Throughput and Delay Optimization of Linear Network Coding in Wireless Broadcast

Mingchao Yu

October 2016

A thesis submitted for the degree of Doctor of Philosophy
of the Australian National University

For my parents and grandparents.

Declaration

Substantial majority of this research work is accomplished by the candidate, Mingchao Yu, independently, under the supervision of Associate Professor Parastoo Sadeghi. Besides, valuable ideas and suggestions have been received from Dr. Neda Aboutorab (in Chapter 4) and Associate Professor Alex Sprintson (in Chapter 5 and 6).

Mingchao Yu

Acknowledgements

I would like to dedicate my heartfelt appreciation to my supervisor Associate Professor Parastoo Sadeghi for her enormous supports during my Ph.D. candidature. It was her who led me to the fascinating world of network coding. It was her generous financial support that made my academic visit to the U.S. possible. Without her supervision, I would hardly realize my academic achievements and self-value. Just before my thesis submission, I was granted the Chinese Government Award for Outstanding Self-financed Students Abroad. I am so proud to win this award as a student of Associate Professor Parastoo Sadeghi, the best supervisor ever.

I am indebted to my better half, Yupei Lyu. She enlightened my life with her gentle and kindness. She always has her faith in me, and always encourages me under adversities. She fully supported my work and has never made a complaint on my negligence of the family. I must have exhausted all my luck in my entire life to have met her and married her.

Finally, my grateful thanks to my parents and grandparents. They are my spiritual anchor when I am abroad. Their company and selfless supports have always been with me.

Mingchao Yu

Research School of Engineering, ANU, Canberra

ming.yu@anu.edu.au

Abstract

Linear network coding (LNC) is able to achieve the optimal throughput of packet-level wireless broadcast, where a sender wishes to broadcast a set of data packets to a set of receivers within its transmission range through lossy wireless links. But the price is a large delay in the recovery of individual data packets due to network decoding, which may undermine all the benefits of LNC. However, packet decoding delay minimization and its relation to throughput maximization have not been well understood in the network coding literature.

Motivated by this fact, in this thesis we present a comprehensive study on the joint optimization of throughput and average packet decoding delay (APDD) for LNC in wireless broadcast. To this end, we reveal the fundamental performance limits of LNC and study the performance of three major classes of LNC techniques, including instantly decodable network coding (IDNC), generation-based LNC, and throughput-optimal LNC (including random linear network coding (RLNC)).

Various approaches are taken to accomplish the study, including 1) deriving performance bounds, 2) establishing and modelling optimization problems, 3) studying the hardness of the optimization problems and their approximation, 4) developing new optimal and heuristic techniques that take into account practical concerns such as receiver feedback frequency and computational complexity.

Key contributions of this thesis include:

- a necessary and sufficient condition for LNC to achieve the optimal throughput of wireless broadcast;
- the NP-hardness of APDD minimization;
- lower bounds of the expected APDD of LNC under random packet erasures;
- the APDD-approximation ratio of throughput-optimal LNC, which has a value of between 4/3 and 2. In particular, the ratio of RLNC is exactly 2;
- a novel throughput-optimal, APDD-approximation, and implementation-friendly LNC technique;
- an optimal implementation of strict IDNC that is robust to packet erasures;

- a novel generation-based LNC technique that generalizes some of the existing LNC techniques and enables tunable throughput-delay tradeoffs.

Contents

Notation and terminology

Notation

α	coding coefficient
β	approximation ratio of an approximation algorithm
$\boldsymbol{C}$	the coding coefficient matrix of a set of coded packets
D	average packet decoding delay
$\underline{D}$	lower bound of average packet decoding delay
$\mathcal{E}$	a set of edges or hyperedges or elements
$\mathbb{F}_q$	a finite field of size q
$\mathcal{G}$	an undirected graph
$\boldsymbol{G}$	a generation of data packets
$\mathcal{H}$	a hypergraph
$\mathcal{I}$	the set of all independent sets of a matroid
$\boldsymbol{I}$	an independent set in $\mathcal{I}$
K	the number of data packets in a block
$\mathcal{M}$	a coding set, which is a set of data packets
N	the number of receivers
$\mathcal{P}$	a block of data packets

$\mathbf{p}$	a data packet
$\boldsymbol{P}$	a partition of the packet block $\mathcal{P}$
P_e	the system's packet erasure probability
$P_{e,n}$	the packet erasure probability of Receiver-n
$\mathcal{R}$	the set of all receivers
r	a receiver
$\boldsymbol{S}$	a network coding solution
τ	the set of targeted receivers of a data packet
U	the number of coded transmissions (block completion time)
$\mathbf{u}$	a packet erasure pattern
$\mathcal{V}$	the set of all vertices of a graph or hypergraph
$\boldsymbol{v}$	a vertex of a graph or hypergraph
$\boldsymbol{\omega}$	the set of data packets wanted by a receiver
w	the number of data packets wanted by a receiver, i.e., $w = \lvert\boldsymbol{\omega}\rvert$
$\mathcal{W}$	the set of all receivers' Wants sets
$\boldsymbol{X}$	a network coded packet

Terminology

ACK	acknowledgement
APDD	average packet decoding delay
ARQ	automatic–repeat-request
BCT	block completion time
FC	fountain codes
IC	index coding

IDNC	instantly decodable network coding
LNC	linear network coding
MDS	maximum distance separation
NC	network coding
RLNC	random linear network coding
SFM	state feedback matrix

Chapter 1

Introduction

1.1 A Brief History of Network Coding

In traditional wired networks, the main function of intermediate nodes, such as routers, is to forward each incoming data flow to its destination. When there are multiple incoming flows and the outbound link of the intermediate node has a low capacity, a flow may have to wait a significant amount of time before it is forwarded. Consequently, the node, together with its outbound link, become the bottleneck of the network, and cause network congestion.

For instance consider the network topology below. Two sources wish to send 1 bit of data (x_1 and x_2) to their respective sinks through wired links, all having a capacity of 1 bit per transmission. In this instance, intermediate Node 1 is the bottleneck node, as it has to first forward x_1 and then x_2, which incurs extra waiting time to Sink 2. We also note that, although there are direct links between Source 1 and Sink 2 and between Source 2 and Sink 1, these links are not helpful.

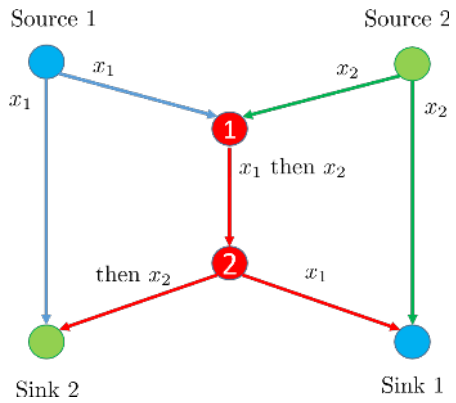


Figure 1.1: A butterfly network with a bottleneck node.

To alleviate the tension, certain congestion control methods can be applied to different parts of the network. On one hand, the source may reduce its transmission rate upon congestion. For example, transmission control protocol (TCP) of the source may halve its transmission rate, or even back-off if congestion persists [1]. On the other hand, intermediate nodes may prioritize the incoming flows for better quality of service (QoS) [2]. Typical metrics include data type (e.g., assign higher priority to video streaming and lower priority to file downloading) and price (e.g., assign higher priority to the flows for subscribed users).

However, congestion control is not the optimal solution because it is not able to increase the overall network throughput, but may even reduce it sometimes. It was a common assumption that it is hardly possible to improve network throughput without infrastructure upgrade, until the invention of network coding (NC).

The year of 2000 witnessed the official birth of NC. In their ground-breaking paper [3], Ahlswede *et al.* proposed that intermediate nodes do not just forward data flows but mix them together using certain coding techniques, which is now widely known as network coding. Based on the idea of NC, the function of Node 1 in Fig. 1.1 can be modified to that in Fig. 1.2: instead of forwarding x_1 and then x_2, Node 1 now generates a binary XOR of the two bits, i.e., $x_1 \oplus x_2$, which is still 1 bit. This NC bit is then sent to both sinks using one transmission. Since Sink 1 and Sink 2 have overheard x_2 and x_1 through the direct link from the other source, respectively, they can decode their wanted bit by solving a set of two simple linear equations. For example, for Sink 1 the equations are:

$$\begin{cases} y_1 = x_2 \\ y_2 = x_1 \oplus x_2 \end{cases}$$

where y_1 and y_2 represent the two bits it has received.

Ahlswede *et al.* then proved the optimality of NC by showing that it can achieve the min-cut-max-flow capacity [3] of wired multicast. Explicitly, they showed that NC allows every demanding sink to receive information from the source at the maximum rate, which is equal to the minimum sum capacity of the sets of links that cut off this sink from the source. However, they did not specify how NC should be implemented to optimally achieve the network capacity region, which is a core NC problem.

The optimal NC for wired multicast remained unclear until Li *et al.* proposed the concept of linear network coding (LNC) [4] in 2003. In LNC, the data unit is a vector over a certain finite field $\mathbb{F}_q$, and is called a message or, in practical terms, a data packet. Data packets are linearly combined with coefficients chosen

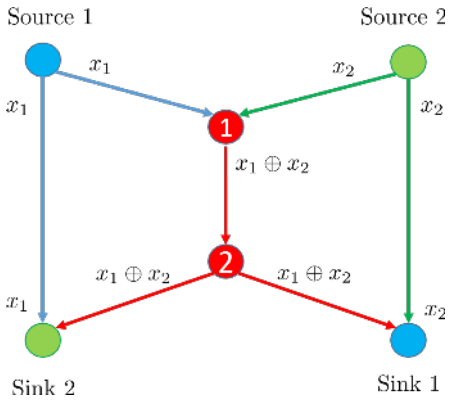


Figure 1.2: Network coding breaks the capacity bottleneck of the butterfly network.

from $\mathbb{F}_q$ to generate NC packets for transmissions. Li *et al.* proved that LNC suffices to achieve the min-cut-max-flow capacity of wired multicast. This result was confirmed again by Koetter *et al.* through proposing an algebraic framework of network coding [5].

The next milestone in the NC literature is the invention of random linear network coding (RLNC) [6] by Ho *et al.* in 2005, for it enables optimal, robust, yet fully decentralized implementations of NC. In RLNC, each node simply generates random linear combinations of all incoming packets using randomly picked coefficients from $\mathbb{F}_q$. It was proved that RLNC is asymptotically capacity-achieving in wired multicast when the size of $\mathbb{F}_q$ is sufficiently large [6].

For other networks where the connections are not multicast, it has been proved in [5] that their capacity optimization using NC is generally an NP-complete problem[1]. It has also been shown by Dougherty *et al.* in [8] that for certain non-multicast networks, the NC capacity achieved by linear NC is strictly less than non-linear NC.

For the general NC optimization problem in arbitrary networks, useful mathematical equivalences have been found. It has been shown by Dougherty *et al.* in [9] that the NC problem is closely related to matroid theory, a branch of abstract mathematics that studies the independence of elements [10]. It has been proved by Rouayheb *et al.* in [11] that both the NC problem and the represen-

[1]NP stands for "non-deterministic polynomial time". An NP-complete problem cannot be optimally solved using an algorithm whose processing time is polynomial in the size of the input of the problem, unless P=NP (P stands for "polynomial time") [7]. But solutions to an NP-complete problem, once provided, can be verified using polynomial-time algorithms. If polynomial-time verification is also not possible, then the problem becomes NP-hard. [7]

tation problem of matroids can be reduced to an index coding (IC) problem. In IC, there is a single source and multiple sinks that are directly connected to the source. Every sink has already received a subset of the messages held by the source as its *side information* [11–15], and still wants one [11–14] or some [15] of the remaining messages. We will revisit IC in Section 1.4.4.

Although a sound theoretical basis has been established for NC in wired networks, it could be difficult to integrate NC into existing wired networks such as the Internet. The main reason is that NC requires special network topologies that enable overhearing: as was explained in previous examples, in order to decode the received NC packet from a link, a sink must overhear some side information from other link(s). The side information could be combinations of data packets it wants (intra-session NC) and/or combinations of data packets it does not want (inter-session NC) [16]. The other reason is that every intermediate node, especially the bottleneck node, must be able to perform NC, which may require hardware and protocol upgrades.

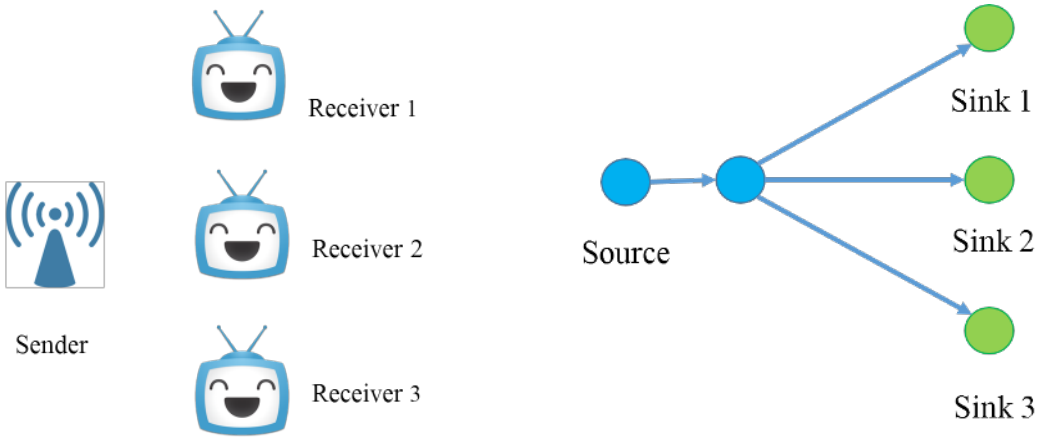
Whilst wired networks' inherent features hinder the implementation of NC, there is another class of networks that intrinsically enables side information and does not impose the existence of intermediate nodes, namely, wireless networks. In the next section, we will review wireless NC.

1.2 Wireless Network Coding: New Challenges

A basic wireless network consists of a sender and a set of wireless receivers within the transmission range of the sender. Due to the broadcast nature of the wireless media, every transmitted packet from the sender can be heard by every receiver. To fully exploit this feature, in this thesis we will use wireless networks for broadcast, a scenario where every receiver wants all the data packets held by the sender. We note that broadcast is a subset of multicast.

From a network topology point of view, a wireless network is equivalent to a wired network where the source is connected to an intermediate node, and this node is connected to every sink through a different link (for example see Fig. 1.3). Hence, all the NC theorems and implementations developed for wired networks should be readily applicable to wireless networks. For example, it is straightforward that LNC suffices to achieve the optimal capacity of wireless multicast and broadcast.

However, there are some significant differences between wireless networks and



(a) wireless network (b) wired network

Figure 1.3: A basic wireless network and its wired equivalence.

their wired counterpart. The most noticeable one is that wireless channels are much more lossy than wired links [17]: packets sent through wireless channels may be erased due to fading and interference and other imperfect channel conditions, which is much less likely to happen in wired links and can be efficiently treated with error correction codes therein. To compensate for packet losses in wireless networks, packet retransmissions are needed.

Consequently, a typical problem setting in wireless broadcast is that each receiver has only received a subset of a block of data packets, and still wants all the remaining data packets. Efficiently recovering these missing data packets is the design goal of coding techniques, and calls for the solving of several challenges.

A core challenge is to minimize the number of retransmissions. This is because the number of retransmissions is inversely related to the system throughput, which is measured by the average number of packets delivered per transmission, and is a common alternative optimization goal to network capacity in practical wireless systems.

Another challenge is to minimize feedback cost. Due to the presence of packet erasures, receiver feedback must be collected for reliable data delivery [18, 19].[2] For example, each receiver should at least send one Acknowledgement (ACK) notification upon reception of all wanted packets from the current block. Otherwise, the sender cannot decide when to stop sending the current block. However, it could be expensive to collect feedback in wireless networks due to the

[2]There are also feedback-free techniques such as forward-error-control codes. But they have non-zero decoding error probabilities and, thus, are out of the scope of this thesis.

bandwidth limitation of the uplink and the energy limitation of the receivers. One such example is wireless sensor networks, where receivers are remote sensors with very limited energy and access to the network. Therefore, the amount of feedback should be minimized in practical implementations of wireless NC techniques [18–21, 21–24].

Besides throughput and feedback, the exploding demand for wireless video streaming raises an important challenge that has been largely overlooked in wired NC techniques, namely, minimizing average packet decoding delay (APDD). NC packets must be decoded before being usable, which only happens after the reception of a certain number of NC packets. This incurs decoding delay of individual data packets [18, 25]. A large APDD is acceptable for applications where data packets are only useful as a whole (e.g., file downloading), but is undesirable or even unacceptable when individual data packets are useful or have a hard deadline (e.g. multi-layer image transmission and video streaming [26–28]).

Moreover, since wireless devices usually have limited energy and hardware resources, the computational complexity of applying NC is a non-negligible aspect in designing wireless NC techniques [29,30]. Some major sources of computational complexity are the complexity of making coding decisions and the complexity of performing the encoding and decoding.

In summary, optimizing throughput, APDD, feedback, and computational complexity are the main challenges in the design of NC techniques for wireless broadcast. Among them, throughput and APDD are the main performance metrics, whilst feedback and computational complexity are implementation costs. In particular, the challenge of optimizing throughput can be solved by LNC.

Therefore, the main theme of this thesis is a comprehensive study on the throughput and APDD optimization for wireless broadcast using LNC. Implementation costs will also be considered and minimized whenever applicable. Following this theme, in this thesis we will present a four-part study. In the rest of this introduction, we will first outline the scope and structure of the thesis. We will then review the literature, and then summarize our contributions.

Before moving on, we remark, as a side note without diving into the details, that studying wireless NC can motivate the design of over-the-top (OTT) NC solutions for wired networks [31]. In such solutions, NC is only applied at the servers and sinks, and the underlying network is treated as a wireless-like media and unaltered. Examples of commercialized products are mainly peer-to-peer (P2P) based, such as Microsoft® Avalanche for its content distribution (e.g., Windows-10 updates) [32, 33] and UUUSee® for video-on-demand (VOD) [34].

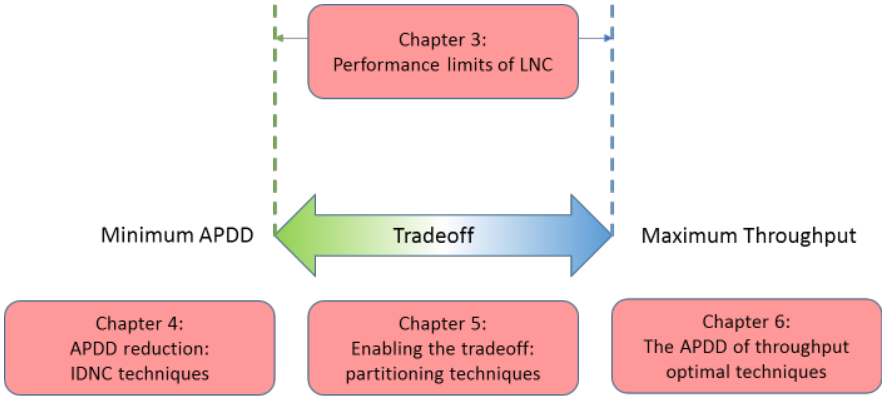


Figure 1.4: **Thesis structure.**

1.3 Thesis Scope and Structure

The fundamental problem addressed in this thesis is:

Problem 1.1

What is the achievable throughput and APDD performance of LNC in wireless broadcast and how to achieve them?

In order to comprehensively solve this problem, we will study LNC's spectrum in terms of throughput and APDD performance, which includes the individual limits of the two, as well as their tradeoff. To this end, we will formally model the wireless broadcast system in Chapter 2, and then conduct a four-part study. The thesis structure is sketched in Fig. 1.4, and is elaborated as follows:

- Chapter 3 studies the throughput and APDD performance limits of LNC, as well as the hardness to find the optimal strategies to achieve such limits.

  Our approach is to study LNC in its abstract form using mathematical theories such as matroid theory (a branch of mathematics that captures and generalizes linear independence in vector space [10]) and hypergraph theory. No specific LNC techniques will be studied. Therefore, results obtained from this chapter apply to all LNC techniques;

- Chapter 4 studies a class of LNC techniques that aim at reducing APDD.

  This class is well known as instantly decodable network coding (IDNC). We will mainly focus on a subclass called strict IDNC (S-IDNC). We will study its throughput and APDD performance limits, compare it with the more general class, and then develop its optimal and heuristic implementations.

- Chapter 5 studies the achievable tradeoff between throughput and APDD.

  By partitioning a large set of data packets into smaller ones before applying LNC, it is possible to tune the tradeoff between throughput and APDD. Such techniques are well-known as generation-based LNC techniques. We will establish and study the optimal partitioning problem, develop its implementations, and evaluate its performance.

- Chapter 6 studies the APDD of throughput-optimal LNC techniques.

  In order to obtain general results, we will apply an abstract model of such LNC techniques rather than specific ones. We will then develop a new throughput-optimal LNC techniques that aims at minimizing APDD, and compare its performance with existing ones, such as the classic random linear network coding (RLNC).

We will then conclude the thesis in Chapter 7.

In summary, we have briefly outlined the theme of each chapter. To provide deeper insights and to stress the contribution of this thesis, in the next section we will review the current art of the aforementioned classes of LNC techniques, as well as elaborate the knowledge gaps. Closing these gaps will be the main focus of the corresponding chapter.

1.4 Network Coding for Wireless Broadcast: A Review

To motivate the development of NC techniques for wireless broadcast, we first briefly discuss a classic uncoded wireless retransmission technique called Automatic-Repeat-reQuest (ARQ). When ARQ is applied, the sender retransmits in each transmission the data packet requested by the most receivers [19].

Although easily implementable, ARQ is not throughput-optimal when there are multiple receivers. This is because the retransmitted data packets are useless to the receivers who have already received them. These receivers' requests can only be addressed in later retransmissions, which may increase the total number of retransmissions and result in a throughput loss.

1.4.1 Throughput-optimal Linear Network Coding

The iconic technique in this class is RLNC. Every RLNC packet is a random linear combination of all data packets in a block. For a receiver who is missing w data packets out of a block of K data packets and has received the rest, RLNC allows this receiver to decode all its w missing data packets w.h.p. (with high probability) upon the reception of any w RLNC packets. Hence, if this receiver does want all its w missing data packets, which is the case in wireless broadcast, then every RLNC packet is useful to it w.h.p. until it has decoded these w packets. Consequently, RLNC is asymptotically throughput-optimal in wireless broadcast.

RLNC also minimizes the amount of receiver feedback. Due to randomized encoding, the sender does not need receivers' packet reception state to make coding decisions. It only requires one block completion ACK from each receiver, and will stop broadcasting the current block when all receivers have acknowledged.

However, RLNC has two drawbacks. The first one is a large APDD. Generally speaking, none of the wanted w data packets can be decoded out until the receiver has received at least w RLNC coded packets and has completed a block decoding procedure. Consequently, every data packet experiences a decoding delay of at least w. The other drawback is a high computational complexity: $\mathcal{O}(w^3)$ to solve a set of w linear equations. The complexity will be further increased when a large finite field size is applied to increase the successful decoding probability.

Various efforts have been made to reduce the computational complexity of RLNC. It has been suggested in [29] that a systematic phase is applied before RLNC, where all data packets are broadcast uncoded once. [29] also suggested binary coding to further reduce the computational complexity at the cost of a graceful degradation in throughput. RLNC with random sparse coefficients (namely, coefficients are more likely to be zeros) [35–38] have been proposed to substantially reduce the computational complexity with graceful degradation on the throughput. There are also throughput-optimal techniques with sparse deterministic coefficients [39], which generate coded packets by collecting receiver feedback after every transmission and solving a hitting-set problem.

However, to the best of out knowledge, all existing throughput-optimal LNC techniques require block decoding in general and, thus, does not reduce APDD. There have not been any analytical results on their APDD performance, nor any attempts to reduce it. This is partly because APDD has not been a major design concern of this class, and partly because their encoding process does not provide

much design flexibility for APDD minimization.

To address the above two drawbacks of RLNC, another class of LNC techniques was invented to enable instant packet decodings under the binary field, which we now review.

1.4.2 Instantly Decodable Network Coding (IDNC)

Instant packet decodings refer to the case where a receiver instantly decodes a data packet upon the reception of one NC packet. For example, in Fig. 1.2, Sink 1 (resp. Sink 2) can instantly decode x_1 (resp. x_2) upon the reception of $x_1 \oplus x_2$ if it already has x_2 (resp. x_1).

LNC techniques that aim at providing such packet decodings are called instantly decodable network coding (IDNC) [21, 26, 27, 40–46]. The idea is to send in each transmission the binary XOR of a selected subset of data packets. Since both coding and decoding are operated under the binary field, IDNC techniques are also computationally friendly. Hence, IDNC techniques are very attractive in delay-sensitive applications where the receivers have limited computational resources, such as video streaming to mobile receivers [26, 27].

The main limitation of IDNC is a generally sub-optimal throughput performance because an instantly decodable packet to a subset of receivers may be useless to some other receivers. It has been shown that IDNC is asymptotically throughput-optimal if there are at most three receivers or the block size is infinite [26]. But the exact characterization of the throughput of IDNC with a larger number of receivers is unknown in the literature. The other limitation is its dependence on receiver feedback to make coding decisions.

Besides receiving instantly decodable or useless packets, a subset of receivers may receive non-instantly decodable packets. For example, if in Fig. 1.2 there is a Sink 3 that has not received any bit, then Sink 3 will find $x_1 \oplus x_2$ useful (in the sense that it contains new bits that Sink 3 does not know) but not instantly decodable. IDNC techniques that prohibit the transmission of non-instantly decodable packets are called strict IDNC (S-IDNC) [40, 43], whilst those allow such packets are called general IDNC (G-IDNC) [26, 27, 41, 42]. S-IDNC transmissions can thus be thought of as a subset of G-IDNC transmissions.

Both G-IDNC and S-IDNC have been extensively studied [21, 23, 24, 26, 27, 40–43, 47]. However, most results are heuristic, including performance analysis guided by some developed theories, as well as implementations. The only performance that has been optimized is the throughput of S-IDNC. However, this optimization

requires solving an NP-hard graph coloring problem [43], and the optimal value does not have a closed-form expression yet. Besides, there has not been any optimization results on the APDD of S-IDNC. Similarly, the best throughput and APDD of G-IDNC are unknown, too.

In summary, IDNC techniques trade throughput off for lower APDD by encoding only a subset of data packets, whilst RLNC trades APDD off for optimal throughput by encoding all data packets together. There is another class of LNC techniques that somewhat sits between RLNC and IDNC: in this class, RLNC is applied to different subsets of data packets from the current block separately. We refer to this class as generation-based LNC techniques and will review it next for its potential in APDD reduction and achieving a better throughput-delay tradeoff.

1.4.3 Generation-based Linear Network Coding

Generation-based LNC techniques [48–53] were first introduced to reduce the decoding computational complexity of RLNC. The idea is to partition a block of data packets into small generations, and then apply RLNC to these generations separately. Consequently, each generation requires solving a smaller set of linear equations than applying RLNC without partitioning. Decoding computational complexity is thus reduced at the cost of a graceful degradation in throughput.

This class may also reduce APDD because data packets are now decoded per generation rather than per whole block [53]. By tuning the generation size, it is even possible to tune the throughput and APDD performance. However, most of the existing results on this class do not provide much insight on APDD minimization, partly because this is not their application focus, and partly because they do not collect receiver feedback for partitioning. Hence, studying and optimizing the APDD of feedback-assisted generation-based LNC is a new research topic.

1.4.4 Related Coding Techniques

Besides NC, there are also other coding techniques applicable to wireless broadcast. For the completeness of this literature review, we briefly remark on two that are closely related to NC.

Index Coding

With proper reduction, the throughput optimization problem of LNC in wireless broadcast can be reduced to a very well studied retransmission minimization problem of IC [11–14]. The approach is to split every receiver that wants multiple data packets in wireless broadcast to multiple (virtual) receivers that only want one data packet in IC.

However, APDD minimization has not been considered in the IC literature. Moreover, most works in the IC literature assume no packet erasures, which is generally not the case in wireless broadcast. Therefore, the problems we will solve in this thesis are different from those in the IC literature, and may provide new insights into the problems in the IC context.

Fountain Codes

Fountain codes (FC) [54–56] are also asymptotically throughput-optimal codes that can work equally well in wireless broadcast as RLNC. With appropriate pre-coding of the input data packets, FC are able to offer linear time encoding and decoding complexity per data packet. However, FC are not necessarily throughput-optimal in more complex networks with relays, because the relays have to decode and re-encode. Moreover, FC in general do not offer as much flexibility in the design and ease of implementation as NC, e.g., when APDD is to be minimized.

According to the above literature review, it is clear that APDD minimization has been largely overlooked in the literature. There has not been a comprehensive study on it, nor any optimization or approximation techniques[3] for it. Moreover, the interplay between APDD, throughput, feedback, and computational complexity has not been well understood in wireless broadcast. These gaps motivated this thesis. In the next section, we will summary the contributions of this thesis.

1.5 Contributions

In this thesis, we will solve Problem 1.1 and close the aforementioned knowledge gaps by applying a wide range of mathematics theories such as matroid theory, graph theory, hypergraph theory, finite field theory, and stochastic processes. Our main contributions are as follows:

[3] A technique is a β-approximation technique of APDD if its APDD performance is at most β times of the minimum APDD.

1. [**Chapter 3**] We deduce a necessary and sufficient condition for LNC to achieve the optimal throughout of wireless broadcast under any given $\mathbb{F}_q$;

2. [**Chapter 3**] We prove the NP-hardness of using LNC for APDD minimization;

3. [**Chapter 3**] We derive closed-form lower bounds of the expected APDD of LNC in wireless broadcast;

4. [**Chapter 4**] We prove that S-IDNC is not able to approximate the minimum APDD in general. But it is able to optimize both throughput and APDD when there are at most three receivers. We develop optimal and heuristic S-IDNC algorithms.

5. [**Chapter 5**] We establish the optimal partitioning problem and prove its NP-hardness. We develop a heuristic partitioning algorithm that achieves local Pareto-optimal[4] throughput-delay tradeoff;

6. [**Chapter 6**] We prove that all throughput-optimal LNC techniques are APDD-approximation techniques with a ratio of between 4/3 and 2. In particular, the approximation ratio of RLNC is exactly 2;

7. [**Chapter 6**] We develop a throughput-optimal and APDD-approximation LNC technique that: 1) always provides instant packet decodings; 2) has an approximation ratio of strictly smaller than 2; 3) uses a polynomial-time encoding algorithm; and 4) does not require intensive receiver feedback.

We also conduct extensive simulations to verify the proposed theorems and properties, and to demonstrate the superiority of the new techniques over existing ones. Most of the results of this thesis have been presented in academic papers, including 7 published ones and 1 under preparation:

1. M. Yu, N. Aboutorab, P. Sadeghi,"From instantly decodable to random linear network coded broadcast," *IEEE Trans. Comm.*, vol. 62, no. 11, pp. 3943–3955, 2014.

2. M. Yu, P. Sadeghi, N. Aboutorab,"Performance characterization and transmission schemes for instantly decodable network coding in wireless broadcast," *Euro J. Advances in Signal Processing*, vol. 94, 2015.

[4]A tradeoff between two metrics is Pareto-optimal if neither metric can be improved without sacrificing the other one.

3. M. Yu, P. Sadeghi, A. Sprintson,"The benefit of limited feedback to generation-based random linear network coding in wireless broadcast," in *Proc. IEEE Global Communications Conference (GLOBECOM) Workshop* , 2016.

