are taken from $GF(2^8)$, then the coefficients overhead for a generation of size n is n^2 bytes for NGNC, and $2n^2 + \frac{n^2}{8} + n - 1$ for SGNC as specified by (4.8).

In this experiment after a peer completes the download of a file, the network coding coefficients overhead caused by the downloaded file and the redundant pieces is calculated. Results corresponding to the file sizes of 320MB, 640MB, 1.2GB, and 5.1GB are depicted in Fig.4.16, Fig.4.17, Fig.4.18, and Fig.4.19, respectively. The results show the following: 1) the network coding coefficients overhead does not exceed 6% for all the network coding schemes, file sizes, and sharing scheduling policies, and 2) coefficients overhead incurred by SGNC is slightly larger than that of NGNC.

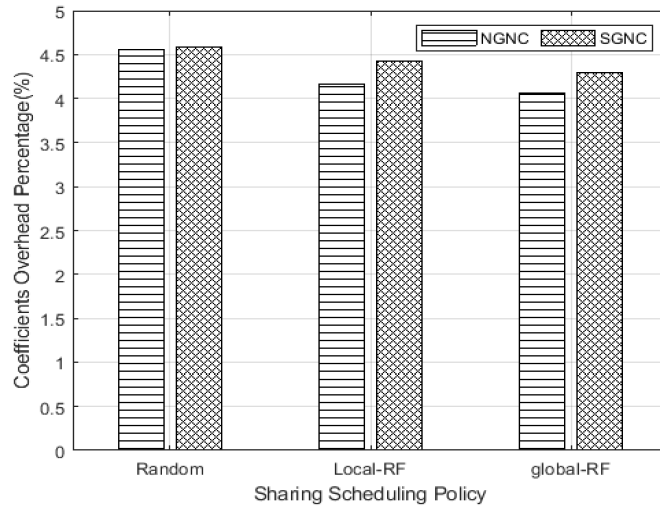


Figure 4.16: Network coding coefficients overhead percentage of 320MB file

While SGNC has the maximum coefficients overhead, however, if we consider the

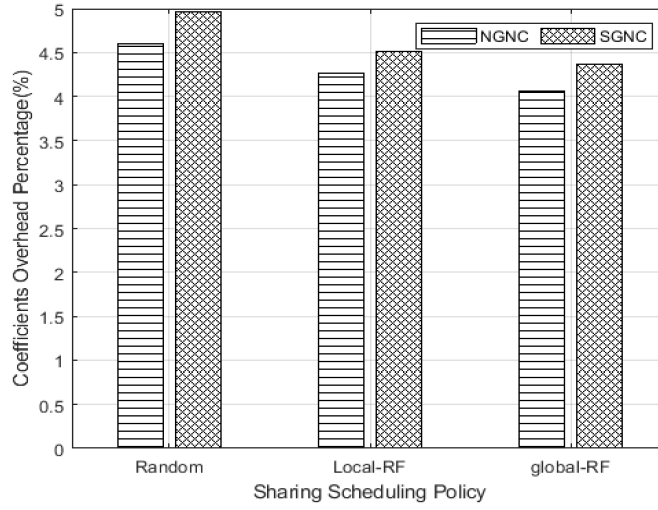


Figure 4.17: Network coding coefficients overhead percentage of 640MB file

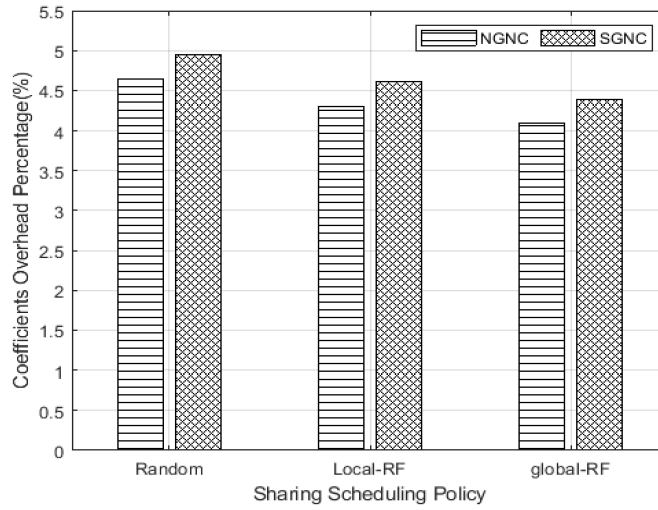


Figure 4.18: Network coding coefficients overhead percentage of 1.2GB file

overall overhead which is the cumulative overhead caused by network coding coefficients and the duplicate pieces at the receiver side, then SGNC incurs the minimum overall overhead. Fig.4.20 shows the results of the overall overhead for NGNC and SGNC for different file sizes based on global rarest-first scheduling policy. It is clear that SGNC has the lowest overhead among all network coding schemes.

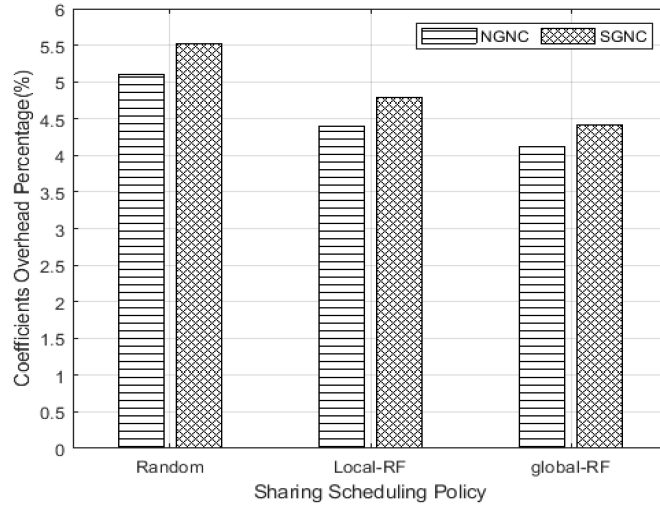


Figure 4.19: Network coding coefficients overhead percentage of 5.1GB file

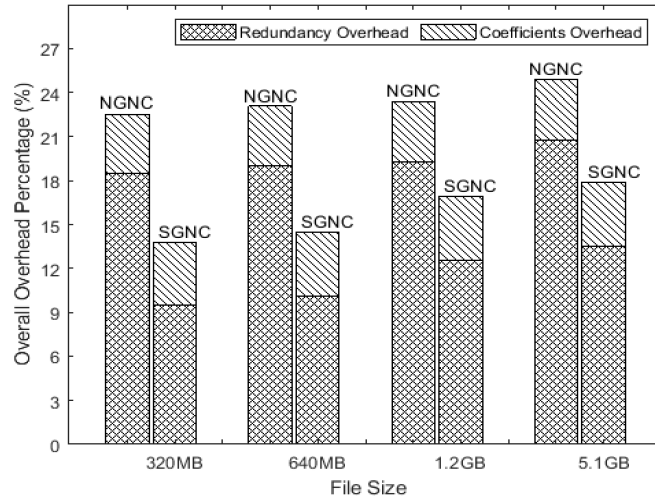


Figure 4.20: The overall overhead based on global-rarest first scheduling policy

4.6 Chapter Summary

In this chapter two contributions are presented. First, we model the problem of P2P content distribution as the deck of cards probability problem and we analyze it based on this model for both baseline and network coding P2P content distribution systems. Second, we propose a novel procedure referred to herein Super generation network

coding (SGNC) which is a network coding scheme that boosts the robustness against churn in P2P networks, while maintaining roughly the computational cost relative to generation network coding. In addition, we evaluate the proposed method and compare it with normal generation network coding, overlapped generation network coding, and baseline BitTorrent. The experiments consider file sizes ranging from 320MB to 5GB, and utilize three piece scheduling policies namely: random, local rarest first (RF), and global RF. The aforementioned schemes are evaluated in terms of availability, overhead, download time, and decodability. Simulation results show that SGNC can improve the robustness of the network, minimize the download time, incurs low overhead, and speeds up the decodability which in turns alleviates the central processing unit (CPU) load.

CHAPTER 5

ONE ITEM NETWORK CODING VECTORS

In this chapter we present an efficient method for compressing network coding vectors to a very small constant value. The method is based on modular arithmetic and prime numbers, and is influenced by Chinese remainder theorem (CRT). It is referred to as the One item network coding vectors or OINCV.

5.1 Problem Statement

With RLNC, the coded packet of a generation of size n is a linear combination of n plain packets such that the coded packet contains the original data plus a vector of coefficients (ζ), a.k.a coding vector, as a header. The size of the coding vector depends on Galois Field (GF) size, generation size, and the coding vector representation. On the receiver side, it is needed to receive $n \times n$ coefficients matrix, a.k.a the transfer matrix, along with n coded packets to solve a system like in (5.1) and recover the

original packets. The total coefficients overhead of RLNC is $n \log_2 q$ and $n^2 \log_2 q$ per packet and generation, respectively, where q is the GF size.

$$\begin{bmatrix} P_{c1} \\ P_{c2} \\ \vdots \\ P_{cn} \end{bmatrix} = \underbrace{\begin{bmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,n} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ e_{n,1} & e_{n,2} & \cdots & e_{n,n} \end{bmatrix}}_{n \times n \text{ coefficients matrix}} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} \quad (5.1)$$

Although the coefficients overhead can be considered small compared to large data packets, there are some cases in which this overhead can be significant and consumes noticeable network bandwidth and power of the devices. For instance, in wireless sensor networks (WSN) where packets are short in length and devices have limited power, the coefficients overhead can quickly become prohibitive [21]. Even in wired networks if only standard maximum transmission units (MTU), such as 1500 bytes are permitted, the header fraction can be too high [22]. Moreover, in P2P content distribution networks where huge sized files are shared, e.g., 10 Gigabytes (GB) or larger, the overall coefficients overhead incurred is in hundreds of Megabytes (MB). The following three examples demonstrate cases where the coefficients overhead is significant.

Example 1 *Consider a wireless sensors network that contains multi-sources which want to send 50 packets to a sink. If network coding is applied over $GF(2^8)$, then the coding vector is $50 \mathbb{F}_{256}^{50}$ symbols which translates to 50 bytes. But in a such network the overall packet size is about 100 Bytes, e.g. zigbee packet, which means that the*

overhead ratio is 50%.

Example 2 Consider a wired network that uses $MTU = 1500$ Bytes as a packet size. In such network the generation size is typically large. Hence if we assume $n = 512$ and the coefficients are drawn from $GF(2^8)$, then the coding vector is 512 bytes which is more than one third of the MTU.

Example 3 Consider P2P content distribution network that uses 16KB as a packet size and shares a 10GB File. Usually in such scenario the best generation size is 20MB, then the file is segmented to $10GB/20MB = 512$ generations. Each block in the generation is 16KB, then number of blocks in a generation is $20MB/16KB = 1280$ blocks. Thus, the coefficients matrix is 1280×1280 per generation and if the coefficient are drawn from $GF(2^8)$ and each represented as 1 byte, then the total coefficients overhead of the file is $1280B \times 1280B \times 512 = 800MB$ per the file.

5.2 Methods in Literature

Many approaches in the literature are proposed to address the problem of network coding coefficients overhead. Most of these approaches are based on using sparse coding vectors, i.e., each coding vector contains only small numbers of nonzero coefficients, and thus the coding vectors can be compressed.

Mahdi. J et.al. in [22] propose first compressing coefficients vectors scheme which is based on error correcting codes and using parity check matrices. The scheme considers m nonzero coefficients, where $m \ll n$. The scheme results in coefficients overhead

equals to $2m$ per packet. S.Li et. al. in [101] improve the compressing scheme proposed in [22] by utilizing a method of erasure codes and adding a segment ID to the header to enumerate the IDs of the m nonzero coefficients. The overall compression rate is improved and coefficients overhead becomes $m + n/\log q$. Further, the coefficients overhead becomes $m + (\log_2 n)/\log_2 q$ when a list decoding [102] method is used, however this requires a GF with large size and more complicated decoding algorithm. With ordinary GF and efficient decoding algorithms the coefficients overhead becomes close to:

$$m - \frac{m^2}{n} + \frac{(\log_2 n)}{\log_2 q}. \quad (5.2)$$

Danilo.G et. al. in [103] propose another sparse method that only allows selecting coefficients from a small set, Q , of $GF(q)$ elements. Specifically, Q is a subset of the primitive elements of $GF(q)$, and can be agreed upon among the network nodes in advance. Thus, the sender, instead of sending the m coefficients values as an n -coding vector, it can send only a bit vector that contains the coefficients indexes in Q and their positions in the coding vector. The overall coefficients overhead of this scheme is $m(\log_2 |Q| + \log_2 n)$ bits.

Ye.L et. al. in [100], propose a method that split the coefficients overhead into two vectors. The first one contains the m coefficients values. The second vector is an n -bit vector of 0's and 1's. At the receiver, the 1's are replaced with the m coefficients. The overall coefficients overhead of this scheme is

$$m \log(q)/8 + \lceil n/8 \rceil \text{ bytes}. \quad (5.3)$$

A scheme that is very related to our work is proposed in [104]. The scheme reduces the coefficients overhead per packet to be one symbol based on Vandermonde matrices. In the Vandermonde matrix, the entire row can be generated using a single seed. However, this scheme has two limitations. First, because of the cyclic characteristic of the finite field, the generation size is restricted to be $n \leq GF(q)$ which means that in practical scenarios the generation size cannot exceed 32 packets. Second, recoding at intermediate nodes is highly likely to generate linearly dependent coding vectors. Another scheme that uses a single seed which is transferred from the sender to the receiver is proposed in [105]. However, this method requires synchronization among the participant nodes and it completely lacks the recodability at intermediate nodes.

Nilloofar.Y et.al. in [106] propose revolve codes (ReC) which are based on drawing a common element from a large GF, namely $q \geq 2^{16}$ to guarantee low probability of linear dependency, and another n elements from $GF(2^2)$. The small elements are interpreted as one item t and sent along with the large element as a compressed coefficients vector. The receiver generates the complete coefficients vector by repeatedly shifting the small elements item and each time adding to the large element. The overall coefficients overhead of this method is $(\log_2 q + n * t)$.

Most of the above studies are unreliable in terms of recoding or\and generating linearly independent vectors, and not scalable. In specific they do not support wide range of generation sizes. Moreover, the most reliable methods of these studies are sparse, which usually requires more complex recoding strategies to retain the structure of the sparse code [106], [100].

5.3 Chapter Contributions

We contribute by the following:

- We propose a new method, based on modular arithmetic and prime numbers referred to as the One Item Network Coding Vectors (OINCV). The method efficiently compresses the coefficients vectors to a tiny constant value that often does not exceed 4 bytes.
- The proposed method is dense, recodable, and reliable in the sense that it can be real alternative to RLNC without any limitations.
- The proposed method is scalable and it can be used over a wide range of applications and devices, i.e., from wireless sensors networks to high performance computers.
- Both theoretical analysis and simulation are provided to prove the correctness and features of the proposed method.
- Theoretically, the proposed method is compared with all state of art studies in terms of coefficients overhead ratio, and encoding cost.
- Experimentally, the proposed method is compared with all reliable studies in terms of coefficients overhead ratio, download time, throughput, and packets drop rate. The experiments are run based on three different packet sizes namely 128 bytes, 1500 bytes, and 16KB.

5.4 Preliminaries and Design Model

Let P be a prime and $\mathbb{Z}_P = \{0, 1, \dots, P-1\}$, if addition and multiplication are defined in $\mathbb{Z}_P$ as being modulo P , $\mathbb{Z}_P$ can be defined as prime GF of order P , $GF(P)$. On the other hand, $GF(2^r)$ or $GF(q)$ where $q = 2^r$ and $r \geq 2$ is referred to as an extended GF. However, elements in the extended GF are presented as polynomials of degree $r-1$ and arithmetic operations are defined as being modulo irreducible primitive polynomial $P(X)$ of degree r ; such primitive polynomials are analogous to the prime number in $\mathbb{Z}$.

Herein, both prime and extended GF are used. $GF(2^r)$ and $GF(q)$ are used interchangeably and when $GF(P)$ and $GF(q)$ are mentioned together, all $GF(P)$ elements are sub-elements of $GF(q)$ and P is always the largest prime number less than q . Also, P is always notated by big P , while little p_i notation is used to refer to the other primes in $GF(P)$.

The Chinese remainder theorem (CRT) is used to solve a set of congruent equations with one variable but different moduli, which are relatively prime.

Theorem 5.1 (The Chinese remainder theorem (CRT)) *Let $p_1, p_2, \dots, p_n$ are relatively prime positive integers, then the following equations have a unique solution*

modulo $M = p_1 * p_2 * \dots * p_n$.

$$X \equiv a_1 \pmod{p_1}$$

$$X \equiv a_2 \pmod{p_2}$$

$$\vdots$$

$$X \equiv a_n \pmod{p_n}$$

The proof and details about this theorem can be found in [107], [108].

Definition 5.1 (Code Density [109]:) *It is referred to as the ratio of non-zero elements in an coefficients vector relative to the size of the vector. It is calculated by:*

$$d(c) = \frac{\sum_{i=1}^n e_i \neq 0}{n} = \frac{m}{n} \quad (5.4)$$

Full density can be achieved by choosing a coefficient for each cell in the coefficients vector based on random uniform distribution. On the other hand, sparse density can be achieved by allowing several zeros to be positioned in the coefficients vector. Full density is desirable as it yields high degree of innovation amongst packets, while it results in higher coefficients overhead. In practice, sparsity is widely used as it allows the coefficients vector to be compressed and hence yields reduced coefficients overhead. However, the higher the sparsity, i.e., code density becomes closer to 0, the higher the risk to get linearly dependent vectors which leads to even worse overhead. Herein, we set the code density to 0.4, i.e., $m = n/2.5$.

In what follows, we propose dense coding vectors technique called One item net-

work coding vectors (OINCV) based on modular arithmetic and prime numbers. It incurs only 2, 3 or maximum 4 bytes overhead per packet.

5.4.1 Primarily step : generating the primes

For a generation of size n , n primes are generated independently in each network node such that the n generated primes are identical in all the nodes.

Definition 5.2 (Generation Moduli Set (GMS)) *For a network coding generation of size n that runs GF operations, e.g., addition and multiplication, over $GF(2^r)$, the generation primes set is defined as the set of the largest consecutive $n - 1$ primes picked up from $GF(P)$ besides P itself, such that $GMS = \{p_1, p_2, \dots, p_{n-1}, P\}$.*

For example and without loss of generality the GMSs for 4-packets generation that operates over different GFs are listed in Table 5.1. The reasons behind selecting the

Table 5.1: Examples of GMSs

Extended GF	Prime GF	GMS
$GF(64)$	$GF(61)$	$\{49, 53, 59, 61\}$
$GF(128)$	$GF(127)$	$\{107, 109, 113, 127\}$
$GF(256)$	$GF(251)$	$\{233, 239, 241, 251\}$

largest n primes are twofold: 1) to make the nodes synchronized, namely all the nodes generate the same GMS without the need to transfer the primes and 2) to leverage the items of the coding vector, namely, the modulo primes operation will give more items combination within a coding vector, thereby reducing linearly dependent vectors probability.

We emphasize that $GF(P)$ is used to creating GMS, and generating or regenerating

coefficients vectors, while the needed operations by network coding are still performed over $GF(q)$.

5.4.2 Encoding

Definition 5.3 (Traversed Coefficient (I)) *is an integer number selected randomly by the sender such that it is a 2, 3, or 4 bytes and then it is sent as a header of the coded packet to generate the whole coefficients vector on the receiver side.*

Definition 5.4 (Allowed Selection Range (ASR)) *is a set where the traversed coefficients are allowed to be selected. e.g., the set of integer numbers in the range $[P + 1, 2^{16} - 1]$, $[P + 1, 2^{24} - 1]$, or $[P + 1, 2^{32} - 1]$. An appropriate ASR is chosen based on the generation size*

For instance, for a RLNC with generation size of 32, it can use $GF(P = 251)$ to have a GMS with 32 primes and the RLNC operations can be run over $GF(q = 256)$, and hence, the appropriate ASR for such system is in range $[252, 2^{16} - 1]$. Encoding begins by generating a proper GMS that fits with the generation size, thereafter, the sender selects a traversed coefficient I from an ASR set, then, a vector of integers is constructed as follows:

$$[I \bmod p_1, I \bmod p_2, \dots, I \bmod p_{n-1}, I \bmod P]$$

These vector elements are $GF(P)$ integers therefore they can easily be transformed to the corresponding $GF(q)$ elements, such that the coding coefficients vector becomes



Figure 5.1: Two network coding packets: (a) RLNC's packet, and (b) OINCV's packet as:

$$\zeta = [e_1, e_2, \dots, e_{n-1}, e_n]$$

To compose the coded packet, each coefficient from ζ is multiplied by its corresponding plain packet and then packets are combined, exactly as in RLNC. However, instead of sending the complete coding coefficients vector ζ , the sender sends only the traversed coefficient I along with the payload. For example assume a generation size of 32 packets and RLNC is applied over $GF(2^8)$. Fig. 5.1 shows how OINCV can reduce the network coding overhead from 32 bytes for the RLNC's packet to only 2 bytes for OINCV.

5.4.3 Decoding

Once the receiver received the coded packet, it uses the traversed coefficient to construct the coefficients vector ζ , in the same way the sender does. The receiver keeps filling the coefficients matrix until it is a full rank, and then solves using Gaussian elimination to get the original generation packets. The steps are presented in Fig.5.2.

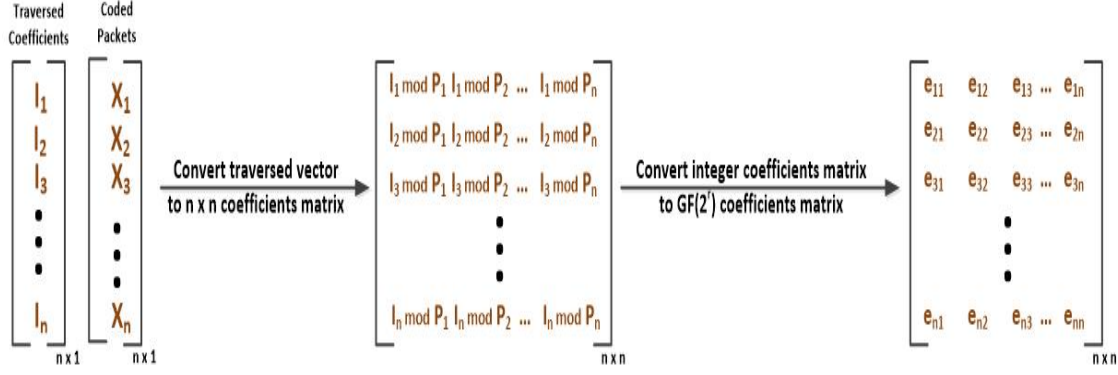


Figure 5.2: Decoding process at the receiver side

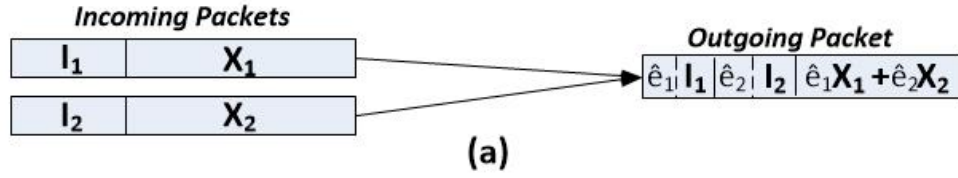
5.4.4 Recoding

An intermediate node, e.g. a router, recodes the incoming packets by randomly selecting two packets out of all the received packets and new two coding coefficients ($\hat{e}_1$ and $\hat{e}_2$) from $GF(2^8)$. The intermediate node then multiplies each coefficient by its corresponding packet's payload. Thereafter, linearly combines the new payloads together to produce the outgoing packet. Two packets are sufficient since the proposed method is fully dense. On the other hand, the overhead part is composed by attaching the new selected coefficients ($\hat{e}_1, \hat{e}_2$), along with the original traversed coefficients (I_1, I_2) of the two selected packets to the outgoing packet as shown in Fig. 5.3 (a). The reason for attaching the coefficient as opposed to combining them is that if ($\hat{e}_1, \hat{e}_2$) are multiplied by (I_1, I_2), then (I_1, I_2) are almost impossible to be recovered on the receiver side. Thus, attaching these two coefficients reduces the extra overhead from n to a maximum 10 bytes as will be shown shortly.

When the receiver receives the recoded packet, it uses the traversed coefficients to construct the coding coefficients vectors as shown in the previous subsection. Next, each attached coding coefficient ($\hat{e}_i$) is multiplied by its corresponding coefficients

vector $((I_1))$. Next, the two coding vectors are linearly combined to produce one vector. Finally, this vector is added as a row vector to the coefficients matrix. The complete process is illustrated in Fig .5.3 (b). The payload part of the recoded packet is not touched on the receiver side and it is added as is to the coded packets column.

Router



Receiver

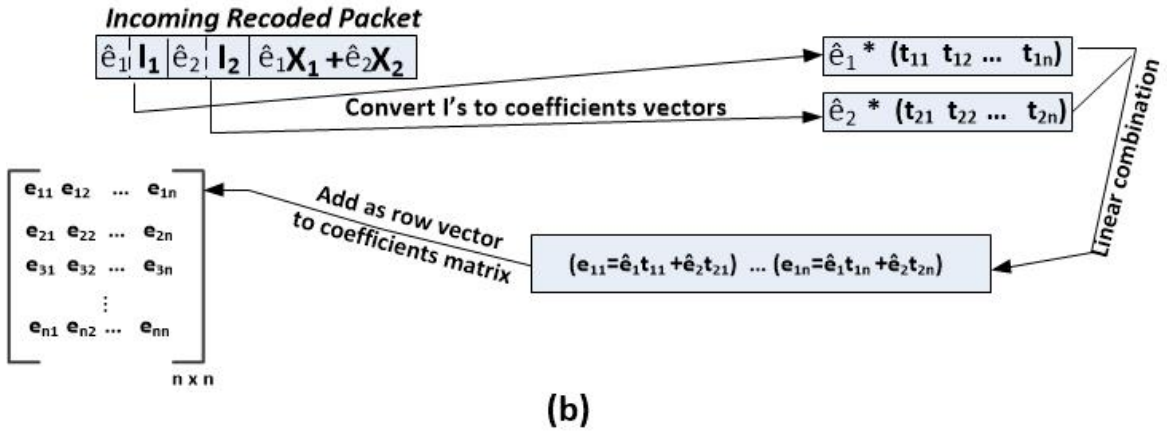


Figure 5.3: (a) Recoding process at an intermediate node and (b) Decoding of a recoded packet at the receiver.

Thus, if the recoding is required, then the process is accomplished with an extra coefficients overhead cost which increases as the number of the packets to be recoded together increases. However, since the proposed method coefficients vectors are fully dense, then only two coded packets are enough to produce a good recoded packet and this highly reduces the extra overhead cost. The extra overhead cost will be discussed in the next section

Two essential questions about the proposed method arise. First, since the servers are not co-located and have no coordination, and the traversed coefficients are selected randomly, then selecting the same traversed coefficient by different two servers leads to constructing two identical coefficients vectors and as a consequence two linearly dependent vectors. Hence what is the probability that two or more servers select the same traversed coefficient? The second question is will the two different traversed coefficients, I_1 and I_2 , generate the same coefficients vectors after the modulo operation over the GMS? These questions are addressed and analyzed in the next section.

5.5 Theoretical Analysis and Proof of Correctness

In this section, we provide theoretical analysis and prove the correctness of the proposed method. In addition, we compare the proposed method with all the previous studies in terms of coefficients overhead and encoding cost.

5.5.1 Analysis of network coding coefficients overhead

If recoding is not required, then the coding coefficients overhead incurred by OINCV is a very tiny constant per packet regardless of the generation size, namely, it is 2, 3, or 4 bytes per packet and $2n$, $3n$, $4n$ bytes per generation. In case of recoding, the analysis considers that a recoded packet is composed by combining two coded packets only and the receiver is one-hop away from the server, i.e, only one intermediate node is assumed between the sender and the receiver. Thus in the recoding case, the overhead contains two traversed coefficients plus two selected coefficients $(\hat{e}_1, \hat{e}_2)$, at

the intermediate node, whose size is 2 bytes and hence the overall coefficients overhead incurred by the proposed method per packet becomes 6, 8, or 10 bytes.

Whether recoding is required or not, the proposed method compared to the other state of art studies can reduce the coefficients overhead to a very large extent. Recall that $m = \lceil n/2.5 \rceil$, Fig. 5.4 (a), Fig. 5.4 (b), and Fig. 5.4 (c) present results for theoretical coefficients overhead per generation (plotted on log scale), when the recoding is not required, for small, moderate, and large generations sizes, respectively. The evaluation is performed for OINCV using 2, 3, and 4 bytes traversed coefficient, referred to as OINCV2, OINCV3, and OINCV4, respectively. The figure shows that OINCV always incurs the minimum overhead and the difference in the overhead value between OINCV and the other methods increases as the generations sizes get larger. Moreover, the figures show that the proposed method is viable for all the generations sizes, while some methods are vanished for moderate and large sizes.

Figure. 5.5 shows the same results with recoding. Although the method has been proposed in [104] has less coefficient overhead when the generation size is 16, it is prone to extreme linearly dependent if recoding is required and it is viable only when $n \leq 32$ [104].

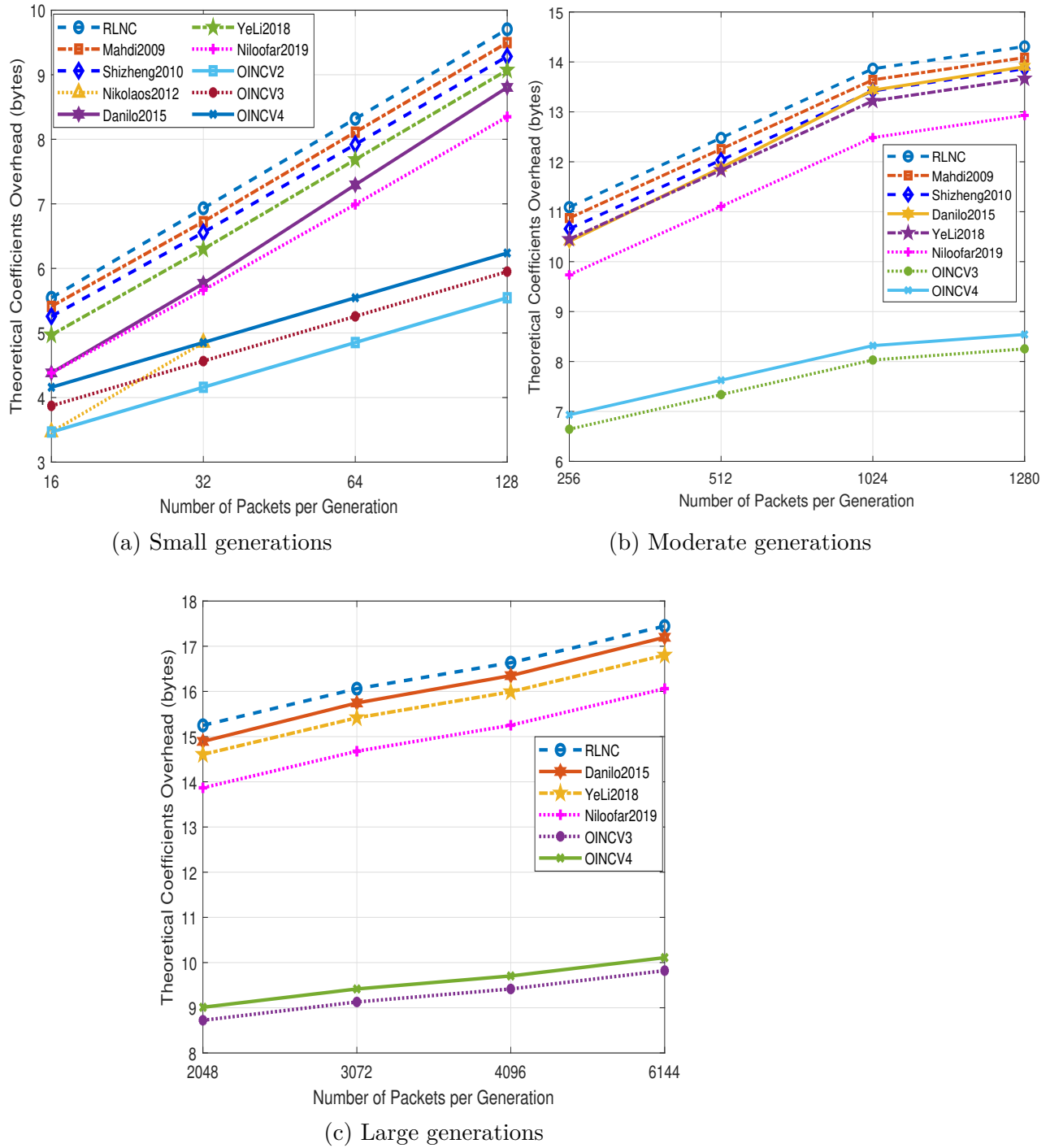
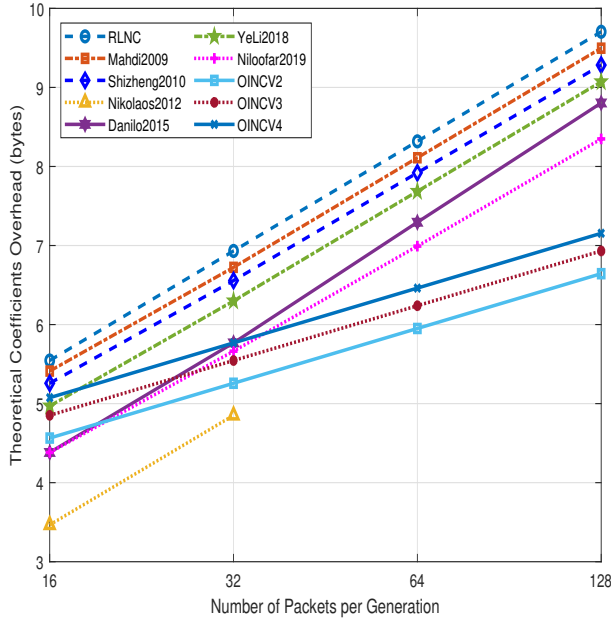
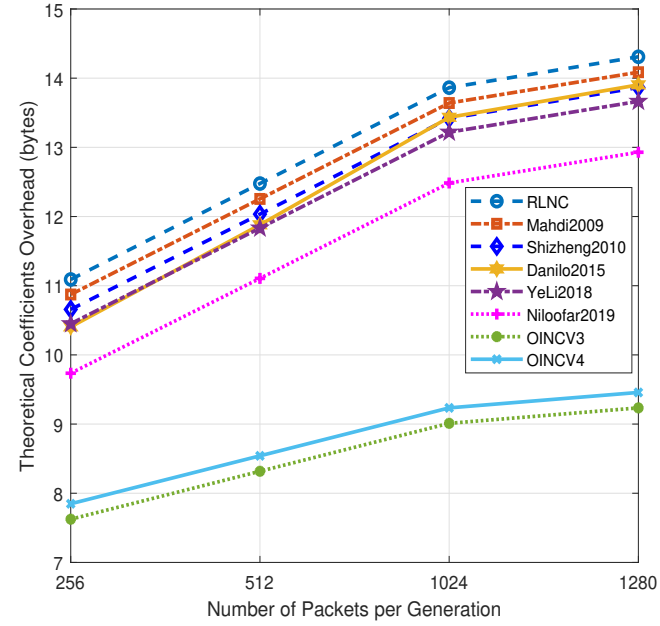