4. M. Yu, A. Sprintson, P. Sadeghi, "On minimizing the average packet decoding delay in wireless network coded broadcast," in *Proc. IEEE Int. symp. Network Coding (NetCod)*, 2015.

5. M. Yu, P. Sadeghi, N. Aboutorab,"On deterministic linear network coded broadcast and its relation to matroid theory," in *Proc. IEEE Information Theory Workshop (ITW)*, 2014.

6. P. Sadeghi, M. Yu, N. Aboutorab,"On throughput-delay tradeoff of network coding for wireless communications," (invited paper) in *Proc. IEEE Int. Symp. Information Theory and its Applications (ISITA)*, 2014.

7. M. Yu, N. Aboutorab, P. Sadeghi,"Rapprochement between instantly decodable and random linear network coding," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, 2013.

8. M. Yu, Alex Sprintson, P. Sadeghi,"On the packet decoding delay in wireless network coded broadcast," to be submitted to *IEEE Trans. Wireless Comm.*

Chapter 2

Modeling Linear Network Coded Wireless Broadcast

In this chapter, we review how linear network coding (LNC) can be applied in wireless broadcast systems for performance improvements. We will introduce the basic system settings, demonstrate the general implementation of LNC, define the performance measures, and briefly introduce two LNC techniques that aim to optimize these measures.

2.1 System Model

2.1.1 Basic Settings

We consider a wireless broadcast system depicted in Fig. 2.1. It involves one sender and a set of N receivers, denoted by $\mathcal{R} = \{r_n\}_{n=1}^{N}$. The sender holds a block of K data packets, denoted by $\mathcal{P} = \{\mathbf{p}_k\}_{k=1}^{K}$, and wishes to deliver them to all receivers. All data packets are modelled as equal-length vectors over a given finite field $\mathbb{F}_q$, where q is a power of a prime. In the simplest setting, q is equal to 2 and all data packets are sequences of binary bits.

Time is slotted. In each time slot, the sender broadcasts a packet to all receivers. Due to imperfect wireless media, each receiver r_n either misses the packet with a probability of $P_{e,n}$, or correctly receives it with a probability of $1 - P_{e,n}$. In other words, the wireless downlink from the sender to each receiver r_n is subject to independent and Bernoulli distributed packet erasures with an erasure probability of $P_{e,n}$. This is a common model for wireless erasure broadcast channels [19, 21, 57].

We can think of the K data packets as the K orthogonal bases of a K-dimensional knowledge space. A receiver is able to retrieve these K bases iff it can reproduce this space. To this end, it will need a set of K linearly independent vectors of this space, where each vector is either a basis or a linear combination of the bases. In LNC context, such a combination is called a *NC coded packet*. It is denoted by $\boldsymbol{X}$ and takes a form of:

$$\boldsymbol{X} = \sum_{\mathbf{p} \in \mathcal{M}} \alpha_k \mathbf{p}_k \tag{2.1}$$

where $\{\alpha_k\}$ are non-zero coding coefficients chosen from $\mathbb{F}_q$, and $\mathcal{M} \subseteq \mathcal{P}$ is the set of data packets with non-zero coding coefficients. We call $\mathcal{M}$ the coding set of $\boldsymbol{X}$ and call $\boldsymbol{X}$ a coded packet of $\mathcal{M}$. At a high level, every LNC technique is a method of choosing $\mathcal{M}$ and $\{\alpha_k\}$. Upon the reception of sufficient data and coded packets, the K-dimensional knowledge space can be reproduced, and thus all the K bases can be retrieved through solving linear equation(s).

2.1.2 Transmission Phases

Optimally choosing $\mathcal{M}$ and $\{\alpha_k\}$ is trivial in the initial phase of the broadcast. The sender can simply broadcast every data packet uncoded once using K time slots. This phase is called the *systematic transmission phase*. In this phase, every packet transmission is *innovative* to every receiver, where:

Definition 2.1

A (data or coded) packet is innovative to a receiver r_n if it allows r_n to increase the dimension of its knowledge space by one. In other words, this packet is linearly independent of the set of packets that r_n already has.

Therefore, every transmission in this phase is *throughput-optimal*, where:

Definition 2.2

The transmission of a (data or coded) packet is throughput-optimal if the packet is innovative to every receiver who is still missing packets.

Hence, throughout-optimality is achieved in the systematic transmission phase.

Due to packet erasures, by the end of the systematic transmission phase, each receiver will have received a subset of the data packets and still want the remaining data packets. The packet reception state of all receivers can be summarized by an $N \times K$ state-feedback-matrix (SFM) $\boldsymbol{A}$, where $\boldsymbol{A}(n,k) = 1$ means r_n wants $\mathbf{p}_k$,

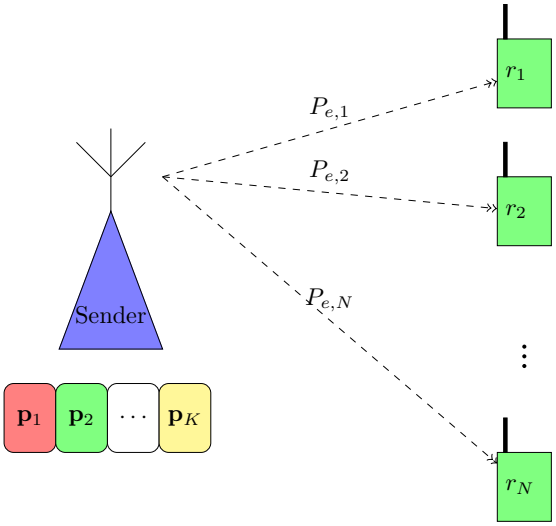


Figure 2.1: **The wireless broadcast of K data packets to N receivers.**

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$
r_1	1	0	0
r_2	0	0	1
r_3	0	1	0

Figure 2.2: **An example of state feedback matrix $\boldsymbol{A}$**

and $\boldsymbol{A}(n,k) = 0$ means r_n already has $\mathbf{p}_k$. The set of data packets wanted by r_n is called the *Wants* set of r_n and is denoted by $\boldsymbol{\omega}_n$. The size of $\boldsymbol{\omega}_n$ is denoted by w_n. The set of $\boldsymbol{\omega}_n$ of all receivers is denoted by $\mathcal{W}$. The set of receivers who want $\mathbf{p}_k$ is called the *Target* set of $\mathbf{p}_k$ and is denoted by τ_k. The size of τ_k is denoted by t_k. An example of SFM is demonstrated in Fig. 2.2, in which $\boldsymbol{\omega}_1 = \{\mathbf{p}_1\}$ and $\tau_2 = \{r_3\}$.

Although the transmission of every data packet is throughput-optimal in the systematic transmission phase, retransmitting them after this phase will generally incur throughput loss, as the data packets are not innovative to the receivers who already have them. To see this, consider the following example.

Example 2.1

Consider the SFM in Fig. 2.2. Receiver r_1, r_2, and r_3 only want $\mathbf{p}_1$, $\mathbf{p}_3$, and $\mathbf{p}_2$, respectively, and have received all the remaining data packets. Retransmitting $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$ separately will cost 3 transmissions, but a coded packet of $\boldsymbol{X} = \mathbf{p}_1 \oplus \mathbf{p}_2 \oplus \mathbf{p}_3$, where $\oplus$ is the binary XOR operator, will allow all receivers to decode their missing data packets upon the reception of $\boldsymbol{X}$. For example, r_1 can decode $\mathbf{p}_1$, as $\mathbf{p}_1 = \boldsymbol{X} \oplus \mathbf{p}_2 \oplus \mathbf{p}_3$.

Hence, after the systematic transmission phase, the sender applies an LNC technique and transmits coded packets until the broadcast is complete, i.e., until all receivers have retrieved all the K data packets. These coded transmissions constitute the second phase of the broadcast, called the *coded transmission phase*. In the next chapter, we will study the achievability of throughput-optimality in this phase.

2.1.3 Classes of Linear Network Coding Techniques

To generate the coded packet in each coded transmission, the sender needs to select the coding set $\mathcal{M}$ and the coding coefficients $\{\alpha_k\}$. The way they are selected broadly classifies LNC techniques into two types:

- Random techniques, in which $\mathcal{M}$ is randomly selected from $\mathcal{P}$, and/or $\{\alpha_k\}$ are randomly selected from $\mathbb{F}_q$;
- Deterministic techniques, in which $\mathcal{M}$ and/or $\{\alpha_k\}$ are deterministically selected (from $\mathbb{F}_q$ for $\{\alpha_k\}$).

For example, the RLNC technique sets $\mathcal{M} = \mathcal{P}$, and chooses $\{\alpha_k\}_{k=1}^{K}$ uniformly at random from $\mathbb{F}_q$. For another example, the instantly decodable network coding (IDNC) technique strategically chooses $\mathcal{M}$, and then adds all data packets in $\mathcal{M}$ together under the binary field $\mathbb{F}_2$. We will discuss these two techniques in more detail by the end of this section.

2.1.4 Receiver Feedback

Some LNC techniques rely on the packet reception state of receivers to make coding decisions. This information is accessible through receiver feedback, which is assumed to be erasure-free and delay-free. At the minimum, both random and deterministic LNC techniques require one round of feedback from every receiver

after this receiver has retrieved all data packets. Upon the reception of this feedback from all receivers, the sender can complete the broadcast of the current block. Besides, deterministic LNC techniques require one round of feedback immediately after the systematic transmission phase to construct the SFM. In addition, the sender may also collect feedback during the coded transmission phase. The explicit feedback frequency of different LNC techniques will be discussed in later chapters.

In summary, we consider a two-phase wireless broadcast of K data packets to N receivers through wireless channels that are subject to independent packet erasures. In the systematic transmission phase, data packets are transmitted uncoded once. Then in the coded transmission phase, coded packets are transmitted, where each is a linear combination of the data packets generated using an LNC technique. Receivers send feedback at an appropriate frequency (depending on the LNC technique) to assist the sender in making coding decisions. They solve linear equation(s) to retrieve all data packets.

As demonstrated in Example 2.1, applying LNC after the systematic transmission phase can significantly improve the broadcast efficiency compared with uncoded packet retransmission schemes, such as Automatic-Repeat-reQuest (ARQ) [19]. In this thesis, we consider two fundamental performance measures, namely, throughput and packet decoding delay, which are defined next.

2.2 Throughput and Decoding Delay Measures

In this section, we define the throughput and packet decoding delay measures. For each measure we will introduce an LNC technique that aims to optimize it.

2.2.1 Throughput and RLNC

We denote by U the total number of transmissions in the coded transmission phase in the broadcast of a block of K data packets, and call it the block completion time (BCT). U is inversely related to throughput, because the system spends a total of $K + U$ transmissions to broadcast a set of K data packets, yielding a throughput of $\frac{K}{K+U}$ packet per transmission. Hence, maximizing throughput is equivalent to minimizing U.

In order to understand how U can be minimized, we study the decoding condition of each receiver. In the coded transmission phase, the coded packets each receiver r_n has received can be represented by a *coding coefficient matrix* $\boldsymbol{C}$:

Definition 2.3

For a receiver r_n with u received coded packets, its coding coefficient matrix $\boldsymbol{C}_n$ is an $u \times K$ matrix under $\mathbb{F}_q$, where $\boldsymbol{C}_n(i,k)$ is the coefficient of $\mathbf{p}_k$ in the i-th coded packet:

$$\boldsymbol{C}_n = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,K} \\ \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,K} \\ \vdots & \ddots & \ddots & \vdots \\ \alpha_{u,1} & \alpha_{u,2} & \cdots & \alpha_{u,K} \end{bmatrix}_{u \times K} \tag{2.2}$$

We can then easily prove the following decoding condition:

Condition 2.1

In order to allow r_n to decode all the data packets in $\boldsymbol{\omega}_n$, the set of columns of $\boldsymbol{C}_n$ indexed by $\boldsymbol{\omega}_n$ must have a rank of w_n.

For example, if r_n wants $\boldsymbol{\omega}_n = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$, then the first three columns of $\boldsymbol{C}_n$ must have a rank of 3, so that r_n can decode $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$ through solving a set of 3 linear equations.

To satisfy this condition, the number of received coded packets must satisfy $u \geqslant w_n$, because otherwise with $u < w_n$ rows, the rank of the columns can at most be u, but never be w_n. The minimum value of u is w_n and is achieved iff every received coded packet is innovative (defined in Definition Definition 2.1) to r_n.

Therefore, the BCT U is minimized when every receiver r_n can decode all its w_n wanted data packets after receiving w_n coded packets. This requires that every transmitted coded packet must be innovative to every receiver who is still missing packets. Consequently, we have the concept of throughput-optimal LNC technique:

Definition 2.4

A LNC technique is throughput-optimal if the transmission of every coded packet generated by it is throughput-optimal.

A well-known throughput-optimal LNC technique in the literature is the RLNC technique, first introduced by Ho *et. al.* in their celebrated paper [6] in 2006. Every RLNC coded packet is a random linear combination of all data packets in $\mathcal{P}$ with coefficients chosen uniformly at randomly from $\mathbb{F}_q$. There-

fore, $\boldsymbol{C}_n$ is a matrix with randomly valued entries. When $\mathbb{F}_q$ is sufficiently large, RLNC asymptotically ensures that any $w_n \times w_n$ sub-matrix of $\boldsymbol{C}_n$ has a rank of w_n. Therefore, upon the reception of any w_n RLNC coded packets, every receiver r_n can decode its w_n wanted data packets by solving a set of w_n linear equations provided by these w_n RLNC coded packets.

Due to its random nature of coding, RLNC has an additional advantage that it does not require intermediate feedback after the systematic transmission phase and during the coded transmission phase. But there are two main problems in using RLNC. The first one is high decoding computational load:

Definition 2.5

Each receiver performs Gaussian eliminations to solve linear equations. The computational load of solving a set of w linear equations is $\mathcal{O}(w^3)$ operations.

We can then easily show that the maximum computational load of a receiver r_n is $\mathcal{O}(w_n^3)$. This maximum is reached by RLNC because r_n has to solve a set of w_n linear equations in the RLNC decoding process. Hence, RLNC requires the highest decoding computational load among all LNC techniques. The above decoding process also implies the second and main problem of RLNC, namely, RLNC is inefficient in terms of packet decoding delay, which we now define.

2.2.2 Average Packet Decoding Delay (APDD) and IDNC

When LNC is applied, it is likely to be the case that a receiver has to collect a certain number of coded packets before being able to decode any individual data packet. This feature is acceptable if the receivers are only interested in the block of data packets as a whole. But this is undesirable in applications where individual data packets are useful, such as image transmissions and video streaming [26, 28, 57]. Therefore, we are interested in the decoding delay of the individual data packets. We measure this performance through the average packet decoding delay (APDD), denoted by D:

$$D = \frac{1}{T} \sum_{\forall n,k:\boldsymbol{A}_{n,k}=1} u_{n,k} \tag{2.3}$$

where $u_{n,k}$ is the index of the coded transmission when r_n decodes $\mathbf{p}_k$, and $T = \sum_{k=1}^{K} t_k$ is the total number of targeted receivers of all data packets, which is also

the number of "1"s in $\boldsymbol{A}$. Note that in this thesis we do not consider ordered packets. Thus, all the data packets have the same decoding priority.

We note that D is the overall APDD of $\boldsymbol{A}$ under a realization of the coded transmission phase. Applying the same idea, we can also calculate the APDD experienced by each receiver r_n, denoted by D_n:

$$D_n = \frac{1}{w_n} \sum_{\forall k: \mathbf{p}_k \in \boldsymbol{\omega}_n} u_{n,k} \tag{2.4}$$

We further note that there are also other measures of packet decoding delay in the literature. A common one is that a receiver experiences one unit increase of decoding delay if it has received a coded packet, but cannot decode any new data packet from it [58]. However, this measure does not consider the exact decoding delay of each data packet, and thus cannot fully reflect the packet decoding delay performance of LNC techniques. To see this, consider the following example:

Example 2.2

Consider two receivers: r_1 decodes one wanted data packet in the first and third coded transmissions, respectively, but cannot decode in the second coded transmission; r_2 decodes one wanted data packet in the second and third coded transmissions, respectively, but cannot decode in the first coded transmission. According to the above measure, the packet decoding delay experienced by both r_1 and r_2 is 1. However, it is obvious that r_1 has faster packet decodings. If our measure is applied, the APDD of r_1 and r_2 is 2 and 2.5, respectively.

It is intuitive that the key to reducing D is to reduce the number of coded packets that each receiver r_n needs to collect before being able to decode individual data packets. In the best case scenario, r_n can *instantly decode* a wanted data packet using only one coded packet. To this end, the coding set of this coded packet must contain only one wanted data packet of r_n, i.e., $|\mathcal{M} \cap \boldsymbol{\omega}_n| = 1$. A well-known class of deterministic LNC techniques that aim at designing such coded packets is called IDNC. The coded packet we have generated in Example 2.1 is an IDNC coded packet.

Since receivers are able to decode by using only one coded packet, the decoding computational load of IDNC techniques is much lower than RLNC. Since IDNC techniques make deterministic coding decisions, they require intermediate feedback from receivers after the systematic transmission phase and during the coded transmission phase. The main problem of IDNC techniques is that they

are not necessarily throughput-optimal because there may exist a subset of receivers with their $|\boldsymbol{\omega}_n \cap \mathcal{M}| = 0$, who will find the corresponding coded packet non-innovative.

2.3 Conclusion

In this chapter we have established the network coded wireless broadcast system considered in this thesis. We have also introduced the main performance measures that will be optimized in this thesis, including block completion time U and average packet decoding delay D. We have also shed some light on how could they be reduced by using specific LNC techniques such as RLNC and IDNC.

However, before starting to tackle the optimization problems using any specific LNC techniques, it is important to first understand the general performance limits of LNC techniques in terms of U and D. Hence, in the next chapter, we will study the minimum possible U and D that LNC techniques can achieve when there are no packet erasure in the coded transmission phase.

Chapter 3

Fundamental Limits of Linear Network Coded Wireless Broadcast

This chapter focuses on the fundamental performance limits of linear network coding (LNC) in wireless broadcast. The two performance measures are the block completion time U and the average packet decoding delay D defined in the last chapter. Although U and D vary under different LNC techniques, there exist lower bounds of U and D that no LNC technique can break. Studying the values and achievability of these bounds is important for understanding the fundamental limits of LNC.

To this end, we will first study the lower bounds of U and D for any given state feedback matrix (SFM) by assuming no packet erasures in the coded transmission phase. The results will also serve as fundamental limits in the presence of random packet erasures. Then by further assuming random packet erasures, we will derive lower bounds of the expected D, and study its relation with system parameters, including the number of data packets and receivers and the packet erasure probability. We will not study the expected U, as this is well understood in the literature, e.g., by studying the distribution of U of RLNC [59] under random packet erasures, as RLNC is able to asymptotically minimizes U.

3.1 Throughput

Given a SFM $\boldsymbol{A}$, we denoted by U_{min} the minimum possible block completion time (BCT) that LNC techniques can achieve. The value of U_{min} is straightforward: let $w_{\max}$ be the largest size of the Wants sets of all receivers, i.e., $w_{\max} \triangleq \max\{w_1, \cdots, w_n\}$, we have that $U_{min} = w_{\max}$ because each receiver r_n

needs at least w_n coded packets to decode its w_n wanted data packets.

We then study the achievability of U_{min} under a finite field $\mathbb{F}_q$ by considering the coefficient matrix of a set of U_{min} coded packets:

$$\boldsymbol{C} = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,K} \\ \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,K} \\ \vdots & \ddots & \ddots & \vdots \\ \alpha_{U_{min},1} & \alpha_{U_{min},2} & \cdots & \alpha_{U_{min},K} \end{bmatrix}_{U_{min} \times K} \tag{3.1}$$

According to the decoding condition defined in Condition 2.1 in the last chapter, a receiver r_n can only decode all its wanted data packets from $\boldsymbol{C}$ if the columns of $\boldsymbol{C}$ indexed by $\boldsymbol{\omega}_n$ are linearly independent, for all $n \in [1, N]$. Hence, the achievability of U_{min} is translated into the existence of some $\boldsymbol{C}$ under $\mathbb{F}_q$ such that its columns satisfy some linear independence constraints imposed by $\mathcal{W} \triangleq \{\boldsymbol{\omega}_n\}_{n=1}^{N}$. This problem is closely related to a field in abstract mathematics called matroid theory [60], which has been extensively used to characterize the broader index coding problem. We will first briefly introduce matroid theory, and then develop its connection with our problem, which is the main contribution of this section.

3.1.1 Preliminaries of Matroid Theory

A matroid M is an ordered pair $(E, \mathcal{I})$. E is a finite set of elements called the *ground set*. $\mathcal{I}$ is a family of subsets of E called *independent sets*. An independent set is denoted by I and its rank is equal to its cardinality, i.e., $r(I) = |I|$, where $r(\cdot)$ is a rank function. A maximal independent set is called a basis. All bases of a matroid have the same size and rank. Their rank is also the rank of M, denoted by r_M. On the other hand, all the subsets of E not in $\mathcal{I}$ are dependent sets. The meaning of independence can be visualized through a representation of matroid called matrix matroid.

Definition 3.1

An $r_M \times |E|$ matrix $\boldsymbol{C}$ over $\mathbb{F}_q$ represents a matroid $M(E, \mathcal{I})$ and is called a matrix matroid if the columns of $\boldsymbol{C}$ indexed by any $I \in \mathcal{I}$ are linearly independent, and those not indexed by I are linearly dependent.

A matroid is *q-representable* if it has a matrix representation $\boldsymbol{C}$ over $\mathbb{F}_q$.

Example 1. *Consider a matroid M with $E = \{1, 2, 3\}$ and $\mathcal{I} = \{1, 2, 3, (1, 2), (2, 3)\}$. Its bases are $(1, 2)$ and $(2, 3)$. Its rank is thus $r_M = 2$. M is $2-$representable by*

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Figure 3.1: The matrix representation of a matroid with $E = \{1, 2, 3\}$ and $\mathcal{I} = \{1, 2, 3, (1, 2), (2, 3)\}$.

the 2×3 binary matrix $\boldsymbol{C}$ in Fig. 3.1(a), because column sets (1,2) and (2,3) are linearly independent sets according to $\mathcal{I}$, while (1,3) is a dependent set.

There is a special type of matroid called *uniform matroid*, denoted by $\mathcal{U}_K^{r_M}$. It is a matroid that 1) has K elements, 2) has a rank of r_M, and 3) every size-r_M subset of the elements is a basis.

3.1.2 Achievability of U_{min} v.s. Matroid Representability

Recall that U_{min} is achieved iff there exists a $U_{min} \times K$ matrix $\boldsymbol{C}$ under $\mathbb{F}_q$ such that its columns indexed by any $\boldsymbol{\omega}_n \in \mathcal{W}$ are linearly independent. Compare this condition with the definition of matrix matroid, we reach the following theorem:

Theorem 3.1:

A sufficient condition for the achievability of U_{min} under $\mathbb{F}_q$ is that the uniform matroid $\mathcal{U}_K^{U_{min}}$ is q-representable. This condition becomes also necessary if $\mathcal{W}$ contains all the bases of $\mathcal{U}_K^{U_{min}}$.

Proof. This theorem holds because the $\mathcal{I}$ of $\mathcal{U}_K^{U_{min}}$ contains all the size-1 to size-U_{min} subsets of the K elements. Since every receiver r_n wants at most U_{min} out of K data packets, we have $\boldsymbol{\omega}_n \in \mathcal{I}$ for every $n \in [1, N]$, indicating that the decoding condition on $\boldsymbol{C}$ required by r_n can be satisfied by the matrix matroid of $\mathcal{U}_K^{U_{min}}$. Moreover, if every set of U_{min} data packets are wanted by a different receiver, then $\mathcal{W}$ contains all the bases of $\mathcal{U}_K^{U_{min}}$. In this case, the only $\boldsymbol{C}$ that achieves U_{min} is the matrix matroid of $U_K^{U_{min}}$. □

Example 2. *Uniform matroid $\mathcal{U}_4^2$ has a ground set $E = \{1, 2, 3, 4\}$ and a rank of 2. Every one-element and two-element subset of E is an independent set. Consider an SFM with $K = 4$ data packets and $N = 6$ receivers. Each receiver wants a different pair of two data packets, and thus $U_{min} = 2$. Both the representability of $\mathcal{U}_4^2$ and the achievability of U_{min} requires a 2×4 matrix $\boldsymbol{C}$ such that every two columns of $\boldsymbol{C}$ are linearly independent. This implies that all the 4 columns must be distinct.*

Table 3.1: $U_K^{U_{min}}$ is q-representable iff $K \leqslant K'$.

U_{min}	K'	restriction on q
1	no	no
2	$q+1$	no
3	$q+1$	q odd
	$q+2$	q even

U_{min}	K'	restriction on q
4	5	$q \leqslant 3$
	$q+1$	$q \geqslant 4$
5	6	$q \geqslant 4$
	$q+1$	$q \geqslant 5$

This requirement is not feasible under the binary field $\mathbb{F}_2$, because there are only 3 distinctive length-2 non-zero columns under $\mathbb{F}_2$ as shown below. Consequently, the 4-th column must repeat a previous column, which fails the requirement. Hence, $\mathcal{U}_4^2$ is not 2-representable and U_{min} cannot be achieved over $\mathbb{F}_2$. On the other hand, $\mathcal{U}_4^2$ is q-representable over any $q \geqslant 3$, and thus U_{min} can be achieved as well.

$$\boldsymbol{C} = \begin{bmatrix} 0 & 1 & 1 & ? \\ 1 & 0 & 1 & ? \end{bmatrix} \tag{3.2}$$

Theorem 3.1 indicates that the achievability of U_{min} under $\mathbb{F}_q$ can be translated into the q-representability of uniform matroid $\mathcal{U}_K^{U_{min}}$. However, this problem is largely open: studies on the q-representability of $\mathcal{U}_K^{U_{min}}$ is only complete for $U_{min} \leqslant 5$ [60], with results summarized in Table 3.1. For other values of U_{min}, it is not clear under what $\mathbb{F}_q$ $\mathcal{U}_K^{U_{min}}$ is q-representable. Moreover, there has not been a polynomial-time algorithm that optimally finds the matrix representation of a matroid [60].

Despite the openness of the problem, all existing results support a general conjecture that uniform matroids are more likely to be representable over large $\mathbb{F}_q$. For example, the well known maximum distance separation (MDS) conjecture in coding theory argues, after some translation [61], that $q \geqslant K - 1$.

Therefore, the achievability of U_{min} is an open problem in general and does not have an optimal algorithm for its solutions. But U_{min} could be asymptotically achieved over large $\mathbb{F}_q$. For example, random linear network coding (RLNC) chooses coding coefficients uniformly at random from a sufficiently large $\mathbb{F}_q$, so that in the $U_{min} \times K$ coefficient matrix of any U_{min} coded packets, any U_{min} columns are linearly independent with a high probability [6].

3.2 Average Packet Decoding Delay

Given a SFM $\boldsymbol{A}$, we denoted by $D_{\min}$ the minimum average packet decoding delay (APDD) that LNC can achieve even without packet erasures. Unlike U_{min}, the value of $D_{\min}$ has not been studied in the literature. In this section, we will prove that it is NP-hard to find and achieve $D_{\min}$. We will then study the best D that LNC techniques can be *expected* to achieve on average under random packet erasures.

For the proof of the NP-hardness of finding and achieving $D_{\min}$, our approach is to first derive a *lower bound* of $D_{\min}$, then prove the NP-hardness of achieving this lower bound. This result will indicate the NP-hardness of finding $D_{\min}$, because otherwise by finding $D_{\min}$, we can immediately determine the achievability of the lower bound. The NP-hardness of finding $D_{\min}$ will further indicate that it is NP-hard to achieve $D_{\min}$, because otherwise by achieving it, we can immediately find its value.

We start with presenting our newly developed lower bound on D, which can only be achieved by a *perfect LNC solution*.

3.2.1 The Perfect LNC Solution

A set of ordered coded packets $\{\boldsymbol{X}_u\}_{u=1}^{U}$ is called an LNC solution and is denoted by $\mathcal{S}$ if, upon the reception of all these coded packets, every receiver can decode all its wanted data packets. Then,

Definition 3.2

An LNC solution $\mathcal{S}$ is called a perfect solution and is denoted by $\mathcal{S}_p$ if it allows every receiver r_n to decode a wanted data packet in every transmission, until r_n has decoded all its wanted data packets.

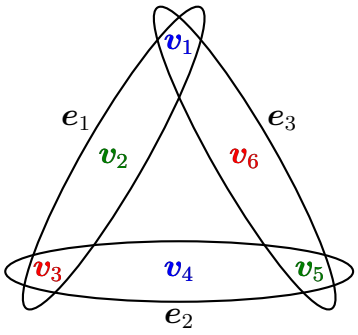
According to its definition, $\mathcal{S}_p$ is throughput-optimal, for every transmission is throughput-optimal. More importantly, $\mathcal{S}_p$ offers the ideal packet decodings. Its average packet decoding delay (APDD), denoted by D_0, is thus a lower bound on $D_{\min}$, and is calculated as:

$$D_0 = \frac{1}{\sum_{n=1}^{N} w_n} \sum_{n=1}^{N} \sum_{u=1}^{w_n} u \tag{3.3}$$

$$= \frac{\sum_{n=1}^{N} w_n^2}{2\sum_{n=1}^{N} w_n} + \frac{1}{2}, \tag{3.4}$$

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	$\mathbf{p}_4$	$\mathbf{p}_5$	$\mathbf{p}_6$
r_1	1	1	1	0	0	0
r_2	0	0	1	1	1	0
r_3	1	0	0	0	1	1

(a) $\boldsymbol{A}$



(b) $\mathcal{H}$

Figure 3.2: The hypergraph model $\mathcal{H}$ of an SFM $\boldsymbol{A}$

It is clear that D_0 can only be achieved if $\mathcal{S}_p$ exists. The natural question in this context is: *Does a perfect solution $\mathcal{S}_p$ exist for a given SFM $\boldsymbol{A}$?* In the next subsection, by using a reduction from the strong hypergraph coloring problem, we will prove that this question is NP-complete to answer.

3.2.2 The Hardness of APDD Minimization: A Hypergraph Coloring Approach

A hypergraph $\mathcal{H}$ is defined by a pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices, and $\mathcal{E}$ is the set of hyperedges. Every hyperedge $\boldsymbol{e} \in \mathcal{E}$ is a subset of $\mathcal{V}$ with size $|\boldsymbol{e}| \geqslant 1$. $\mathcal{H}$ can be used to model the packet reception instance. For each data packet $\mathbf{p}_k$ we generate a vertex $\boldsymbol{v}_k \in \mathcal{V}$, and for each receiver r_n we generate a hyperedge $\boldsymbol{e}_n \in \mathcal{E}$ that is incident to the vertices/packets wanted by r_n, i.e., let $\boldsymbol{e}_n = \boldsymbol{\omega}_n$. An example of SFM and its hypergraph model are demonstrated in Fig. 3.2. Similarly, given any hypergraph $\mathcal{H}$, we can also generate an SFM $\boldsymbol{A}$. Hence, there is a bijection between $\boldsymbol{A}$ and $\mathcal{H}$.