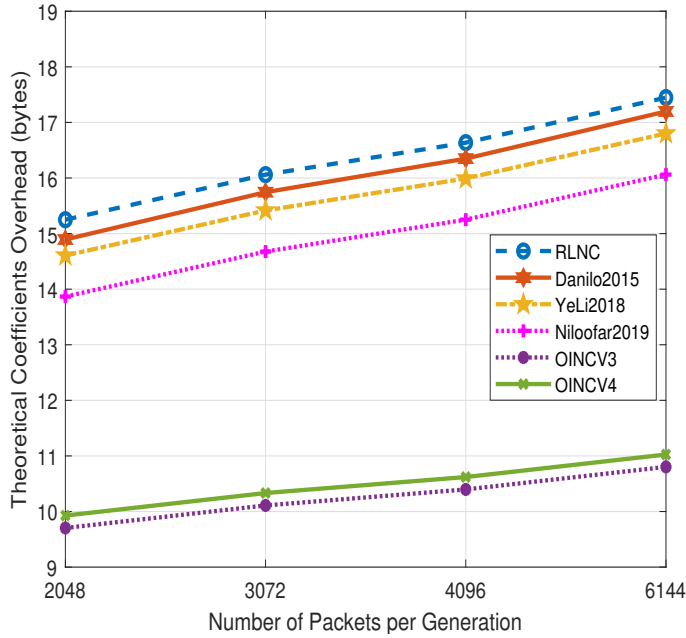
Figure 5.4: Log scale plot of network coding coefficients overhead without recoding



(a) Small generations



(b) Moderate generations



(c) Large generations

Figure 5.5: Log scale plot of network coding coefficients overhead with recoding

5.5.2 Analysis of selecting different values from ASR

To decode a generation of n packets successfully, it needs n distinguishable traversed coefficients to be selected from ASR. we examine the probability of selecting n distinguishable traversed coefficients.

Proposition 5.1 *The probability that n traversed coefficients selected from an ASR are distinguishable is given by:*

$$P_{dist}(I_1, I_2, \dots, I_n) = \prod_{i=1}^n \frac{|ASR| - (i - 1)}{|ASR|}. \quad (5.5)$$

Proof. For the first selection, the probability that the traversed coefficient, I_1 , is distinguishable is equal to 1, $P_{dist}(I_1) = 1$. For the second selection, $P_{dist}(I_2) = \frac{|ASR|-1}{|ASR|}$, for the third selection, $P_{dist}(I_3) = \frac{|ASR|-2}{|ASR|}$ and so on for the next selections. Finally, for the n -th selection, $P_{dist}(I_n) = \frac{|ASR|-(n-1)}{|ASR|}$. Since these are independent events, then the overall probability of selecting distinguishable traversed coefficient from the ASR for a generation of size n is given by:

$$P_{dist}(I_1) \times P_{dist}(I_2) \times \dots \times P_{dist}(I_n) = \prod_{i=1}^n \frac{|ASR|-(i-1)}{|ASR|} \quad \blacksquare$$

Figure. 5.6 shows the probability of selecting distinguishable traversed coefficients based on (5.5) and by simulation for different generation sizes. Also, traversed coefficients of sizes 2, 3, and 4 bytes are considered and denoted in the figure as OINCV2, OINCV3, OINCV4, respectively. For the simulation, each experiment has been run for 1000 repetitions then the average result is taken. From the figure it can be concluded

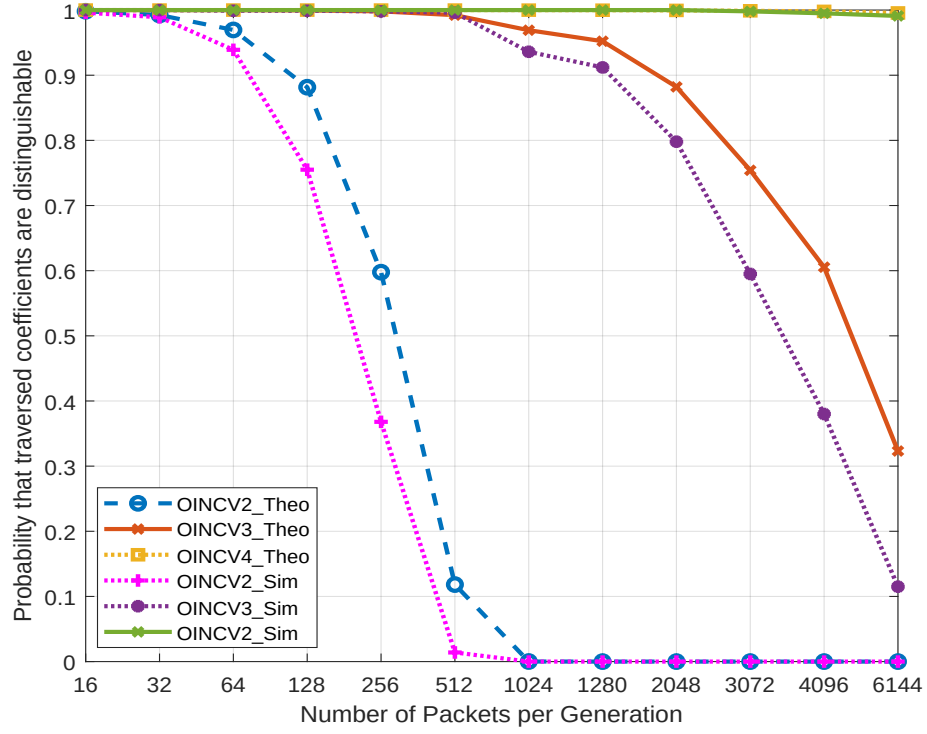


Figure 5.6: Probability that traversed Coefficients are distinguishable; theoretical vs simulation

that the 2 bytes traversed coefficient can be only used with small generations sizes , i.e., size ≤ 32 , since GMSs for these generations can be picked up from $GF(P = 251)$ and the GF operations can be run over $GF(2^8)$ which is compatible with computer arithmetic which stores and sends the data as multiple of 8-bit words. Larger generations, with size ≥ 64 , must not consider 2 bytes traversed coefficient for the following reasons: First, they need that their GF operations run over GFs greater than $GF(2^8)$, for example, there is no 64 primes less than 256 and thus a 64 packets generation needs to use $GF(2^9)$ which leads to chunk a large packet into a 9 bits blocks which in turn violates the computer and network convenience, and the second reason is that the probability of getting distinguishable traversed coefficients is decreased as the generations get larger as shown in the figure. When the traversed coefficient is of size 3

or 4 bytes, the operations coefficients are generated over $GF(2^{16})$ since the GMS is picked up from $GF(P = (2^{16} - 15) = 65521)$ and the figure shows that the 3 bytes traversed coefficient can be used with moderate generations, i.e., between 64 and 512, and the 4 bytes traversed coefficient can be used with large generations, indeed, 4 bytes traversed coefficient can be used for any generation at the cost of extra one or two bytes overhead.

Theorem 5.2 *The arbitrarily selected n traversed coefficients from an appropriate ASR are distinguishable with high probability.*

Proof. The proof is constructive by substituting in (5.5) and the observations from Fig.5.6.

- for small generations that contain 32 packets or less, if ASR ranges in $[252, 2^{16} - 1]$, then $|ASR| = 65284$ and $P_{dist} \geq 0.99$.
- for moderate generations that contain packets between 64 and 512, if ASR ranges in $[65522, 2^{24} - 1]$, then $|ASR| = 8323086$ and $P \geq 0.99$.
- for large generations that contain packets between 1024 and 6144, if ASR ranges in $[65522, 2^{32} - 1]$, then $|ASR| = 4294901774$ and $P \geq 0.99$.
- Moreover, all these calculations are supported by simulation as shown in Fig. 5.6.

■

5.5.3 Analysis of generating two identical coding vectors

We showed in the previous subsection that the probability of selecting two identical traversed coefficients can be minimized to approach 0. In this subsection, we address the questions, do two different traversed coefficients modulo the same GMS generate the same coding coefficients vector? what the probability that two or more different coding coefficients vectors are linear dependent?

Theorem 5.3 *Two different traversed coefficients modulo the same GMS do not generate identical coefficients vectors.*

Proof. This can be proved by contradiction. Assume we have a coding coefficients vector $\zeta = [e_1 e_2 \dots e_n]$ which is generated by a traversed coefficient (I_1) modulo GMS.

step1: Assume that there is another traversed coefficient (I_2) that can generate the same coding coefficients vector ζ .

step2: Let ζ vector be written as a system of simultaneous linear congruence:

$$I_1 \equiv e_1 \pmod{p_1}$$

$$I_1 \equiv e_2 \pmod{p_2}$$

$$\vdots$$

$$I_1 \equiv e_n \pmod{P}$$

since $p_1, p_2, \dots, P$ are relatively prime, then utilizing the CRT the above system has a unique solution which is I_1 . As a consequence, I_2 can not generate ζ unless $I_2 = (p_1 \times p_2 \times \dots \times P) + I_1$ but since this value is far larger than I_1 , it is out of ASR scope.

■

By Theorem 5.2 and 5.3 we have proved the viability and the reliability of OINCV method. Finally and for the sake of completeness, we analyze the probability that two or more different coefficients vectors generated by the proposed method are linearly dependent.

It is well known that an $n \times n$ matrix over $GF(2^r)$ is full rank with high probability for large r , i.e., $r \geq 8$ and the proof is mathematically given by [110]:

$$\prod_{i=0}^{n-1} \frac{q^n - q^i}{q^{n \times n}} \quad (5.6)$$

Since $GF(P)$ contains at least 98% of $GF(2^r)$ elements, for $8 \leq r \leq 16$, then an $n \times n$ matrix over $GF(P)$ is full rank with almost the same high probability.

Lemma 5.1 *An $n \times n$ coefficients square matrix with random choice of its elements from $GF(P)$ is full rank with high probability, w.h.p and mathematically is given by:*

$$\prod_{i=0}^{n-1} 1 - \frac{P^i}{P^n}. \quad (5.7)$$

Proof. By building up the $n \times n$ matrix from the scratch. The first row can be any n -tuple vector over $GF(P)$, therefore we have $P^n - 1$ choices, excluding the zero vector. The second row must be linearly independent from the first one and since there are P linearly dependent vectors of the first row, then we have $P^n - P$ choices. For the third row to be linearly independent from the first and the second rows, we have $P^n - P^2$ and so on. In general, for the i^{th} row to be linearly independent from the previous $(i - 1)$ -rows, we have $P^n - P^{i-1}$ choices. Once we build the matrix by this

way, we know that its rows are linearly independent and thus the matrix is invertible.

Hence, the number of invertible matrices is given by

$$(P^n - 1) \times (P^n - P) \times \dots \times (P^n - P^{n-1}) = \prod_{i=0}^{n-1} (P^n - P^i) \quad (5.8)$$

The overall $n \times n$ matrices over $GF(P)$ is given by:

$$P^{n \times n} \quad (5.9)$$

divides (5.8) by (5.9)

$$\begin{aligned} &= \frac{\prod_{i=0}^{n-1} (P^n - P^i)}{P^{n \times n}} \\ &= \frac{(P^n - 1) \times (P^n - P) \times \dots \times (P^n - P^{n-1})}{P^{n \times n}} \\ &= \frac{P^n(1 - \frac{1}{P^n}) \times P^n(1 - \frac{P}{P^n}) \times \dots \times P^n(1 - \frac{P^{n-1}}{P^n})}{P^{n \times n}} \\ &= (1 - \frac{1}{P^n}) \times (1 - \frac{P}{P^n}) \times \dots \times (1 - \frac{P^{n-1}}{P^n}) \end{aligned}$$

$$= \prod_{i=0}^{n-1} 1 - \frac{P^i}{P^n}.$$

I

5.5.4 Analysis of encoding cost

Definition 5.5 (Encoding Cost:) *This refers to the time needed to compose an encoded packet from plain packets. Precisely, this time involves the addition and multiplication operations over $GF(2^r)$ needed to generate a such packet.*

Since in the proposed method the GF operations are run over $GF(2^{16})$ for generations sizes ≥ 64 , thus the whole data packet is represented by fewer GF symbols than the other methods which usually run GF operations over $GF(2^8)$ or $GF(2^4)$. This leads to minimizing the number of addition and multiplication operations. For instance, for a packet of size 1500 bytes if the used GF is $GF(2^8)$, then the packet is presented by 1500 symbols which means that an encoded packet needs $1500n$ addition and multiplication operations. On the other hand, if the used GF is $GF(2^{16})$, then the same packet is presented by 750 symbols which in turn minimizes the number of addition and multiplication operations to $750n$.

Although the cost for a single operation over $GF(2^{16})$ is more expensive than the cost of the same operation over $GF(2^8)$, the encoding cost of the proposed method is always less than that for RNLC and RLNC-like methods. This is because the number of operations is minimized to the half and most software and hardware implementations are using lookup tables for multiplication which makes the time per an operation is almost the same even when the field sizes are different.

Figure. 5.7, Table 5.2 , and Table 5.3 present results of encoding cost measured in seconds of 128 and 1500 bytes packets for generations range between 64 to 1280. The results show that whether the lookup tables are used or not, OINCV has less encoding

cost than RLNC.

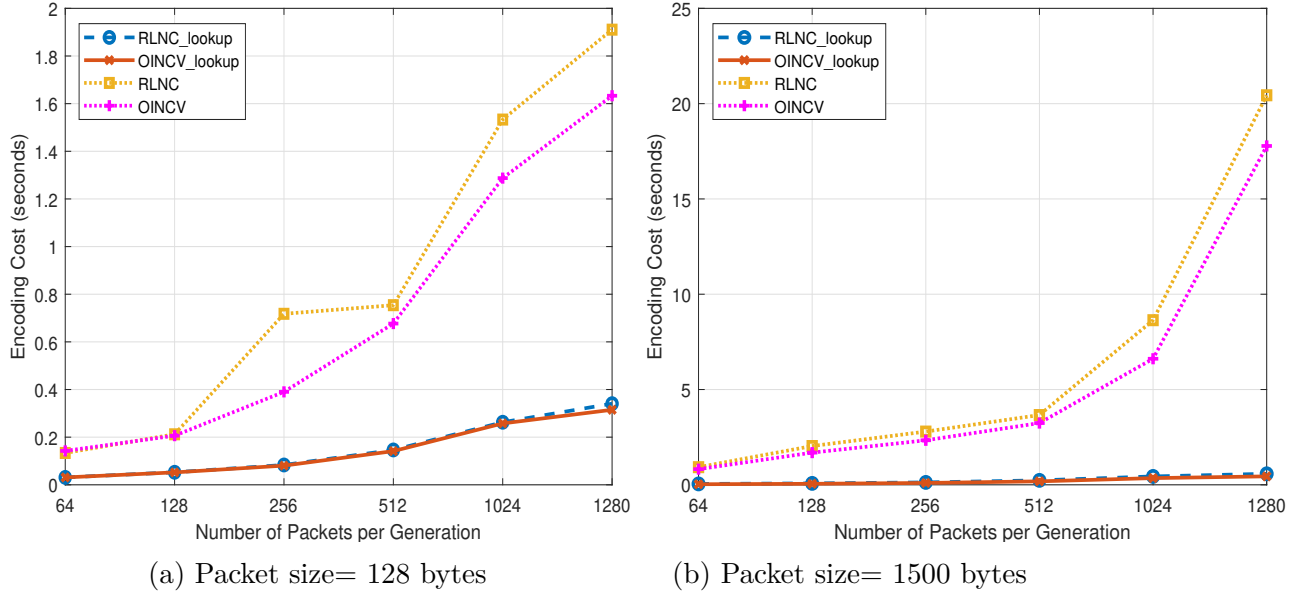


Figure 5.7: Encoding cost comparison between RLNC and OINCV

Table 5.2: Encoding cost (secs) of a 128 bytes packet

Generation Size	64	128	256	512	1024	1280
RLNC	0.133	0.212	0.717	0.754	1.533	1.910
OINCV	0.132	0.206	0.390	0.676	1.287	1.633
RLNC _{LT}	0.031	0.052	0.083	0.146	0.262	0.340
OINCV _{LT}	0.031	0.052	0.080	0.141	0.257	0.314

Table 5.3: Encoding cost (secs) of a 1500 bytes packet

Generation Size	64	128	256	512	1024	1280
RLNC	0.924	2.034	2.793	3.660	8.637	20.439
OINCV	0.821	1.689	2.335	3.239	6.614	17.78
RLNC _{LT}	0.046	0.074	0.125	0.232	0.440	0.578
OINCV _{LT}	0.029	0.053	0.097	0.181	0.351	0.442

5.6 Experimental Work

In this section, we consider a dissemination protocol that has multi servers which disseminate their data by performing network coding to one or multi receivers. The servers have no coordination among themselves. Also, we consider different file sizes that vary from 16 KB to 20 GB to cover wide range of applications and networks from wireless sensors to large scale P2P content distribution networks. OINCV is compared with the most reliable state of the art methods which are RLNC of S.Li et. al. [101], and the method of Ye.L et. al.[100].