We then introduce some related concepts from the hypergraph theory. A hypergraph is $r-$uniform if the size of all hyperedges is r, i.e., $|\boldsymbol{e}| = r$ for every $\boldsymbol{e} \in \mathcal{E}$. A size-$k$ strong coloring of $\mathcal{H}$ is a partition of $\mathcal{V}$ into k subsets $\{\mathcal{V}_i\}_{i=1}^k$, such that $|\mathcal{V}_i \cap \boldsymbol{e}| \leqslant 1$ for every $\boldsymbol{e} \in \mathcal{E}$. In other words, if we assign k colors to $\{\mathcal{V}_i\}_{i=1}^k$, respectively, every color appears at most once in every hyperedge. It can be proved (given in Appendix A) that the hypergraph coloring problem is intractable:

Lemma 3.1:

It is NP-complete to determine whether an $r-$uniform hypergraph is size-r strong colorable or not, for any $r \geqslant 3$.

This lemma indicates the hardness of finding $\mathcal{S}_p$:

Corollary 3.1:

It is NP-complete to determine whether there exists a perfect solution for a given $\boldsymbol{A}$.

Proof. Here we only need to prove that an r-uniform hypergraph is size-r strong colorable iff there exists a perfect LNC solution of the corresponding $\boldsymbol{A}$, in which every receiver wants r data packets. If this is proved, then according to Lemma 3.1, it is NP-complete to determine the existence of a perfect solution for such $\boldsymbol{A}$. This will indicate that the problem is NP-complete under general $\boldsymbol{A}$.

First, we prove that a size-r strong coloring $\{\mathcal{V}_i\}_{i=1}^r$ of $\mathcal{H}$ implies a perfect solution $\mathcal{S}_p$ of $\boldsymbol{A}$. Since for every hyperedge it holds that $|\boldsymbol{e}_n| = r$ and there are r colors, we have $|\mathcal{V}_i \cap \boldsymbol{e}_n| = 1$. Let $\{\mathcal{M}_i\}_{i=1}^r$ be the sets of packets corresponding to $\{\mathcal{V}_i\}_{i=1}^r$. Then, we have $|\mathcal{M}_i \cap \boldsymbol{\omega}_n| = 1$ for every receiver r_n. Hence, the linear sum of all data packets from $\mathcal{M}_i$ is a coded packet $\boldsymbol{X}_i$ that allows every receiver to immediately decode a wanted data packet. Therefore, $\{X_i\}_{i=1}^r$ together form a perfect solution $\mathcal{S}_p$ of $\boldsymbol{A}$.

Next, we prove that a perfect solution $\mathcal{S}_p$ of $\boldsymbol{A}$ implies a size-r strong coloring $\{\mathcal{V}_i\}_{i=1}^r$ of $\mathcal{H}$. Since every receiver wants r data packets, $\mathcal{S}_p$ contains r coded packets $\{X_i\}_{i=1}^r$. In order to allow every receiver to decode one wanted data packet from X_i, the coding set $\mathcal{M}_i$ of X_i must contain exactly one wanted data packet of every receiver, i.e., $|\mathcal{M}_i \cap \boldsymbol{\omega}_n| = 1$. Let $\{\mathcal{V}_i\}_{i=1}^r$ be the sets of vertices corresponding to $\{\mathcal{M}_i\}_{i=1}^r$, it holds that $|\mathcal{V}_i \cap \boldsymbol{e}_n| = 1$ for every hyperedge. Thus, $\{\mathcal{V}_i\}_{i=1}^r$ is a size-r strong coloring of $\mathcal{H}$.

Therefore, a size-r strong coloring of an r-uniform hypergraph is equivalent to a perfect solution of the corresponding $\boldsymbol{A}$. Then according to Lemma 3.1, Corollary 3.1 is true. □

Example 3.1

The SFM in Fig. 3.2(a) has a perfect solution that contains three coded packets: $\boldsymbol{X}_1 = \mathbf{p}_1 \oplus \mathbf{p}_4$, $\boldsymbol{X}_2 = \mathbf{p}_2 \oplus \mathbf{p}_5$, and $\boldsymbol{X}_3 = \mathbf{p}_3 \oplus \mathbf{p}_6$, where $\oplus$ is the binary XOR operator. Then, by coloring $\{\boldsymbol{v}_1, \boldsymbol{v}_4\}$, $\{\boldsymbol{v}_2, \boldsymbol{v}_5\}$, and $\{\boldsymbol{v}_3, \boldsymbol{v}_6\}$ in the corresponding hypergraph $\mathcal{H}$ using three different colors, we obtain a size-3 strong coloring of $\mathcal{H}$, as shown in Fig. 3.2(b).

Since D_0 can only be achieved by a perfect solution $\mathcal{S}_p$, an optimal algorithm that finds $D_{\min}$ will be able to determine the existence of a perfect solution through comparing $D_{\min}$ with D_0. According to Corollary 3.1, this decision is NP-complete to make, and thus it is NP-hard to find $D_{\min}$:

Theorem 3.2

It is NP-hard to find $D_{\min}$ for a given SFM $\boldsymbol{A}$.

This theorem also indicates that there is no network coding technique that can achieve $D_{\min}$ using a deterministic polynomial-time coding algorithm unless **P=NP**.

3.2.3 APDD Approximation

Since $D_{\min}$ is NP-hard to achieve, we are interested in its approximation. An LNC technique is said to be a β-approximation technique of $D_{\min}$ if its APDD is at most β (inclusive) times of $D_{\min}$. Such a technique, upon its existence, will be highly appreciated for providing guaranteed APDD performance.

However, being able to approximate $D_{\min}$ is not sufficient. In practice, the minimum APDD of LNC varies under different packet erasure patterns. A stronger approximation technique should be able to approximate the minimum APDD of LNC under any packet erasure patterns. But since there are potentially infinite number of such patterns, it is impossible to examine all of them in a brute-force manner. Instead, it is more meaningful to study the minimum expected APDD under any given packet erasure probability and define APDD-approximation techniques as follows:

Definition 3.3

An LNC technique is said to be a β-approximation technique of the APDD minimization problem if for any given SFM $\boldsymbol{A}$ and any packet erasure probability $\{P_{e,n}\}_{n=1}^{N} \geqslant 0$, the expected APDD using this technique is at most β times of the minimum expected APDD of LNC.

Due to the NP-hardness of finding $D_{\min}$ when $\{P_{e,n}\}_{n=1}^{N} = 0$, it is also NP-hard to find the minimum APDD of LNC under a more general setting on packet erasure probabilities. To solve this problem, in the next subsection we will derive lower bounds of the minimum expected APDD of LNC. These bounds will serve as the APDD performance limits of LNC under wireless broadcast, and will also help with verifying APDD-approximation techniques. This is because if a technique's APDD is at most β times of the lower bound, then its approximation ratio is *at most* β.

3.2.4 Lower bounds of the Expected APDD

In this subsection, we derive lower bounds of the minimum expected APDD of all LNC techniques under random packet erasures. To this end, we will extend the concept of "perfect LNC solution" to "perfect LNC technique", and study what APDD performance can be expected from this perfect technique under random packet erasures. The results will serve as lower bounds of the expected APDD of all LNC techniques. In particular, we are interested in two types of expectations:

- The expected APDD of a given SFM using LNC. The expectation will be the APDD averaged over all possible erasure patterns in the coded transmission phase;
- The overall expected APDD of the wireless broadcast system using LNC. The expectation will be the APDD averaged over all possible erasure patterns in both the systematic and coded transmission phases. The resultant expected APDD will reflect the overall APDD performance in terms of system parameters, including the number of data packets K, the number of receivers N, and the packet erasure probabilities $\{P_{e,n}\}_{n=1}^{N}$.

We first extend the concept of perfect LNC solution to the concept of perfect LNC technique under random packet erasures.

Definition 3.4

A perfect LNC technique allows every receiver to decode a wanted data packet whenever this receiver successfully receives a coded packet generated using this technique.

. Similar to the perfect LNC solution, a perfect LNC technique is also throughput-optimal, and also offers the ideal packet decodings.

In order to derive the expected APDD of all receivers when the perfect LNC technique is applied, we first derive the expected APDD of a single receiver. Let $\underline{D}_n$ denote the APDD of receiver r_n under an arbitrary erasure pattern when the perfect LNC technique is applied in the coded transmission phase. (Here the underline means lowed bound.) Then, by averaging over all possible erasure patterns, we obtain the expected $\underline{D}_n$ of r_n, which is given in the following theorem.

Lemma 3.2

When coded transmissions are subject to random packet erasures with a probability of $P_{e,n}$, the expected $\underline{D}_n$ of a receiver r_n who wants w_n data packets is:

$$E[\underline{D}_n|\text{when } r_n \text{ wants } w_n \text{ data packets}] = \frac{w_n + 1}{2(1 - P_{e,n})} \tag{3.5}$$

Its proof is given in Appendix B. In the rest of this subsection, we will simplify the notation for such conditional expectations to a form of $E[a|(b, c, \cdots)]$, which stands for the expectation of variable a when the values of variables $b, c, \cdots$ are given. For example, the expectation in (3.5) can be simplified to $E[\underline{D}_n|w_n]$.

By applying this lemma, we can immediately obtain the following theorem:

Theorem 3.3

Let $E[\underline{D}|\{w_n\}_{n=1}^N]$ be the expected APDD of a given SFM $\boldsymbol{A}$ when the perfect LNC technique is applied, where $\{w_n\}_1^N$ are known from $\boldsymbol{A}$. Then,

$$E[\underline{D}|\{w_n\}_{n=1}^N] = \frac{\sum_{n=1}^N \frac{w_n^2+w_n}{2(1-P_{e,n})}}{\sum_{n=1}^N w_n} \tag{3.6}$$

This is a lower bound of the expected APDD of $\boldsymbol{A}$ when LNC is applied.

Proof. The APDD of an SFM can be calculated as the weighted average of the

APDD of each receiver:

$$E[\underline{D}|\{w_n\}_{n=1}^N] = \frac{E[\underline{D}_1|w_1]\cdot w_1 + E[\underline{D}_2|w_2]\cdot w_2 + \cdots + E[\underline{D}_N|w_N]\cdot w_N}{w_1 + w_2 + \cdots + w_N}$$
$$= \frac{\sum_{n=1}^N \frac{w_n^2+w_n}{2(1-P_{e,n})}}{\sum_{n=1}^N w_n}$$

Since the perfect LNC technique minimizes the expected APDD of every receiver, it also minimizes the expected APDD of the SFM. □

Then, by noting that the SFM $\boldsymbol{A}$ itself is the consequence of random packet erasures in the systematic transmission phase, it is possible to derive the expectation of $E[\underline{D}|\{w_n\}_{n=1}^N]$, i.e., $E[E[\underline{D}|\{w_n\}_{n=1}^N]]$, by averaging over all possible $\boldsymbol{A}$. The result is a more general lower bound:

Theorem 3.4

Given system parameters K, N, and P_e, the overall APDD performance of the perfect LNC technique is $E[\underline{D}]$, where

$$E[\underline{D}] = \frac{1}{2(1-P_e)}\left(1 + E\left[\frac{\sum_{n=1}^N w_n^2}{\sum_{n=1}^N w_n}\right]\right), \quad \{w_n\}_{n=1}^N \sim B(K, P_e) \tag{3.7}$$
$$\approx \frac{KP_e - P_e + 2}{2 - 2P_e}, \quad \text{when } N \text{ is sufficiently large} \tag{3.8}$$

This is a lower bound of the minimum expected APDD of LNC techniques.

Here (3.7) is obtained by letting $\{P_{e,n}\} = P_e$ in (3.6) and taking the expectation. Each w_n follows a binomial distribution of $B(K, Pe)$ because the packet erasures follow a Bernoulli distribution defined by P_e. We will prove in Appendix E that $E\left[\frac{\sum_{n=1}^N w_n^2}{\sum_{n=1}^N w_n}\right] \approx KP_e - P_e + 1$ when N is sufficiently large, which will prove the estimation in (3.8).

According to (3.8), our proposed lower bound is independent of the number of receivers. This observation is confirmed by our simulations. The results for $K = 15$ data packets, $N \in [5, 100]$ receivers, and packet erasure probabilities of $\{P_{e,n}\}_{n=1}^N = 0.2$ are shown in Fig. 3.3, which also affirm the accuracy of the proposed estimation.

We also note that, although the above theorem is derived by assuming a homogeneous packet erasure probability of $\{P_{e,n}\}_1^n = P_e$, the result can be easily extended to heterogeneous cases: when packet erasure probabilities are different among receivers, we can apply the smallest (resp. largest) probabilities to derive

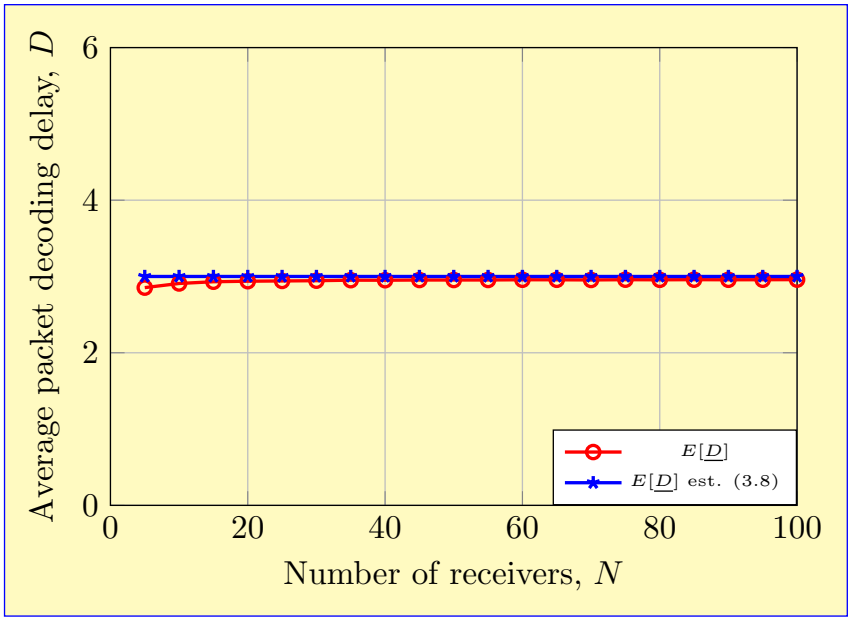


Figure 3.3: **The accuracy of the proposed estimation of the lower bound of APDD.**

a lower (resp. upper) bound of $E[\underline{D}]$. The results will also be independent of the number of receivers.

3.3 Conclusion

In conclusion, in this chapter we have shown that 1) it is an open problem to determine the achievability of the minimum block completion time (BCT) U_{min} for a given $\mathbb{F}_q$; and 2) it is NP-hard to achieve the minimum APDD $D_{\min}$. We have also derived two lower bounds of the minimum expected APDD of LNC. Motivated by these results, in the rest of this thesis we will design and study LNC techniques that can efficiently reduce APDD with graceful or no degradation in BCT.

Chapter 4

Instantly Decodable Network Coding

In the last chapter, we proposed the concept of perfect LNC solution, which allows *every* receiver to instantly decode a wanted data packet in each coded transmission, and thus minimizes the APDD. However, it is NP-complete to even determine whether a perfect LNC solution exists or not. Therefore, it is desirable if at least *a subset of* receivers can instantly decode a wanted data packet in each coded transmission. This feature is achieved by a class of LNC techniques called instantly decodable network coding (IDNC). In this chapter, we will provide a comprehensive study on IDNC, such as its optimal performance, optimal/heuristic algorithms, and packet transmission schemes. Our main focus will be a sub-class called strict IDNC (S-IDNC). We will also compare S-IDNC with a more general class, namely, generalized IDNC (G-IDNC).

4.1 Introduction

IDNC techniques enable instant packet decodings by sending carefully generated coded packets, each being the binary sum of a selected subset of data packets. Since both coding and decoding are operated under $\mathbb{F}_2$, IDNC techniques are also computationally friendly. Hence, IDNC techniques are very attractive in delay-sensitive applications with limited computational resources , such as video streaming to mobile receivers [26, 27].

IDNC coded packets may have three different types of decodability at different receivers. Consider the state-feedback-matrix (SFM) $\boldsymbol{A}$ in Fig. 4.1. There are 3 data packets, $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$, and 3 receivers, $\{r_1, r_2, r_3\}$. r_1 wants all the 3 data

packets. r_2 has received $\mathbf{p}_1$ and wants $\mathbf{p}_2$ and $\mathbf{p}_3$. r_3 has received $\mathbf{p}_1$ and $\mathbf{p}_2$ and wants $\mathbf{p}_3$. Denote by $\bigoplus$ the binary XOR operation. An IDNC coded packet of $\mathbf{p}_1 \oplus \mathbf{p}_2$: 1) is non-instantly decodable to r_1, as it contains 2 wanted data packets of r_1; 2) is instantly decodable to r_2, as it contains 1 wanted data packet of r_2; 3) is non-innovative to r_3, as it contains no wanted data packets of r_3.

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$
r_1	1	1	1
r_2	0	1	1
r_3	0	0	1

Figure 4.1: An instance of SFM

The restriction on non-instantly decodable packets separates IDNC techniques into two variations. The first one, called S-IDNC [40, 43, 62], prohibits the transmissions of non-instantly decodable packets to any receiver. Effectively, each coded packet can include at most one wanted data packet of every receiver. The second one, called G-IDNC, removes this restriction for more coding opportunities.

Therefore, S-IDNC can be thought of as a sub-class of G-IDNC, in the sense that every valid S-IDNC coded packet is also a valid G-IDNC coded packet. Although G-IDNC has been extensively studied under various wireless broadcast settings, including basic ones [12, 58, 58, 63–68] and those with limited/lossy feedback [23, 24] or with hard deadline [41], most developed algorithms are heuristics. The optimal G-IDNC in terms of throughput and APDD are still unknown or intractable due to prohibitively large computational complexity. Hence in this chapter, we aim to understand the performance limits and optimal implementations of a sub-class of G-IDNC, namely, S-IDNC. This will provide new insights into the more general G-IDNC class.

So far, studies on performance limits and implementations of S-IDNC have been quite limited in both breath and depth. S-IDNC was graphically modeled in [43], which also proved that the minimum clique partition solution of the associated graph can be a S-IDNC solution that minimizes the block completion time (BCT). However, this solution does not take into account the APDD performance and the robustness of coded transmissions to erasures. S-IDNC has shown to be asymptotically throughput optimal when there are up to three receivers or when the number of data packets approaches infinity [26], but the general relation be-

tween the throughput of S-IDNC and system parameters, such as the number of data packets and receivers and the values of packet erasure probabilities, has not been characterized before. In addition and to the best of our knowledge, the minimum APDD of S-IDNC is still unknown. Moreover, although G-IDNC schemes dealing with limited/lossy feedback has been developed [23, 24], there have not been S-IDNC transmission schemes that can work with intermittent receiver feedback, but only some studies for certain G-IDNC with heuristic results. Another unaddressed problem is a systematic performance comparison between S-IDNC and G-IDNC.

In this chapter, we study the above problems and provide the following contributions:

1. We characterize the throughput performance limits of S-IDNC. Specifically, we derive a closed-form expression for its expected minimum BCT in terms of the number of packets and receivers and their erasure probabilities;

2. We prove that it is NP-hard to minimize the APDD of S-IDNC. We prove that S-IDNC is not an APDD-approximation technique in general, but is the perfect LNC technique that minimizes both BCT and APDD when there are at most three receivers;

3. We introduce the concept of packet multiplicity, which measures the robustness of data packets against packet erasures. We develop optimal/heuristic algorithms that find S-IDNC solutions with the optimal/very good BCT and with high packet multiplicities. These solutions perform as well or better than the minimum clique partition solution found in [43]. We also design S-IDNC transmission schemes under full and intermittent receiver feedback.

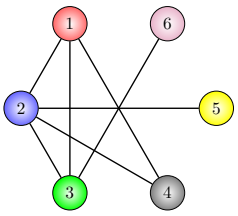
4. We also provide new results on the relation between S-IDNC and G-IDNC. For example, we prove the equivalence between the chromatic number of S- and G-IDNC graphs.

4.2 Modeling IDNC

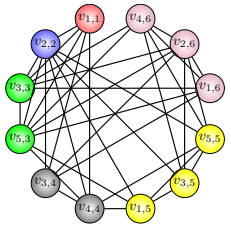
The broadcast starts with the systematic transmission phase, during which all data packets in the packet block $\mathcal{P}$ are broadcast uncoded once. Then IDNC techniques are applied in the coded transmission phase. Each IDNC coded packet is the binary sum of the data packets in a selected coding set $\mathcal{M} \subseteq \mathcal{P}$, i.e.,

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	$\mathbf{p}_4$	$\mathbf{p}_5$	$\mathbf{p}_6$
r_1	1	0	0	0	1	1
r_2	0	1	0	0	0	1
r_3	0	0	1	1	1	0
r_4	0	0	0	1	0	1
r_5	0	0	1	0	1	0

(a) SFM $\boldsymbol{A}$



(b) S-Graph $\mathcal{G}_s$



(c) G-Graph $\mathcal{G}_g$

Figure 4.2: An example of SFM and its S- and G-IDNC graphs. Vertices representing different data packets are colored differently.

$\boldsymbol{X} = \bigoplus_{\mathbf{p}_k \in \mathcal{M}} \mathbf{p}_k$. $\boldsymbol{X}$ has three possible types of decodability at the receivers $\{r_n\}_{n=1}^{N}$:

Definition 4.1

1. An IDNC coded packet $\boldsymbol{X}$ is instantly decodable to receiver r_n if its $\mathcal{M}$ contains exactly one data packet from the Wants set $\boldsymbol{\omega}_n$ of r_n, i.e., if $|\mathcal{M} \cap \boldsymbol{\omega}_n| = 1$.
2. An IDNC coded packet $\boldsymbol{X}$ is non-instantly decodable to receiver r_n if its $\mathcal{M}$ contains two or more data packets from the Wants set $\boldsymbol{\omega}_n$ of r_n, , i.e., if $|\mathcal{M} \cap \boldsymbol{\omega}_n| > 1$.
3. An IDNC coded packet $\boldsymbol{X}$ is non-innovative for receiver r_n if its $\mathcal{M}$ contains no data packets from the Wants set $\boldsymbol{\omega}_n$ of r_n, , i.e., if $|\mathcal{M} \cap \boldsymbol{\omega}_n| = 0$. Otherwise, it is innovative.

S-IDNC prohibits the transmission of any non-instantly decodable coded packets to any receiver. In other words, every coded packet is either instantly decodable or non-innovative to each receiver. This restriction implies that any two data packets wanted by the same receiver cannot be coded together. We thus have the concept of conflicting and non-conflicting data packets:

Definition 4.2

Two data packets $\mathbf{p}_i$ and $\mathbf{p}_j$ conflict if at least one receiver wants both of them, i.e., if $\exists n : \{\mathbf{p}_i, \mathbf{p}_j\} \subseteq \boldsymbol{\omega}_n$. Otherwise they do no conflict.

A S-IDNC coding set is thus a set of pairwise non-conflicting data packets. The conflicting state between all data packets can be represented by an undirected graph $\mathcal{G}_s(\mathcal{V}, \mathcal{E})$. Each vertex $\boldsymbol{v}_i \in \mathcal{V}$ represents a data packet $\mathbf{p}_i$. Two

vertices $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are connected by an edge $\boldsymbol{e}_{i,j} \in \mathcal{E}$ if $\mathbf{p}_i$ and $\mathbf{p}_j$ do not conflict. Thus, every complete subgraph of $\mathcal{G}_s$, a.k.a., a clique, represents a S-IDNC coding set. In the rest of the chapter, we will use the terms "coding set" and "clique" interchangeably, and denote both by $\mathcal{M}$.

The main limitation of S-IDNC is that a coded packet which is instantly decodable to a large subset of receivers may be prohibited because it is non-instantly decodable to a small subset of receivers. In the second type of IDNC, called generalized IDNC (G-IDNC), the restriction on non-instantly decodable packets is removed for more coding opportunities.[1]

G-IDNC can also be graphically modeled [63]. The difference is that, in the G-IDNC graph $\mathcal{G}_g(\mathcal{V}, \mathcal{E})$, a data packet $\mathbf{p}_k$ wanted by different receivers are individually represented by different vertices $\boldsymbol{v}_{n,k}$, for all $\boldsymbol{A}(n,k) = 1$. Consequently, the number of vertices in $\mathcal{G}_g$ is equal to the number of "1"s in $\boldsymbol{A}$. Two vertices $\boldsymbol{v}_{m,i}$ and $\boldsymbol{v}_{n,j}$ are connected by an edge if: 1) $i = j$, or 2) if $\mathbf{p}_i \notin \boldsymbol{\omega}_n$ and $\mathbf{p}_j \notin \boldsymbol{\omega}_m$. In the first case, $\mathbf{p}_i = \mathbf{p}_j$, and thus by sending $\mathbf{p}_i$ both r_m and r_n can decode their identical wanted data packet. In the second case, by sending $\mathbf{p}_i \oplus \mathbf{p}_j$, r_m and r_n can decode $\mathbf{p}_i$ and $\mathbf{p}_j$, respectively, because they already have $\mathbf{p}_j$ and $\mathbf{p}_i$, respectively. Similar to S-IDNC, every clique of $\mathcal{G}_g$ represents a G-IDNC coding set.

We note that a S-IDNC coded packet is always a G-IDNC coded packet, but the reverse is not necessarily true. Below is such an example:

Example 4.1

Consider the SFM and its S- and G-IDNC graphs in Fig. 4.2. The G-IDNC graph indicates that $(\boldsymbol{v}_{1,1}, \boldsymbol{v}_{5,3}, \boldsymbol{v}_{4,4})$ is a clique. The corresponding G-IDNC coding set is $(\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4)$, and thus $\boldsymbol{X}_g = \mathbf{p}_1 \oplus \mathbf{p}_3 \oplus \mathbf{p}_4$ is a G-IDNC coded packet. $\boldsymbol{X}_g$ is instantly decodable to r_1, r_4, r_5 because they only want one data packet from $\boldsymbol{X}_g$. $\boldsymbol{X}_g$ is non-instantly decodable to r_3 because r_3 wants both $\mathbf{p}_3$ and $\mathbf{p}_4$. $\boldsymbol{X}_g$ is non-innovative for r_2. Due to the existence of r_3, $\boldsymbol{X}_g$ is not a S-IDNC coded packet. The S-IDNC graph indicates that $(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3)$ is a clique. The corresponding coding set is $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$, and thus $\boldsymbol{X}_s = \mathbf{p}_1 \oplus \mathbf{p}_2 \oplus \mathbf{p}_3$ is a S-IDNC coded packet, which can be verified to also correspond to clique $(\boldsymbol{v}_{1,1}, \boldsymbol{v}_{2,2}, \boldsymbol{v}_{3,3}, \boldsymbol{v}_{5,3})$ in the G-IDNC graph.

[1] In traditional G-IDNC, non-instantly decodable packets will be discarded by the receivers. Storing such packets may further improve throughput with extra computational cost. This problem is beyond the scope of this thesis. Please refer to [47] for a fairly recent treatment.

We also have the notion of IDNC solution:

Definition 4.3

A set of U IDNC coding sets $\{\mathcal{M}_u\}_{u=1}^{U}$ is called an IDNC solution if, upon the reception of the corresponding IDNC coded packets $\{\boldsymbol{X}_u\}_{u=1}^{U}$, every receiver can decode all its wanted data packets. A S-IDNC solution is denoted by $\mathcal{S}_s$. A G-IDNC solution is denoted by $\mathcal{S}_g$.

We further denote by $\mathbb{S}_s$ (resp. $\mathbb{S}_g$) the set of all S-IDNC (resp. G-IDNC) solutions of a given SFM. It holds that $\mathbb{S}_s \subseteq \mathbb{S}_g$.

Example 4.2

For the SFM in Fig. 4.2, by partitioning $\mathcal{G}_s$ into three disjoint cliques, we can obtain, among others, an S-IDNC solution of $\mathcal{S}_s = \{(\mathbf{p}_1, \mathbf{p}_4), (\mathbf{p}_2, \mathbf{p}_5), (\mathbf{p}_3, \mathbf{p}_6)\}$. This solution is also a G-IDNC solution obtained by partitioning the G-IDNC graph $\mathcal{G}_g$ into $\{(\boldsymbol{v}_{1,1}, \boldsymbol{v}_{3,4}, \boldsymbol{v}_{4,4}), (\boldsymbol{v}_{2,2}, \boldsymbol{v}_{1,5}, \boldsymbol{v}_{3,5}, \boldsymbol{v}_{5,5}), (\boldsymbol{v}_{3,3}, \boldsymbol{v}_{5,3}, \boldsymbol{v}_{1,6}, \boldsymbol{v}_{2,6}, \boldsymbol{v}_{4,6})\}$

We denote by U_s and D_s the minimum BCT and the minimum APDD across all S-IDNC solutions when there are no packet erasures. The definitions of U_g and D_g for G-IDNC follow similarly. Although the actual BCT U and APDD D of the coded transmission phase vary according to the IDNC solutions, transmission schemes, and erasure patterns, it always holds that $U \geqslant U_s$ and $D \geqslant D_s$ if S-IDNC is applied. Therefore, U_s and D_s reflect the performance limits of S-IDNC. Hence, we will first study these limits in the next section, and then design S-IDNC transmission schemes and coding algorithms in Sections 4.4 and 4.5, respectively.

4.3 Performance Limits and Properties

In this section, we study performance limits and properties of S-IDNC and compare it with G-IDNC.

4.3.1 Minimum Block Completion Time

We first study the throughput limit of S-IDNC, measured by the minimum BCT U_s. It has been proved that U_s is equal to the size of the minimum clique par-

tition solution[2] of $\mathcal{G}_s$ [43]. The S-IDNC solution corresponding to the minimum clique partition solution is denoted by $\mathcal{S}_c = \{\mathcal{M}_u\}_{u=1}^{U_s}$, where $\{\mathcal{M}_u\}_{u=1}^{U_s}$ are disjoint IDNC coding sets that together cover all data packets in $\mathcal{P}$. The above equivalence holds because of the following property:

Property 4.1

Removing any vertex from the S-IDNC graph will not change the connectivity of the remaining vertices.

This property holds because vertices in $\mathcal{G}_s$ represent different data packets. Due to this property, to remove all vertices from $\mathcal{G}_s$ (i.e., to complete the broadcast), at least $|\mathcal{S}_c|$ cliques must be removed, which yields $U_s = |\mathcal{S}_c|$.

According to graph theory, $|\mathcal{S}_c|$ is equal to the chromatic number[3] $\chi(\overline{\mathcal{G}}_s)$ of the complementary graph $\overline{\mathcal{G}}_s$, which has the same vertex set as $\mathcal{G}_s$, but has opposite vertex connectivity. We thus have $U_s = \chi(\overline{\mathcal{G}}_s)$.

However, it is NP-hard to find the chromatic number of a given graph [69], and thus it is NP-hard to find U_s. Under this situation, bounds and approximations on the chromatic number of a given graph have been well studied in the graph theory literature [69–71]. They provide some insights into the U_s of a given SFM. Thus, we will not further investigate the U_s of a given SFM. Instead, we are interested in the probabilistic characterization of U_s, because $\mathcal{G}_s$ is the consequence of random packet erasures in the systematic transmission phase. Specifically, we address the following problem:

Problem 4.1

What is the relation between U_s and system parameters, including the number K of data packets, the number N of receivers, as well as the packet erasure probabilities $\{P_{e,n}\}_{n=1}^{N}$?

Under the assumption that packet erasures are independently and Bernoulli distributed at each receiver r_n with an erasure probability of $P_{e,n}$, a similar question has already been introduced and answered for the RLNC technique. It has been shown in [25, 72, 73] that the BCT of RLNC scales as $\mathcal{O}(\ln(N))$ when K is a constant. Consequently, the throughput of RLNC vanishes with increasing

[2]The minimum clique partition solution of a graph $\mathcal{G}$ is the minimum set of disjoint cliques of $\mathcal{G}$ that together cover all the vertices.

[3]The chromatic number of a graph $\mathcal{G}$ is the minimum number of colors to color the vertices so that any two connected vertices have different colors.

number of receivers N. To prevent zero throughput, it has been proved in [74] that K should scale faster than $\ln(N)$.

Since the throughput of RLNC is optimal, it cannot be exceeded by the throughput of S-IDNC. Hence, we can infer that the throughput of S-IDNC should also follow a vanishing behavior with increasing N. However, its rate and specific dependence on system parameters have not been characterized in the literature. In this subsection, we answer this question through the following theorem:

Theorem 4.1

The mean of the minimum BCT $E[U_s]$ over all possible SFMs is a function of the block size K, the number of receivers N, and packet erasure probability $\{P_{e,n}\}_{n=1}^{N}$:

$$E[U_s] = -K\left(\frac{1}{2} + o(1)\right)\sum_{n=1}^{N}\log_K(1 - P_{e,n}^2) \tag{4.1}$$

where $o(1)$ is a small term that approaches zero with increasing K.

Proof. Our approach is to model the complementary S-IDNC graph $\overline{\mathcal{G}}_s$ as a random graph with i.i.d. edge generating probability. Recall that two vertices in $\overline{\mathcal{G}}_s$ are connected if the two data packets conflict, i.e., if at least one receiver has missed both packets. Therefore, the generating probability of every edge, denoted by P_c, is calculated as:

$$P_c = 1 - \prod_{n=1}^{N}(1 - P_{e,n}^2), \tag{4.2}$$

Then, the key is to prove that different edges are generated independently. We first consider the independence between two adjacent edges. Without loss of generality let us consider the generation of $\boldsymbol{e}_{1,2}$ and $\boldsymbol{e}_{1,3}$, two edges that are adjacent via $\boldsymbol{v}_1$, and are incident to $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$, respectively. We denote by $P(\boldsymbol{e}_{1,2})$ the probability that $\boldsymbol{e}_{1,2}$ is generated. It holds that $P(\boldsymbol{e}_{1,2}) = P(\boldsymbol{e}_{1,3}) = P_c$ in (4.2). We further denote by $P(\boldsymbol{e}_{1,2}|\boldsymbol{v}_1)$ the probability that $\boldsymbol{e}_{1,2}$ is generated conditioned on that $\boldsymbol{v}_1$ is generated. We then argue the following relations:

1. $P(\boldsymbol{e}_{1,2}, \boldsymbol{v}_1) = P(\boldsymbol{e}_{1,2})$, because the generating of $\boldsymbol{e}_{1,2}$ already indicates that $\boldsymbol{v}_1$ is generated. Similarly, we also have $P(\boldsymbol{e}_{1,3}, \boldsymbol{v}_1) = P(\boldsymbol{e}_{1,3})$;

2. $P(\boldsymbol{v}_1|\boldsymbol{e}_{1,2}, \boldsymbol{e}_{1,3}) = 1$ because of the same reason as above;

3. $P(\boldsymbol{e}_1, \boldsymbol{e}_2|\boldsymbol{v}_1) = P(\boldsymbol{e}_1|\boldsymbol{v}_1) \cdot P(\boldsymbol{e}_2|\boldsymbol{v}_1)$, because if $\boldsymbol{v}_1$ is already generated, the

generating of $\boldsymbol{e}_1$ (resp. $\boldsymbol{e}_2$) only depends on whether $\mathbf{p}_2$ (resp. $\mathbf{p}_3$) is wanted by some of the receivers who want $\mathbf{p}_1$. Since wanting $\mathbf{p}_2$ and $\mathbf{p}_3$ are independent events for every receiver, the generating of $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ is independent conditioned on that $\boldsymbol{v}_1$ is generated;

4. $P(\boldsymbol{v}_1)^2 \approx P(\boldsymbol{v}_1) \approx 1$, and the accuracy increases quickly with increasing number of receivers N. This is because $P(\boldsymbol{v}_1)$ is the probability that at least one receiver has missed $\mathbf{p}_1$ in the systematic transmission phase. It has a value of $1 - \prod_{n=1}^{N}(1 - P_{e,n})$, which quickly approaches to 1 with increasing N.

Then, to prove that $\boldsymbol{e}_{1,2}$ and $\boldsymbol{e}_{1,3}$ are generated independently, we only need to show that $P(\boldsymbol{e}_{1,2}, \boldsymbol{e}_{1,3}) = P(\boldsymbol{e}_{1,2}) \cdot P(\boldsymbol{e}_{1,3})$:

$$\begin{aligned} P(\boldsymbol{e}_{1,2}, \boldsymbol{e}_{1,3}) &= \frac{P(\boldsymbol{e}_{1,2}, \boldsymbol{e}_{1,3}|\boldsymbol{v}_1) \cdot P(\boldsymbol{v}_1)}{P(\boldsymbol{v}_1|\boldsymbol{e}_{1,2}, \boldsymbol{e}_{1,3})} \qquad (4.3) \\ &= P(\boldsymbol{e}_{1,2}|\boldsymbol{v}_1) \cdot P(\boldsymbol{e}_{1,3}|\boldsymbol{v}_1) \cdot P(\boldsymbol{v}_1) \\ &\approx P(\boldsymbol{e}_{1,2}|\boldsymbol{v}_1) \cdot P(\boldsymbol{e}_{1,3}|\boldsymbol{v}_1) \cdot P(\boldsymbol{v}_1)^2 \\ &= P(\boldsymbol{e}_{1,2}, \boldsymbol{v}_1) \cdot P(\boldsymbol{e}_{1,3}, \boldsymbol{v}_1) \\ &= P(\boldsymbol{e}_{1,2}) \cdot P(\boldsymbol{e}_{1,3}), \end{aligned}$$

where (4.3) follows Bayes' rule. Hence, the generation of $\boldsymbol{e}_{1,2}$ and $\boldsymbol{e}_{1,3}$ are asymptotically independent of each other.

On the the other hand, it is intuitive that two disjoint edges in $\overline{\mathcal{G}}_s$ are generated independently. Therefore, we can assume that all edges in $\overline{\mathcal{G}}_s$ are generated independently.

Consequently, $\overline{\mathcal{G}}_s$ can be modeled as an Erdõs-Rényi random graph [75], which has K vertices and i.i.d. edge generating probability of P_c. Fig. 4.3 compares the mean number of edges (with a value of $K(K-1)/2 \cdot P_c$) of our proposed random graph model and the simulated average number of edges in $\overline{\mathcal{G}}_s$. Our model shows virtually no deviation under all considered values of N and K.

From graph theory, given K and P_c, almost every random graph $\overline{\mathcal{G}}_s$ has a chromatic number of [76]:

$$\chi(\overline{\mathcal{G}}_s) = \frac{K}{\log K}\left(\frac{1}{2} + o(1)\right)\log\frac{1}{1-P_c}. \qquad (4.4)$$

Since $U_s = \chi(\overline{\mathcal{G}}_s)$, the above value is the mean of U_s. By substituting (4.2) into (4.4) we obtain (4.1). □

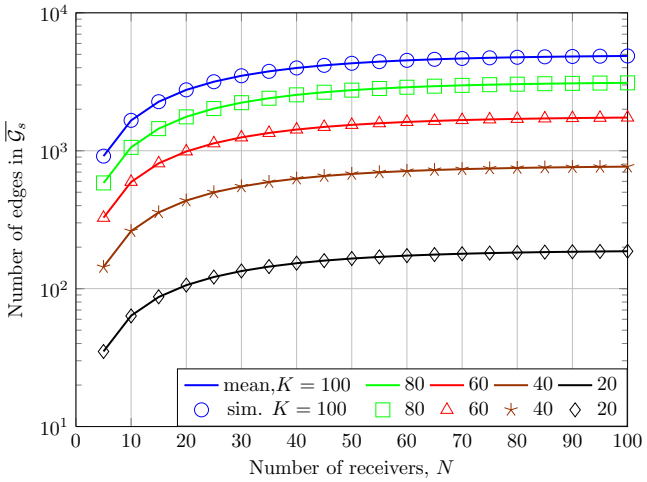


Figure 4.3: The mean and simulated number of edges in $\overline{\mathcal{G}}_s$ when $\{P_{e,n}\}_{n=1}^{N} = 0.2$ and $K \in [20, 100]$.

Theorem 4.1 has the following important corollary:

Corollary 4.1

The mean of the minimum BCT $E[U_s]$ of S-IDNC increases almost linearly with the number of receivers when all receivers experience similar packet erasure probabilities.

This corollary can be proved by letting $\{P_{e,n}\}_{n=1}^{N} = P_e$, which will transform (4.1) into a linear function of N. We can thus conclude that the throughput of S-IDNC degrades linearly with increasing number of receivers. Such degradation on throughput is common among linear network coding techniques that primarily aim to reduce packet decoding delay [20].

4.3.2 Minimum APDD

Unlike U_s, to the best of our knowledge there is no existing hardness result on finding the minimum APDD D_s of S-IDNC. In this subsection, we address it through the following theorem and then propose an upper bound on D_s.

Theorem 4.2

It is NP-hard to find the minimum APDD D_s of S-IDNC.

This theorem holds because the perfect LNC solution $\mathcal{S}_p$ we have proposed in the last chapter is indeed a perfect S-IDNC solution, which offers the minimum

possible APDD D_0 over all LNC techniques. If it is not NP-hard to find D_s, then by comparing D_s with D_0, we can easily determine the existence of $\mathcal{S}_p$, which contradicts the fact that it is NP-complete to determine the existence of $\mathcal{S}_p$.

In addition, we can prove the following theorem:

Theorem 4.3

S-IDNC is not an APDD-approximation technique in general.

Proof. We prove this theorem by providing a counter example.

Consider an instance that has a set of K data packets, and for every pair of two data packets there is a receiver that wants both of them. This implies a conflict between every pair of data packets. Hence, all K data packets must be broadcast uncoded, which requires a BCT of K. The resulted APDD is thus $D_s = (K+1)/2$.

On the other hand, since every receiver only wants two data packets, by applying a throughput-optimal LNC technique such as RLNC, the BCT will be 2 and, thus, the APDD will also be 2. Hence, the ratio between the APDD of S-IDNC and the minimum is at least $(K+1)/4$, which increases with K and, thus, is unbounded in this instance. This contradicts with the requirement that an APDD-approximation technique must provide a maximum ratio of β, where β is a constant. □

We note that the term "in general" in the above theorem means that the statement is only true if no additional conditions are applied. Under certain parameter settings, however, S-IDNC is able to achieve the minimum APDD of LNC techniques:

Theorem 4.4

When there are at most three receivers, S-IDNC technique is a perfect LNC technique that minimizes both BCT and APDD.

Proof. It has been proved in [26] that S-IDNC is throughput-optimal when there are at most three receivers. In this case, every S-IDNC packet is innovative to every receiver. Since every innovative S-IDNC packet is instantly decodable, S-IDNC allows every receiver to decode a wanted data packet in every transmission. According to Definition 3.4, S-IDNC is a perfect LNC technique in this case. □

The proof of Theorem 4.3 indicates that S-IDNC is unable to approximate the minimum APDD mainly because of its large BCT. This motivates us to design

S-IDNC implementations that minimize its BCT. Before we move on, we would like to compare the performance limits of S-IDNC with G-IDNC.

4.3.3 S-IDNC vs. G-IDNC

In this subsection, we address the following problem:

Problem 4.2

How does S-IDNC compare with G-IDNC?

We first note that the NP-hardness of finding D_s also holds for D_g. This is because the perfect S-IDNC solution $\mathcal{S}_p$ is also the best possible G-IDNC solution. For the throughput, we first present an equivalence between S- and G-IDNC graphs (proved in Appendix C):

Theorem 4.5:

The minimum clique partition solutions of S-IDNC and G-IDNC graphs have the same size. In other words, $\chi(\overline{\mathcal{G}}_s) = \chi(\overline{\mathcal{G}}_g)$.

This theorem, together with Corollary 4.1, indicates that $\chi(\overline{\mathcal{G}}_g)$ also increases almost linearly with N when all receivers experience similar erasure probabilities. However, the above theorem does not imply $U_s = U_g$. This is because G-IDNC does not have Property 4.1. Explicitly, by removing a vertex from $\mathcal{G}_g$, more edges and larger cliques may be generated, and thus minimum BCT U_g can be smaller than $\chi(\overline{\mathcal{G}}_g)$ of the original G-IDNC graph $\mathcal{G}_g$ [64]. One such example is demonstrated in the Appendix D. We thus have $U_g \leqslant U_s$. We note, however, that a systematic way of finding U_g other than brute-force search remains open.

4.4 S-IDNC Transmission Schemes

Given an SFM after the systematic transmission phase, the simplest transmission scheme for the coded transmission phase is to first find a S-IDNC solution $\mathcal{S}_s = \{\mathcal{M}_u\}_{u=1}^{U}$, and then transmit the corresponding coded packets $\{\boldsymbol{X}_u\}_{u=1}^{U}$ repeatedly until all receivers have decoded all wanted data packets. However, this scheme is inefficient, because it ignores the online coding opportunities emerging during the coded transmission phase. In order to access such opportunities and make online coding decisions, the sender must regularly collect feedback from the receivers about their packet reception state. We consider transmission schemes with two different types of feedback frequency, namely:

1. fully-online feedback: feedback is collected after every coded transmission. However, this could be costly in wireless communications;

2. semi-online feedback: feedback is only collected after transmitting a complete S-IDNC solution;

To be able to design such schemes, two questions need to be answered:

Problem 4.3

1. What is the optimization objective for performance improvement?
2. which coded packets should the sender send to achieve it?

Before addressing these questions, we first highlight some challenges:

Remark 4.1

Under random packet erasures, a reasonable measure of throughput is the mean BCT $E[U]$ of the coded transmission phase. However, it is intractable to minimize $E[U]$. To see this, let us consider the stochastic shortest path (SSP) method [63]. In SSP method, the state space comprises the current SFM $\boldsymbol{A}$, all its successors, and an absorbing state of all-zero SFM $\boldsymbol{A}_0$. Thus, the state space has a prohibitively large size with a value of 2^T, where T is the number of "1"s in $\boldsymbol{A}$. Moreover, the action space for each state is the set of all its cliques/coding sets, which is NP-hard to find [77]. Then, $E[U]$ is recursively minimized by examining all the states and the associated actions. Such examination is necessary, because the packet erasures can take any pattern and are not predictable. Therefore, $E[U]$ is intractable to minimize. To overcome this difficulty, we will reduce the action space by focusing on the set of actions that belong to the shortest possible path between $\boldsymbol{A}$ and $\boldsymbol{A}_0$. In other words, we will focus on the set of actions that are able to complete the broadcast using U_s transmissions with a non-zero probability. We also note that this reduction is not possible for G-IDNC because there has not been a systematic way of finding U_g.

Remark 4.2

It is intractable to minimize the mean APDD $E[D]$ of the coded transmission phase due to the NP-hardness of finding D_s, because otherwise under the setting where $P_e = 0$, the minimum $E[D]$ is equal to D_s. To overcome this difficulty, we will give higher priority to the minimization of BCT. In other words, we first minimize BCT, and then among the resultant coding decisions, we choose the one that minimizes APDD. Our prioritization reflects the motivation of using linear network coding, that is, to achieve better throughput performance. It also provides bounded APDD as we have discussed after Theorem 4.4. Our simulations will confirm that D generally decreases with decreasing U.

4.4.1 Fully-online Transmission Scheme

In this scheme, the action space of a given $\boldsymbol{A}$ comprises all the S-IDNC coding sets of $\boldsymbol{A}$. The cost of each action is one, for it consumes one transmission. BCT U is thus equal to the number of transitions (a.k.a. path length or distance) between $\boldsymbol{A}$ and $\boldsymbol{A}_0$. It is obvious that the shortest distance between $\boldsymbol{A}$ and $\boldsymbol{A}_0$ is U_s.

According to Remark 4.1, we propose to choose an action/coding set that belongs to the shortest path from $\boldsymbol{A}$ to $\boldsymbol{A}_0$. This choice guarantees that, upon the reception of the coded packet at all interested receivers, the shortest distance between the updated state $\boldsymbol{A}'$ and $\boldsymbol{A}_0$ is minimized with a value of $U_s - 1$. To this end, the coding set must belong to a minimum clique partition solution $\mathcal{S}_c$. Otherwise, the shortest distance between $\boldsymbol{A}'$ and $\boldsymbol{A}_0$ will still be U_s.

We then reduce APDD by forcing the coding set to be maximal (and thus serving the maximal number of receivers). However, cliques in a minimum clique partition solution are not necessarily maximal. Hence, we further require the coding set to belong to a set of U_s maximal cliques that together cover all the data packets. This set is also a S-IDNC solution and is denoted by $\mathcal{S}_m$.

In conclusion, we propose the following coding set $\mathcal{M}_f$ for fully-online transmission scheme:

Definition 4.4

Given an SFM instance, the preferred fully-online coding set $\mathcal{M}_f$ is the most wanted coding set in $\mathcal{S}_m$, where $\mathcal{S}_m$ is a S-IDNC solution that contains U_s maximal cliques.

4.4.2 Semi-online Transmission Scheme

In this scheme, the action space of a given $\boldsymbol{A}$ becomes the set of all S-IDNC solutions $\mathbb{S}_s$ of $\boldsymbol{A}$, and the cost of each action is the solution size $|\mathcal{S}_s|$, which is equal to the length of a semi-online transmission round. The total cost is thus equal to BCT U.

According to Remark 4.1, we propose to minimize the expected cost of the shortest path between $\boldsymbol{A}$ and $\boldsymbol{A}_0$. The shortest path includes only one transition, representing the event that every coding set of the chosen solution $\mathcal{S}_s$ is received by all the interested receivers after only one semi-online round. Denote the probability of this event by P_s. Then the expected cost is $|\mathcal{S}_s|/P_s$, where P_s is calculated as:

$$P_s = \prod_{k=1}^{K} \prod_{r_n \in \tau_k} (1 - P_{e,n}^{d_k}) \tag{4.5}$$

Here d_k is called the packet multiplicity and is defined below.

Definition 4.5

The multiplicity d_k of data packet $\mathbf{p}_k$ is the number of coding sets in $\mathcal{S}_s$ that comprise $\mathbf{p}_k$.

We note that the minimum clique partition solution $\mathcal{S}_c$ is not a preferred semi-online S-IDNC solution. Although $\mathcal{S}_c$ offers the smallest solution size ($|\mathcal{S}_c| = U_s$), it does not maximize P_s because every data packet has a multiplicity of only one due to disjoint cliques in $\mathcal{S}_c$. In contrast, the $\mathcal{S}_m$ we have proposed for the fully-online case has the same size as $\mathcal{S}_c$, but can offer a higher P_s due to possibly overlapping maximal cliques.

We still wish to answer the following question before choosing $\mathcal{S}_m$ as our preferred semi-online S-IDNC solution:

Problem 4.4

Is there a S-IDNC solution that, though large in its size, is high in packet multiplicities, so that P_s is maximized?

An explicit answer to this question is difficult to obtain, because it requires the examination of all the solutions of size greater than U_s. Such search is computationally costly and does not provide any insight into this question. Moreover, a larger solution is unlikely to provide higher packet multiplicities due to the following property of S-IDNC solutions:

Property 4.2

Every coding set in a S-IDNC solution comprises at least one data packet with a multiplicity of one.

This property holds because if every data packet in a coding set has a multiplicity of greater than one, then this coding set can be removed from the solution without affecting the completeness of the solution. Due to the above property, a S-IDNC solution $\mathcal{S}_s$ has at least $|\mathcal{S}_s|$ data packets with a multiplicity of only one. According to (4.5), these unit-multiplicity data packets reduce P_s the most. Hence, S-IDNC solutions with a larger size may have more unit-multiplicity data packets than $\mathcal{S}_m$, and thus are not preferable.

Therefore, we choose $\mathcal{S}_m$ for throughput improvement. Then, by taking into account our secondary optimization objective, i.e., the APDD, we define our preferred semi-online S-IDNC solution as follows:

Definition 4.6

Given an SFM instance, the preferred semi-online S-IDNC solution is $\mathcal{S}_m$, which comprises a set of U_s maximal coding sets. These sets are sorted for transmission in the descending order of their numbers of targeted receivers to minimize the APDD.

A flow-chart of the proposed two transmission schemes is presented in Fig. 4.4. Both the fully- and semi-online IDNC schemes require finding $\mathcal{S}_m$. Since packet multiplicity is not a concern in graph theory, algorithms that find $\mathcal{S}_m$ do not exist in the graph theory literature. Hence, we will design algorithms dedicated to S-IDNC in the next section. Before moving on, we briefly compare S-IDNC and G-IDNC under the above two transmission schemes.

4.4.3 S-IDNC vs. G-IDNC

With fully-online feedback, the sender can update the G-IDNC graph $\mathcal{G}_g$ and add new edges to it after every transmission. The throughput of G-IDNC is

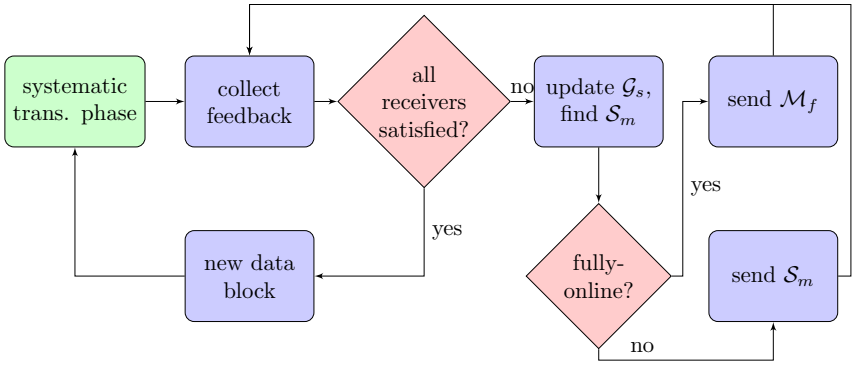


Figure 4.4: **The fully- and semi-online transmission schemes.**

thus at least as good as S-IDNC. But the price is a higher computational load, because G-IDNC graph is much larger tha S-IDNC graph ($\mathcal{O}(NK)$ v.s. $\mathcal{O}(K)$). On the other hand, it has been proved in [24] that, when receiver feedback is not available, the best strategy for the sender is not to update $\mathcal{G}_g$ (compared with certain probabilistic update strategy). Therefore, during a semi-online transmission round, the sender only sends the minimum clique partition solution of $\mathcal{G}_g$, which, according to Theorem 4.5, has the same size as the minimum clique partition solution of S-IDNC.

4.5 S-IDNC Coding Algorithms

The two transmission schemes we proposed in the last section require finding $\mathcal{S}_m$, a S-IDNC solution that contains U_s maximal coding sets. In this section, we develop its optimal and heuristic algorithms.

4.5.1 Optimal S-IDNC Coding Algorithm

Our optimal algorithm finds $\mathcal{S}_m$ in two steps:

1. *Find all the maximal coding sets (maximal cliques):* This problem is NP-hard in graph theory [77], and thus cannot be optimally solved by an algorithm whose computational complexity is a polynomial of the number K of data packets. However, it can be solved by exponential algorithms such as Bron-Kerbosch (B-K) algorithm [77]. We use B-K algorithm to optimally find the group of all maximal cliques and denote the group by $\mathcal{A}$.

2. *Find $\mathcal{S}_m$ from $\mathcal{A}$:* We propose a branching algorithm in Algorithm 4.1. The intuition behind this algorithm is that, if a data packet $\mathbf{p}_k$ belongs to d_k

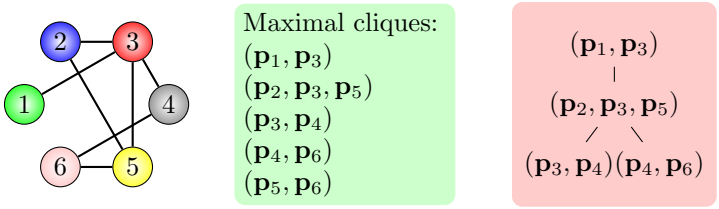


Figure 4.5: An example of the 2-step optimal S-IDNC coding algorithm.

maximal coding sets in $\mathcal{A}$, then one of these d_k maximal coding sets must be included in $\mathcal{S}_m$ for the completeness of $\mathcal{S}_m$. In the extreme case where $d_k = 1$, the sole maximal coding set that contains $\mathbf{p}_k$ must be included in $\mathcal{S}_m$.

Algorithm 4.1 Optimal S-IDNC solution search

1: **input:** the group of all maximal coding sets, $\mathcal{A}$;
2: **initialization:** a set $\mathcal{B}$ of solutions, $\mathcal{B}$ only contains an empty solution $\mathcal{S} = \emptyset$. A counter $u = 1$;
3: **while** no solution in $\mathcal{B}$ contains all data packets, **do**
4: **while** there is a solution in $\mathcal{B}$ with size $u - 1$, **do**
5: Denote this solution by $\mathcal{S} = \{\mathcal{M}_i\}_{i=1}^{u-1}$. Denote the data packets included in $\mathcal{S}$ by $\mathcal{P} = \bigcup_{i=1}^{u-1}\{\mathcal{M}_i\}$ and all data packets not included in $\mathcal{S}$ by $\overline{\mathcal{P}} = \mathcal{P}_K \backslash \mathcal{P}$. Also denote the maximal coding sets not included in $\mathcal{S}$ by $\overline{\mathcal{S}} = \mathcal{A} \setminus \mathcal{S}$;
6: Pick from $\overline{\mathcal{P}}$ the data packet $\mathbf{p}$ that has the smallest multiplicity d under $\overline{\mathcal{S}}$. Denote the d coding sets which contain $\mathbf{p}$ by $\mathcal{M}'_1, \cdots, \mathcal{M}'_d$;
7: Branch $\mathcal{S}$ into d identical solutions, $\mathcal{S}'_1, \cdots, \mathcal{S}'_d$. Then, add $\mathcal{M}'_1, \cdots, \mathcal{M}'_d$ to these solutions, respectively. The size of every new solution is u;
8: **end while**
9: $u = u + 1$;
10: **end while**
11: Output the solutions in $\mathcal{B}$ that contain all data packets.

B-K algorithm and Algorithm 1 constitute our optimal S-IDNC coding algorithm. It is optimal because it exhaustively finds all the valid $\mathcal{S}_m$ from all the maximal coding sets. Among these solutions, we can choose the one that optimizes a secondary criteria, such as the one offering the smallest $D_{\mathcal{S}}$, or the largest P_S, calculated using (4.5). Below is an example of our optimal coding algorithm.

Example 4.3

Consider the instance in Fig. 4.5. In Step-1, we find all the maximal cliques: $\mathcal{A} = \{(\mathbf{p}_1, \mathbf{p}_3), (\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_5), (\mathbf{p}_3, \mathbf{p}_4), (\mathbf{p}_4, \mathbf{p}_6), (\mathbf{p}_5, \mathbf{p}_6)\}$. Then in Step-2:

1. Initially, $\mathcal{S} = \emptyset$, $\overline{\mathcal{S}} = \mathcal{A} \setminus \mathcal{S} = \mathcal{A}$, and the set of data packets not included in $\mathcal{S}$ is $\overline{\mathcal{P}} = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6\}$. Since $\mathbf{p}_1$ is only included in $(\mathbf{p}_1, \mathbf{p}_3)$ and $\mathbf{p}_2$ is only included in $(\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_5)$, these two coding sets must be added to $\mathcal{S}$. Hence, $\mathcal{S} = \{(\mathbf{p}_1, \mathbf{p}_3), (\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_5)\}$ after the first two iterations;

2. In the third iteration, $\overline{\mathcal{S}} = \mathcal{A} \setminus \mathcal{S}$=$\{(\mathbf{p}_3, \mathbf{p}_4), (\mathbf{p}_4, \mathbf{p}_6), (\mathbf{p}_5, \mathbf{p}_6)\}\}$, and the set of data packets not included in $\mathcal{S}$ is $\overline{\mathcal{P}} = \{\mathbf{p}_4, \mathbf{p}_6\}$. Since $\mathbf{p}_4$ has a multiplicity of 2 under $\overline{\mathcal{S}}$ due to $(\mathbf{p}_3, \mathbf{p}_4)$ and $(\mathbf{p}_4, \mathbf{p}_6)$, we branch $\mathcal{S}$ into two successors: $\mathcal{S}_1 = \{(\mathbf{p}_1, \mathbf{p}_3),(\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_5),(\mathbf{p}_4, \mathbf{p}_5)\}$ and $\mathcal{S}_2 = \{(\mathbf{p}_1, \mathbf{p}_3),(\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_5),(\mathbf{p}_4, \mathbf{p}_6)\}$. Since $\mathcal{S}_2$ contains all data packets and there are no other branching opportunities, the algorithm stops and outputs $\mathcal{S}_2$ as $\mathcal{S}_m$.

4.5.2 Hybrid S-IDNC Coding Algorithm

Algorithm 4.1 is memory demanding, because the number of candidate solutions grows exponentially with branching. Thus, we propose a heuristic alternative to it. The idea is to iteratively maximize the number of data packets included in $\mathcal{S}_m$. The algorithm is given in Algorithm 4.2.

Algorithm 4.2 Hybrid S-IDNC solution search

1: **input:** the group of all maximal coding sets, $\mathcal{A}$, packet block $\mathcal{P}_K$;
2: **initialization:** an empty solution $\mathcal{S} = \emptyset$, a counter $u = 1$;
3: **while** $\mathcal{S}$ does not contain all data packets, **do**
4: Add to $\mathcal{S}$ the coding set $\mathcal{M}$ in $\mathcal{A}$ that contains the largest number of data packets in $\mathcal{P}_K$;
5: Remove data packets in $\mathcal{M}$ from $\mathcal{P}_K$;
6: $u = u + 1$;
7: **end while**
8: Output the solution $\mathcal{S}$.

B-K algorithm and Algorithm 4.2 constitute our hybrid S-IDNC coding algorithm. It produces only one S-IDNC solution, with no guarantee on the solution

size. It is still computational expensive due to B-K algorithm, which is exponential. Thus, we develop a polynomial time heuristic S-IDNC coding algorithm next.

4.5.3 Heuristic S-IDNC Coding Algorithm

Algorithm 4.3 is a simple algorithm that heuristically finds the maximum (the largest maximal) clique of a graph. The intuition behind this algorithm is that, a vertex is very likely to be in the maximum clique if it is incident by the largest number of edges. Variations of this algorithm have been developed in the literature [43, 58, 63]. But this algorithm has not been applied to finding a complete S-IDNC solution, and its computational complexity has not been identified yet.

Algorithm 4.3 Heuristic maximum clique search

1: **input**: graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$;
2: **initialization**: an empty vertex set $\mathcal{V}_{\text{keep}}$;
3: **while** $\mathcal{G}$ is not empty **do**
4: add to $\mathcal{V}_{\text{keep}}$ the vertex $\boldsymbol{v}$ which has the largest number of edges incident to it;
5: update $\mathcal{G}$ by deleting $\boldsymbol{v}$ and all the vertices not connected to $\boldsymbol{v}$ (*These vertices can be ignored because they cannot be part of the target clique, which contains* $\boldsymbol{v}$);
6: **end while**
7: Output $\mathcal{V}_{\text{keep}}$. ($\mathcal{V}_{keep}$ *is a maximal clique, because vertices in it are pair-wise connected, and no more vertices can be added.*)

The computational complexity of Algorithm 4.3 is polynomial in the number of data packets K. The highest computational cost occurs when the input graph is complete, i.e., when all vertices are connected to each other. In this case, only one vertex will be removed in each iteration. Thus, the number of remaining vertices in iteration-i will be $K - i$, $\forall i \in [0, K-1]$. Then, to find the vertex with the largest number of incident edges, we need $K - i$ comparisons. The total computational cost is thus in the order of $\sum_{i=0}^{K-1} K - i = K(K-1)/2$. Hence, the computational complexity of Algorithm 3 is at most $\mathcal{O}(K^2)$.