The evaluation considers the following performance metrics: coding coefficients overhead measured in KB, number of dropped packets at the receiver side, download time measured in terms of number of rounds, and throughput measured as number of packets per round. The remaining details needed for each network scenario are given in the respective subsections.

5.6.1 Wireless Sensors Network Case Study

In this case scenario, we consider the following settings:

- Data of sizes 16KB, 32KB ,64KB, and 128KB are shared by two servers to one client who is zero, one, or two hops away from the servers. The servers share their packets generation by generation, namely, sequential scheduling policy. The client sends acknowledgement only when it is received all generation's packets.

- We consider a Zigbee packet which is 128 bytes but since the packet header is 28 bytes we have only 100 bytes under our control.
- We assume generation size of 32 packets [21], link bandwidth of 2 packets/round, the loss probability due to channel erasure is $\epsilon = 0.2$, and the acknowledgement is likely to get lost with probability $\alpha = 0.1$.

Based on these settings, we can specify the packet payload and coefficients overhead, and compute accordingly how many generations the data must be divided for each of the considered methods. For instance, when the data file size is 16KB and the RLNC method is used, then the coding coefficients overhead per packet is equal to the generation size, i.e., 32 bytes, and thus the remaining 68 bytes are for the payload. To specify the number of generations, first the data size is divided by the payload to get the number of packets, $\lceil 16KB/68B \rceil = 241$ packets, then the number of packets is divided by the generation size, $\lceil 241/32 \rceil = 8$ generations. For the methods proposed in [101] and [100], first the coefficients overhead is calculated according to (5.2) and (5.3), respectively, then the other specifications are determined. For OINCV the coefficients overhead is 2 bytes since GMS can be generated from $GF(2^{16})$, and therefore I can be selected from $[252, 2^{16} - 1]$. Table 5.4 summarizes the specification for all methods. The number following the OINCV word indicates the traversed coefficient size, e.g., *OINCV2* means OINCV that uses 2 bytes traversed coefficient.

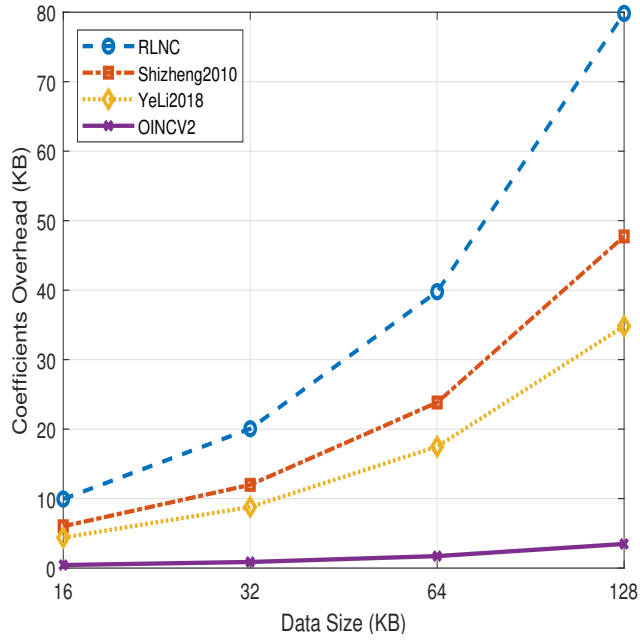
Figure. 5.8(a) shows results of the coefficients overhead in KB; the results indicate that the proposed method always incurs the minimum overhead beginning with 0.43 KB and ending with 3.5 KB, while the overhead for next best method begins with

4.3 KB and ends with almost 35 KB. Figure. 5.8(b) shows the download time results in terms of rounds. Since the proposed method has the minimum overhead, then the majority of the packet is occupied by the payload which leads to split the whole data file into fewer packets and generations, and hence less download time. Figure. 5.8(c) shows the throughput results measured in packets/round. Although the bandwidth of the channel is 2 packets/around, the maximum throughput (ideal) is 1.6 packets/round since $\epsilon = 0.2$. The OINCV throughput is 1.48 packets/round which is the maximum amongst the others and the closest to the ideal throughput. Figure. 5.8(d) shows results of the number of dropped packets at the receiver side. Basically, packets are dropped at the receiver side when packets of a generation are received while the generation is already completed. This action happens because of the loss of the acknowledgement or/and the absence of servers coordination. Since OINCV has always the minimum number of generations and thus fewer acknowledgement to be sent, it has always the minimum number of dropped packets.

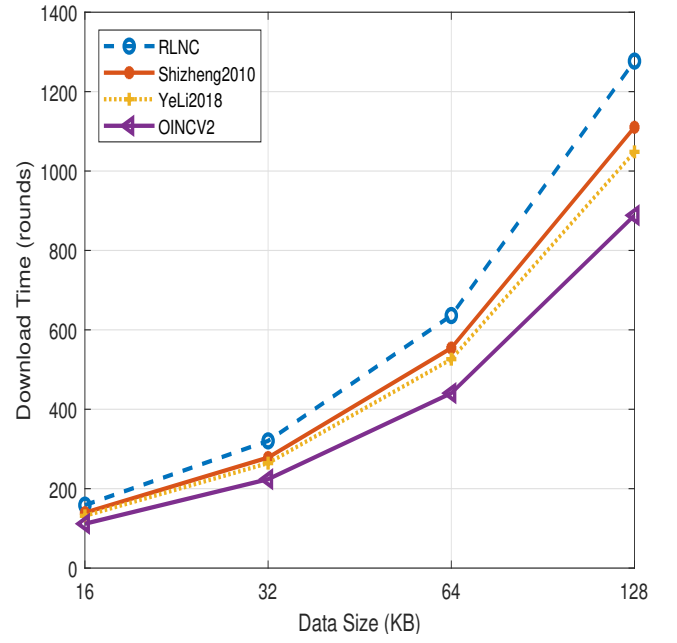
Because OINCV overhead is constant and, in contrast to the other methods, it is not affected by the generation size, it turns out that the generation size of 32 is not optimal choice for OINCV. Indeed, OINCV can work on generation size of 100 packets and perform better with extra one byte overhead within a packet. Figure. 5.9

Table 5.4: Specifications for different NC methods

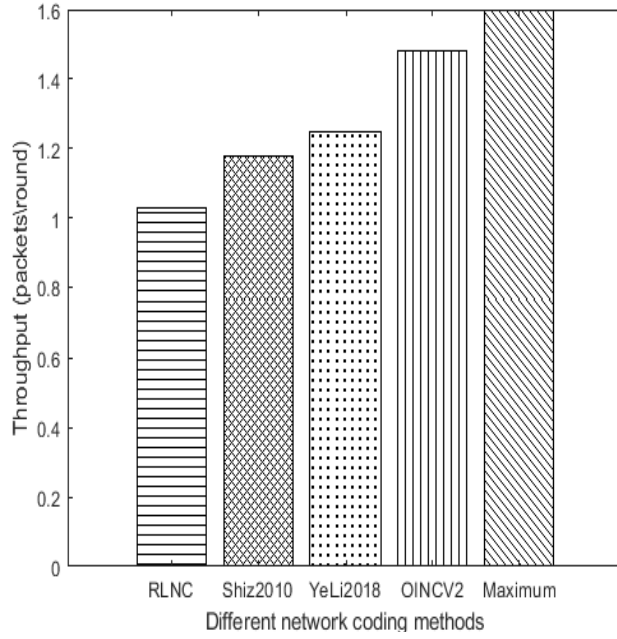
NC Methods	Packet Payload (Bytes)	Packet Overhead (Bytes)	Number of generations			
			16KB	32KB	64KB	128KB
RLNC	68	32	8	16	31	61
Shizheng2010	78	22	7	14	27	53
YeLi2018	83	17	7	13	25	50
OINCV2	98	2	6	11	21	42



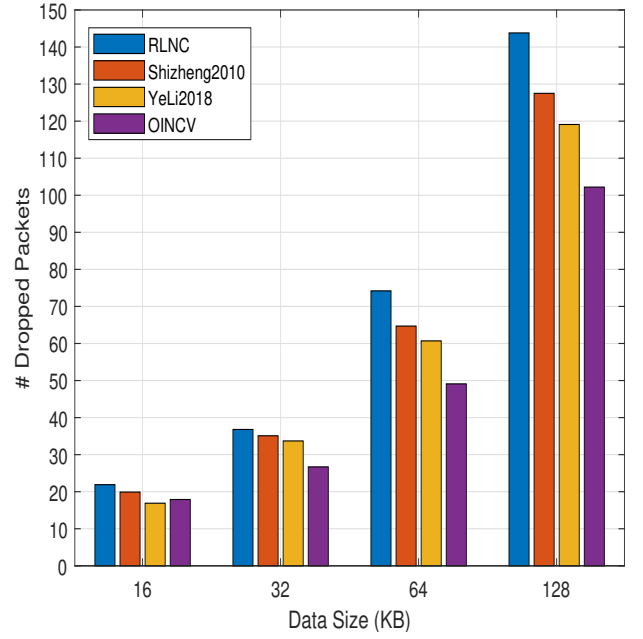
(a) Coefficients Overhead



(b) Download Time



(c) Throughput



(d) Dropped Packets

Figure 5.8: Performance metrics of 0-hops WSN for different NC methods

compares OINCV of 32 packets generation size with OINCV of 100 packets generation size, referred as $OINCV_{opt}$ in terms of coefficients overhead, download time, through-

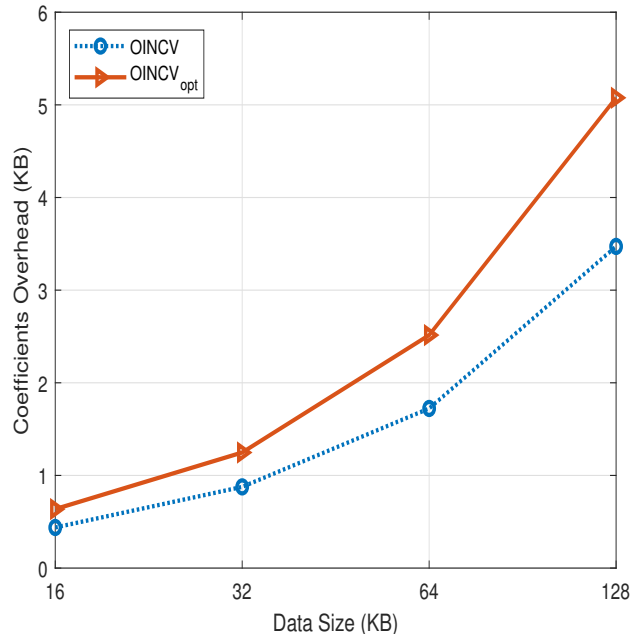
put, and number of dropped packets. Results show that the coefficients overhead of $OINCV_{opt}$ is slightly larger than OINCV and this is because for a generation size of 100 the traversed coefficient (I) can be selected up to 2^{24} which yields to 3 bytes coefficients overhead per packet. The results also show that all the other metrics are improved, e.g., throughput reaches 1.54 packets/round and the number of dropped packets is reduced to at least 60%.

Minimum Coordination among the servers

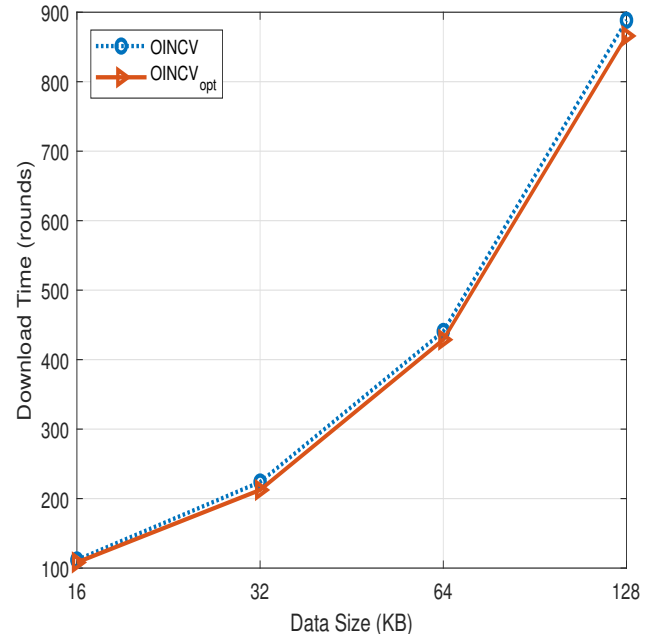
Further to reduce the coefficients overhead and maintain the generation size equals to 100, a traversed coefficient with 2 bytes can still be used but with GMS generated from $GF(P = 1021)$ and thus ASR ranges in $[1022, 2^{16} - 1]$. However, this ASR range for 100 packets generation may lead to a probability equals up to 0.2 that two or more selected traversed coefficients are identical since the servers have zero coordination among themselves.

Minimum coordination assumes that the servers can narrowly cooperate with each other such that the ASR is divided into subsets and each subset contains 100 values, e.g., $[1022, 1121]$, $[1122, 1221]$... etc. Then if each server is specified to one subset to select its traversed coefficients, the probability of selecting identical traversed coefficient approaches to 0. For ASR ranges in $[1022, 2^{16} - 1]$, $|ASR| = 64513$, and generation size equals to 100, this scheme supports up to 645 servers.

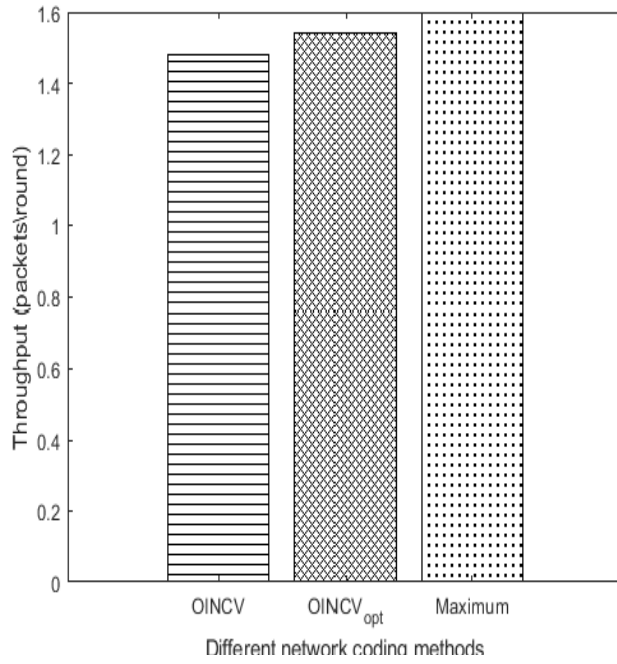
Figure. 5.10 compares OINCV of 32 packets generation size, $OINCV_{opt}$, and OINCV of 100 packets generation that uses 2 bytes traversed coefficients, referred as $OINCV_{optPlus}$, for the same metrics. Results show that the coefficients overhead is