A simple heuristic algorithm to find the minimum clique partition solution $\mathcal{S}_c$ of $\mathcal{G}_s$ is to iteratively apply Algorithm 3 to find and remove cliques from $\mathcal{G}_s$. As we have discussed already, $\mathcal{S}_c$ is not preferable due to its low packet multiplicity.

Hence, we modify this simple algorithm to heuristically find $\mathcal{S}_m$. The idea is to maximize the clique found in each iteration by adding more vertices to it whenever possible, which will increase the multiplicities of the added vertices/packets. The details are presented in Algorithm 4.4.

Algorithm 4.4 Heuristic S-IDNC solution search

input: a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$;
initialization: an empty vertex set $\mathcal{V}_{\text{covered}}$, a working graph $\mathcal{G}_w = \mathcal{G}$, and a counter $i = 0$;
while $\mathcal{V}_{\text{covered}} \neq \mathcal{V}$ **do**
  Find the maximum clique in $\mathcal{G}_w$ using Algorithm 4.3. Denote it by $\mathcal{M}_i$;
  Find the vertices in $\mathcal{V}_{\text{covered}}$ which are connected to $\mathcal{M}_i$. Denote their set by $\mathcal{V}_i$ (*They are the candidate vertices that could be added to* $\mathcal{M}_i$.);
  Generate a subgraph of $\mathcal{G}$ whose vertex set is $\mathcal{V}_i$. Denoted this subgraph by $\mathcal{G}'_i(\mathcal{V}_i, \mathcal{E}_i)$;
  Find the maximum clique in $\mathcal{G}'_i$ using Algorithm 1, denoted it by $\mathcal{M}'_i$ (*All vertices in* $\mathcal{M}'_i$ *are connected to each other and thus can all be added to* $\mathcal{M}_i$.);
  Update $\mathcal{V}_{\text{covered}}$ by adding vertices in $\mathcal{M}_i$ into it;
  Update $\mathcal{G}_w$ by removing $\mathcal{M}_i$ from it;
  Update $\mathcal{M}_i$ as $\mathcal{M}_i = \mathcal{M}_i \cup \mathcal{M}'_i$ (*The new clique is at least as large as the old one, and thus provides higher packet multiplicity*);
  $i = i + 1$;
end while

Below is an example:

Example 4.4

Consider the graph $\mathcal{G}_s$ in Fig. 4.2(b). In the first two iterations, the algorithm will choose $\mathcal{M}_1 = (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_4)$ and $\mathcal{M}_2 = (\mathbf{p}_3, \mathbf{p}_6)$, respectively. In the third iteration, $\mathcal{V}_{\text{covered}} = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_6\}$ and the algorithm can only choose $\mathcal{M}_3 = (\mathbf{p}_5)$. Among all the data packets in $\mathcal{V}_{\text{covered}}$, $\mathbf{p}_2$ can be added to $\mathcal{M}_3$. Thus $\mathcal{M}_3 = \{\mathbf{p}_2, \mathbf{p}_5\}$. The algorithm then stops and outputs $\mathcal{S}_m = \{(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_4), (\mathbf{p}_3, \mathbf{p}_6), (\mathbf{p}_2, \mathbf{p}_5)\}$.

In conclusion, we proposed an optimal coding algorithm that exhaustively finds all the possible $\mathcal{S}_m$, and also proposed its hybrid and heuristic alternatives. Both the optimal and hybrid algorithms are exponential-time algorithms

in terms of K due to their use of B-K algorithm, while the heuristic algorithm is a polynomial-time one. The output $\mathcal{S}_m$ is used as the S-IDNC solution for the semi-online transmission scheme. If fully-online transmission scheme is applied, the transmitted coding set $\mathcal{M}_f$ is chosen from $\mathcal{S}_m$.

4.6 Simulations

In this section, we present the simulated throughput and decoding delay performance of S-IDNC (abbreviated as S- in the figures) under different scenarios, including under fully- and semi-online transmission schemes, and under the use of optimal, hybrid, and heuristic coding algorithms (abbreviated in the figures as Fully-, Semi-, Opt., Hybr., and Heur., respectively).

We also compare S-IDNC with RLNC and G-IDNC. For RLNC, we assume a sufficiently large finite field, so that its throughput is almost surely optimal and serves as a benchmark. For G-IDNC, although its best performance is at least as good as S-IDNC (as we have explained in Section 4.3.3), this advantage will not necessarily be reflected in our simulation results. This is because there has not been any optimal G-IDNC algorithm. Instead, we apply a heuristic algorithm (abbreviated as Heur. G- in the figures) proposed in [63], which aims at minimizing the block completion time. This aim coincides with our optimization priorities for S-IDNC in Remark 4.1, namely, to minimize the block completion time first.

We conduct four sets of simulations. In all simulations we apply a block size of $K = 15$. In the first three sets, we fix $P_e = 0.2$ and set the number of receivers $N \in [5, 40]$. In the fourth set, we fix $N = 15$ and set $P_e \in [0.05, 0.4]$.

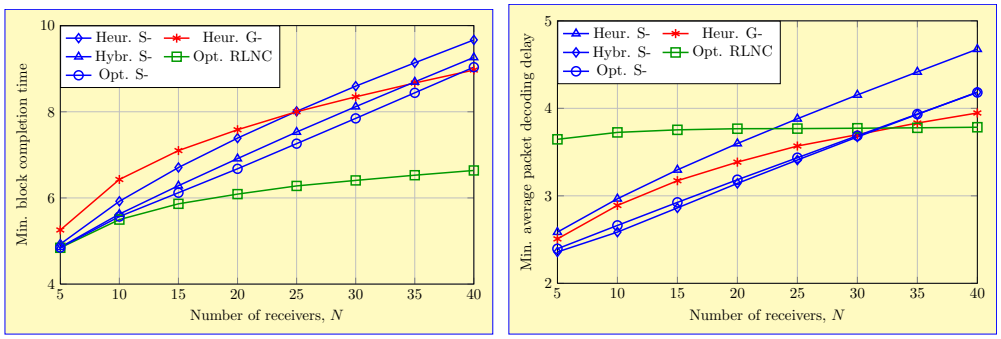
The purposes of the four sets of simulations are as follows. The first set compares the performance limits of the three techniques. The results are presented in Fig. 4.6. The second (resp. third) set of simulations compares the throughput decoding delay performance under fully-online (resp. semi-online) transmission schemes. The efficiency of feedback reduction using the semi-online scheme is also studied. The results are presented in Fig. 4.7 - 4.9. We note that the performance of RLNC is the same under both schemes, because RLNC is feedback-free. In the fourth set, we evaluate the performance of our hybrid algorithm under different packet erasure probabilities and compare it with fully-only heuristic G-IDNC and RLNC. The results are presented in Fig. 4.10.

Our observations on S-IDNC are as follows:

- According to Fig. 4.6, the absolute minimum block completion time of S-IDNC increases almost linearly with N. This result matches Corollary 4.1;
- According to Fig. 4.7 and 4.8, the fully-online transmission scheme always provides better throughput and decoding delay performance than the semi-online one. But the performance gain is marginal when N is small;
- On the other hand, the semi-online transmission scheme requires much less feedback than the fully-online one. Statistics from Fig. 4.9 indicate that most of the broadcast can be finished after 5 semi-online rounds, implying that only 5 rounds of feedback are collected. In contrast, since the fully-online scheme collets feedback after every transmission, it requires up to 13 rounds of feedback according to Fig. 4.7. Therefore, the semi-online scheme is a good alternative to the fully-online scheme in terms of both performance and feedback cost when the number of receivers is not too large;
- According to Fig. 4.6-4.10, the optimal coding algorithm always provides better throughput performance than its hybrid and heuristic alternatives. This result verifies our choice of $\mathcal{S}_m$ for throughput improvement, because only the optimal coding algorithm can always produce $\mathcal{S}_m$, which has $|\mathcal{S}_m| = U_s$;
- However, the optimal coding algorithm does not necessarily minimize the APDD. For example, in Fig. 4.6(b), the hybrid algorithm provides slightly smaller APDD than the optimal one when there are no packet erasures and when the number of receivers is $N \leqslant 15$;
- The performance gap between the optimal and hybrid algorithms is always marginal, and is much smaller than their gap with the heuristic one. Hence, the hybrid algorithm strikes a good balance between performance and computational load.

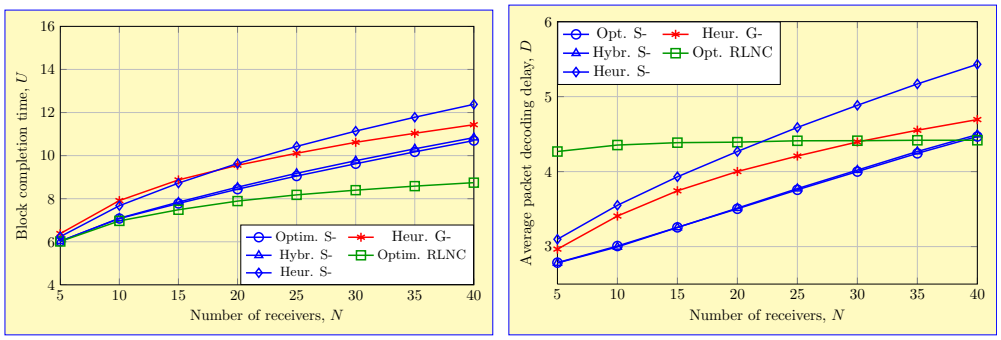
A cross comparison of RLNC, S-, and G-IDNC shows that:

- The throughput of RLNC is always the best. The throughput of S-IDNC is very close to RLNC when the number of receivers is small. Their gap increases with N;
- In general, the APDD of both S- and G-IDNC is better than RLNC. This advantage only vanishes when the block completion time of S- and G-IDNC becomes much larger than RLNC, which takes place when N is much larger



(a) BCT (b) APDD

Figure 4.6: The BCT and APDD performance limits of S- and G-IDNC, as well as RLNC.



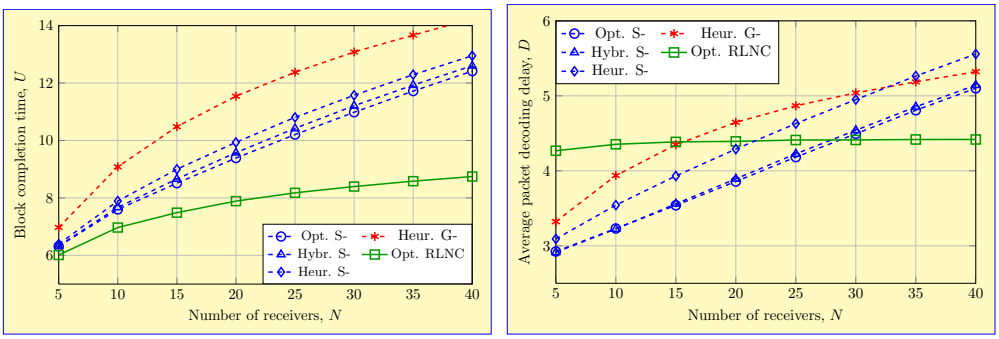
(a) BCT (b) APDD

Figure 4.7: The BCT and APDD performance of the fully-online transmission scheme when different coding algorithms are applied. It is compared with the performance of heuristic fully-online G-IDNC and RLNC.

than K;

- With a moderate number of receivers $N = 15$, the APDD of S-IDNC is better than RLNC under all simulated values of packet erasure probability;
- There is no clear winner between the performance of heuristic G-IDNC and optimal S-IDNC. We can expect that G-IDNC will outperform S-IDNC if its optimal coding algorithm is developed.

In summary, our simulations verified our theorems, propositions, and algorithms. They also demonstrated that, if we are concerned with both throughput and decoding delay performance, S-IDNC is a good alternative to RLNC when the number of receivers is not too large.



(a) BCT (b) APDD

Figure 4.8: The BCT and APDD performance of the semi-online transmission scheme when different coding algorithms are applied. It is compared with the performance of heuristic semi-online G-IDNC and RLNC.

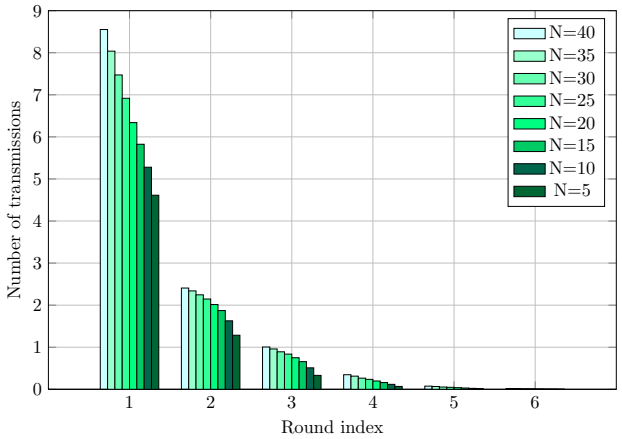
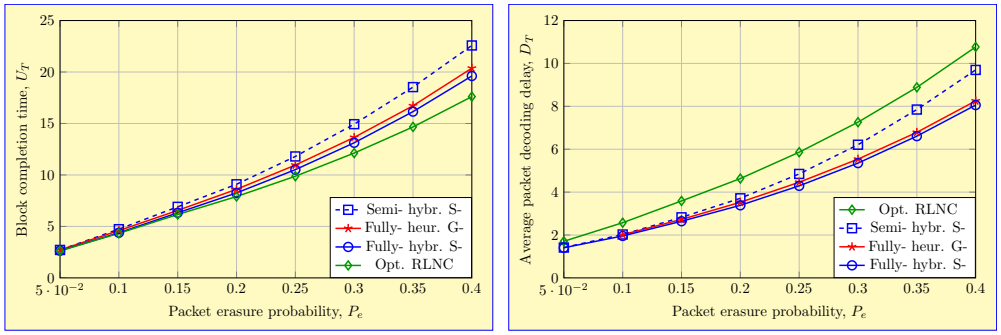


Figure 4.9: The length of each semi-online round when the optimal coding algorithm is applied.

4.7 Conclusion

In this chapter, we showed that, although aiming at minimizing APDD, S-IDNC is not able to approximate the minimum APDD of LNC or maximize the throughput, unless there are at most 3 receivers. Indeed, by using a random graph model, we showed that the BCT of S-IDNC increases with increasing number of receivers. By introducing the concept of perfect S-IDNC solution, we proved the NP-hardness of minimize the APDD of S-IDNC. We derived an upper bound on APDD and showed that minimizing the IDNC solution size can effectively reduce APDD.

Even though, S-IDNC is still very appealing in practice for its instant packet



(a) Block completion time (b) Average packet decoding delay

Figure 4.10: The BCT and APDD performance of hybrid S-IDNC with different feedback frequencies and under different P_e. It is compared with the performance of heuristic fully-online G-IDNC and RLNC.

delivery, low computational complexity, and low memory requirement. We thus have developed its optimal and heuristic implementations. By applying stochastic shortest path method, we showed that it is intractable to make optimal coding decisions in the presence of random packet erasures. We then used heuristic objective functions to determine the preferred coded packet(s) to send when fully- or semi-online receiver feedback is collected. We developed optimal and heuristic S-IDNC coding algorithms that minimize the solution size and increase packet multiplicity. We also compared S-IDNC with G-IDNC by proving the equivalence between the chromatic number of the complementary S-IDNC and G-IDNC graphs.

Our work provides news understandings of S-IDNC. It will facilitate the extension of S-IDNC to applications in other network settings, such as cooperative data exchange and distributed data storage.

Chapter 5

Generation-based Techniques: Enabling the Tradeoff Between Throughput and APDD

From previous chapters, we know that IDNC and RLNC offer different tradeoffs between throughput and APDD performance: while RLNC is throughput optimal, IDNC is able to offer lower APDD than RLNC when the number of receivers is not large. This observation raises a fundamental question:

Problem 5.1

What are the achievable throughput-delay tradeoffs of LNC techniques in wireless broadcast?

In this chapter, we will tackle this question through studying a class of LNC techniques called generation based techniques.

5.1 Introduction

The general answer to the aforementioned question would require identifying the complete spectrum of the throughput-delay tradeoffs, together with the LNC techniques that can achieve them. However, this has been a very challenging problem. First, to the best of our knowledge, only limited explicit results exist on the throughput-delay relationship of certain LNC techniques [65, 74]. Moreover, it is not clear how the throughput and packet decoding delay of different LNC techniques relate to each other. Finally, only a limited number of LNC schemes

exist in the literature and finding the complete tradeoff spectrum for LNC remains an open problem.

In this chapter, we tackle this problem through developing a new generation-based LNC technique. Explicitly, after the systematic transmission phase, the sender collects feedback from the receivers and partitions the packet block $\mathcal{P}$ into several generations accordingly, where each generation contains a different subset of data packets. Then in the coded transmission phase, the sender constructs the coded packets only using data packets from the same generation. Coded packets of different generations are transmitted separately following a generation scheduling (which will be specified later) until all receivers have decoded all data packets.

By partitioning $\mathcal{P}$ into generations, we effectively split the SFM $\boldsymbol{A}$ into a set of smaller instances of SFM and then complete the broadcast of each instance separately. The following example demonstrates how such partition can help reduce the APDD of RLNC:

- Consider the SFM in Fig. 5.1(a). Its BCT is 4. If RLNC is applied to all the 8 data packets in the block, the minimum APDD is also 4;
- By partitioning the packet block into two generations $\{\mathbf{p}_1, \cdots, \mathbf{p}_4\}$ and $\{\mathbf{p}_5, \cdots, \mathbf{p}_8\}$, we obtain two small instances of SFM $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. Since both instances require a minimum BCT of 2, the minimum BCT is still 4 in total. On the other hand, both receivers can decode two data packets after two RLNC coded packets of $\{\mathbf{p}_1, \cdots, \mathbf{p}_4\}$ are transmitted. Thus, the minimum decoding delay of $\{\mathbf{p}_1, \cdots, \mathbf{p}_4\}$ is reduced to 2, while the decoding delay of $\{\mathbf{p}_5, \cdots, \mathbf{p}_8\}$ is still 4. In total the minimum APDD is reduced to 3;
- The APDD can be further reduced by partitioning the packet block into 4 generations, each containing two data packets, as in Fig. 5.1(b). In total, the minimum BCT is still 4, but the minimum APDD can be further reduced to 2.5. We also note that the two data packets in each generation form an S-IDNC coding set of the original SFM.

The above example reveals two motivations of developing such a technique to analyze the throughput-delay tradeoff of LNC techniques. First, this technique will generalize several existing LNC techniques. For example, we will show that RLNC and IDNC are special cases of this technique. Second, this technique enables efficient reduction in APDD and decoding computational load. This is

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	$\mathbf{p}_4$	$\mathbf{p}_5$	$\mathbf{p}_6$	$\mathbf{p}_7$	$\mathbf{p}_8$
r_1	1	1	0	0	1	1	0	0
r_2	0	0	1	1	0	0	1	1
	$\boldsymbol{A}_1$				$\boldsymbol{A}_1$			

(a) Partition into 2

	$\mathbf{p}_1$	$\mathbf{p}_5$	$\mathbf{p}_2$	$\mathbf{p}_6$	$\mathbf{p}_3$	$\mathbf{p}_7$	$\mathbf{p}_4$	$\mathbf{p}_8$
r_1	1	0	1	0	1	0	1	0
r_2	0	1	0	1	0	1	0	1
	$\boldsymbol{A}_1$		$\boldsymbol{A}_2$		$\boldsymbol{A}_3$		$\boldsymbol{A}_4$	

(b) Partition into 4

Figure 5.1: Partition the packet block into 2 (then 4) generations. The minimum BCT is not changed. But the minimum APDD is reduced.

because: 1) packet decodings of different generations are independent, which means that data packets in earlier generations can be decoded sooner; and 2) such decodings generally require solving less linear equations.

Generation-based techniques have been the subject of many studies, e.g., [48–52], under various network settings. Most of the previous studies, however, focused on settings with no receiver feedback. In such settings, the sender is unaware of what packets are missing at each receiver, hence it cannot utilize this information to optimize the partition. Also, due to the absence of intermediate feedback, the sender is unaware of whether a generation has been decoded by all receivers or not. It can only repeatedly send coded packets of all generations until all receivers have decoded the complete packet block and signal the completion. This could incur redundant coded packets of generations that have already been decoded by all receivers. A feedback assisted scheme was proposed in [78] to reduce redundant transmissions, but its setting is different. This scheme assumes delayed feedback, which will incur redundant transmissions after the receivers have acknowledged block completion. Intermediate feedback is thus collected to predict block completion. This scheme, however, applies RLNC to the whole packet block, and thus still suffers from heavy computational load and large packet decoding delays.

The main drawback of generation-based techniques is that they can result in a larger BCT, and, as a result, lower throughput. As the first step to solve this problem, for each generation we construct coded combinations of its data packets using random coefficients selected from a sufficiently large finite field. This is equivalent to applying RLNC to each generation separately, and thus guaranteeing the optimal throughput within each generation. We will refer to this technique as generation-based RLNC [48–52].

Then, the main task is to design generation-based RLNC schemes that are

able to reduce APDD and computational load with no or graceful degradation on the overall throughput. To this end, two problems need to be addressed:

Problem 5.2

- How to partition the packet block into generations based on the SFM?
- How to schedule the transmissions of the generations?

In the next section, we will formally define the partition problem.

5.2 Problem Formulation

The input is the state-feedback-matrix (SFM) $\boldsymbol{A}$ after the systematic transmission phase. Based on the SFM, the sender then partitions the packet block $\mathcal{P}$ into a set of M disjoint *generations*, denoted by $\boldsymbol{P} = \{\boldsymbol{G}_m\}_{m=1}^{M}$. We define the *rank* γ_m of a generation $\boldsymbol{G}_m$ as the maximum number of packets in $\boldsymbol{G}_m$ wanted by one receiver, i.e.,

$$\gamma_m = \max_{r_n \in R} \sum_{\mathbf{p}_k \in \boldsymbol{G}_m} \boldsymbol{A}(n,k). \tag{5.1}$$

For example, for the SFM in Fig. 5.2(a), generation $\{\mathbf{p}_2, \mathbf{p}_3\}$ has a rank of 2 because both are wanted by r_1.

Then in the coded transmission phase, the sender broadcasts a *random-coded* packet of a generation in each transmission. That is, each coded packet is a linear combination of data packets from this generation, with coefficients chosen uniformly at random from a sufficiently large finite field $\mathbb{F}_q$. Hence, we effectively apply RLNC [6] to each generation. RLNC asymptotically guarantees linear independency between coded packets, so that any receiver who wants γ_m packets from $\boldsymbol{G}_m$ can decode these packets after receiving γ_m coded packets of $\boldsymbol{G}_m$ with high probability. Note that this decoding condition can also be achieved by deterministic codes such as maximum-distance-separation (MDS) codes and fountain codes [55, 56].

Consequently, an LNC solution $\mathcal{S}$ contains $\sum_{m=1}^{M} \gamma_m$ coded packets, where the first γ_1 coded packets are generated from $\boldsymbol{G}_1$, the next γ_2 coded packets are generated from $\boldsymbol{G}_2$, and so forth. To minimize the computational load and reduce the APDD, while also maintaining a good throughput performance, we propose the following minimum partitioning (MP) problem:

Problem 5.3

Given an SFM $\boldsymbol{A}$ and a constant γ, find the minimum value of M, for which there exists a partition of the packet block $\mathcal{P}$ into M disjoint generations $\boldsymbol{P} = \{\boldsymbol{G}_m\}_{m=1}^{M}$ such that the rank of each generation $\boldsymbol{G}_m$ does not exceed γ, i.e., $\gamma_m \leq \gamma$ for $m \in [1, M]$.

Minimizing the number of generations M under a constrained γ is an important goal due to the following reasons:

1. The lower values of M will result in smaller BCT and, as a result, higher throughput. In particular, we need $U = \sum_{m=1}^{M} \gamma_m \leqslant M\gamma$ transmissions in the coded transmission phase to satisfy all receivers even in the absence of packet erasures. In more practical settings with the presence of packet erasures, more transmissions may be needed, but its number will yet still depend on M: lower values of M typically require a smaller number of transmissions.

2. The lower values of M reduce the overall decoding computational load. Indeed, every receiver can decode its wanted data packets from any generation by solving a set of at most γ linear equations, which requires $\mathcal{O}(\gamma^3)$ operations. Hence, the total decoding computational load is bounded to $\mathcal{O}(M\gamma^3)$ operations.

3. The lower values of M reduce the APDD, because otherwise the increase in U due to larger M will incur large decoding delay of data packets in the last few generations.

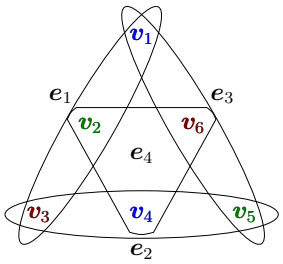
Motivated by the above importance of Problem MP, in this chapter we will aim to solve it. Our immediate concern is: *how hard is this problem?*

5.3 The Hardness of the Minimum Partitioning Problem

In this section, we analyze the hardness of Problem MP by constructing a reduction from a hypergraph coloring problem. As was discussed in Chapter 3, a hypergraph $\mathcal{H}$ is defined by a pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices, and $\mathcal{E}$ is the set of hyperedges. Every hyperedge $\boldsymbol{e} \in \mathcal{E}$ is a subset of $\mathcal{V}$ with size $|\boldsymbol{e}| \geqslant 1$. By identifying each vertex $\boldsymbol{v}_k$ with a data packet $\mathbf{p}_k$, and letting each hyperedge

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	$\mathbf{p}_4$	$\mathbf{p}_5$	$\mathbf{p}_6$
r_1	1	1	1	0	0	0
r_2	0	0	1	1	1	0
r_3	1	0	0	0	1	1
r_4	0	1	0	1	0	1

(a) SFM $\boldsymbol{A}$



(b) Hypergraph $\mathcal{H}$

Figure 5.2: A state feedback matrix and its hypergraph representation. The 1-regular coloring of $\mathcal{H}$ requires a minimum of 3 colors.

$\boldsymbol{e}_n$ incident to the set of data packets/vertices wanted by receiver r_n, we obtain an equivalence between $\mathcal{H}$ and $\boldsymbol{A}$. For example, the SFM in Fig. 5.2(a) can be generated from the hypergraph depicted in Fig. 5.2(b), with 6 data packets/vertices and 4 receivers/hyperedges.

A hypergraph is $\omega-$uniform if for every hyperedge $\boldsymbol{e} \in \mathcal{E}$ it holds that $|\boldsymbol{e}| = \omega$. A γ-regular coloring of $\mathcal{H}$ with M colors is a partition $\{\mathcal{V}_m\}_{m=1}^{M}$ of $\mathcal{V}$ that satisfies $|\mathcal{V}_m \cap \boldsymbol{e}_n| \leqslant \gamma$ for each $\boldsymbol{e}_n \in \mathcal{E}$. In other words, it is a coloring of the vertices with M colors, such that each color appears at most γ times in every hyperedge. The minimum number M of colors required for a γ-regular coloring of $\mathcal{H}$ is called the γ-chromatic number of $\mathcal{H}$. It was shown in [79,80] that the problem of finding the γ-chromatic number for a given hypergraph is NP-hard. Moreover, it is also not approximable, namely, the chromatic number cannot be approximated within a constant ratio by any polynomial-time algorithm.

We now establish the equivalence between a γ-regular coloring of $\mathcal{H}$ and a partition $\boldsymbol{P}$ on $\mathcal{P}$ given the associated $\boldsymbol{A}$.

Let $\{\mathcal{V}_m\}_{m=1}^{M}$ be a γ-regular coloring of the instance $\mathcal{H}$ of the γ-regular coloring problem. Then, let $\boldsymbol{P} = \{\boldsymbol{G}_m\}_{m=1}^{M}$ be a partition of $\mathcal{P}$ such that for $m = 1, \ldots, m$, a generation $\boldsymbol{G}_m$ includes packets that correspond to vertices in $\mathcal{V}_m$. It is easy to verify that $\boldsymbol{P}$ is a valid solution to Problem MP with M generations such that the rank of each generation is bounded by γ.

Similarly, for any partition $\boldsymbol{P} = \{\boldsymbol{G}_m\}_{m=1}^{M}$ of Problem MP, we can construct a γ-regular coloring solution $\{\mathcal{V}_m\}_{m=1}^{M}$ with M colors, in which each set $\mathcal{V}_m$ includes vertices that correspond to packets in $\boldsymbol{G}_m$. Thus, an optimal algorithm that minimizes M for Problem MP can be used to find the $\gamma-$chromatic number of a hypergraph, which is an NP-hard problem and is not approximable. We summarize our results in the following theorem:

Theorem 5.1

Problem MP is NP-hard and not approximable.

We note that our reduction can also be used to establish the hardness of several special cases of our problem based on other results in graph theory. For example, it is an NP-hard problem to find 2-regular coloring with two colors for a 3-uniform hypergraph [80, 81]. This implies that, in the special case in which every receiver wants $w = 3$ data packets, it is NP-hard to find a partition with $M = 2$ generations whose ranks are at most 2. In addition, it is NP-hard to find the chromatic number of hypergraphs in the special case of 1-regular coloring [82]. This implies that our problem is intractable even for the special case of $\gamma = 1$, in which every generation can include at most one missing packet for any receiver [83]. Such generations indeed form a partition of the S-IDNC graph $\mathcal{G}_s$. Thus, by using our technique we can verify the hardness result of finding the minimum BCT U_s of S-IDNC (was discussed in Chapter 4.3) using a different approach.

5.4 Partition Algorithms

Since Problem MP is hard to optimize and hard to approximate, it is hard to find a partition solution with the optimal throughput-delay tradeoff, in which both the BCT and APDD are minimized. However, it is possible to design heuristic algorithms that find partition solutions with the following two properties:

Property 5.1

Optimal generation order: generations should be ordered and transmitted according to decreasing number of their target receivers. Mathematically, $\sum_{\mathbf{p}_k \in \boldsymbol{G}_m} t(k) \geqslant \sum_{\mathbf{p}_k \in \boldsymbol{G}_n} t(k)$ for $m < n$.

Here $t(k)$ is the number of receivers which want $\mathbf{p}_k$. This order is optimal in the sense that it minimizes APDD compared with other orders: any swap between two optimally ordered generations will reduce the number of data packets being decoded sooner.

Property 5.2

Irreducibility: no data packet from a later generation $\boldsymbol{G}_n$ can be relocated to an earlier generation $\boldsymbol{G}_m$ $(n > m)$ without increasing γ_m.

This property is important because otherwise by relocating this data packet, we could obtain a better partition solution with lower APDD without increasing BCT.

Therefore, a partition solution satisfying the above two properties achieves local Pareto-optimal throughput-delay tradeoff, namely, any relocation of a data packet or a generation will incur an increase in BCT and/or APDD. In this section, we will develop two heuristic algorithms and compare their performance.

Recall that when $\gamma = 1$, a partition solution $\boldsymbol{P} = \{\boldsymbol{G}_m\}_{m=1}^{M}$ is indeed a clique partition $\mathcal{S}_c = \{\mathcal{M}_m\}_{m=1}^{M}$ of the S-IDNC graph $\mathcal{G}_s$. Hence, $\{\boldsymbol{G}_m\}_{m=1}^{M}$ can be iteratively found by using the maximum weighted clique search algorithm in Algorithm 4.3 in Chapter 4. By weighing each vertex $\boldsymbol{v}_k$ in $\mathcal{G}_s$ with the the number of targeted receivers (t_k) of the corresponding data packet $\mathbf{p}_k$, it is guaranteed that earlier cliques/generations will target more receivers. Property 5.1 is thus satisfied.

When $\gamma > 2$, as a natural extension to the $\gamma = 1$ case, we first propose to construct the generations by using S-IDNC coding sets $\{\mathcal{M}_m\}_{m=1}^{M}$ as the building blocks. For each generation $\boldsymbol{G}_m$, we iteratively allocate S-IDNC coding sets to it until it is saturated, in the sense that no more S-IDNC coding sets can be allocated to $\boldsymbol{G}_m$ without breaking the rank limit γ. The details of the algorithm is presented in Algorithm 5.1.

Algorithm 5.1 S-IDNC based Partitioning

Initialize: S-IDNC solution $\mathcal{S}_c = \{\mathcal{M}_m\}_{m=1}^{M}$, generation rank γ, a counter $m = 0$;
while $\mathcal{S}_c$ is nonempty **do**
  $m \leftarrow m + 1$, create an empty generation $\boldsymbol{G}_m$;
  while there exists a coding set $\mathcal{M}$ in $\mathcal{S}_c$ that would not increase the rank of $\boldsymbol{G}_m$ to $\gamma + 1$ **do**
    if there exist an $\mathcal{M}$ in $\mathcal{S}_s$ that would not increase γ_m by one **then**
      add to $\boldsymbol{G}_m$ the most popular $\mathcal{M}$ that satisfies this condition;
    else
      add to $\boldsymbol{G}_m$ the most popular $\mathcal{M}$ in $\mathcal{S}_s$ and set $\gamma_m \leftarrow \gamma_m + 1$;

9: **end if**
10: remove the chosen $\mathcal{M}$ from $\mathcal{S}_s$;
11: **end while**
12: **end while**
13: Order the first $m-1$ generations in the descending order of their number of target receivers.

We note, however, that the partition solution produced by this algorithm is not necessarily irreducible. This is because it operates on S-IDNC coding sets instead of individual data packets. It could be the case that no more S-IDNC coding sets can be allocated to a generation without breaking the rank limit γ, but a subset of data packets in this coding set can be.

Therefore, we propose in Algorithm 5.2 a direct partition algorithm, which shares the same idea as the S-IDNC based algorithm, but operates on individual data packets instead of S-IDNC coding sets.

The partition solution $\boldsymbol{P}$ generated from Algorithm 5.2 achieves local Pareto-optimal throughput-delay tradeoff. The reason is that all the first $M-1$ generations have a rank of γ and are saturated, in the sense that no more data packets can be added to them without breaking the rank limit γ.

Algorithm 5.2 Direct Partitioning

1: Initialize: SFM $\boldsymbol{A}$, generation rank γ, a counter $m = 0$
2: **while** $\mathcal{P}$ is nonempty **do**
3: $m \leftarrow m + 1$, create an empty generation $\boldsymbol{G}_m$;
4: **while** there exists a data packet in $\mathcal{P}$ that would not increase the current rank of $\boldsymbol{G}_m$, γ_m, to $\gamma + 1$ **do**
5: **if** there exists a packet in $\mathcal{P}$ that would not increase γ_m by one **then**
6: add to $\boldsymbol{G}_m$ the most popular data packet that satisfies this condition;
7: **else**
8: add the most popular packet in $\mathcal{P}$ to $\boldsymbol{G}_m$ and set $\gamma_m \leftarrow \gamma_m + 1$;
9: **end if**
10: remove the chosen packet from $\mathcal{P}$;
11: **end while**
12: **end while**
13: Order the first $m-1$ generations according to their number of target receivers.

We conclude this section by remarking the importance of collecting one round of feedback after the systematic transmission phase to the partition algorithms:

Remark 5.1

Our partition algorithms are enabled by collecting one round of feedback after the systematic transmission phase. Without this feedback, the sender can only perform a blind partition, namely, $\mathcal{P}$ is partitioned into M generations, each containing $\frac{K}{M}$ successive data packets. It can be expected that the BCT and APDD performance of our partition solution is much better than such blind partition solution, and the gain is brought by collecting only one round of feedback after the systematic transmission phase. In Section 5.6, we will numerically demonstrate the benefit of this feedback.

With the partition solution produced, we now discuss how to broadcast it.

5.5 Generation Scheduling Strategies

5.5.1 Scheduling Without Feedback

Given a partition solution $\boldsymbol{P} = \{\boldsymbol{G}_m\}_{m=1}^{M}$, there are several possible scheduling strategies to broadcast it. When receiver feedback is not available during the coded transmission phase, there are two common strategies, known as "random" and "round-robin". When the random strategy is applied, for each transmission the sender randomly chooses a generation and then broadcasts a random-coded packet of this generation [51]. When the round-robin strategy is applied, in the i-th coded transmission the $\text{Mod}(i, M)$-th generation is chosen, where $\text{Mod}(x, y)$ outputs the reminder of $\frac{x}{y}$. Consequently, any M consecutive coded transmissions will contain one coded packet of every generation [52]. Both strategies will stop when the sender has received the block completion feedback from all receivers.

Although easily implementable, the above two strategies are inefficient in terms of throughput. As long as there is one generation that has not been decoded by all receivers, all the generations, including those already decoded by all receivers, will be broadcast. Coded packets of decoded generations are non-innovative to *every* receiver, and thus are completely redundant.

One way to reduce redundant coded packets is to use overlapped generations, which allows receivers to decode some data packets in a generation $\boldsymbol{G}_m$ using coded packets of other generations overlapped with $\boldsymbol{G}_m$ [49,50,52]. Compared to

the disjoint counterpart, this approach is able to reduce the expected BCT under the presence of random packet erasures at a price of an increase in the minimum BCT. The gain, however, will reduce with decreasing values of P_e, and will become negative when P_e is close to zero. Hence, we will focus on disjoint generations, and avoid redundant coded packets through collecting receiver feedback.

5.5.2 Scheduling With Feedback

In order to completely avoid the redundant coded packets, we consider collecting receiver feedback during the coded transmission phase. In this section, we propose two feedback-assisted generation scheduling strategies, and then analyze their advantages and disadvantages.

Our first strategy is called "sequential". The coded transmission phase is segmented into M sequential rounds. In the m-th sequential round, random-coded packets of $\boldsymbol{G}_m$ are continuously transmitted until all receivers have decoded all data packets in $\boldsymbol{G}_m$ and send feedback.

Our second strategy is called "semi-online". We segment the coded transmission phase into several semi-online rounds. In each semi-online round, all M generations are broadcast, where for $\boldsymbol{G}_m$, γ_m random-coded packets of it are broadcast. Thus, each semi-online round includes $\sum_{m=1}^{M} \gamma_m$ coded transmissions, and it is possible that the broadcast will complete after only one semi-online round. After each semi-online round, every receiver updates the sender with the number of random-coded packets it has already received from each generation. The sender can thus calculate the number of random-coded packets that each receiver still wants from each generation, and then can update $\{\gamma_m\}_{m=1}^{M}$ accordingly for the next round. Below is an example of how γ_m is updated after a semi-online round.

Example 5.1

Consider three receivers $\{r_1, r_2, r_3\}$ that want 2, 3, and 4 data packets from generation $\boldsymbol{G}_1$, respectively. Hence, $\gamma_1 = 4$ random-coded packets of $\boldsymbol{G}_1$ are broadcast in the first semi-online round, along with coded packets of other generations. Assume that after the first semi-online round, r_1, r_2, r_3 have received 2, 1, and 3 random-coded packets of $\boldsymbol{G}_1$, respectively. They still want 0, 2, and 1 coded packets to decode $\boldsymbol{G}_1$, respectively. γ_1 is thus updated to 2. In the second semi-online round, 2 random-coded packets of $\boldsymbol{G}_1$ will be broadcast, along with coded packets of other generations.

5.5.3 Analysis and Comparison

Our proposed strategies have the following property:

Property 5.3

Given a partition solution, the sequential and semi-online strategies offer the same mean BCT, and thus the same mean throughput.

This is because by collecting feedback, both sequential and semi-online strategies never waste any transmission on generations that have been decoded by all receivers. This means that they both guarantee the optimal throughput within each generation. Indeed, from a statistical point of view, the semi-online strategy only "swaps" the coded transmissions for each generation in the sequential strategy.

However, such swaps are able to reduce the APDD of the semi-online strategy compared to the sequential one. This is because, in the sequential strategy, all the target receivers of a sub-generation $\boldsymbol{G}_m$ have to wait until the sequential round for $\boldsymbol{G}_{m-1}$ is complete, even if this round is hindered by only one receiver that experiences a bad channel.

The primary advantage of the sequential strategy over the semi-online strategy is that it offers tunable throughput and APDD during the broadcast of a data block. After the completion of one generation, the sender can adaptively partition the remaining data packets in the block, either using the same γ, or applying a different γ. A larger γ will generally trade APDD off for higher throughput, while a smaller γ will generally trade throughput off for lower APDD. For example, when APDD reduction becomes the primary concern during the broadcast, the system can easily switch to IDNC by setting $\gamma = 1$ in all remaining sequential rounds. An example of three adaptive sequential rounds when the S-IDNC based partition algorithm is applied is plotted in Fig. 5.3. In the 1st (resp. 2nd, 3d) round, a generation that contains 2 (resp. 1, $U_s - 3$) S-IDNC coding sets is broadcast. Such adaption is not possible under the semi-online strategy, as all the generations will have already been transmitted after the first semi-online round.

The sequential strategy requires M rounds of feedback from every receiver. The generation size γ can also be adjusted to fit the system's specific feedback frequency if there is any. On the other hand, when the semi-online strategy is applied, the broadcast could be completed after m semi-online rounds with a non-zero probability for any positive integer m. Hence, the amount of feedback

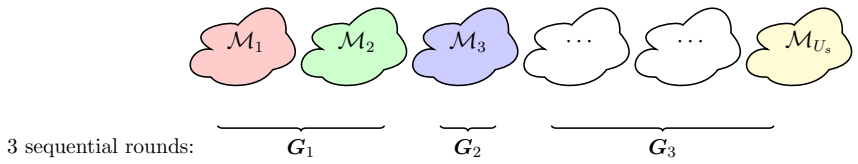


Figure 5.3: An example of adaptive sequential rounds.

rounds required by the semi-online strategy is lower bounded by 1 and cannot be upper bounded due to stochastic erasures.

In conclusion, both the sequential and semi-online strategies are optimal generation scheduling strategies that offer the same mean throughput performance. They offer different advantages such as low APDD, adaptive throughout-delay tradeoff, and low feedback costs. Hence, there is no clear winner between the sequential and the semi-online strategies. Which one to adopt depends on the application.

5.6 Simulations

In this section, we conduct three sets of simulations:

1. We demonstrate the best throughput-delay tradeoff achieved by the S-IDNC based and direct partition algorithms. To this end, in this set of simulations only, we assume that the coded transmission phase is erasure-free. The results are shown in Fig. 5.4;

2. We compare the BCT and APDD performance of the sequential and semi-online generation scheduling. The results are shown in Fig. 5.5;

3. We demonstrate the performance improvement brought by collecting only one round of feedback after the systematic transmission phase, as remarked in Remark 5.1. To demonstrate this gain, we assume no intermediate feedback during the coded transmission phase. The simulation results are shown in Fig. 5.6;

In all the three sets of simulations, we consider the broadcast of $K = 20$ data packets to a various number N of receivers. The packet erasure probability is $P_e = 0.2$ for all receivers.

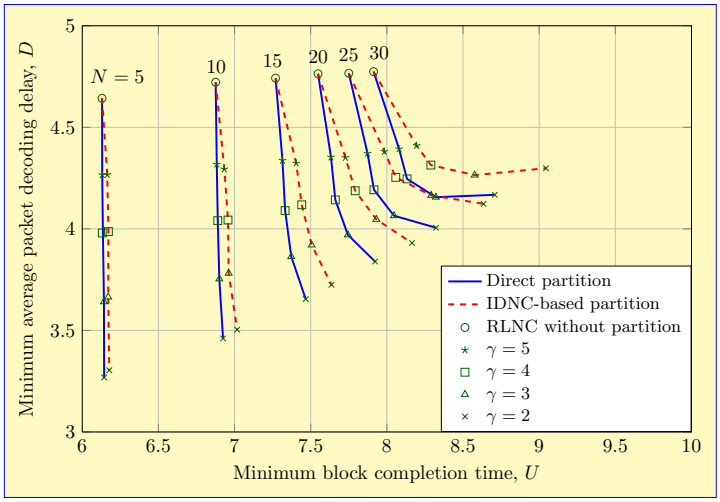


Figure 5.4: **The throughput-delay tradeoff of the proposed partition algorithms.**

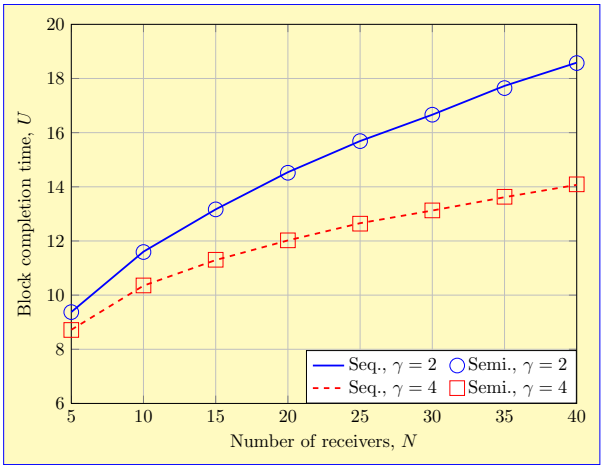
5.6.1 Best Throughput-delay Tradeoff

In this subsection, we demonstrate the best throughput-delay tradeoff achieved by the S-IDNC based and direct partition algorithms with $\gamma \in [2, 4]$. We also add, as a benchmark, the throughput-delay tradeoff achieved by the RLNC technique without partitioning. The results are shown in Fig. 5.4.

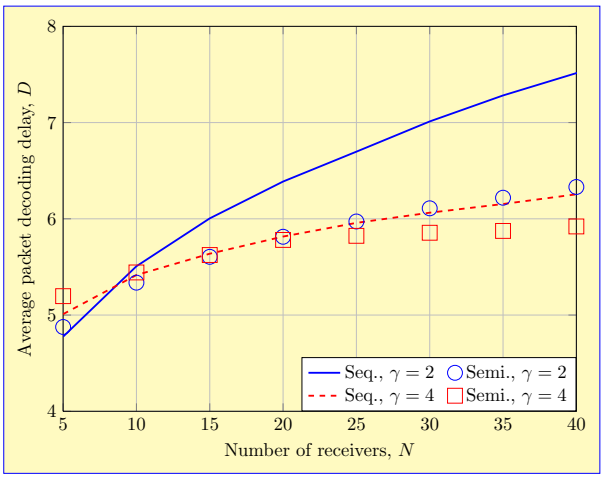
Our main observation is that both partition algorithms always offer lower APDD than the RLNC technique without partitioning. The reduction is up to 30%. On the other hand, the increase in BCT due to partitioning is negligible when the number of receivers N is small, and is only 11% when $N = 30$.

We also observe that the BCT and APDD of the direct algorithm is similar to the S-IDNC based one when N is small, but becomes better with increasing N. This is mainly due to the fact that S-IDNC based partition solutions are not irreducible as the direct partition solutions. Another reason is that the performance of S-IDNC degrades quickly with increasing N (so does S-IDNC based partitioning), as we have already shown in Chapter 4. Despite this, both algorithms provide much better throughput-delay tradeoff than the RLNC technique without partitioning when N is not too large.

Due to the superiority of the direct partition algorithm over the S-IDNC based one, in the remaining simulations we will apply the direct partition algorithm.



(a) BCT

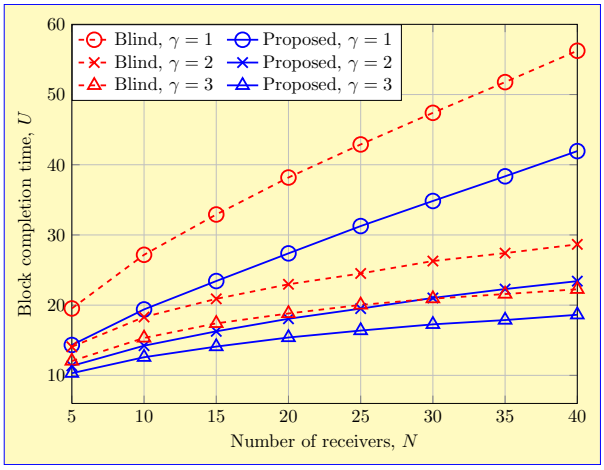


(b) APDD

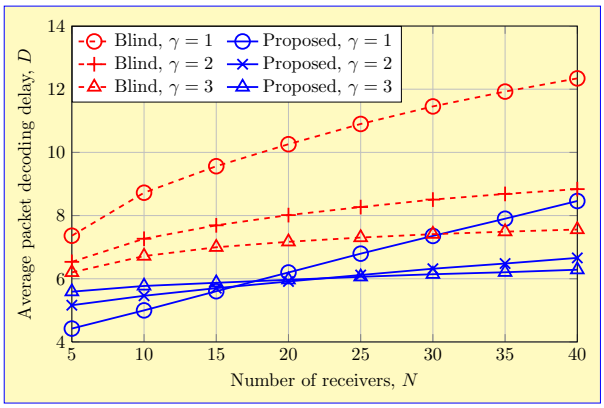
Figure 5.5: The performance of the sequential and semi-online generation scheduling strategies.

5.6.2 Generation Scheduling Strategies

In this subsection, we compare the performance of the sequential and semi-online generation scheduling strategies. We apply two values of the generation rank γ: 2 and 4. Simulation results are shown in Fig. 5.5. Our first observation is that the sequential and semi-online strategies always share the same mean throughput performance. This result matches Property 5.3. The second observation is that the APDD of the semi-online strategy outperforms the sequential one. This gap reduces with increasing γ. This is because with increasing γ the performance of the partition solution will approach the traditional RLNC, whose APDD is not affected by the way how feedback is collected.



(a) BCT



(b) APDD

Figure 5.6: Performance improvements with one round of feedback.

5.6.3 The Benefit of Collecting One Round of Feedback

As we have remarked in Remark 5.1, in this subsection we demonstrate the performance improvement brought by collecting only one round of feedback after the systematic transmission phase. In order to demonstrate the benefit of collecting only one round of feedback, we assume no intermediate feedback during the coded transmission phase. Under this assumption, the sequential strategy is not applicable because it relies on intermediate feedback to initiate a new generation. We thus apply the semi-online strategy. The only modification here is that we are not able to update $\{\gamma_m\}_{m=1}^{M}$ after each semi-online round due to the absence of receiver feedback. We also note that, when the blind partition is applied, the sender is unaware of the rank of each generation. Hence, in each semi-online round only one coded packet of every blindly produced generation is transmitted.

The simulation results are shown in Fig. 5.6. We observe that by collecting only one round of feedback, our algorithm can reduce BCT by 30% and APDD by 40% compared to the one without feedback. The improvement decreases with increasing γ, because both techniques will converge to the traditional RLNC without partitioning.

5.7 Conclusion

In conclusion, in this chapter we proposed a feedback-assisted generation-based LNC technique. By simply tuning the generation rank γ, this technique enables a wide range of throughput-delay tradeoffs, as well as computational load and feedback frequency. We proved the NP-hardness of the optimal partition problem, and proposed a partitioning algorithm that achieves local Pareto-optimal throughput-delay tradeoffs. We also designed transmission schemes with fully- and semi-online receiver feedback. They provide various advantages such as lower APDD and online tradeoff adaptation. The performance of the proposed generation-based technique is strictly better than those without feedback.

An interesting feature of the proposed technique is that it generalizes S-IDNC and RLNC through changing the values of γ. Then since S-IDNC is not an APDD-approximation technique in general, the proposed technique is not an APDD-approximation technique *in general*, too. However, it is too early to conclude that the proposed technique is not able to approximate APDD *under any settings*. This is because RLNC is also a special case of our technique and its APDD has not been studied yet. Hence in the next chapter, we will study the APDD performance of the class of LNC techniques that RLNC belongs to, namely, throughput-optimal LNC techniques.

Chapter 6

On the APDD of Throughout-optimal Techniques

In previous chapters we observed that both S-IDNC technique and the proposed generation-based technique do not approximate the APDD with a guaranteed ratio in general. The main cause of their unbounded APDD is their sub-optimality in terms of throughput. When throughput is sacrificed for lower APDD, the BCT increases accordingly. This may incur significant delay to the data packets decoded in the final stages of the broadcast, which will, in turn, increase APDD.

We are thus motivated to study in this chapter the APDD performance of throughput-optimal LNC techniques, and to design new techniques.

6.1 Introduction

As we have already reviewed in Section 1.4.1, throughput-optimal LNC techniques do not provide early packet decodings in general, nor are they designed for APDD minimization. However, their optimal throughput minimizes BCT. This imposes an upper bound on the decoding delay of each data packet and, thus, may yield a guaranteed APDD that is within a constant multiple of the minimum APDD.

Therefore, in this chapter we aim at quantifying the APDD performance of throughput-optimal LNC techniques and then trying to better approximate APDD while maintaining throughput optimality.

To this end, we conduct a two-part study. In the first part, we study the worse APDD performance of all throughput-optimal LNC techniques. Then by comparing the results with the lower bounds of APDD derived in Chapter 3 and by making examples, we will prove that all throughput-optimal LNC techniques

approximate APDD with a ratio of between $\frac{4}{3}$ and 2. In particular, those not offering early packet decodings, such as RLNC, have a ratio of exactly 2.

Results from the first part motivated our second part of work (Section 6.3 and 6.4): to develop a new throughput-optimal LNC technique that guarantees early packet decodings, so that the approximation ratio can be reduced while maintaining throughput optimality. We implement our technique under different feedback frequencies, and prove that its performance does not degrade even if feedback is not collected after every transmission. We also conduct extensive simulations in Section 6.5. The results show that our technique provides the lowest APDD compared with some existing techniques under all parameter settings.

6.2 The APDD Performance of Throughput-optimal Techniques

In this section, we will derive the APDD approximation ratio β of throughput-optimal LNC techniques by deriving an upper bound of their expected APDD. The underlying rationale is that, if this upper bound is at most β times of the lower bound of the minimum expected APDD, then all throughput-optimal LNC techniques have a ratio of at most β.

In order to derive this upper bound, we introduce the concept of *the worst throughput-optimal LNC technique*:

Definition 6.1

The worst throughput optimal LNC technique does not allow any receiver r_n to decode any wanted data packets until it has received w_n coded packets, for $n \in [1, N]$.

Obviously, this technique has the worst expected APDD among all throughput-optimal LNC techniques. Any other throughput-optimal LNC techniques that offer early packet decodings will have a lower expected APDD.

An example of the worst throughput optimal LNC technique is RLNC, for it is asymptotically throughput optimal and does not offer early packet decodings in general. Therefore, the expected APDD of RLNC is equal to the upper bound of the expected APDD of throughput-optimal LNC techniques. Similar to Section 3.2.4, we consider two types of expectation:

- The expected APDD for a given SFM;

- The overall expected APDD as a function of system parameters, including the number of data packets K, the number of receivers N, and the packet erasure probabilities $\{P_{e,n}\}_{n=1}^{N}$.

When the worst throughput-optimal LNC technique, e.g., RLNC, is applied, a receiver who wants w_n data packets is able to decode them in one go after receiving w_n coded packets, which is expected to take $\frac{w_n}{1-P_e}$ coded transmissions. Hence, the mean APDD experienced by r_n is:

$$E[D_{R,n}|w_n] = \frac{w_n}{1-P_{e,n}} \tag{6.1}$$

where the "R" in the subscript stands for RLNC. By comparing (3.5) and (6.1) we observe:

$$\frac{E[D_{R,n}|w_n]}{E[\underline{D}_n|w_n]} = \frac{2w_n}{w_n+1} \leqslant 2 \tag{6.2}$$

Therefore, if RLNC is applied, the expected APDD of every receiver r_n is at most twice of the corresponding lower bound. This implies that the expected APDD of the SFM is also at most twice of the corresponding lower bound. To see this, we denote by $E[D_{R,n}|\{w_n\}_{n=1}^{N}]$ the expected APDD of a given SFM when RLNC is applied. Then,

$$\begin{aligned} E[D_R|\{w_n\}_{n=1}^{N}] &= \frac{E[D_{R,1}|w_1]w_1 + E[D_{R,2}|w_2]w_2 + \cdots + E[D_{R,N}|w_N]}{w_1+w_2+\cdots+w_N} \\ &= \frac{\sum_{n=1}^{N} \frac{w_n^2}{1-P_{e,n}}}{\sum_{n=1}^{N} w_n} \end{aligned} \tag{6.3}$$

By comparing (6.3) and (3.6), we can easily show that the expected APDD of using RLNC is at most twice of the lower bound and, thus, is also at most twice of the minimum expected APDD. Hence, the approximation ratio of RLNC is at most 2, i.e., $\beta \leqslant 2$. Then, if we can find an instance of the APDD minimization problem in which the expected APDD of using RLNC is exactly twice of the minimum expected APDD, we can prove the following theorem:

Theorem 6.1

RLNC is a 2-approximation technique of the APDD minimization problem when coded transmissions are subject to random packet erasures.

Proof. Recall that the approximation ratio of a technique is the largest ratio among all combinations of $\boldsymbol{A}$ and $\{P_{e,n}\}_{n=1}^{N}$. Here we only need to provide an

combination of $\boldsymbol{A}$ and $\{P_{e,n}\}_{n=1}^{N}$, for which the expected APDD of using RLNC is twice of the minimum APDD. This instance will imply that $\beta \geqslant 2$. This, together with the fact that $\beta \leqslant 2$ for RLNC, will prove that $\beta = 2$.

Consider an instance with $2K$ data packets and 2 receivers. Receiver 1 wants $\{\mathbf{p}_1, \cdots, \mathbf{p}_K\}$ and has $\{\mathbf{p}_{K+1}, \cdots, \mathbf{p}_{2K}\}$, whereas receiver 2 wants $\{\mathbf{p}_{K+1}, \cdots, \mathbf{p}_{2K}\}$ and has $\{\mathbf{p}_1, \cdots, \mathbf{p}_K\}$. Further assume that the packet erasure probabilities are $\{P_{e,n}\}_{n=1}^{2} = 0$. Then if RLNC is applied, all data packets are decoded after K transmissions, and thus $D_R = K$ in this instance. On the other hand, if we send the binary XOR of $\mathbf{p}_u$ and $\mathbf{p}_{u+K}$, i.e., send $\mathbf{p}_u \oplus \mathbf{p}_{u+K}$ in the u-th transmission for $u \in [1, K]$, both receivers can decode a data packet in each transmission. This achieves the minimum APDD with a value of $D_{\min} = \frac{K+1}{2}$. Consequently, the approximation ratio of RLNC in this instance is $\frac{2K}{K+1}$, which approaches 2 with increasing K. □

Then since RLNC is the worst technique in its class, we obtain:

Theorem 6.2

The approximation ratio of all throughput-optimal LNC techniques is at most 2.

Similarly, we can prove a lower bound of β for all throughput-optimal LNC techniques:

Theorem 6.3

The approximation ratio of all throughput-optimal LNC techniques is at least $\frac{4}{3}$.

Proof. Similar to the proof of Theorem 6.1, to prove that $\beta \geqslant \frac{4}{3}$ it suffices to show an instance in which the APDD of all throughput-optimal LNC techniques is $\frac{4}{3}$ times of the minimum.

Consider an instance $\boldsymbol{A}$ of the APDD minimization problem with 2 data packets and N receivers. After the systematic transmission phase, r_1 only wants $\mathbf{p}_1$, r_2 only wants $\mathbf{p}_2$, and all the remaining $N-2$ receivers want both $\mathbf{p}_1$ and $\mathbf{p}_2$. Further assume that $\{P_{e,n}\}_{n=1}^{N} = 0$. When N is sufficiently large, the APDD is minimized if $\mathbf{p}_1$ and $\mathbf{p}_2$ are transmitted separately using two transmissions. This yields $D_{\min} = 1.5$. On the other hand, if throughput-optimal techniques are applied, $\mathbf{p}_1$ and $\mathbf{p}_2$ must be combined in the first transmission in order to serve both r_1 and r_2, which does not allow the remaining $N-2$ receivers to

decode in the first transmission. These receivers can only decode in the second transmission. Consequently, the APDD is 2 when N is sufficiently large, which is $\frac{4}{3}$ times of the minimum. □

We now derive the overall expected APDD of RLNC, which is an upper bound of the overall expected APDD of all throughput-optimal LNC techniques:

Theorem 6.4

Given system parameters K, N, and P_e, the overall expected APDD performance of RLNC is:

$$E[D_R] = \frac{1}{1-P_e} E\left[\frac{\sum_{n=1}^{N} w_n^2}{\sum_{n=1}^{N} w_n}\right], \quad \{w_n\}_{n=1}^{N} \sim B(K, P_e) \tag{6.4}$$

$$\approx 1 + \frac{KP_e}{1-P_e}, \quad \text{when } N \text{ is sufficiently large} \tag{6.5}$$

This theorem indicates that, when all receivers experience the same packet erasure probability, the APDD performance of RLNC is robust to the increasing number of receivers. This property is verified by our simulation results. The results for $K = 15$ data packets, $N \in [5, 100]$ receivers, and packet erasure probabilities of $\{P_{e,n}\}_{n=1}^{N} = 0.2$ are plotted in Fig. 6.2. The results also affirm the accuracy of our estimation in 6.5. Moreover, the overall expected APDD of RLNC is less then twice of the lower bound of the minimum expected APDD of LNC techniques. This again confirms the approximation ability of RLNC and, thus, all throughput-optimal LNC techniques.

In conclusion, in this section we proved that the APDD-approximation ratio of all throughput-optimal LNC techniques lies between $\frac{4}{3}$ and 2. Particularly, those do not provide early packet decodings are 2-approximation techniques. Indeed, to the best of our knowledge, all existing throughput-optimal LNC techniques, including RLNC, are such techniques. Then the question is, *can this ratio be further reduced?* In the next two sections, we will affirmatively answer this question by developing a new throughput-optimal LNC technique.