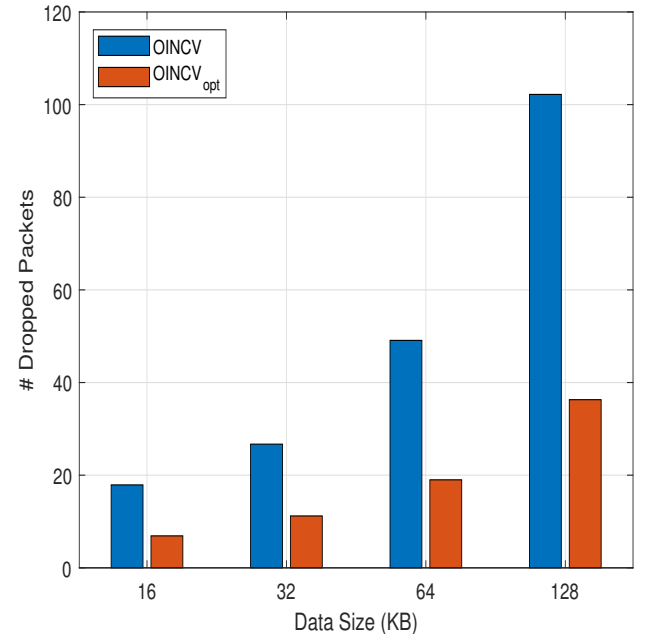
(a) Coefficients Overhead



(b) Download Time



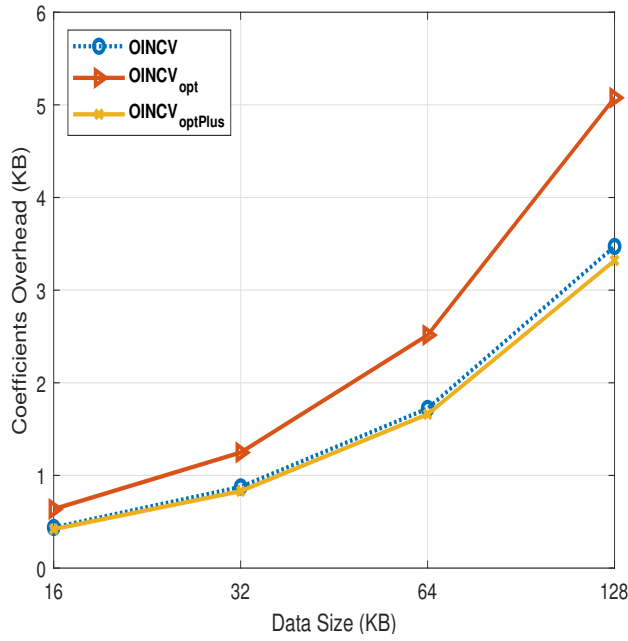
(c) Throughput



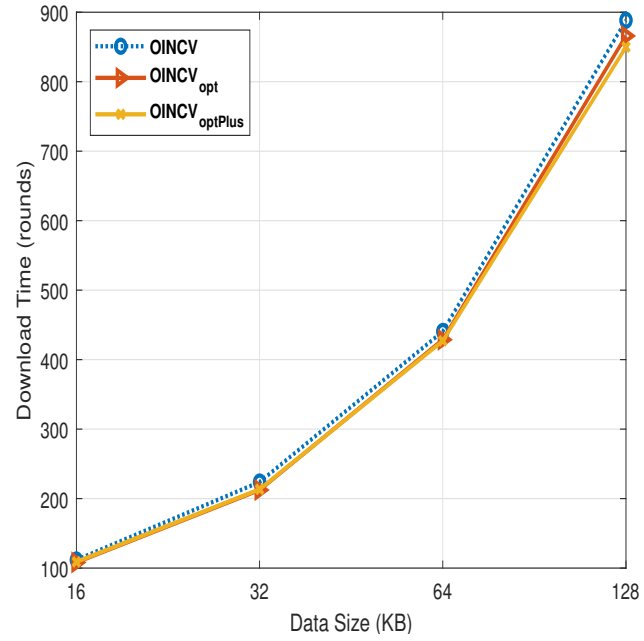
(d) Dropped Packets

Figure 5.9: Performance metrics of 0-hops WSN; OINCV vs. $OINCV_{opt}$

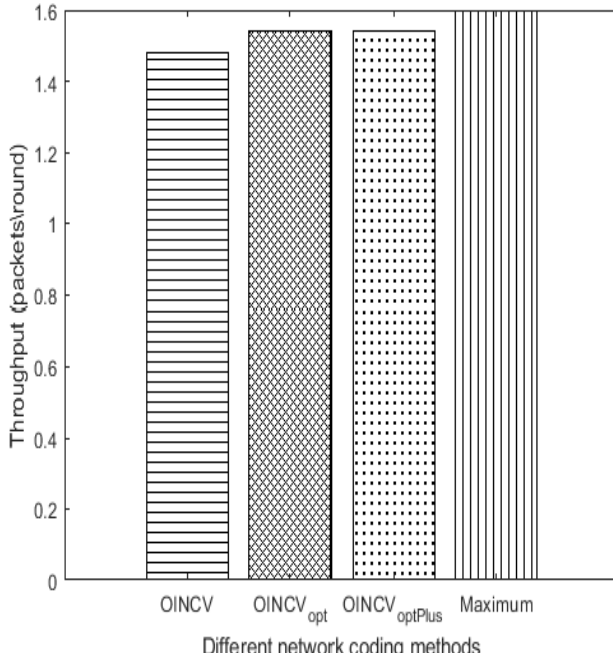
maintained at the same level of $OINCV_2$ while the other performance metrics are maintained at the same level of $OINCV_{opt}$ or even improved.



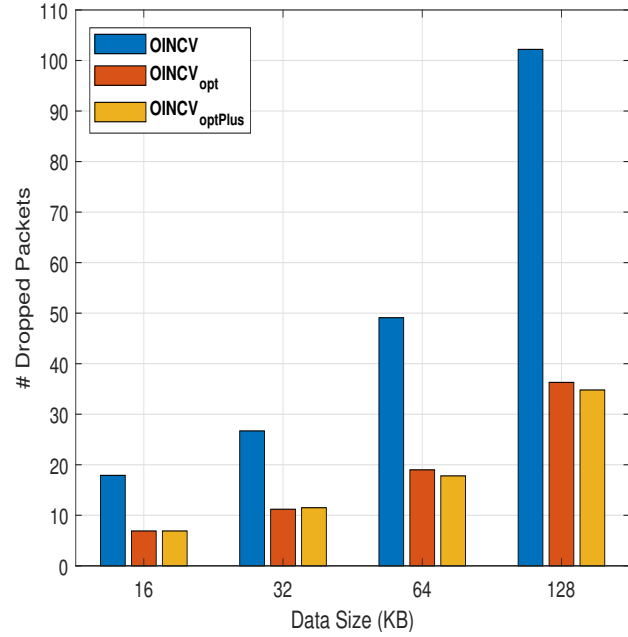
(a) Coefficients Overhead



(b) Download Time



(c) Throughput



(d) Dropped Packets

Figure 5.10: Performance metrics of 0-hops WSN; OINCV vs. $OINCV_{opt}$ vs. $OINCV_{optPlus}$

Recoding

In this section, we repeat the previous experiments with the same settings except now the servers are 2-hops away from the receiver as shown in Fig. 5.11, and thus

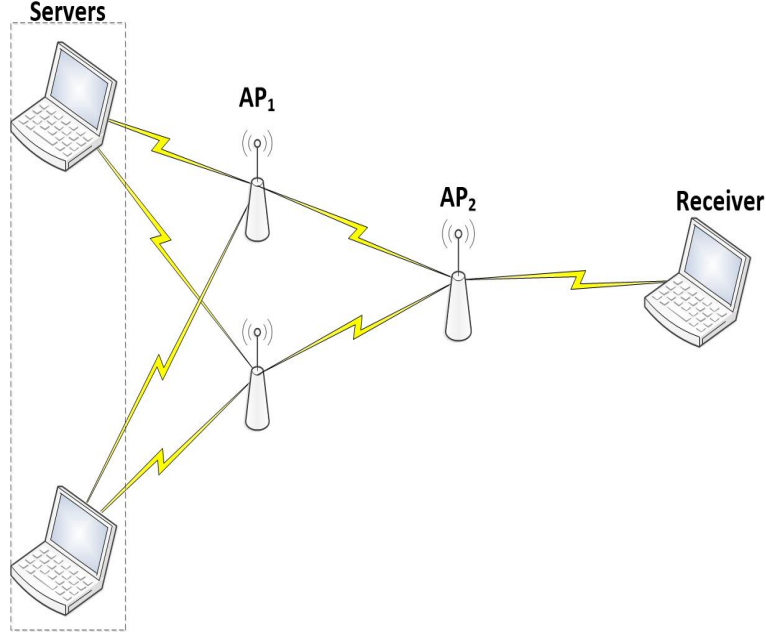


Figure 5.11: 2-hops WSN with two servers and one receiver.

recoding is required. We assume that the links between the servers and access points (APs), and the ones between the AP and the receiver have the loss probability of $\epsilon = 0.2$. It is also assumed that the links between the three APs are error free. Table 5.5 summarizes the new specifications for OINCV and $OINCV_{opt}$, considering that recoding of the other methods does not need any extra cost.

Figure. 5.12 presents results of the new network topology for the same metrics. Naturally, the performance degrades because of the existence of intermediate nodes between the servers and the receiver. The results show that although OINCV has extra cost when the recoding is required, it still outperforms the other methods for

Table 5.5: New Specifications of OINCV and $OINCV_{opt}$

	Packet Payload (Bytes)	Packet Overhead (Bytes)	Number of generations			
			16KB	32KB	64KB	128KB
OINCV2	92	8	6	12	23	45
$OINCV_{opt}$	90	10	2	4	8	15

all the metrics. Indeed, OINCV can outperform the other methods as long as the route between the servers and the receiver is less than or equal to 11 hops as shown in Fig. 5.13. When the number of hops is greater than or equal to 12, the method proposed by [100] begins to outperform OINCV.

Effect of Sensitivity Analysis on OINCV

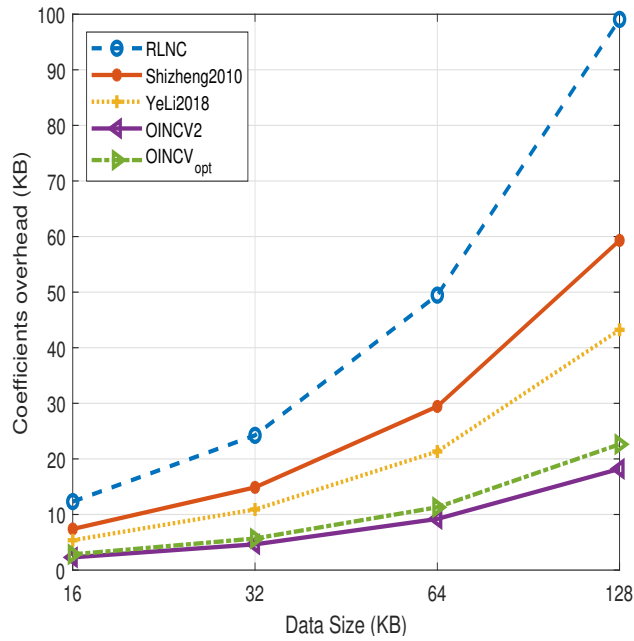
By varying the loss probability (ϵ), the effect of sensitivity analysis is shown in Fig 5.14.

5.6.2 Ethernet Protocol Network

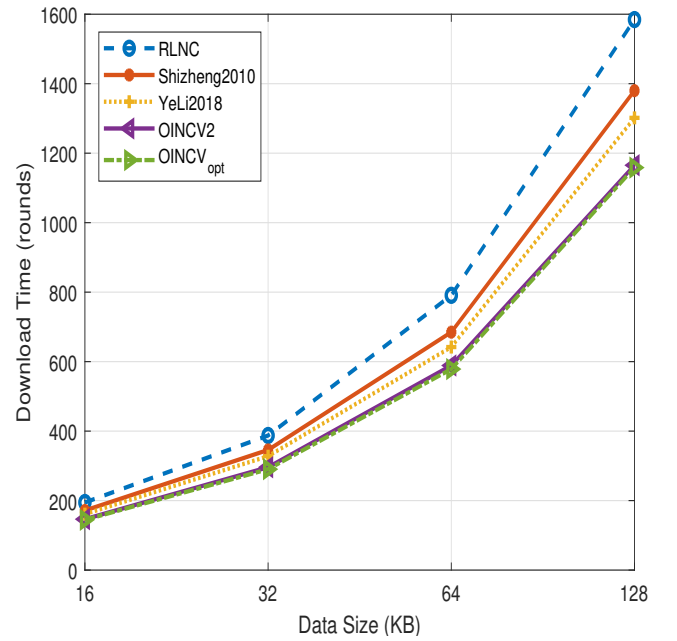
In this case scenario, we consider the following settings:

- File of sizes 50MB, 100MB, 250MB, 500MB are shared by 4 servers to one client who is zero hops away from the servers. The servers share their packets generation by generation, namely, sequential scheduling policy. The client sends acknowledgement only when it is received all generation's packets.
- We consider Ethernet/IP MTU packet size which is 1500 bytes, and the upload rate for each server is up to 10 packets/round.
- We assume generation sizes of 256 and 512 packets.

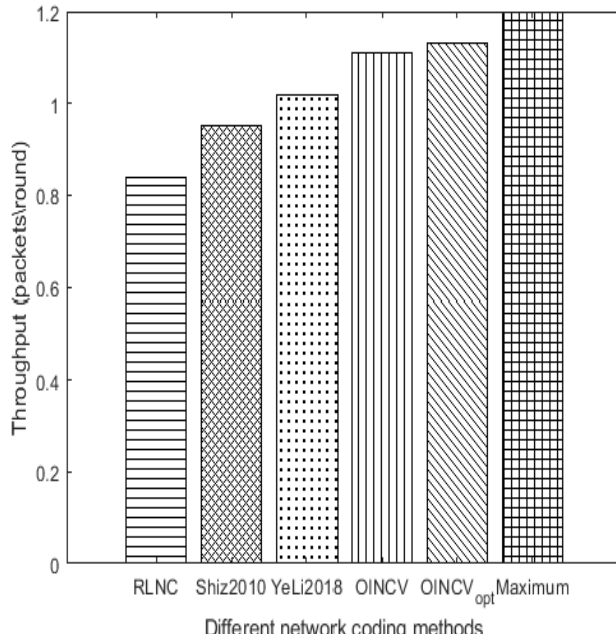
Based on these settings and as shown in the previous section, Table 5.6 presents the specifications of packet payload and coefficients overhead, and number of generations of each network coding method when the generation size is 256. Figure. 5.15(a) shows results of the coefficients overhead in MB; the results show that the OINCV overhead



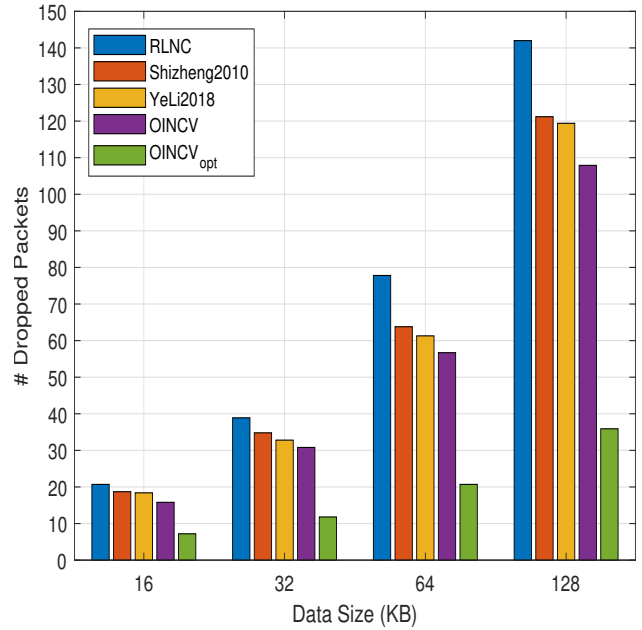
(a)



(b)



(c)



(d)

Figure 5.12: Performance metrics of WSN for different network coding vectors methods with recoding.

does not exceed 1 MB for all the files, while the next best method overhead reaches 50 MB. Figure. 5.15(b) shows the download time results in terms of rounds. Since

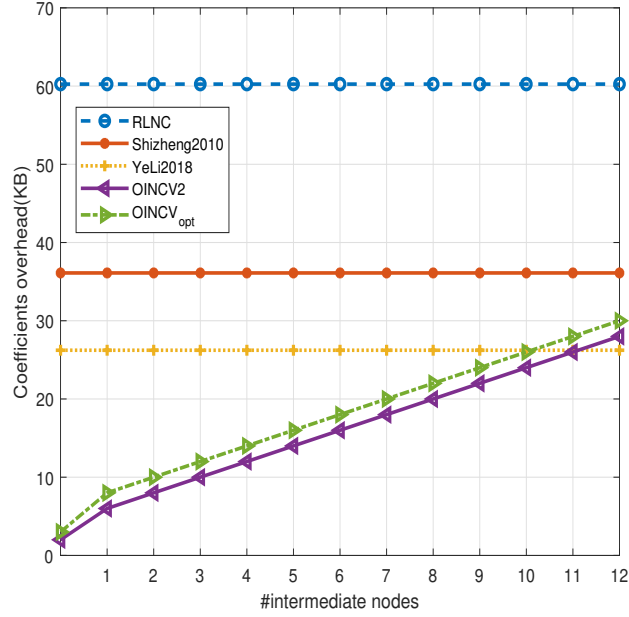
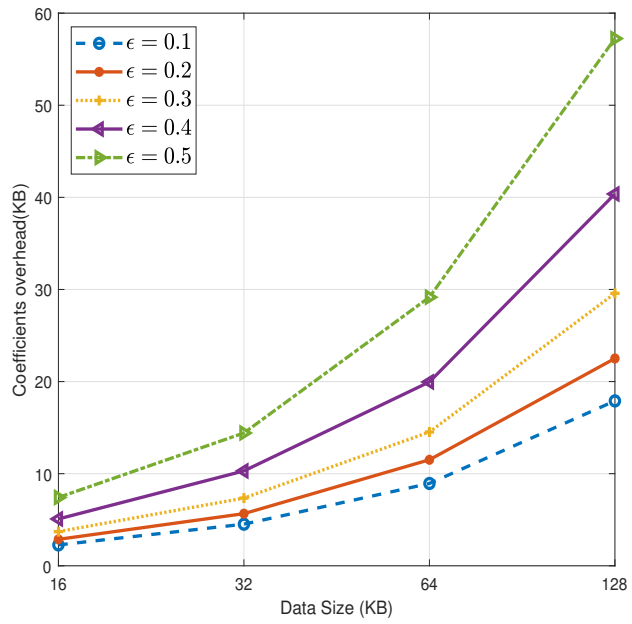