6.3 A New Throughput-optimal APPD-approximation LNC Technique

In this section, we design an LNC technique whose coded packets are: 1) innovative to every receiver and thus guarantee throughput optimality, and 2) always

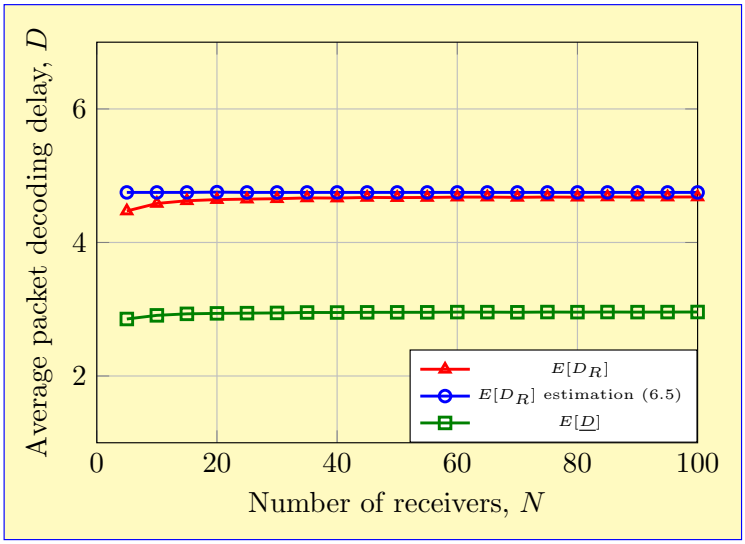


Figure 6.1: The proposed estimation of $E[D_R]$ and its comparison to the simulated value, and the lower bound $E[\underline{D}]$ of all LNC techniques derived in Section 3.2.4.

able to offer early packet decodings. Such a technique will be able to reduce the APDD from RLNC and, thus, reduce the approximation ratio. To describe our proposed design, we first generalize the definition of Wants sets in Chapter 3:

Definition 6.2

The Wants set $\mathcal{W}_n$ of a receiver r_n is the set of data packets it has not decoded yet.

Under this definition, $\mathcal{W}_n$ after the systematic transmission phase is the same as defined in Chapter 3, i.e., it is still the set of data packets that r_n has not received after this phase. Then during the coded transmission phase, wanted data packets decoded by r_n will be removed from $\mathcal{W}_n$, whereas those included in the received coded packets but not yet decodable will remain in $\mathcal{W}_n$.

The Wants sets, or equivalently the packet reception state, of all receivers can be represented using the following hypergraph.

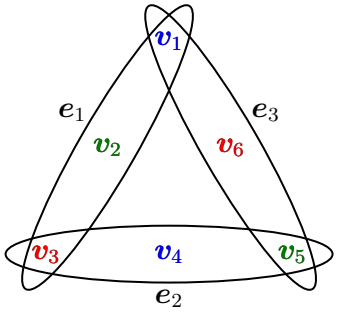
Definition 6.3

In the hypergraph model $\mathcal{H}(\mathcal{V}, \mathcal{E})$ of the receivers' packet reception state, each vertex $\boldsymbol{v} \in \mathcal{V}$ represents a data packet, i.e., $\boldsymbol{v}_k \leftrightarrow \mathbf{p}_k$. Each hyperedge $\boldsymbol{e} \in \mathcal{E}$ represents the Wants set of a receiver, i.e, $\boldsymbol{e}_n \leftrightarrow \mathcal{W}_n$, by connecting the data packets/vertices not yet decoded and hence still included in $\mathcal{W}_n$.

An example of SFM and its hypergraph representation is given in Fig. 6.2.

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	$\mathbf{p}_4$	$\mathbf{p}_5$	$\mathbf{p}_6$
$\mathbf{r}_1$	1	1	1	0	0	0
$\mathbf{r}_2$	0	0	1	1	1	0
$\mathbf{r}_3$	1	0	0	0	1	1

(a) SFM $\boldsymbol{A}$



(b) Hypergraph model $\mathcal{H}$

Figure 6.2: An example of SFM A and its hypergraph representation $\mathcal{H}$.

It is the same as Fig. 3.2, except that the concept of wanted data packets is generalized. Our hypergraph model $\mathcal{H}$ has two important properties, which are related to throughput optimality and early packet decodings, respectively:

Property 6.1

The coding set of any throughput-optimal coded packet must form a vertex cover of $\mathcal{H}$.

Here a vertex cover of $\mathcal{H}$ is a subset $\mathcal{V}'$ of $\mathcal{V}$ that is incident to every hyperedge, i.e., $|\boldsymbol{e}_n \cap \mathcal{V}'| > 0$ for $n \in [1, N]$. This property holds because no matter what linear network coding technique is applied, to guarantee the innovation of the coded packet $\boldsymbol{X}$ to every receiver, the coding set $\mathcal{M}$ of $\boldsymbol{X}$ must comprise at least one wanted data packet of every receiver. Thus, the associated vertex set $\mathcal{V}'$ of $\mathcal{M}$ must be incident to every hyperedge, and thus forms a vertex cover.

We further denote by $\mathcal{V}_c$ a minimal vertex cover of $\mathcal{H}$, which is a vertex cover that is not the superset of any other smaller vertex cover. Every minimal vertex cover has the property that it uniquely covers at least one hyperedge, i.e., $\exists n : |\mathcal{V}_c \cap \boldsymbol{e}_n| = 1$. This is because otherwise we can remove any single vertex from $\mathcal{V}_c$ to obtain a smaller vertex cover. Denote by $\mathcal{M}_c$ the coding set corresponding to $\mathcal{V}_c$, it holds that $\exists n : |\mathcal{M}_c \cap \boldsymbol{\omega}_n| = 1$. i.e., there exists at least one receiver who only wants one data packet from $\mathcal{M}_c$. Upon receiving a coded packet of $\mathcal{M}_c$, this receiver can *instantly* decode that data packet. Thus, $\mathcal{H}$ has the property below:

Property 6.2

A coded packet of the coding set $\mathcal{M}_c$ of any minimal vertex cover $\mathcal{V}_c$ in $\mathcal{H}$ always offers at least one instant packet decoding opportunity.

The above two properties of $\mathcal{H}$ underpin the core of our proposed technique:

Proposition 6.1

In each coded transmission, the sender finds a minimal vertex cover $\mathcal{V}_c$ of $\mathcal{H}$, sends a linear coded packet of the packets in $\mathcal{V}_c$, and then updates $\mathcal{H}$. Upon the reception of a coded packet, every receiver attempts to decode data packets through Gaussian eliminations.

We now present a handy example before discussing the details.

Example 6.1

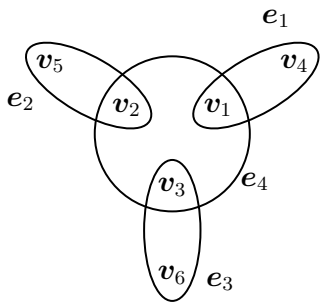
Consider the SFM in Fig. 6.3(a), with its hypergraph $\mathcal{H}$ plotted in Fig. 6.3(b). For simplicity we assume erasure-free transmissions. Then, the coded transmission phase of our technique is as follows:

1. A minimal vertex cover of $\mathcal{H}$ is $\mathcal{V}_c = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$. Thus, $\boldsymbol{X} = \alpha_1\mathbf{p}_1 + \alpha_2\mathbf{p}_2 + \alpha_3\mathbf{p}_3$ is sent, where $\{\alpha_1, \alpha_2, \alpha_3\}$ are non-zero coefficients chosen from $\mathbb{F}_q$. Since receivers r_1, r_2, r_3 only want $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ from $\boldsymbol{X}$, respectively, they can decode them from $\boldsymbol{X}$. Hence, $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, $\boldsymbol{v}_3$ are removed from $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, $\boldsymbol{e}_3$, respectively. On the other hand, r_4 cannot decode any data packet. It only holds $\boldsymbol{X}$, and thus $\boldsymbol{e}_4$ is still incident to $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$. The updated graph $\mathcal{H}'$ is plotted in Fig. 6.3(c);
2. A minimal vertex cover of $\mathcal{H}'$ is $\mathcal{V}_c' = \{\boldsymbol{v}_1, \boldsymbol{v}_4, \boldsymbol{v}_5, \boldsymbol{v}_6\}$. Thus, $\boldsymbol{X}' = \alpha_1'\mathbf{p}_1 + \alpha_4'\mathbf{p}_4 + \alpha_5'\mathbf{p}_5 + \alpha_6'\mathbf{p}_6$ is sent. Since receivers r_1, r_2, r_3, r_4 only want $\mathbf{p}_4$, $\mathbf{p}_5$, $\mathbf{p}_6$, $\mathbf{p}_1$ from $\boldsymbol{X}'$, respectively, they can decode them from $\boldsymbol{X}'$. Hence, $\boldsymbol{v}_4$, $\boldsymbol{v}_5$, $\boldsymbol{v}_6$, $\boldsymbol{v}_1$ are removed from $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, $\boldsymbol{e}_3$, $\boldsymbol{e}_4$, respectively. Since receivers $\{r_1, r_2, r_3\}$ are satisfied, $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$ are completely removed from $\mathcal{H}$. On the other hand, r_4 holds a linear equation of $\alpha_2\mathbf{p}_2 + \alpha_3\mathbf{p}_3 = \boldsymbol{X} - \alpha_1\mathbf{p}_1$, where the value of the RHS is known to r_4. Since r_4 still wants $\{\mathbf{p}_2, \mathbf{p}_3\}$, $\boldsymbol{e}_4$ is incident to $\{\boldsymbol{v}_2, \boldsymbol{v}_3\}$. The updated graph $\mathcal{H}''$ is plotted in Fig. 6.3(d).
3. A minimal vertex cover of $\mathcal{H}''$ is $\boldsymbol{v}_2$. Thus, $\mathbf{p}_2$ is sent alone, which will allow r_4 to obtain $\mathbf{p}_2$, and then decode $\mathbf{p}_3$ from equation $\alpha_2\mathbf{p}_2 + \alpha_3\mathbf{p}_3 = \boldsymbol{X} - \alpha_1\mathbf{p}_1$. The broadcast will then be completed.

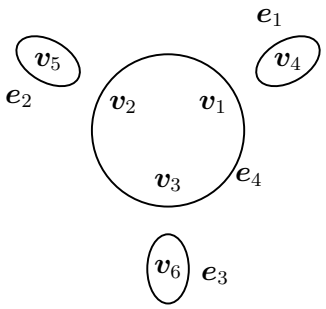
To complete the details of our technique, three questions need to be answered:

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	$\mathbf{p}_4$	$\mathbf{p}_5$	$\mathbf{p}_6$
$\mathbf{r}_1$	1	0	0	1	0	0
$\mathbf{r}_2$	0	1	0	0	1	0
$\mathbf{r}_3$	0	0	1	0	0	1
$\mathbf{r}_4$	1	1	1	0	0	0

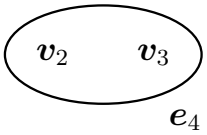
(a) SFM $\boldsymbol{A}$



(b) Original $\mathcal{H}$



(c) updated $\mathcal{H}'$ after sending $\alpha_1\mathbf{p}_1 + \alpha_2\mathbf{p}_2 + \alpha_3\mathbf{p}_3$



(d) updated $\mathcal{H}''$ after sending $\alpha'_1\mathbf{p}_1 + \alpha'_4\mathbf{p}_4 + \alpha'_5\mathbf{p}_5 + \alpha'_6\mathbf{p}_6$

Figure 6.3: **Hypergraph update**

Problem 6.1

1. How to find the minimal vertex covers?
2. How to choose the coding coefficients of the coded packets?
3. How and when to update the hypergraph?

The answer to the first question reveals an advantage of our technique: minimal vertex covers can be easily found using polynomial-time algorithms. We propose in Algorithm 6.1 such an algorithm, which aims at maximizing the number of uniquely covered hyperedges, so that the number of receivers who can instantly decode a data packet is large. Given $\mathcal{H}$, our algorithm finds $\mathcal{V}_c$ by adding to $\mathcal{V}_c$ the vertex in $\mathcal{H}$ with the largest degree (i.e., incident to the largest number of hyperedges), and then discarding this vertex and its incident hyperedges before finding the next such vertex. We note that our technique does not require finding the minimum (the smallest minimal) vertex cover, which is an NP-hard problem. Moreover, the minimum vertex cover does not necessarily maximize the number of uniquely covered hyperedges, and thus may not even be a desired solution to

Algorithm 6.1 Find a minimal hypergraph vertex cover

Input: A hypergraph $\mathcal{H}(\mathcal{V}, \mathcal{E})$, an empty vertex set $\mathcal{V}_c$;
Weigh each vertex with the number of hyperedges incident to it; (*The weight of a vertex is indeed the number of receivers which want the corresponding data packet.*)
while there are still vertices in $\mathcal{H}$ **do**
  Add to $\mathcal{V}_c$ the vertex with the largest weight;
  Update $\mathcal{H}$ by: 1) removing this vertex from $\mathcal{H}$; 2) removing from $\mathcal{H}$ all hyperedges incident to this vertex; 3) removing from $\mathcal{H}$ all vertices that do not have any hyperedge incident to them;
end while
Output $\mathcal{V}_c$ as a minimal hypergraph vertex cover.

our problem.

For the second question, any throughput-optimal coefficient selection technique is acceptable and is guaranteed to result instantly decodable packets for all receivers r_n such that $|\mathcal{V}_c \cap \boldsymbol{e}_n| = 1$. For the sake of consistency and simplicity, in this thesis we select them uniformly at random from a sufficiently large $\mathbb{F}_q$. In other words, we generate random-coded packets of the minimal vertex cover coding sets.

Therefore, our technique is a framework that generalizes throughput-optimal LNC techniques. The traditional RLNC technique is indeed an inefficient implementation of this framework in terms of APDD: it always chooses the set of all vertices $\mathcal{V}$ as the vertex cover. Since $\mathcal{V}$ is not necessarily minimal, the resultant coded packets do not guarantee instant packet decodings.

For the third question, hypergraph update is very simple when feedback is available after each coded transmission: each receiver r_n only needs to feed back which data packets it has decoded. Then, the sender updates $\mathcal{H}$ by simply removing the associated vertices from $\boldsymbol{e}_n$. In the absence of such feedback, the core problems are how often should feedback be collected, and how to update the hypergraph when feedback is not always available, which we will study in the next section.

6.4 Broadcast under Different Feedback Frequencies

In our technique, each receiver will send at least two rounds of feedback: 1) after the systematic transmission phase, and 2) after it has decoded all the data packets in $\mathcal{P}$. The first round of feedback is necessary to acquire $\mathcal{H}$ so that an *opportunistic* throughput-optimal LNC can be devised by the sender. Otherwise, the only guaranteed vertex cover is the set of all vertices $\mathcal{V}$, implying that the traditional RLNC technique must be applied to achieve optimal throughput. The second round of feedback is necessary for indicating block completion. In addition to the above two compulsory rounds of feedback, we also consider intermediate feedback during the coded transmission phase. Depending on the frequency of intermediate feedback, we consider three different transmission schemes:

1. Fully-online scheme, where feedback is collected after every coded transmission;
2. Semi-online scheme, where feedback is collected after every transmission *round*. Each round contains several coded transmissions (to be specified later);
3. Off-line scheme, where there is no intermediate feedback during the coded transmission phase.

We will not focus on the fully-online scheme, because hypergraph update is straightforward under this scheme, and this scheme evidently provides the best APDD performance among the three. However, it could be expensive or even impossible to collect feedback after every transmission. For example, in time-division-duplex systems the sender has to stop and listen to the feedback [84]. We are thus motivated to study the semi-online and off-line schemes. In particular, we will address the following two problems:

Problem 6.2

1. How to update the hypergraph in the semi- and off-line schemes when feedback is not always available?
2. How does the performance of semi- and off-line schemes compare with the fully-online scheme?

We start with presenting in Algorithm 6.2 our semi-online hypergraph updating algorithm. We will then prove that its performance is the same as the fully-online scheme, and then extend it to design the off-line scheme.

Algorithm 6.2 Hypergraph update in a semi-online round

1: input: a hypergraph $\mathcal{H}_u$ generated according to receiver feedback;
2: let a constant N_e be the number of hyperedges of $\mathcal{H}_u$;
3: **while** the number of hyperedges of $\mathcal{H}_u$ is still N_e **do**
4: find a minimal vertex cover of $\mathcal{H}_u$ using Algorithm 6.1;
5: generate a coded packet $\boldsymbol{X}_u$ using this minimal vertex cover and send;
6: let $\mathcal{H}_{u+1}$ be the updated hypergraph under the assumption that $\boldsymbol{X}_u$ is received by all receivers;
7: $u = u + 1$;
8: **end while**
9: the current semi-online round is completed. Collect feedback and start the next round.

According to Algorithm 6.2, a semi-online round starts from collecting one round of receiver feedback. After that, the sender iteratively sends a coded packet and *blindly* (i.e., without the help of receiver feedback) updates the hypergraph by assuming the coded packet is received by all receivers. This process stops when the blind update reduces the number of hyperedges.[1] In this case, the sender will collect receiver feedback to correctly update the hypergraph, and start the next semi-online round.

The purpose of stopping when the number of hyperedges is reduced is to avoid any loss on throughput optimality and degradation on APDD performance compared with the fully-online one. It is motivated by the following property of hypergraphs:

Property 6.3

If $\mathcal{H}_u$ is a subgraph of $\mathcal{H}'_u$ with the same number of hyperedges, then any minimal vertex cover $\mathcal{V}_c^u$ of $\mathcal{H}_u$ is also a minimal vertex cover of $\mathcal{H}'_u$. On the other hand, $\mathcal{V}_c^u$ may not be a vertex cover of $\mathcal{H}'_u$ when $\mathcal{H}_u$ is a subgraph of $\mathcal{H}'_u$ with less hyperedges.

Proof. It is intuitive that $\mathcal{V}_c^u$ is also a vector cover of $\mathcal{H}'_u$ when $\mathcal{H}_u$ and $\mathcal{H}'_u$ have the same number of hyperedges. We now prove that it is a *minimal* one of $\mathcal{H}'_u$: If it was not, then there exists a set $\mathcal{V}' \subset \mathcal{V}_c^u$ that is a minimal vertex cover of

[1] Note that the removal of a hyperedge means, in our context, a receiver is assumed by the sender to have fully decoded all its wanted packets.

$\mathcal{H}'_u$. Since $\mathcal{H}_u$ is a subgraph of $\mathcal{H}'_u$, $\mathcal{V}'$ is also a minimal vector cover of $\mathcal{H}_u$, which contradicts with the fact that its superset $\mathcal{V}_c^u$ is a minimal vector cover of $\mathcal{H}_u$. Hence, such $\mathcal{V}'$ does not exist and, thus, $\mathcal{V}_c^u$ is a minimal vector cover of $\mathcal{H}'_u$.

On the other hand, if $\mathcal{H}_u$ has less hyperedges than $\mathcal{H}'_u$, then the hyperedges that only exist in $\mathcal{H}'_u$ may not be covered by $\mathcal{V}_c^u$, indicating that $\mathcal{V}_c^u$ may not be a vertex cover of $\mathcal{H}'_u$. □

This property yields an important property of the minimal vertex covers found by the sender during a semi-online round:

Property 6.4

Let $\mathcal{H}_u$ be the hypergraph blindly updated by the sender using Algorithm 6.2 before the u-th transmission during a semi-online round. Let $\mathcal{H}'_u$ be the hypergraph representation of the actual packet reception state of the receivers before the u-th transmission. A minimal vertex cover $\mathcal{V}_c^u$ of $\mathcal{H}_u$ is also a minimal vertex cover of the actual hypergraph $\mathcal{H}'_u$.

Proof. Here we only need to prove that $\mathcal{H}_u$ is a subgraph of $\mathcal{H}'_u$ with the same number of hyperedges during a semi-online round.

Assume that feedback is collected before the u-th transmission. Denote by L the number of transmissions in the semi-online round starting from $\boldsymbol{X}_u$. We study the relation between two sets of hypergraphs:

- $\{\mathcal{H}_i\}_{i=u}^{u+L-1}$, the set of hypergraphs used by the sender to generate $\{\boldsymbol{X}_i\}_{i=u}^{u+L-1}$. Among them, $\{\mathcal{H}_i\}_{i=u+1}^{u+L-1}$ are generated using our proposed semi-online hypergraph update algorithm in Algorithm 6.2;
- $\{\mathcal{H}'_i\}_{i=u}^{u+L-1}$, the set of hypergraphs of the actual packet reception state before each of the i-th transmission. We note that $\mathcal{H}'_u = \mathcal{H}_u$ due to feedback.

For $i \in [u+1, u+L-1]$, $\mathcal{H}_i$ is always a subgraph of $\mathcal{H}'_i$. This is because every $\boldsymbol{e}_n \in \mathcal{H}_i$ represents what r_n wants after receiving $\{\boldsymbol{X}_j\}_{j=u}^{i-1}$, while every $\boldsymbol{e}'_n \in \mathcal{H}'_i$ represents what r_n actually wants after receiving only a subset of $\{\boldsymbol{X}_j\}_{j=u}^{i-1}$ due to packet erasures. Hence, $\boldsymbol{e}_n \subseteq \boldsymbol{e}'_n$, indicating that $\mathcal{H}_i \subseteq \mathcal{H}'_i$. Moreover, Algorithm 6.2 ensures that they have the same number of hyperedges. □

Property 6.4 indicates that in the semi-online scheme, the sender always finds a minimal vertex cover of the actual hypergraph, which is also the case in the fully-online scheme. Due to the heuristic nature of Algorithm 6.1, there is no

guaranteed winner between the minimal vertex covers found in the two schemes. Thus, the following performance of the semi-online scheme can be expected:

Proposition 6.2

Reducing feedback frequency from fully-online to semi-online using Algorithm 6.2 does not increase APDD or BCT.

Numerical results in Section 6.5 will show that the APDD performance of the two is indistinguishable.

We now remark how to request receiver feedback in the semi-online scheme.

Remark 6.1

We call L the length of a semi-online round. L can be pre-computed at the beginning of each round using Algorithm 6.2. With L calculated, the sender can easily control the time to collect feedback by, e.g., attaching a down-counter l (with an initial value of $l = L$) in the header of every coded packet to inform the receivers to send feedback after l time slots.
With a very small probability of $P_{e,n}^{L}$, all the L coded packets in the current round are lost at a receiver r_n, implying that the sender will not receive feedback from this receiver. In this case, the sender will keep sending random-coded packets of all data packets in $\mathcal{P}$ to maintain optimal throughput. The down-counter is set at $l = 0$, asking the receivers to immediately feed back. Once feedback from all receivers are collected, the sender can initiate the next semi-online round.

Based on the above results, we propose the following off-line LNC scheme to maintain optimal throughput while reducing APDD compared to traditional RLNC. This is at minimal cost of only one round of feedback after the systematic transmission phase compared to traditional RLNC.

According to Algorithm 6.3, after the systematic transmission phase, the sender in the off-line scheme will first apply one semi-online round. It will then apply the classic RLNC scheme to maintain throughput optimality without collecting intermediate feedback. Since packets sent during the semi-online round will provide instant packet decodings, the APDD performance of the proposed off-line scheme is better than applying the class RLNC scheme alone.

In conclusion, all the three proposed schemes are throughput-optimal and able to offer instant packet decodings. The fully- and semi-online ones share the same APDD performance, which is better than the off-line one. In the next section,

Algorithm 6.3 Off-line hypergraph update

1: input: the hypergraph $\mathcal{H}_0$ generated from the receiver feedback collected after the systematic transmission phase;
2: let a constant N_e be the number of hyperedges of $\mathcal{H}_0$. Let $u = 0$;
3: **while** the number of hyperedges of $\mathcal{H}_u$ is still N_e **do**
4: find a minimal vertex cover of $\mathcal{H}_u$ using Algorithm 6.1;
5: generate a coded packet $\boldsymbol{X}_u$ using this minimal vertex cover and send;
6: let $\mathcal{H}_{u+1}$ be the updated hypergraph under the assumption that $\boldsymbol{X}_u$ is received by all receivers;
7: $u = u + 1$;
8: **end while**
9: Keep sending random-coded packets of all packets in $\mathcal{P}$ until all receivers have fully decoded all packets in $\mathcal{P}$ and acknowledged.

we will verify our theorems through simulations.

6.5 Simulations

In this section, we will numerically compare the APDD performance of the proposed technique with some existing techniques, as well as the lower bound of APDD. In total, there are 6 different APDD we will compare. They are abbreviated and explained as follows:

1. "Fully-": the APDD of our technique when fully-online feedback is collected;
2. "Semi-": the APDD of our technique when semi-online feedback is collected;
3. "Off-": the APDD of our technique when no feedback is collected during the coded transmission phase;
4. "RLNC": the APDD of RLNC, operated under a sufficiently large finite field;
5. "$E[\underline{D}]$": the lower bound of APDD. In the simulations it is obtained by assuming that every received coded packet allows all receivers to decode one wanted data packet;
6. "G-IDNC": the APDD of a heuristic implementation of generalized in-

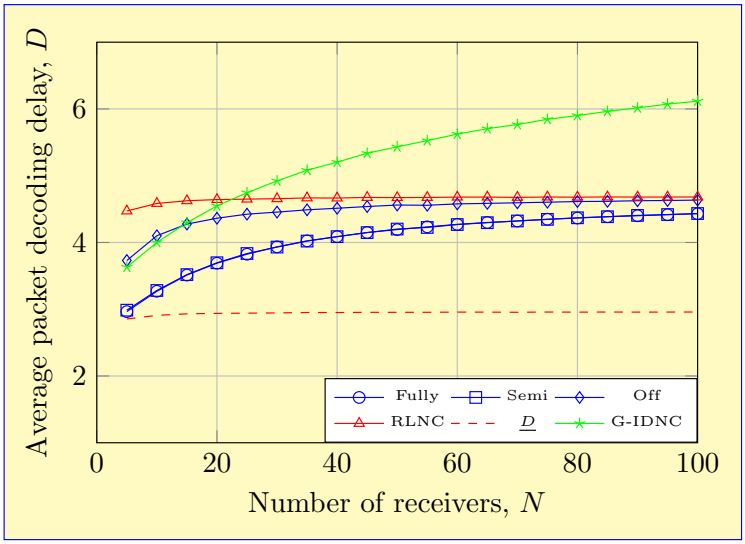


Figure 6.4: The APDD performance of different LNC techniques.

stantly decodable network coding technique [63] when fully-online feedback is collected. We note that there has not been any optimal G-IDNC algorithms in the literature.

In our simulations, there are $K = 15$ data packets, $N \in [5, 100]$ receivers. The packet erasure probabilities are $\{P_{e,n}\}_{n=1}^{N} = 0.2$. For each value of N, we simulate the broadcast of 10^5 packet blocks, and then make average on the APDD in their coded transmissions phase. The simulation results are plotted in Fig. 6.4, from which we observe that:

- The APDD performance of our technique outperforms the existing techniques. This superiority holds even when our technique is implemented under the off-line scheme, in which feedback is not collected during the coded transmission phase;
- The semi- and fully-online schemes share the same performance. This result matches Proposition 6.2. Their performance is better than the off-line one;
- The APDD of RLNC is always within a constant factor of the lower bound, indicating that RLNC is an approximation technique, and so is our technique. On the other hand, the APDD of the heuristic G-IDNC is unbounded, indicating that it is not an approximation technique. We further note that we cannot read the approximation ratio from the figure, because the approximation ratio is the ratio in the worst case scenario, which may not be reflected in the averaged result plot in the figure.

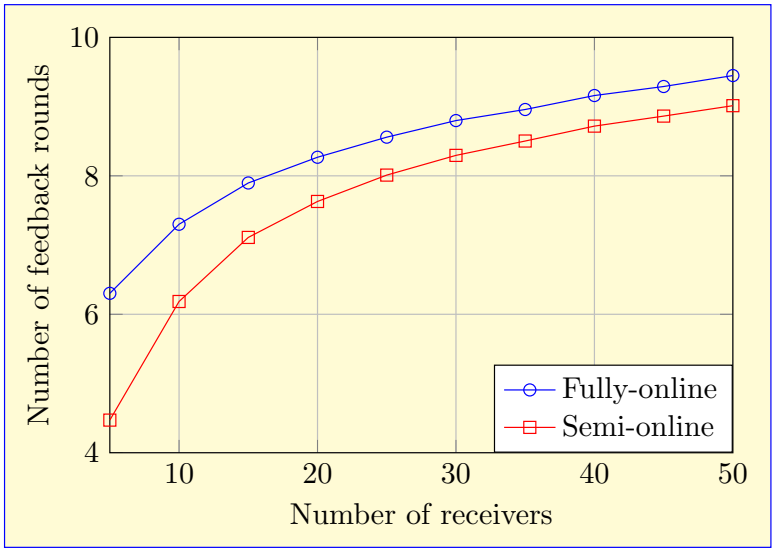


Figure 6.5: The Amount of feedback collected under the fully- and semi-online schemes.

In addition, we compare the amount of feedback collected under fully- and semi-online schemes and the results are plotted in Fig. 6.5. We observe that the semi-online scheme can reduce up to 30% feedback from the fully-online one when the number of receivers is small. The reduction becomes marginal with increasing number of receivers because the semi-online scheme approaches the fully-online one: with increasing number of receivers, the probability of having a receiver who only wants one data packet after a certain semi-online round increases. When this happens, the semi-online scheme has to collect feedback after only one transmission according to Algorithm 6.2, which makes it equivalent to the fully-online scheme.

6.6 Conclusion and Implication on Other Classes of LNC Techniques

In conclusion, by deriving bounds and presenting examples, we proved that all throughput-optimal LNC techniques are APDD-approximation techniques with a ratio of between $\frac{4}{3}$ and 2. Then based on a hypergraph model of receivers' knowledge space, we developed a new throughput-optimal and APDD-approximation LNC technique whose APDD is strictly better than RLNC. This technique is also implementation friendly, for it only uses a polynomial-time encoding algorithm, and does not require intensive receiver feedback.

Our results verify the APDD-approximation capability of S-IDNC claimed in Theorem 4.3 and Theorem 4.4, namely, S-IDNC is generally not an APDD-approximation technique unless there are at most three receivers. In this special case, S-IDNC is able to approximate (indeed achieve) the minimum APDD of LNC because it is throughput-optimal and provides instant packet decodings to all receivers in every transmission, making it a perfect LNC technique.

Our results also indicate that the generation-based LNC technique proposed in Chapter 5 is an APDD-approximation technique at least when there is no throughput loss due to partitioning. The relation between the approximation ratio and the generation rank is an interesting open question for future research.

Chapter 7

Conclusion and Future Work

In this thesis, we have conducted a comprehensive study on the throughput and APDD optimization of packet-level LNC techniques in wireless broadcast. Our study resulted in characterizing the fundamental performance limits of LNC techniques and the performance of three main classes of LNC techniques in the literature, including IDNC, generation-based, and throughput-optimal techniques. Our results indicate that throughput optimization and APDD optimization are not necessarily conflicting goals. Rather, the minimum APDD of LNC techniques can be approximately achieved by throughput-optimal LNC techniques with a guaranteed ratio. It is even possible to simultaneously optimize the two metrics using IDNC techniques under certain settings.

Our core theoretical contribution consists of revealing fundamental performance limits of LNC techniques. We have derived a necessary and sufficient conditions for LNC techniques to achieve the optimal throughput of wireless broadcast. We have proved the NP-hardness of APDD minimization, and have derived informative closed-form bounds of the expected APDD performance of LNC techniques under random packet erasures.

Our core practical contribution is a novel LNC technique that has many appealing properties: 1) it is throughput-optimal; 2) it approximates the minimum APDD of LNC techniques with a ratio of strictly smaller than 2; 3) it is computationally friendly: it only applies a polynomial-time encoding algorithm that does not require solving any NP-hard optimization problems; 4) it does not need intensive receiver feedback to make coding decisions. Consequently, the proposed technique is an outstanding technique both performance-wise and cost-wise. Indeed, our simulation results have showed that its APDD performance is better than many existing LNC techniques for wireless broadcast, including RLNC, un-

der all parameter settings.

We have also made contributions to S-IDNC and generation-based techniques. For S-IDNC, we have proved its limited APDD-approximation capability. We have developed optimal and heuristic coding algorithms that offer high packet multiplicity to fight against packet erasures. We have also found an equivalence between S-IDNC and G-IDNC techniques under certain settings. For generation-based techniques, we have established the optimal partitioning problem and proved its NP-hardness. We have developed a heuristic algorithm that achieves local Pareto-optimal throughput-delay tradeoffs. For both classes, we have designed transmission schemes that work efficiently under different feedback frequencies.