Figure 5.13: Network coding coefficients overhead with recoding; 0 to 12 hops and $\epsilon = 0$

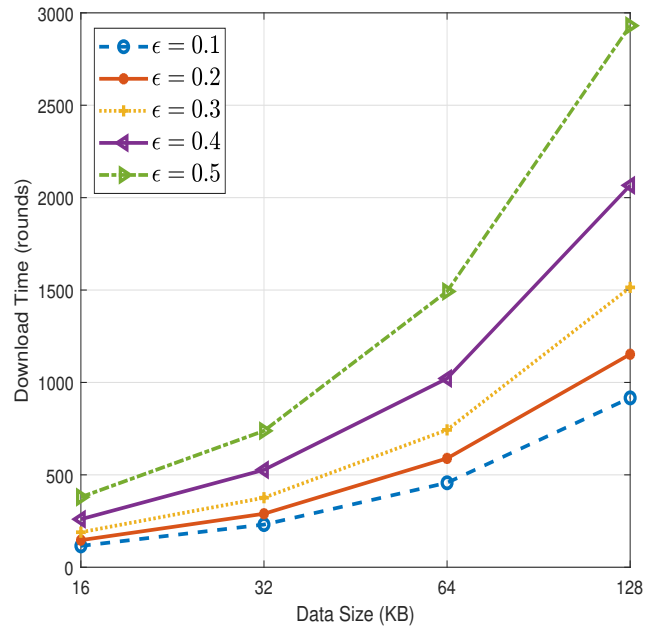
the proposed method has the minimum overhead, then the majority of the packet is occupied by the payload which leads to split the whole data file into fewer packets and generations, and hence less download time. Figure. 5.15(c) shows throughput results measured in packets/round. The results shows that OINCV throughput is 39.5 packets/round which is the maximum amongst the others and the closest to the ideal throughput which is 40 packets/round. Figure. 5.15(d) shows results of the number of dropped packets at the receiver side. Since OINCV has always the minimum number

Table 5.6: Specifications for different NC methods ($n = 256$)

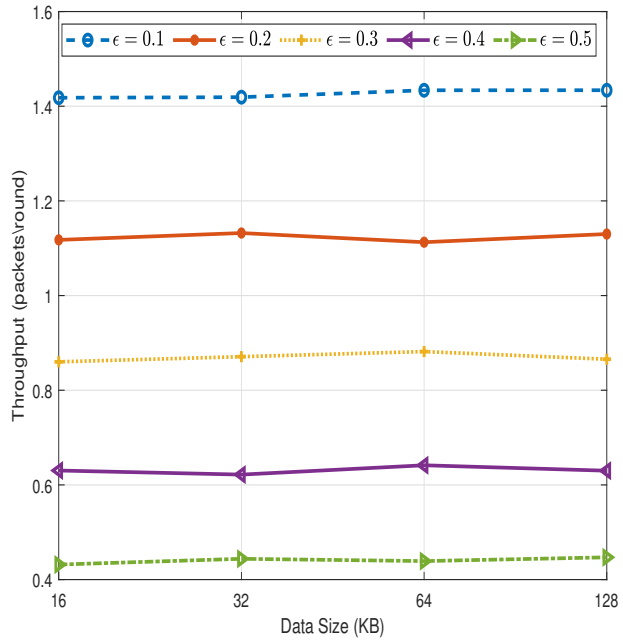
NC Methods	Packet Payload (Bytes)	Packet Overhead (Bytes)	Number of generations			
			50MB	100MB	250MB	500MB
RLNC	1244	256	165	330	824	1647
Shizheng2010	1334	166	154	308	768	1536
YeLi2018	1365	135	151	301	751	1501
OINCV3	1497	3	137	274	685	1369



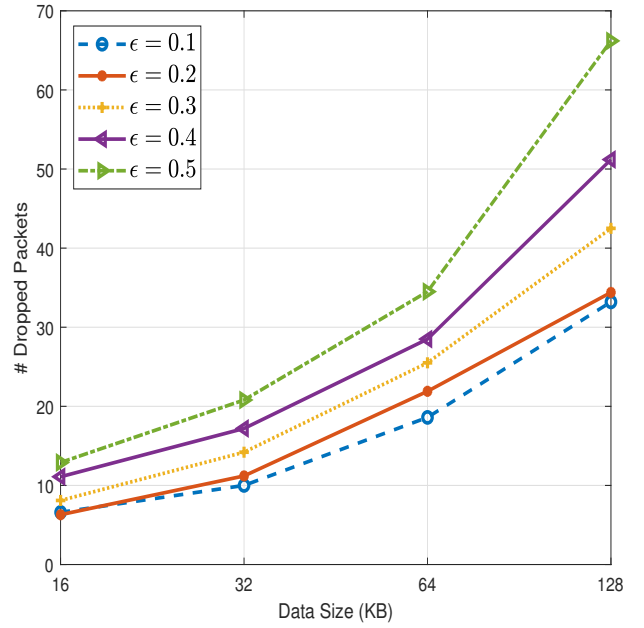
(a) Coefficients Overhead



(b) Download Time



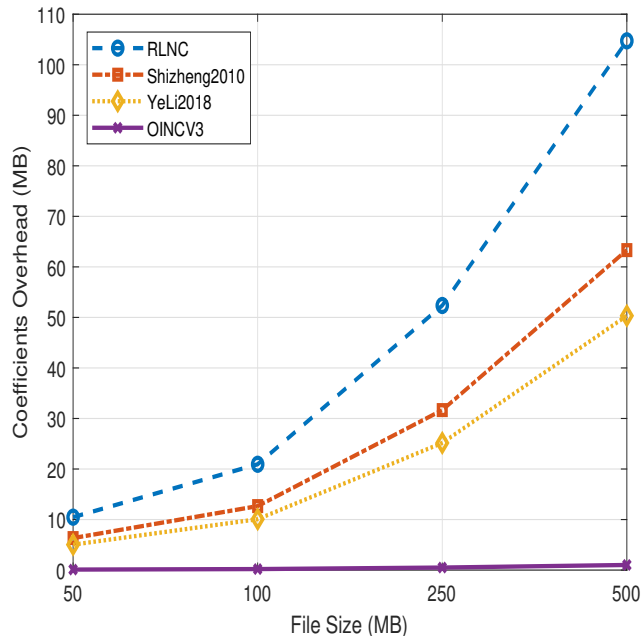
(c) Throughput



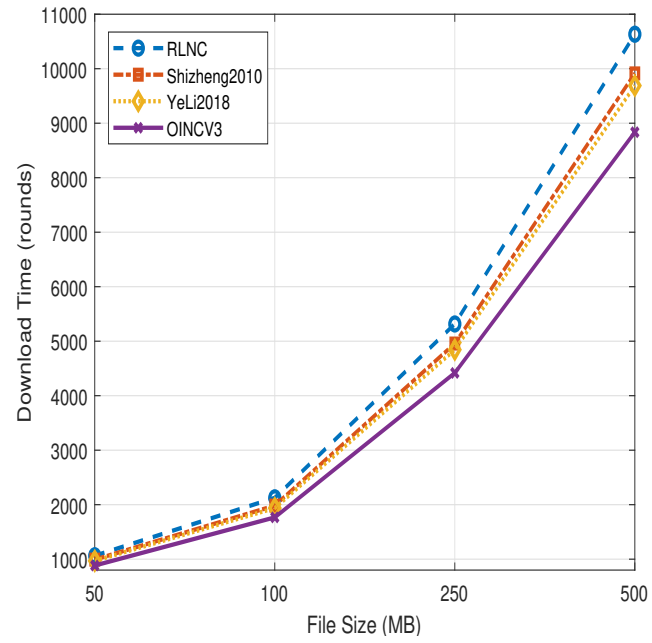
(d) Dropped Packets

Figure 5.14: Effect of sensitivity analysis on OINCV performance metrics.

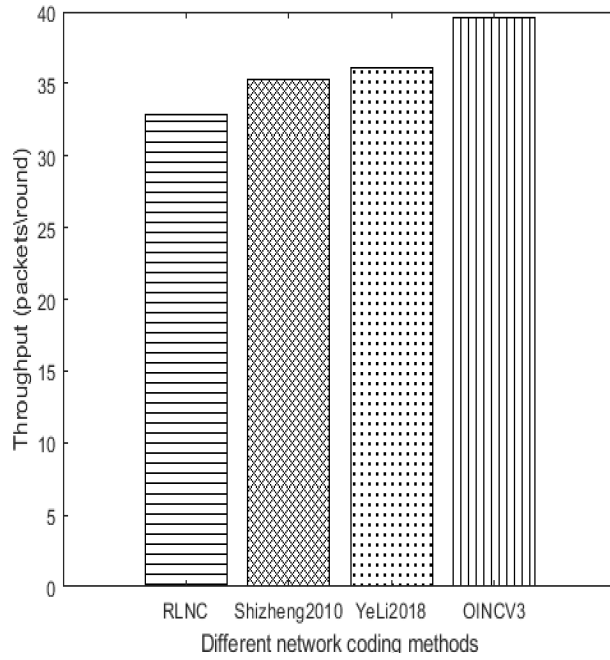
of generations and thus fewer acknowledgement to be sent, it always has the minimum dropped packets.



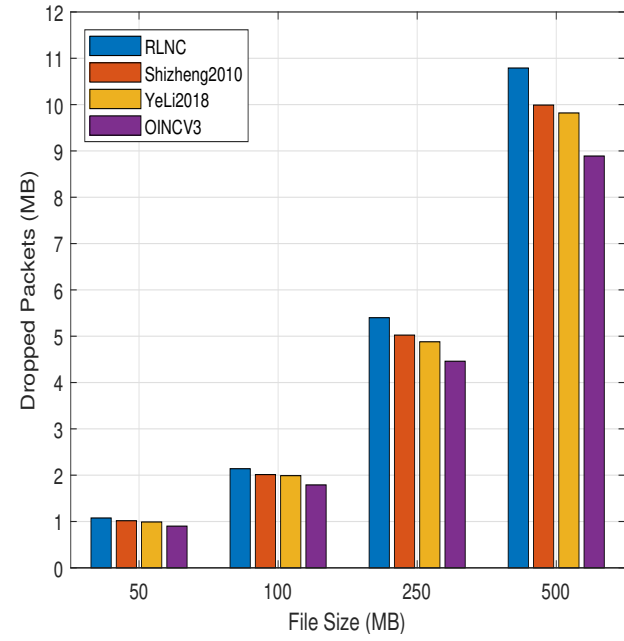
(a) Coefficients Overhead



(b) Download Time



(c) Throughput



(d) Dropped Packets

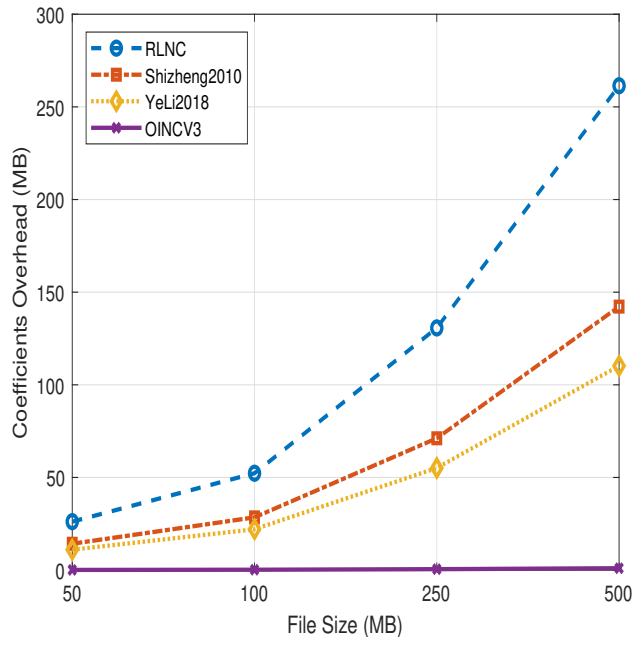
Figure 5.15: Performance evaluation of Ethernet\IP network for different NC methods, $n = 256$

Table 5.7 shows the specifications considering generation size of 512 and Fig. 5.16 depicts related results for the considered metrics. The results show interesting ob-

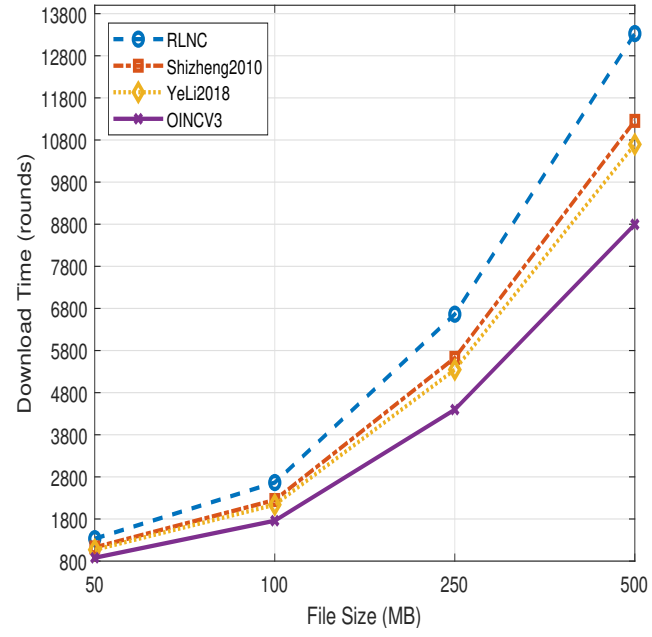
servations. Firstly, although the generation size is doubled the performance of all NC methods expect OINCV degrades for coefficients overhead, download time, and throughput metrics and that is because the coefficients overhead as well as is doubled while the packet size is fixed which yields increasing in number of packets. Secondly, the number of dropped packets at the receiver side for all the methods is improved because the number of generations is reduced and thus fewer acknowledgements to be shared. Thirdly, all the performance metrics of OINCV are improved and that is because the the coefficients overhead size does not change which leads to the same number of packets, and generation size is doubled which leads to more packets included within a generation, and thus the number of generations is reduced to the half. Figure 5.17 shows the same metrics results for variations of OINCV which denoted as *OINCV3_256*, *OINCV3_512*, *OINCV3_1024*, *OINCV4_1024* where the first number refers to the size of the traversed coefficient and the second number refers to the generation size, and for "*OINCV3_1024*" it assumes that there is a minimum coordination among the servers. The results show that as the generation size increases, the performance metrics get improved. Therefore, it is better to expand the generation size as much as possible as long as it is computationally feasible.

Table 5.7: Specifications for different NC methods ($n = 512$)

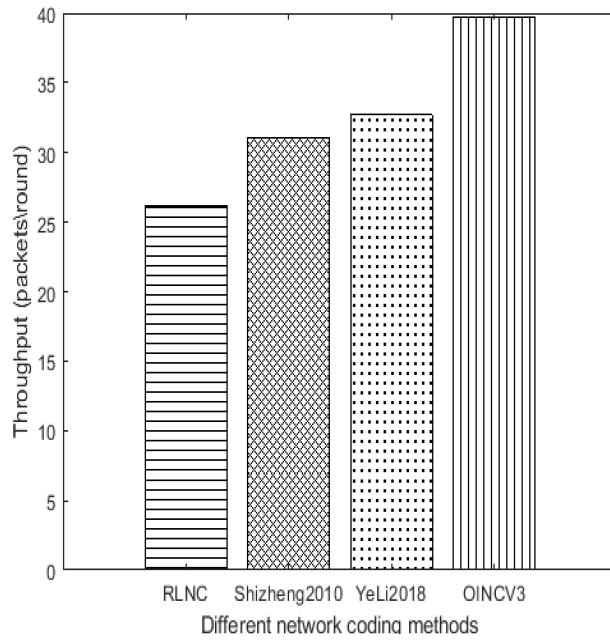
NC Methods	Packet Payload (Bytes)	Packet Overhead (Bytes)	Number of generations			
			50MB	100MB	250MB	500MB
RLNC	988	512	104	208	519	1037
Shizheng2010	1170	330	88	176	438	876
YeLi2018	1231	269	84	167	416	832
OINCV3	1497	3	69	137	343	685



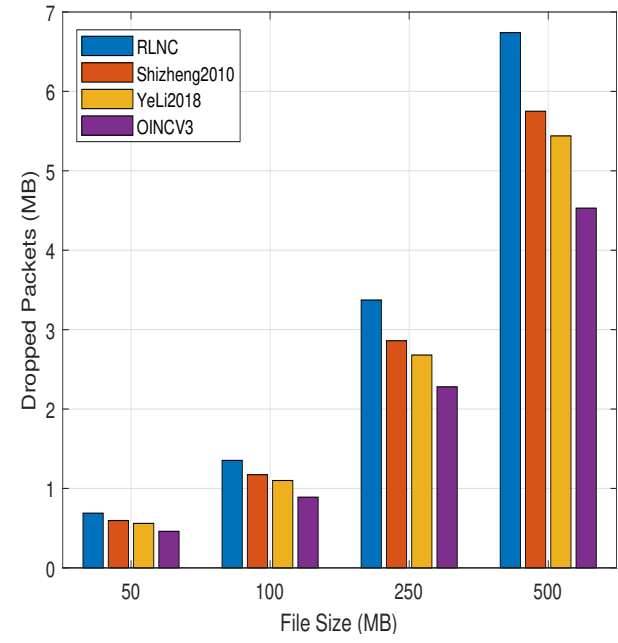
(a) Coefficients Overhead



(b) Download Time

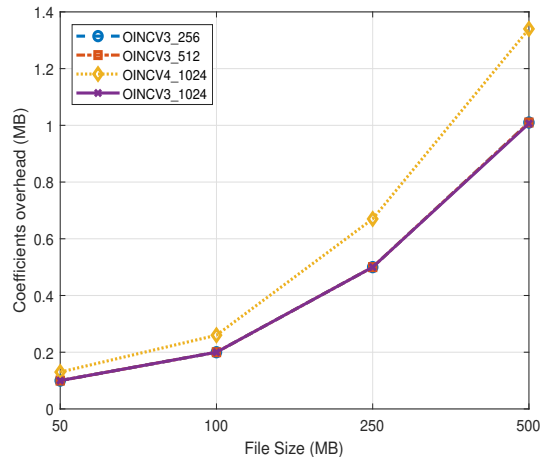


(c) Throughput

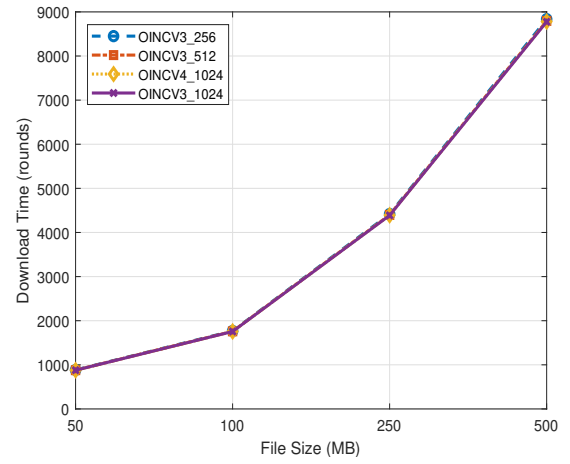


(d) Dropped Packets

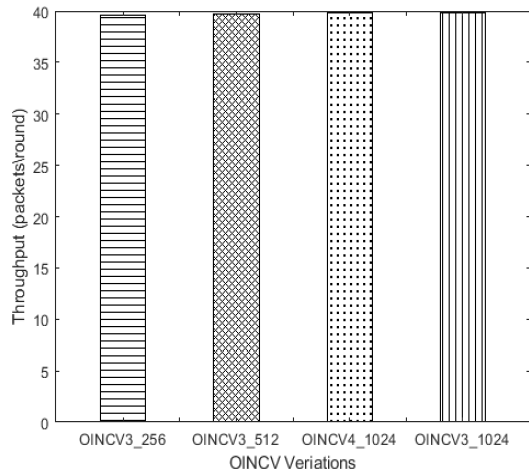
Figure 5.16: Performance evaluation of Ethernet network for different NC methods, $n = 512$



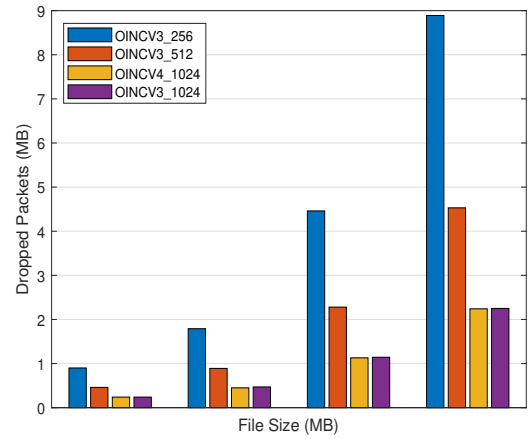
(a) Coefficients Overhead



(b) Download Time



(c) Throughput



(d) Dropped Packets

Figure 5.17: Performance evaluation of Ethernet\IP network for multi OINCV variations.

5.6.3 Large Scale P2P Content distribution network

In this scenario, we are interested in measuring the overall coefficients overhead and the following settings are considered:

- File of sizes 5GB, 10GB, 15GB, and 20GB shared by 4 servers to one client over a reliable channel. The servers share their packets generation by generation, namely, sequential scheduling policy. The client sends acknowledgement only when it is received all the generation's packets.
- We consider packet size of 16KB.
- We assume generation size of 1280 packets, link bandwidth of 8 MB/round that is shared evenly among the servers such that each server can upload 2 MB/round, and the receiver download rate is 8 MB/round.

Based on these setting, results of coefficients overhead in MB are depicted in Fig. 5.18. Since the difference between OINCV and others is huge, log scale is used. The real values are depicted in Table 5.8. The results show the big feature of OINCV that is able to reduce the overall coefficients overhead to a little number of megabytes which does not exceed 6 MB even when the file size is 20 GB. Compared to RLNC, [101], and [100], the performance improved by about 99.7%, 99.5%, and 99.4%, respectively.

Since the coefficients overhead size per packet compared with payload is small and data link rate is high, other metrics like throughput, and download time are not greatly improved by OINCV for this network case.

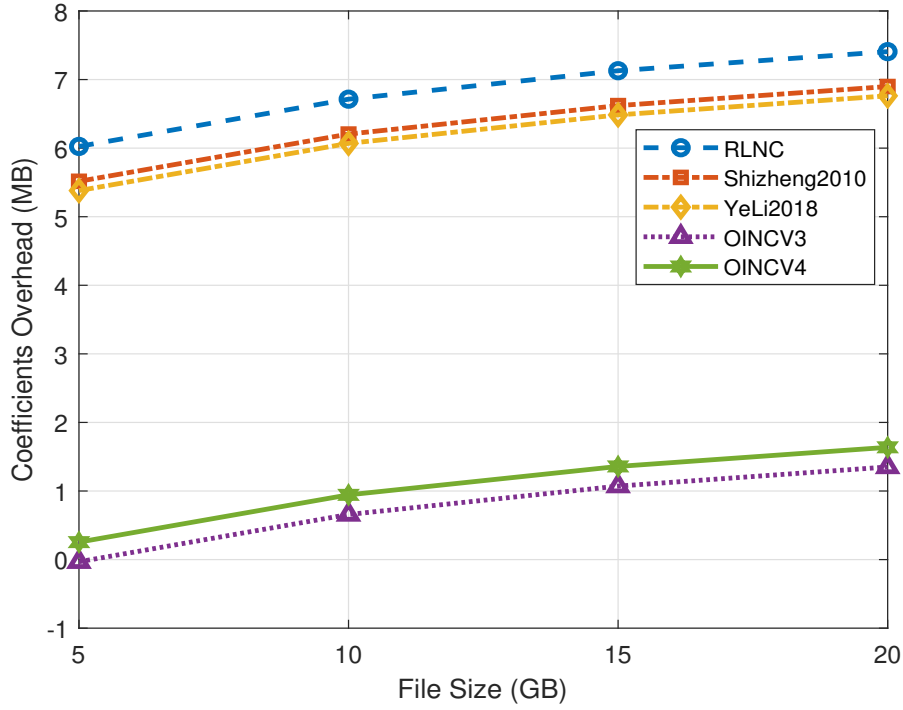


Figure 5.18: log scale of coefficients overhead in P2P content distribution network.

Table 5.8: Coefficients overhead (MB) in P2P content distribution network.

NC Method	File Size			
	5GB	10GB	15GB	20GB
RLNC	413.67	823.8	1247.36	1647.66
Shizheng2010	248.69	495.2	749.88	990.53
YeLi2018	217.18	432.49	654.86	865.02
OINCV3	0.969	1.93	2.92	3.86
OINCV4	1.29	2.57	3.89	5.14

5.7 Chapter Summary

In this chapter, the problem of NC coefficients overhead is revisited. A novel approach based on modular arithmetic and prime numbers, and influenced by Chinese remainder theorem (CRT), referred to as One Item Network Coding Vectors (OINCV),

is proposed to reduce the coefficients overhead by augmenting only one item coefficient of size two or a maximum of four bytes along with the payload packet. The proposed approach successfully addresses all limitations of the previous methods including the limitations on the generation size and density of the packets within the generation, rocoding on the intermediate nodes, and creating innovative coding vectors. Probabilistic theoretical analysis addresses questions about the proposed method and proves its viability and reliability considering generations sizes that vary from 16 to 6144 packets per generation. The experimental work is conducted for three different networks types, i.e., Wireless Sensors Network (WSN) , Ethernet network with 1500 bytes packet size, and large scale P2P file sharing network that uses 16KB packet size. Also, files sizes from 16KB to 20GB are distributed. The results show that the proposed method far outperforms all the previous methods in terms of coding coefficients overhead, download time, throughput, number of dropped packets on the receiver side.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusion

Network Coding (NC) is known as a helpful method for increasing the content availability, accelerating the download process, and robustness against churn in Peer-to-Peer (P2P) networks. Originally, DNC has been proposed and theoretically considered as an optimal solution. However, due to its huge computational overhead, it is not viable for real-world systems. Subsequently, sparse, generations, and overlapped generations network coding schemes are proposed as possible alternatives but at reduced performance compared to that provided by the DNC.

In the first part of this work, we model the problem of P2P content distribution as the deck of cards probability problem and we analyze it based on this model for both baseline and network coding P2P content distribution systems. Based on this analysis, we propose a novel procedure referred to herein Super generation network coding

(SGNC) which is a network coding scheme that boosts the robustness against churn in P2P networks, while maintaining roughly the computational cost relative to generation network coding. In addition, we evaluate the proposed method and compare it with normal generation network coding, overlapped generation network coding, and baseline BitTorrent. The experiments consider file sizes ranging from 320MB to 5GB, and utilize three piece scheduling policies namely: random, local RF, and global RF. The aforementioned schemes are evaluated in terms of availability, overhead, download time, and decodability. Simulation results show that SGNC can improve the robustness of the network, minimize the download time, incurs low overhead, and speeds up the decodability which in turns alleviates the CPU load.

In the second part, the problem of NC coefficients overhead is revisited. A novel approach based on modular arithmetic and prime numbers, and influenced by Chinese remainder theorem (CRT), referred to as One Item Network Coding Vectors (OINCV), is proposed to reduce the coefficients overhead by augmenting only one item coefficient of size two or a maximum of four bytes along with the payload packet. The proposed approach successfully addresses all limitations of the previous methods including the limitations on the generation size and density of the packets within the generation, rocoding on the intermediate nodes, and creating innovative coding vectors. Probabilistic theoretical analysis addresses questions about the proposed method and proves its viability and reliability considering generations sizes that vary from 16 to 6144 packets per generation. The experimental work is conducted for three different networks types, i.e., Wireless Sensors Network (WSN) , Ethernet network with 1500 bytes packet size, and large scale P2P file sharing network that uses 16KB

packet size. Also, files sizes from 16KB to 20GB are distributed. The results show that the proposed method far outperforms all the previous methods in terms of coding coefficients overhead, download time, throughput, number of dropped packets on the receiver side.

6.2 Limitations and Future Work

The main limitation of SGNC was its extra network coding coefficients overhead, which is resolved in the second part of this thesis successfully. Future work includes considering the effect of local and global rarest-first policies over wide area networks with large diameters.

The main limitation of OINCV is the extra coefficients overhead cost when recoding is required. Future work includes proposing methods to mitigate the recoding cost. Another potential and important research direction is to have a real implementation of OINCV on sensors and calculate the energy consumed by the OINCV operations, specifically, from generating GMS until composing the encoded packet. Also for Zigbee network, a deep sensitivity analysis for the erasure probability and the loss probability could be considered.

REFERENCES

- [1] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, “Network information flow,” *IEEE Transactions on information theory*, vol. 46, no. 4, pp. 1204–1216, 2000.
- [2] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, “Chord: a scalable peer-to-peer lookup protocol for internet applications,” *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 1, pp. 17–32, 2003.
- [3] F. N. Nwebonyi, R. Martins, and M. E. Correia, “Reputation based approach for improved fairness and robustness in p2p protocols,” *Peer-to-Peer Networking and Applications*, vol. 12, no. 4, pp. 951–968, 2019.
- [4] V. Pacifici, F. Lehrieder, and G. Dán, “Cache bandwidth allocation for p2p file-sharing systems to minimize inter-isp traffic,” *IEEE/ACM Transactions on Networking (TON)*, vol. 24, no. 1, pp. 437–448, 2016.
- [5] B. Cohen, “The bittorrent protocol specification,” 2008.
- [6] A. Lareida, T. Bocek, M. Waldburger, and B. Stiller, “Rb-tracker: A fully distributed, replicating, network-, and topology-aware p2p cdn,” in *2013*

- IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, 2013, pp. 1199–1202.
- [7] D. Wu, P. Dhungel, X. Hei, C. Zhang, and K. W. Ross, “Understanding peer exchange in bittorrent systems,” in *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 2010, pp. 1–8.
- [8] F. V. Hecht, T. Bocek, and B. Stiller, “B-tracker: Improving load balancing and efficiency in distributed p2p trackers,” in *2011 IEEE International Conference on Peer-to-Peer Computing*. IEEE, 2011, pp. 310–313.
- [9] G. Neglia, G. Reina, H. Zhang, D. Towsley, A. Venkataramani, and J. Danaher, “Availability in bittorrent systems,” in *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE, 2007, pp. 2216–2224.
- [10] C. P. Fry and M. K. Reiter, “Really truly trackerless bittorrent,” *School of Computer Science, Carnegie Mellon University, Tech. Rep*, pp. 06–148, 2006.
- [11] N. J. Navimipour and F. S. Milani, “A comprehensive study of the resource discovery techniques in peer-to-peer networks,” *Peer-to-Peer Networking and Applications*, vol. 8, no. 3, pp. 474–492, 2015.
- [12] J. Luo, B. Xiao, K. Bu, and S. Zhou, “Understanding and improving piece-related algorithms in the bittorrent protocol,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 12, pp. 2526–2537, 2013.

- [13] Y. Hu, D. Dong, J. Li, and F. Wu, “Efficient and incentive-compatible resource allocation mechanism for p2p-assisted content delivery systems,” *Future Generation Computer Systems*, vol. 29, no. 6, pp. 1611–1620, 2013.
- [14] D. M. Chiu, R. W. Yeung, J. Huang, and B. Fan, “Can network coding help in p2p networks?” in *2006 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*. IEEE, 2006, pp. 1–5.
- [15] J.-L. Chiang, Y.-Y. Tseng, and W.-T. Chen, “Interest-intended piece selection in bittorrent-like peer-to-peer file sharing systems,” *Journal of parallel and distributed computing*, vol. 71, no. 6, pp. 879–888, 2011.
- [16] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, “Network coding: an instant primer,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 63–68, 2006.
- [17] H. Terelius and K. H. Johansson, “Peer-to-peer gradient topologies in networks with churn,” *IEEE Transactions on Control of Network Systems*, vol. 5, no. 4, pp. 2085–2095, 2018.
- [18] C. Feng and B. Li, “Network coding for content distribution and multimedia streaming in peer-to-peer networks,” in *Network Coding*. Elsevier, 2012, pp. 61–86.
- [19] Q.-C. Cai and K.-T. Lo, “Two blocks are enough: on the feasibility of using network coding to ameliorate the content availability of bittorrent swarms,”

- IEEE transactions on parallel and distributed systems*, vol. 24, no. 8, pp. 1682–1694, 2012.
- [20] T. Etzion and A. Wachter-Zeh, “Vector network coding based on subspace codes outperforms scalar linear network coding,” *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 2460–2473, 2018.
- [21] L. Keller, E. Atsan, K. Argyraki, and C. Fragouli, “Sensecode: Network coding for reliable sensor networks,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 2, pp. 1–20, 2013.
- [22] M. Jafari, L. Keller, C. Fragouli, and K. Argyraki, “Compressed network coding vectors,” in *2009 IEEE International Symposium on Information Theory*. IEEE, 2009, pp. 109–113.
- [23] A. Stiglic, “Computations with a deck of cards,” *Theoretical Computer Science*, vol. 259, no. 1-2, pp. 671–678, 2001.
- [24] A. Srinivasan and H. Aldharrab, “Xtra extended bit-torrent protocol for authenticated covert peer communication,” *Peer-to-Peer Networking and Applications*, vol. 12, no. 1, pp. 143–157, 2019.
- [25] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, “Improving traffic locality in bittorrent via biased neighbor selection,” in *26th IEEE International Conference on Distributed Computing Systems (ICDCS’06)*. IEEE, 2006, pp. 66–66.