Our cross comparison between the three classes of LNC techniques indicates no clear winner, especially when practical concerns such as feedback frequency and computational complexity are taken into account. For example, although S-IDNC is generally sub-optimal in terms of throughput and APDD, its decoding complexity is the lowest among the three. Moreover, its APDD is much lower than RLNC when the number of receivers is not too large.

Interestingly, our generation-based technique actually achieves an rapprochement between S-IDNC and RLNC techniques. By simply changing the generation rank, our generation-based technique can be tuned into S-IDNC or RLNC techniques, and can provide a large range of performance tradeoffs between those of S-IDNC and RLNC. In other words, all the performance metrics considered in this thesis, including throughput, APDD, feedback frequency, and computational complexity, become tunable in the proposed generation-based technique.

This observation immediately motivates a new research topic on our generation-based technique. Recall that S-IDNC is generally not an APDD-approximation technique but RLNC is, an interesting problem is what is the relation between the generation rank and the approximation ratio. If this problem could be solved, we would be able to profile a complete spectrum of the achievable throughput-delay tradeoffs of LNC techniques.

There are also open problems in the other two classes. One of them is to identify the optimal G-IDNC technique, which could possibly be conquered with the help of our novel hypergraph model of receivers' knowledge space. Another one is to optimize the coding sets in the proposed throughput-optimal technique.

At a higher level, in the future we would like to extend our study to wireless multicast and index coding. We are also interested in extending our study to other applications of network coding such as distributed data storage.

Appendix A

Proof of Lemma 3.1

We prove that it is NP-complete to determine whether an $r-$uniforma hypergraph is size-r strong colorable or not, for any $r \geqslant 3$. Our method is a reduction from the k-coloring problem of graphs.

Given an arbitrary graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. For every edge $\boldsymbol{e}_n = (\boldsymbol{v}_i, \boldsymbol{v}_j)$ we construct a hyperedge $\boldsymbol{e}'_n$ by adding $r-2$ dummy vertices $\boldsymbol{v}^{n,1} \cdots v^{n,r-2}$ to $\boldsymbol{e}_n$. The result is an r-uniform hypergraph $\mathcal{H}(\mathcal{V}', \mathcal{E}')$ that has $|\mathcal{V}'| = |\mathcal{V}| + |\mathcal{E}| \cdot (r-2)$ vertices. If $\mathcal{G}$ can be colored using r colors, then in any hyperedge $\boldsymbol{e}'_n = (\boldsymbol{v}_i, \boldsymbol{v}_j, v^{n,1} \cdots v^{n,r-2})$, $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are colored differently using 2 colors. By assigning the remaining $r-2$ colors to the $r-2$ dummy vertices in $\boldsymbol{e}'_n$ greedily, all vertices in $\boldsymbol{e}'_n$ are colored differently. We thus obtain a size-r strong coloring of $\mathcal{H}$. On the other hand, if $\mathcal{H}$ can be strong colored using r colors, then by removing all the dummy vertices, we obtain an r-coloring of $\mathcal{G}$. It is well known that it is NP-complete to determine whether a graph is r colorable or not, for any $r \geqslant 3$. Hence, it is NP-complete to determine whether an r-uniform hypergraph is size-r strong colorable or not. □

Appendix B

Proof of Lemma 3.2

We first assume that r_n is satisfied after U_n coded transmissions (i.e., obtains all data packets). Let $\mathbf{u}_n$ be the set of time indices when r_n decodes a packet. Intuitively, $\mathbf{u}_n$ contains w_n indices, and U_n is a definite element of $\bullet_n$. We call $\mathbf{u}_n$ the packet reception pattern of r_n. It takes a form of $[u_1, \cdots, u_{w_n-1}, U_n]$.

Let $\mathbf{u}' = [u_1, \cdots, u_{w_n-1}]$, i.e., the set of variable indices. Then the APDD under a given $\mathbf{u}$ is $(\|\mathbf{u}'\| + U_n)/w_n$, where $\|\mathbf{u}'\|$ denotes the sum of all elements in $\mathbf{u}'$. Due to the symmetry of data packets and receivers, it is intuitive that all the possible $\mathbf{u}'$ happens with the same probability. Denote the set of all possible $\mathbf{u}'$ by $\mathcal{U}$, then the expected lower bound under given $\boldsymbol{\omega}_n$ and U_n, denoted by $E[\underline{D}_n|(U_n, w_n)]$, is calculated as:

$$\begin{aligned} E[\underline{D}_n|(U_n, w_n)] &= \frac{1}{|\mathcal{U}|} \sum_{\mathbf{u}' \in \mathcal{U}} \frac{\|\mathbf{u}'\| + U_n}{w_n} \\ &= \frac{U_n}{w_n} + \frac{1}{|\mathcal{U}|} \sum_{\mathbf{u}' \in \mathcal{U}} \frac{\|\mathbf{u}'\|}{w_n} \end{aligned} \tag{B.1}$$

We now show that all $\mathbf{u}'$ are axisymmetric about $U_n/2$. Given $\mathbf{u}' = [u_1, \cdots, u_{w_n-1}]$, by letting $u''_i = U_n - u_i$, the resultant $\mathbf{u}'' = [u''_1, \cdots, u''_{w-1}]$ is the mirror of $\mathbf{u}'$ against $U_n/2$. Obviously, $\mathbf{u}''$ also belongs to $\mathcal{U}$, and it holds that $\|\mathbf{u}'\| + \|\mathbf{u}''\| = (w_n - 1)U_n$. Hence, there are $|\mathcal{U}|/2$ such pairs, and thus the above equation can be simplified to:

$$E[\underline{D}_n|(U_n, w_n)] = \frac{U_n}{w_n} + \frac{1}{2}\frac{(w_n - 1)U_n}{w_n} = \frac{U_n}{2} + \frac{U_n}{2w_n} \tag{B.2}$$

Then, by noting that the expected number of coded transmissions for a receiver to be satisfied is $E[U_n|w_n] = w_n/(1 - P_e)$, we have:

$$E[\underline{D}_n|w_n] = \frac{E[U_n|w_n]}{2} + \frac{E[U_n|w_n]}{2w_n} = \frac{w_n + 1}{2(1 - P_e)} \tag{B.3}$$

which proves Lemma 3.2. ■

Appendix C

Proof of Theorem 4.5

Theorem 4.5 requires the proof of $\chi(\overline{\mathcal{G}}_s) = \chi(\overline{\mathcal{G}}_g)$. Since every S-IDNC solution is also a G-IDNC solution, but a G-IDNC solution is not necessarily a S-IDNC solution, we have $U_s \geqslant U_g$, and thus $\chi(\overline{\mathcal{G}}_s) \geqslant \chi(\overline{\mathcal{G}}_g)$. Hence, here we only need to prove that $\chi(\overline{\mathcal{G}}_s) \leqslant \chi(\overline{\mathcal{G}}_g)$.

We first introduce the concept of *affiliated* S-IDNC graph $\mathcal{G}_{as}$ of a G-IDNC graph $\mathcal{G}_g$, which is construct as follows. Given $\mathcal{G}_g$, which involves K data packets and N receivers, we generate a graph $\mathcal{G}_{as}$ with K vertices, each representing a data packet. We then connect $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ in $\mathcal{G}_{as}$ if for every pair of $\{m, n\} \in [1, N]$, $\boldsymbol{v}_{i,m}$ and $\boldsymbol{v}_{j,n}$ are connected upon their existence in $\mathcal{G}_g$. In other words, we claim that $\mathbf{p}_i$ and $\mathbf{p}_j$ do not conflict if every vertex that represents $\mathbf{p}_i$ in $\mathcal{G}_g$ is connected to every vertex that represents $\mathbf{p}_j$ in $\mathcal{G}_g$.

Given a SFM $\boldsymbol{A}$, we can easily show that its S-IDNC graph $\mathcal{G}_s$ is the same as the affiliated S-IDNC graph $\mathcal{G}_{as}$ of its G-IDNC graph $\mathcal{G}_g$. Hence, our task becomes to prove that $\chi(\overline{\mathcal{G}}_{as}) \leqslant \chi(\overline{\mathcal{G}}_g)$, where $\chi(\overline{\mathcal{G}}_{as}) = U_s$. This statement is true if the following property is true:

Property

After removing any clique $\mathcal{M}_g$ from $\mathcal{G}_g$, the chromatic number of the affiliated S-IDNC graph $\mathcal{G}_{as}$ is reduced by at most one.

Since $\mathcal{G}_g$ is nonempty as long as $\mathcal{G}_{as}$ is nonempty, this property indicates that any clique partition solution of $\mathcal{G}_g$ must have a size of at least $\chi(\overline{\mathcal{G}}_{as})$, which will prove that $\chi(\overline{\mathcal{G}}_{as}) \leqslant \chi(\overline{\mathcal{G}}_g)$. Property 6 can be proved through induction:

1. If $\mathcal{M}_g$ does not contain any conflicting data packets in $\mathcal{G}_{as}$, then $\chi(\overline{\mathcal{G}}_{as})$ is reduced by at most one;

2. If $\mathcal{M}_g$ contains one pair of conflicting data packets in $\mathcal{G}_{as}$, then $\chi(\overline{\mathcal{G}}_s)$ is reduced by at most one;

3. If $\mathcal{M}_g$ already contains m pairs of conflicting data packets in $\mathcal{G}_{as}$, then modifying $\mathcal{M}_g$ to contain one more pair of conflicting data packets in $\mathcal{G}_{as}$ cannot further reduce $\chi(\overline{\mathcal{G}}_{as})$.

The first statement is self-evident, because the set of data packets included in such $\mathcal{M}_g$ is a clique of $\mathcal{G}_{as}$. By removing it, $\chi(\overline{\mathcal{G}}_{as})$ can be reduced by at most one.

To prove the second statement, without loss of generality we assume that the pair of conflicting data packets is $(\mathbf{p}_1, \mathbf{p}_2)$. Then the set of data packets included in $\mathcal{M}_g$ takes a form of $\{\mathcal{M}_s, \mathbf{p}_1, \mathbf{p}_2\}$, where $\mathcal{M}_s$ is the set of pair-wise non-conflicting data packets, and thus is a clique of $\mathcal{G}_{as}$. Since $\mathbf{p}_1$ conflicts with $\mathbf{p}_2$, there exists at least one pair of unconnected vertices in $\mathcal{G}_g$ that represent $\mathbf{p}_1$ and $\mathbf{p}_2$. This pair is not included in $\mathcal{M}_g$, and thus is kept after removing $\mathcal{M}_g$ from $\mathcal{G}_g$. Hence, in the updated affiliated S-IDNC graph $\mathcal{G}'_{as}$, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ exist, and are unconnected. Let the chromatic number of $\mathcal{G}'_{as}$ be U', then the minimum clique partition of $\mathcal{G}'_{as}$ takes a form of $\{\mathcal{M}_1, \cdots, \mathcal{M}_{U'}\}$, which keeps $\mathbf{p}_1$ and $\mathbf{p}_2$ in different coding sets. Then, since $\mathcal{M}_s$ is a clique of $\mathcal{G}_{as}$, $\{\mathcal{M}_s, \mathcal{M}_1, \cdots, \mathcal{M}_{U'}\}$ is a partition of $\mathcal{G}_{as}$ with a size of $U' + 1$. Thus, $U' \geqslant \chi(\overline{\mathcal{G}}_{as}) - 1$, implying that $\chi(\overline{\mathcal{G}}_s)$ is reduced by at most one after removing $\mathcal{M}_g$ from $\mathcal{G}_g$.

The proof of the third statement is similar to the second one, and thus is omitted here. According to the above three statements, no matter how many conflicting data packets are included in $\mathcal{M}_g$, after removing $\mathcal{M}_g$ from $\mathcal{G}_g$, the chromatic number of the affiliated S-IDNC graph $\mathcal{G}_{as}$ is reduced by at most one. Therefore, $\chi(\overline{\mathcal{G}}_g) \geqslant \chi(\overline{\mathcal{G}}_{as})$. Since $\mathcal{G}_{as}$ is the same as $\mathcal{G}_s$, we have $\chi(\overline{\mathcal{G}}_g) \geqslant \chi(\overline{\mathcal{G}}_s)$ and Theorem 3 is proved.

Appendix D

An example of $U_g < U_s$

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	$\mathbf{p}_4$
r_1	1	0	0	1
r_2	1	1	1	0
r_3	0	0	1	1
r_4	0	1	0	1

(a) Original SFM

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	$\mathbf{p}_4$
r_1	1	0	0	0
r_2	1	0	1	0
r_3	0	0	1	0
r_4	0	1	0	1

(b) After sending $\mathbf{p}_2 \oplus \mathbf{p}_4$

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	$\mathbf{p}_4$
r_1	0	0	0	0
r_2	0	0	1	0
r_3	0	0	1	0
r_4	0	0	0	1

(c) After sending $\mathbf{p}_1 \oplus \mathbf{p}_2$

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	$\mathbf{p}_4$
r_1	0	0	0	0
r_2	0	0	0	0
r_3	0	0	0	0
r_4	0	0	0	0

(d) After sending $\mathbf{p}_3 \oplus \mathbf{p}_4$

Figure D.1: An example of the coded transmission phase

Consider the SFM in Fig. D.1(a). Since all the four data packet conflict with each other, SIDNC requires a minimum BCT of $U_s = 4$. However, if we violate the conflict between $\mathbf{p}_2$ and $\mathbf{p}_4$, i.e., send $\mathbf{p}_2 \oplus \mathbf{p}_4$, the conflicts between $\mathbf{p}_1$ and $\mathbf{p}_2$, and between $\mathbf{p}_3$ and $\mathbf{p}_4$, are solved, as there will be no receiver who jointly wants $\{\mathbf{p}_1, \mathbf{p}_2\}$ or $\{\mathbf{p}_3, \mathbf{p}_4\}$. Hence, we only need two more coded transmissions: $\mathbf{p}_1 \oplus \mathbf{p}_2$ and $\mathbf{p}_3 \oplus \mathbf{p}_4$. In total, GIDNC requires a minimum BCT of $U_g = 3$.

Appendix E

Proof of Theorem 3.4

Here we only need to prove that $E\left[\frac{\sum_{n=1}^{N} w_n^2}{\sum_{n=1}^{N} w_n}\right] \approx KP_e - P_e + 1$ when N is sufficiently large. We first expand $E\left[\frac{\sum_{n=1}^{N} w_n^2}{\sum_{n=1}^{N} w_n}\right]$ into its series form:

$$E\left[\frac{\sum_{n=1}^{N} w_n^2}{\sum_{n=1}^{N} w_n}\right] = E\left[\frac{w_1^2}{\sum_{n=1}^{N} w_n}\right] + E\left[\frac{w_2^2}{\sum_{n=1}^{N} w_n}\right] + \cdots + E\left[\frac{w_N^2}{\sum_{n=1}^{N} w_n}\right] \quad \text{(E.1)}$$

Since $\{w_n\}_{n=1}^{N}$ are i.i.d. distributed, the N addends in the above equation have the same value. Thus,

$$E\left[\frac{\sum_{n=1}^{N} w_n^2}{\sum_{n=1}^{N} w_n}\right] = N \cdot E\left[\frac{w_1^2}{\sum_{n=1}^{N} w_n}\right] \quad \text{(E.2)}$$

$$= N \cdot E\left[\frac{w_1^2}{w_1 + \sum_{n=2}^{N} w_n}\right] \quad \text{(E.3)}$$

Then according to the law of larger numbers, the value of $\sum_{n=2}^{N} w_n$ will approach to its mean $(N-1)KP_e$ when N is sufficiently large. Thus,

$$E\left[\frac{\sum_{n=1}^{N} w_n^2}{\sum_{n=1}^{N} w_n}\right] \approx N \cdot E\left[\frac{w_1^2}{w_1 + (N-1)KP_e)}\right], \quad \text{when } N \text{ is sufficiently large} \quad \text{(E.4)}$$

$$= E\left[\frac{w_1^2}{KP_e + \frac{w_1 - KP_e}{N}}\right] \quad \text{(E.5)}$$

$$\approx E\left[\frac{w_1^2}{KP_e}\right], \quad \text{when } N \text{ is sufficiently large} \quad \text{(E.6)}$$

$$
\begin{aligned}
& E\left[\frac{\sum_{n=1}^{N} w_n^2}{\sum_{n=1}^{N} w_n}\right] \\
=& N \cdot E\left[\frac{w_1^2}{w_1 + \sum_{n=2}^{N} w_n}\right] \\
=& N \cdot E\left[\frac{w^2}{w+v}\right], \qquad w \sim Bin\{K, P_e\}, \quad v \sim Bin\{NK-K, P_e\} \qquad \text{(E.7)} \\
=& N \cdot \sum_{v=1}^{NK-K} \sum_{w=1}^{K} \frac{w^2}{w+v} \binom{K}{w} \binom{NK-K}{v} P_e^{w+v} (1-P_e)^{NK-w-v} \qquad \text{(E.8)} \\
=& N \cdot \sum_{w=1}^{K} \frac{w}{w+c} \binom{K}{w} P_e^{w} (1-P_e)^{K-w} \qquad \text{(E.9)}
\end{aligned}
$$

Then, since $w_1 \sim B(K, P_e)$, we have $E[w_1] = KP_e$ and $Var[w_1] = KP_e - KP_e^2$. Hence, we have $E[w_1^2] = E[w_1]^2 + Var[w_1] = K^2P_e^2 + KP_e - KP_e^2$, and thus $E\left[\frac{\sum_{n=1}^{N} w_n^2}{\sum_{n=1}^{N} w_n}\right] \approx KP_e - P_e + 1$. ■

Bibliography

[1] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 4, pp. 303–314, 1998.

[2] J. W. Roberts, "Internet traffic, QoS, and pricing," *Proceedings of the IEEE*, vol. 92, no. 9, pp. 1389–1399, 2004.

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

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

[5] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, 2003.

[6] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[7] M. Sipser, *Introduction to the Theory of Computation, 2nd Edition*. Thomson Course Technology, 2006.

[8] R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of linear coding in network information flow," *IEEE Trans. Inf. Theory*, vol. 51, no. 8, pp. 2745–2759, 2005.

[9] ——, "Network coding and matroid theory," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 388–405, 2011.

[10] J. G. Oxley, *Matroid theory*. Oxford University Press, 2006, vol. 3.

[11] S. El Rouayheb, A. Sprintson, and C. Georghiades, "On the index coding problem and its relation to network coding and matroid theory," *Information Theory, IEEE Transactions on*, vol. 56, no. 7, pp. 3187–3195, 2010.

[12] S. Y. El Rouayheb, M. A. R. Chaudhry, and A. Sprintson, "On the minimum number of transmissions in single-hop wireless coding networks," in *Proc. IEEE Information Theory Workshop (ITW)*, 2007.

[13] M. A. R. Chaudhry and A. Sprintson, "Efficient algorithms for index coding," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM) Workshops*, 2008, pp. 1–4.

[14] Z. Bar-Yossef, Y. Birk, T. Jayram, and T. Kol, "Index coding with side information," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1479–1494, Mar. 2011.

[15] S. Brahma and C. Fragouli, "Pliable index coding: The multiple requests case," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, 2013, pp. 1142–1146.

[16] T. Ho and H. Viswanathan, "Dynamic algorithms for multicast with intra-session network coding," *IEEE Trans. Inf. Theory*, vol. 55, no. 2, pp. 797–815, 2009.

[17] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: practical wireless network coding," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 497–510, 2008.

[18] C. Fragouli, D. Lun, M. Médard, and P. Pakzad, "On feedback for network coding," in *Proc. Annual Conf. Inf. Sci. and Syst. (CISS)*, 2007, pp. 248–252.

[19] J. K. Sundararajan, D. Shah, and M. Médard, "ARQ for network coding," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, 2008.

[20] R. Costa, D. Munaretto, J. Widmer, and J. Barros, "Informed network coding for minimum decoding delay," in *Proc. IEEE Int. Conf. Mobile Ad Hoc and Sensor Syst., (MASS)*, 2008, pp. 80–91.

[21] L. Keller, E. Drinea, and C. Fragouli, "Online broadcasting with network coding," in *in Proc. IEEE Workshop on Network Coding (NetCod)*, 2008.

[22] J. K. Sundararajan, D. Shah, and M. Médard, "Online network coding for optimal throughput and delay – The three-receiver case," in *Proc. IEEE Int. Symp. Information Theory and Its Applications (ISITA)*, 2008, pp. 1–6.

[23] S. Sorour and S. Valaee, "Completion delay reduction in lossy feedback scenarios for instantly decodable network coding," in *Proc. IEEE Int. Symp. Personal Indoor and Mobile Radio Communications (PIMRC)*, 2011, pp. 2025–2029.

[24] ——, "Completion delay minimization for instantly decodable network coding with limited feedback," in *Proc. IEEE ICC*, 2011.

[25] A. Eryilmaz, A. Ozdaglar, M. Médard, and E. Ahmed, "On the delay and throughput gains of coding in unreliable networks," *IEEE Trans. Inf. Theory*, vol. 54, no. 12, pp. 5511–5524, Dec. 2008.

[26] X. Li, C.-C. Wang, and X. Lin, "On the capacity of immediately-decodable coding schemes for wireless stored-video broadcast with hard deadline constraints," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 5, pp. 1094–1105, 2011.

[27] H. Seferoglu and A. Markopoulou, "Video-aware opportunistic network coding over wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 5, pp. 713–728, 2009.

[28] L. Yang, Y. E. Sagduyu, and J. H. Li, "Adaptive network coding for scheduling real-time traffic with hard deadlines," in *Proc. ACM Int. Symp. Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2012, pp. 105–114.

[29] J. Heide, M. V. Pedersen, F. H. P. Fitzek, and T. Larsen, "Network coding for mobile devices - systematic binary random rateless codes," in *Proc. IEEE Int. Conf. Communications (ICC) workshop*, June 2009.

[30] M. Xiao, T. Aulin, and M. Médard, "Systematic binary deterministic rateless codes," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, 2008, pp. 2066–2070.

[31] S. Huang, E. Izquierdo, and P. Hao, "Adaptive packet scheduling for scalable video streaming with network coding," in *Proc. IEEE. Int. Conf. Communications (ICC)*, 2016.

[32] C. Gkantsidis and M. Goldberg, "Avalanche: File swarming with network coding," in *Microsoft Research*, 2005.

[33] R. W. Yeung, "Avalanche: A network coding analysis," *Communications in Information & Systems,* International Press of Boston, vol. 7, no. 4, pp. 353–358, 2007.

[34] Z. Liu, C. Wu, B. Li, and S. Zhao, "UUSee: Large-scale operational on-demand streaming with random network coding," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[35] X. Li, W. H. Mow, and F.-L. Tsang, "Rank distribution analysis for sparse random linear network coding," in *Proc. IEEE Int. Symp. Network Coding (NetCod)*, 2011.

[36] S. Yang and R. W. Yeung, "Batched sparse codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 9, pp. 5322–5346, 2014.

[37] P. Garrido, G. David, and J. Lanza, "Exploiting sparse coding: A sliding window enhancement of a random linear network coding schem," in *Proc. IEEE. Int. Conf. Communications (ICC)*, 2016.

[38] A. Tassi, I. Chatzigeorgiou, and D. E. Lucani, "Analysis and optimization of sparse random linear network coding for reliable multicast services," *IEEE Trans. Commun.*, vol. 64, no. 1, pp. 285 – 259, 2016.

[39] H. Y. Kwan, K. W. Shum, and C. W. Sung, "Generation of innovative and sparse encoding vectors for broadcast systems with feedback," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, 2011, pp. 1161–1165.

[40] P. Sadeghi, R. Shams, and D. Traskov, "An optimal adaptive network coding scheme for minimizing decoding delay in broadcast erasure channels," *EURASIP J. on Wireless Commun. and Netw.*, pp. 1–14, Jan. 2010.

[41] A. Le, A. S. Tehrani, A. G. Dimakis, and A. Markopoulou, "Instantly decodable network codes for real-time applications," in *Proc. IEEE Int. Symp. Network Coding (NetCod)*, 2013.

[42] S. Sorour and S. Valaee, "Minimum broadcast decoding delay for generalized instantly decodable network coding," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, 2010.

[43] E. Rozner, A. P. Iyer, Y. Mehta, L. Qiu, and M. Jafry, "ER: Efficient retransmission scheme for wireless LANs," in *Proc. ACM Int. Conf. Emerging Networking Experiments and Technologies (CoNEXT)*, 2007.

[44] S. Sorour and S. Valaee, "Completion delay minimization for instantly decodable network codes," *IEEE/ACM Trans. Networking*, vol. 23, no. 5, pp. 1553–1567, 2015.

[45] M. S. Karim, P. Sadeghi, S. Sorour, and N. Aboutorab, "Instantly decodable network coding for real-time scalable video broadcast over wireless networks," *Euro. Journ. Advances in Signal Processing*, 2016.

[46] N. Aboutorab and P. Sadeghi, "Instantly decodable network coding for completion time or decoding delay reduction in cooperative data exchange systems," *IEEE Trans. Veh. Technol.*, vol. 65, no. 3, pp. 1212–1228, 2016.

[47] N. Aboutorab, S. Sorour, and P. Sadeghi, "O2-GIDNC: Beyond instantly decodable network coding," in *Proc. IEEE Int. Symp. Network Coding (NetCod)*, 2013, pp. 1–6.

[48] P. Maymounkov, N. J. Harvey, and D. S. Lun, "Methods for efficient network coding," in *Proc. 44-th Allerton Conference*, 2006.

[49] D. Silva, W. Zeng, and F. R. Kschischang, "Sparse network coding with overlapping classes," in *Proc. 5th Workshop on Network Coding, Theory, and Applications (NetCod)*, 2009.

[50] A. Heidarzadeh and A. H. Banihashemi, "Overlapped chunked network coding," in *Proc. IEEE Information Theory Workshop (ITW)*, 2010.

[51] Y. Li, P. Vingelmann, M. V. Pedersen, and E. Soljanin, "Round-robin streaming with generations," in *Proc. IEEE Int. Symp. Network Coding (NetCod)*, 2012.

[52] G. Joshi and E. Soljanin, "Round-robin overlapping generations coding for fast content download," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, 2013.

[53] E. Skevakis and I. Lambadaris, "Decoding and file transfer delay balancing in network coding broadcast," in *Proc. IEEE. Int. Conf. Communications (ICC)*, 2016.

[54] D. J. MacKay, "Fountain codes," *IEE Proceedings-Communications*, vol. 152, no. 6, pp. 1062–1068, 2005.

[55] M. Luby, "Lt codes," in *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, 2002, pp. 271–271.

[56] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[57] J. Barros, R. A. Costa, D. Munaretto, and J. Widmer, "Effective delay control in online network coding," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2009, pp. 208–216.

[58] S. Sorour and S. Valaee, "Minimum broadcast decoding delay for generalized instantly decodable network coding," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2010.

[59] M. Nistor, D. E. Lucani, T. T. Vinhoza, R. A. Costa, and J. Barros, "On the delay distribution of random linear network coding," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 5, pp. 1084–1093, 2011.

[60] J. Oxley, *Matroid Theory*. Oxford University Press, 1992.

[61] S. Ball, "On sets of vectors of a finite vector space in which every subset of basis size is a basis," *Journal of the European Mathematical Society (EMS)*, vol. 14, no. 3, pp. 733–748, 2012.

[62] P. Sadeghi, D. Traskov, and R. Koetter, "Adaptive network coding for broadcast channels," in *Proc. 5th Workshop on Network Coding, Theory, and Applications (NetCod)*, 2009, pp. 80–85.

[63] S. Sorour and S. Valaee, "On minimizing broadcast completion delay for instantly decodable network coding," in *Proc. IEEE. Int. Conf. Communications (ICC)*, 2010.

[64] ——, "Coding opportunity densification strategies for instantly decodable network coding," *IEEE Trans. Commun.*, vol. 61, no. 12, pp. 5077–5089, 2013.

[65] N. Aboutorab, P. Sadeghi, and S. Sorour, "Enabling a tradeoff between completion time and decoding delay in instantly decodable network coded systems," *IEEE Trans. Commun.*, vol. 62, no. 4, pp. 1269–1309, 2014.

[66] N. Aboutorab, P. Sadeghi, and S. E. Tajbakhsh, "Instantly decodable network coding for delay reduction in cooperative data exchange systems," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, 2013, pp. 3095–3099.

[67] M. S. Karim, N. Aboutorab, A. A. Nasir, and P. Sadeghi, "Decoding delay reduction in network coded cooperative systems with intermittent status update," in *Proc. IEEE Information Theory Workshop (ITW)*, 2014, pp. 391–395.

[68] Y. Keshtkarjahromi, H. Seferoglu, R. Ansari, and A. Khokhar, "Content-aware instantly decodable network coding over wireless networks," in *Proc IEEE Int. Conf. Computing, Networking and Communications (ICNC)*, 2015, pp. 803–809.

[69] J. M. Harris, J. L. Hirst, and M. J. Mossinghoff, *Combinatorics and Graph Theory, 2nd Edition*. Springer Press, 2008.

[70] M. Kubale, *Graph Coloring*, ser. Contemporary Mathematics. American Mathematical Society, 2004.

[71] D. P. Gelle, "Problem 5713," *American Mathematical Monthly*, vol. 77, p. 85, 1970.

[72] W. Xiao and D. Starobinski, "Extreme value FEC for reliable broadcasting in wireless networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2009.

[73] M. Ghaderi, D. Towsley, and J. Kurose, "Reliability gain of network coding in lossy wireless networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2008.

[74] B. Swapna, A. Eryilmaz, and N. B. Shroff, "Throughput-delay analysis of random linear network coding for wireless broadcasting," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6328–6341, 2013.

[75] B. Ballobás, *Random Graphs*. Cambridge University Press, 2001.

[76] B. Bollobás, "The chromatic number of random graphs," *Combinatorica*, vol. 8, no. 1, pp. 49–55, 1988.

[77] C. Bron and J. Kerbosch, "Algorithm 457: Finding all cliques of an undirected graph," *Commun. of the ACM*, vol. 16, Sep. 1973.

[78] A. Rezaee, L. Zeger, and M. Médard, "Speeding multicast by acknowledgment reduction technique (SMART)," in *Proc. IEEE Global Telecomm. Conf. (GLOBECOM)*, 2011.

[79] M. Krivelevich and B. Sudakov, "The chromatic numbers of random hypergraphs," *Random Structures and Algorithms*, vol. 12, no. 4, pp. 381–403, 1998.

[80] ——, "Approximate coloring of uniform hypergraphs," *Journal of Algorithms*, vol. 49, no. 1, pp. 2–12, 2003.

[81] Lovàsz, "Coverings and colorings of hypergraphs," in *Proc. 4th S.E. Conf. on Combinatiroics, Graph Theory and Computing*, 1973.

[82] G. Agnarsson and M. M. Halldórsson, "Strong colorings of hypergraphs," in *Approximation and Online Algorithms*. Springer, 2005, pp. 253–266.

[83] M. Yu, P. Sadeghi, and N. Aboutorab, "From instantly decodable to random linear network coding," *IEEE Trans. Commun.*, vol. 62, no. 11, pp. 3943–3955, Oct. 2014.

[84] D. E. Lucani, M. Médard, and M. Stojanovic, "Random linear network coding for time-division duplexing: field size considerations," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, 2009.