- [26] K. Huang, L. Wang, D. Zhang, and Y. Liu, “Optimizing the bittorrent performance using an adaptive peer selection strategy,” *Future Generation Computer Systems*, vol. 24, no. 7, pp. 621–630, 2008.
- [27] S.-Y. Li, R. W. Yeung, and N. Cai, “Linear network coding,” *IEEE transactions on information theory*, vol. 49, no. 2, pp. 371–381, 2003.
- [28] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, “Xors in the air: Practical wireless network coding,” in *ACM SIGCOMM computer communication review*, vol. 36, no. 4. ACM, 2006, pp. 243–254.
- [29] Y.-J. Chen, L.-C. Wang, K. Wang, and W.-L. Ho, “Topology-aware network coding for wireless multicast,” *IEEE Systems Journal*, vol. 12, no. 4, pp. 3683–3692, 2018.
- [30] D. Jiang, Z. Xu, W. Li, and Z. Chen, “Network coding-based energy-efficient multicast routing algorithm for multi-hop wireless networks,” *Journal of Systems and Software*, vol. 104, pp. 152–165, 2015.
- [31] L. Czap, C. Fragouli, V. M. Prabhakaran, and S. Diggavi, “Secure network coding with erasures and feedback,” *IEEE Transactions on Information Theory*, vol. 61, no. 4, pp. 1667–1686, 2015.
- [32] R. Matsumoto and M. Hayashi, “Universal secure multiplex network coding with dependent and non-uniform messages,” *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 3773–3782, 2017.

- [33] H. He, R. Li, Z. Xu, and W. Xiao, “An efficient ecc-based mechanism for securing network coding-based p2p content distribution,” *Peer-to-Peer Networking and Applications*, vol. 7, no. 4, pp. 572–589, 2014.
- [34] M. He, Z. Gong, L. Chen, H. Wang, F. Dai, and Z. Liu, “Securing network coding against pollution attacks in p2p converged ubiquitous networks,” *Peer-to-Peer Networking and Applications*, vol. 8, no. 4, pp. 642–650, 2015.
- [35] D. Xie, H. Peng, L. Li, and Y. Yang, “An efficient privacy-preserving scheme for secure network coding based on compressed sensing,” *AEU-International Journal of Electronics and Communications*, vol. 79, pp. 33–42, 2017.
- [36] T. Li, W. Chen, Y. Tang, and H. Yan, “A homomorphic network coding signature scheme for multiple sources and its application in iot,” *Security and communication networks*, vol. 2018, 2018.
- [37] C. Gkantsidis and P. R. Rodriguez, “Network coding for large scale content distribution,” in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 4. IEEE, 2005, pp. 2235–2245.
- [38] M. Yang and Y. Yang, “Applying network coding to peer-to-peer file sharing,” *IEEE Transactions on Computers*, vol. 63, no. 8, pp. 1938–1950, 2013.
- [39] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, “A random linear network coding approach to multicast,” *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

- [40] Q. T. Sun, X. Yang, K. Long, X. Yin, and Z. Li, “On vector linear solvability of multicast networks,” *IEEE Transactions on Communications*, vol. 64, no. 12, pp. 5096–5107, 2016.
- [41] J. Ebrahimi and C. Fragouli, “Algebraic algorithms for vector network coding,” *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 996–1007, 2011.
- [42] S. Kafaie, Y. P. Chen, O. A. Dobre, and M. H. Ahmed, “Network coding implementation details: A guidance document,” *arXiv preprint arXiv:1801.02120*, 2018.
- [43] L. Keller, “Network Coding Utilities . Contribute to lokeller/ncutils development by creating an account on GitHub,” 2019, original-date: 2015-03-14T07:24:23Z. [Online]. Available: <https://github.com/lokeller/ncutils>
- [44] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, “The use of forward error correction (fec) in reliable multicast,” RFC 3453, December, Tech. Rep., 2002.
- [45] B. Cohen, “Incentives build robustness in bittorrent,” in *Workshop on Economics of Peer-to-Peer systems*, vol. 6, 2003, pp. 68–72.
- [46] F. Azzedin and M. Yahaya, “Modeling bittorrent choking algorithm using game theory,” *Future Generation Computer Systems*, vol. 55, pp. 255–265, 2016.
- [47] M. Medard, “How good is random linear coding based distributed networked storage?” *Proc. NETCOD’05. Riva del Garda, Italy*, 2005.

- [48] Y. Cassuto and A. Shokrollahi, “Online fountain codes with low overhead,” *IEEE Transactions on Information Theory*, vol. 61, no. 6, pp. 3137–3149, 2015.
- [49] S. Deb, M. Médard, and C. Choute, “Algebraic gossip: A network coding approach to optimal multiple rumor mongering,” *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2486–2507, 2006.
- [50] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, “Randomized rumor spreading,” in *Proceedings 41st Annual Symposium on Foundations of Computer Science*. IEEE, 2000, pp. 565–574.
- [51] B. Doerr, C. Doerr, S. Moran, and S. Moran, “Simple and optimal randomized fault-tolerant rumor spreading,” *Distributed Computing*, vol. 29, no. 2, pp. 89–104, 2016.
- [52] N. Wang and N. Ansari, “Downloader-initiated random linear network coding for peer-to-peer file sharing,” *IEEE Systems Journal*, vol. 5, no. 1, pp. 61–69, 2010.
- [53] R. W. Yeung *et al.*, “Avalanche: A network coding analysis,” *Communications in Information & Systems*, vol. 7, no. 4, pp. 353–358, 2007.
- [54] A. Soro and J. Lacan, “Fft-based network coding for peer-to-peer content delivery,” in *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*. IEEE, 2011, pp. 1–5.
- [55] M. Wang and B. Li, “How practical is network coding?” in *2006 14th IEEE International Workshop on Quality of Service*. IEEE, 2006, pp. 274–278.

- [56] T. Li, J. Wang, and J. You, “Using degree-based strategy for network coding in content distribution network,” in *2008 International Conference on Computer and Electrical Engineering*. IEEE, 2008, pp. 487–491.
- [57] G. Ma, Y. Xu, M. Lin, and Y. Xuan, “A content distribution system based on sparse linear network coding,” in *Third Workshop on Network Coding (Netcod 2007)*, 2007.
- [58] C. Ortolfo, C. Schindelhauer, and A. Vater, “Paircoding: Improving file sharing using sparse network codes,” in *2009 Fourth International Conference on Internet and Web Applications and Services*. IEEE, 2009, pp. 49–57.
- [59] A. Vater, “Efficient coding schemes for file sharing networks,” Ph.D. dissertation, Verlag nicht ermittelbar, 2011.
- [60] A. Vater, C. Schindelhauer, and C. Ortolfo, “Tree network coding for peer-to-peer networks,” in *Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures*. ACM, 2010, pp. 114–123.
- [61] C. Ortolfo, C. Schindelhauer, and A. Vater, “Classifying peer-to-peer network coding schemes,” in *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*. ACM, 2009, pp. 310–318.
- [62] T. Shang, T. Peng, Q. Lei, and J. Liu, “Homomorphic signature for generation-based network coding,” in *2016 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 2016, pp. 269–273.

- [63] J. Xu, X. Wang, J. Zhao, and A. O. Lim, “I-swifter: improving chunked network coding for peer-to-peer content distribution,” *Peer-to-peer networking and applications*, vol. 5, no. 1, pp. 30–39, 2012.
- [64] L.-Y. Ren, H.-Y. He, Z. Di, and M. Lei, “Content distribution system based on segmented network coding,” in *2009 International Conference on Apperceiving Computing and Intelligence Analysis*. IEEE, 2009, pp. 258–261.
- [65] S. Tao, J. Huang, Z. Yang, W. Cheng, and W. Liu, “An improved network coding-based cooperative content distribution scheme,” in *ICC Workshops-2008 IEEE International Conference on Communications Workshops*. IEEE, 2008, pp. 360–364.
- [66] P. A. Chou, Y. Wu, and K. Jain, “Practical network coding,” in *Proceedings of the annual Allerton conference on communication control and computing*, vol. 41, no. 1. The University; 1998, 2003, pp. 40–49.
- [67] P. Maymounkov, N. J. Harvey, and D. S. Lun, “Methods for efficient network coding,” in *Proc. 44th Annual Allerton Conference on Communication, Control, and Computing*, 2006, pp. 482–491.
- [68] J. Xu, J. Zhao, X. Wang, and X. Xue, “Swifter: Chunked network coding for peer-to-peer content distribution,” in *2008 IEEE International Conference on Communications*. IEEE, 2008, pp. 5603–5608.
- [69] M. Hundeboll, J. Ledet-Pedersen, G. Sluyterman, T. K. Madsen, and F. H. Fitzek, “Peer-assisted content distribution with random linear network coding,”

- in *2014 IEEE 79th Vehicular Technology Conference (VTC Spring)*. IEEE, 2014, pp. 1–6.
- [70] D. Niu and B. Li, “Analyzing the resilience-complexity tradeoff of network coding in dynamic p2p networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 11, pp. 1842–1850, 2011.
- [71] X. Zhang and B. Li, “On the market power of network coding in p2p content distribution systems,” *IEEE Transactions on Parallel & Distributed Systems*, no. 12, pp. 2063–2070, 2011.
- [72] J.-S. Leu, M.-C. Yu, and H.-C. Yueh, “Improving network coding based file sharing for unstructured peer-to-peer networks,” *Journal of Network and Systems Management*, vol. 23, no. 4, pp. 803–829, 2015.
- [73] Z. Li, B. Li, D. Jiang, and L. C. Lau, “On achieving optimal end-to-end throughput in data networks: Theoretical and empirical studies,” *ECE Technical Report*, 2004.
- [74] C. K. Ngai and R. W. Yeung, “Network coding gain of combination networks,” in *Information Theory Workshop*. IEEE, 2004, pp. 283–287.
- [75] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. M. Tolhuizen, “Polynomial time algorithms for multicast network code construction,” *IEEE Transactions on Information Theory*, vol. 51, no. 6, pp. 1973–1982, 2005.

- [76] Y.-h. Chu, S. G. Rao, S. Seshan, and H. Zhang, “A case for end system multicast,” *IEEE Journal on selected areas in communications*, vol. 20, no. 8, pp. 1456–1471, 2002.
- [77] P. J. Braun, M. Sipos, P. Ekler, and H. Charaf, “Increasing data distribution in bittorrent networks by using network coding techniques,” in *Proceedings of European Wireless 2015; 21th European Wireless Conference*. VDE, 2015, pp. 1–6.
- [78] I. Baumgart, B. Heep, and S. Krause, “Oversim: A flexible overlay network simulation framework,” in *2007 IEEE global internet symposium*. IEEE, 2007, pp. 79–84.
- [79] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, “Making gnutella-like p2p systems scalable,” in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 407–418.
- [80] H. Zeng, J. Huang, S. Tao, and W. Cheng, “A simulation study on network coding parameters in p2p content distribution system,” in *2008 Third International Conference on Communications and Networking in China*. IEEE, 2008, pp. 197–201.
- [81] X. Zhang, J. Liu, B. Li, and Y.-S. Yum, “Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming,” in *Proceedings IEEE 24th*

- Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3. IEEE, 2005, pp. 2102–2111.
- [82] O. Kaiqian, X. Yinlong, M. Guanjin, and Z. Yulin, “Dasher: a peer-to-peer content distribution system based on combined network coding,” in *2009 2nd IEEE International Conference on Broadband Network & Multimedia Technology*. IEEE, 2009, pp. 687–692.
- [83] J. Su, Q. Deng, and D. Long, “Pclnc: A low-cost intra-generation network coding strategy for p2p content distribution,” *Peer-to-Peer Networking and Applications*, vol. 12, no. 1, pp. 177–188, 2019.
- [84] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, “Planetlab: an overlay testbed for broad-coverage services,” *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, 2003.
- [85] A. Montresor and M. Jelasity, “Peersim: A scalable p2p simulator,” in *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*. IEEE, 2009, pp. 99–100.
- [86] M. Halloush and H. Radha, “Network coding with multi-generation mixing: A generalized framework for practical network coding,” *IEEE Transactions on wireless communications*, vol. 10, no. 2, pp. 466–473, 2010.
- [87] X. Wei and D.-Y. Long, “P2p content-propagation mechanism tailored by network coding,” in *2009 International Symposium on Computer Network and Multimedia Technology*. IEEE, 2009, pp. 1–6.

- [88] D. Silva, W. Zeng, and F. R. Kschischang, "Sparse network coding with overlapping classes," in *2009 Workshop on Network Coding, Theory, and Applications*. IEEE, 2009, pp. 74–79.
- [89] D. J. MacKay, "Fountain codes," *IEE Proceedings-Communications*, vol. 152, no. 6, pp. 1062–1068, 2005.
- [90] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 4, pp. 56–67, 1998.
- [91] M. Luby, "Lt codes," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings*. IEEE, 2002, pp. 271–280.
- [92] A. Shokrollahi, "Raptor codes," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2551–2567, 2006.
- [93] A. Heidarzadeh and A. H. Banihashemi, "Analysis of overlapped chunked codes with small chunks over line networks," in *2011 IEEE International Symposium on Information Theory Proceedings*. IEEE, 2011, pp. 801–805.
- [94] Y. Li, E. Soljanin, and P. Spasojevic, "Effects of the generation size and overlap on throughput and complexity in randomized linear network coding," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 1111–1123, 2011.
- [95] B. Tang, S. Yang, Y. Yin, B. Ye, and S. Lu, "Expander graph based overlapped chunked codes," in *2012 IEEE International Symposium on Information Theory Proceedings*. IEEE, 2012, pp. 2451–2455.

- [96] G. Joshi and E. Soljanin, “Round-robin overlapping generations coding for fast content download,” in *2013 IEEE International Symposium on Information Theory*. IEEE, 2013, pp. 2740–2744.
- [97] Y. Li, W.-Y. Chan, and S. D. Blostein, “Network coding with unequal size overlapping generations,” in *2012 International Symposium on Network Coding (NetCod)*. IEEE, 2012, pp. 161–166.
- [98] S. Huss-Lederman, E. M. Jacobson, J. R. Johnson, A. Tsao, and T. Turnbull, “Implementation of strassen’s algorithm for matrix multiplication,” in *Supercomputing’96: Proceedings of the 1996 ACM/IEEE Conference on Supercomputing*. IEEE, 1996, pp. 32–32.
- [99] G. Bilardi and L. De Stefani, “The i/o complexity of strassen’s matrix multiplication with recomputation,” in *Workshop on Algorithms and Data Structures*. Springer, 2017, pp. 181–192.
- [100] Y. Li, S. Zhang, J. Wang, X. Ji, H. Wu, and Z. Bao, “A low-complexity coded transmission scheme over finite-buffer relay links,” *IEEE Transactions on Communications*, vol. 66, no. 7, pp. 2873–2887, 2018.
- [101] S. Li and A. Ramamoorthy, “Improved compression of network coding vectors using erasure decoding and list decoding,” *IEEE Communications Letters*, vol. 14, no. 8, pp. 749–751, 2010.

- [102] V. Guruswami, “List decoding with side information,” in *18th IEEE Annual Conference on Computational Complexity, 2003. Proceedings.* IEEE, 2003, pp. 300–309.
- [103] D. Gligoroski, K. Kravlevska, and H. Øverby, “Minimal header overhead for random linear network coding,” in *2015 IEEE International Conference on Communication Workshop (ICCW).* IEEE, 2015, pp. 680–685.
- [104] N. Thomos and P. Frossard, “Toward one symbol network coding vectors,” *IEEE Communications letters*, vol. 16, no. 11, pp. 1860–1863, 2012.
- [105] C.-C. Chao, C.-C. Chou, and H.-Y. Wei, “Pseudo random network coding design for ieee 802.16 m enhanced multicast and broadcast service,” in *2010 IEEE 71st Vehicular Technology Conference.* IEEE, 2010, pp. 1–5.
- [106] N. Yazdani and D. E. Lucani, “Revolving codes: High performance and low overhead network coding,” in *2019 IEEE 2nd Wireless Africa Conference (WAC).* IEEE, 2019, pp. 1–5.
- [107] D. Pei, A. Salomaa, and C. Ding, *Chinese remainder theorem: applications in computing, coding, cryptography.* World Scientific, 1996.
- [108] K. H. Rosen, *Elementary number theory.* Pearson Education, 2011.
- [109] M. V. Pedersen, J. Heide, and F. H. Fitzek, “Kodo: An open and research oriented network coding library,” in *International Conference on Research in Networking.* Springer, 2011, pp. 145–152.

- [110] J. L. Alperin and R. B. Bell, “The general linear group,” in *Groups and Representations*. Springer, 1995, pp. 39–62.

VITAE

- Name: Anas Ahmad Mohammad Abu-Daqa
- Nationality: Palestinian
- Date of Birth: 29/03/1986
- Email: *anas-ahmed@msn.com*
- Permenant Address: Gaza, 8540400, Palestine
- PhD in Computer Engineering, KFUPM, Dhahran, Saudi Arabia (2015 -2020)
- MSc in Computer Networks, KFUPM, Dhahran, Saudi Arabia (2012 -2014)
- Bsc in Computer Engineering, IUGAZA, Gaza, Palestine (2004-2009).
- Areas of interest: Network Coding, P2P networking, interplanetary file system (IPFS), Cryptography, Data Compression