Efficient Network Coding for Different Network Topologies

by

Ye Li

A thesis submitted to the
Department of Electrical and Computer Engineering
in conformity with the requirements for
the degree of Doctor of Philosophy

Queen's University
Kingston, Ontario, Canada
October 2014

Abstract

Network coding is known to be able to improve a network's throughput and resilience to packet losses. However, in practice high decoding complexity has been a major obstacle that prevents network coding from being widely employed. This thesis considers efficient network coding methods that have low complexity for different types of scenarios.

Simple network topologies are first considered. We propose a systematic network coding approach for networks that consist of one or two two-hop lossy paths. It is proved that, compared to the random linear network coding approach, the proposed scheme requires much less computation in encoding and decoding and also achieves a higher end-to-end rate. It is demonstrated that a judicious use of uncoded packets in these networks may provide considerable gain over random coding.

For networks with complex or changing topologies, we investigate the generation-based strategy to design network codes, in which source packets are grouped into subsets called generations and coding is only performed among packets belonging to the same generation. We design generation-based network codes for two different decoders that may be used in different scenarios to meet specific performance/complexity constraints. The first approach, which is generation-by-generation decoding, decodes

within generations and therefore its complexity can be made linear by properly choosing the generation size. We analyze the decoding process and propose 1) a method to estimate the optimal amount of overlap among generations, which is a key parameter of generation-based network codes and 2) a new code based on unequal-size generations. The proposed code is shown to outperform existing codes in terms of both computational cost and reception overhead. The second approach considers overhead-optimized decoding, which decodes generations jointly and has lower overhead than the generation-by-generation decoder but requires more computation. We provide an overhead-optimized decoder design with low computational cost and then design codes that are sparser than existing codes. The proposed code is shown to have lower overhead and decoding cost.

Acknowledgments

I would like to thank my supervisors, Prof. Steven D. Blostein and Prof. Wai-Yip G. Chan, for their invaluable patience, guidance, encouragement, and help during my PhD study.

I would like to thank all my thesis committee members, Prof. Amir Banihashemi from Carleton University, Prof. Fady Alajaji, Prof. Il-Min Kim, Prof. Evelyn Morin and Prof. James McLellan for their time reading my thesis and for their comments and suggestions which lead to the clarity of the thesis.

I would like to thank my parents for their continuous support and understanding; and my wife, Weiwei Yang, without whose support and encouragement the life in Kingston would be tough.

I thank all my lab-mates, past and present, for their friendship. The PhD program was enjoyable with the good time we had together. They include, but not limited to, Ali Bakhshali, Majid Bavand, Yu Cao, James Falt, Jason Liang, Phillip Oni, Shen Shen, Cong Wang, and Hongfei Wang.

Contents

List of Tables

List of Figures

Acronyms

AP Access Point

ARQ Automatic Repeat reQuest

BEC Binary Erasure Channel

BS Base Station

EV Encoding Vector

FEC Forward Error Correction

G-by-G Generation-by-Generation

GDM Global Decoding Matrix

GE Gaussian Elimination

GEV Generation Encoding Vector

GNC Generation-based Network Coding

GW GateWay

H2T Head-to-Toe

ILP Integer Linear Programming

LDM Local Decoding Matrix

LDPC Low-Density Parity-Check

LP Linear Programming

LT Luby-Transform

MaLPI Maximum Local Potential Innovativeness

ML Maximum-Likelihood

MPTCP Multi-path TCP

OA Overlap-Aware

PB-RAC Precoded Binary Random Annex Code

RAC Random Annex Code

RLNC Random Linear Network Coding

S2HNC Systematic 2-Hop Network Coding

UE User Equipment

uRAC Unequal-size Random Annex Code

List of Important Symbols

B Size of base part of random annex codes

G (Average) generation size in GNC codes

K Number of symbols of a packet

L, L' Number of generations in GNC codes

M Number of source packets

M_A Number of active columns in OA decoding.

M_D Number of solo packets in OA decoding.

M_I Number of inactivated columns in OA decoding

M_O Number of overlapping packets in OA decoding.

S Number of parity-check packets in precoded GNC codes

$\Omega(x)$ Generation-size distribution of GNC codes

δ Non-recoverable fraction of packet of generation-by-generation decoding

ϵ, p_e Erasure probabilities

$\frac{1}{1+\theta}$ Precode rate in GNC codes

$\lambda(x)$ Left-side edge degree distribution in GNC codes

$\mathbf{c}_i$ The i-th parity-check packet generated by precoding

$\mathbf{s}_i$ The i-th source packet.

$\mathcal{G}$ Set of generations

$\mathcal{G}_l$ The l-th generation of a GNC code, consisting of a subset of source packets

$\mathcal{S}$ Set of source packets

$\rho(x)$ Right-side edge degree distribution in GNC codes

ε Reception overhead

Chapter 1

Introduction

1.1 Motivation

Network transmissions play an important role in our lives. A variety of networks are in widespread use which include large ones such as the Internet and cellular networks or much smaller ones such as sensor networks. Reliable transmission of information at a high rate in these networks is always of a central interest. To achieve this, techniques such as automatic repeat request (ARQ), forward error correction (FEC) and hybrid ARQ which is a combination of ARQ and FEC are heavily used in existing systems.

Network coding [2], a technique that has been developing since around 2000, is an emerging approach for improving network throughput. Network coding extends traditional network operations from routing and store-and-forward to more powerful operations that allow for coding information at intermediate nodes. In theory, network coding achieves the max-flow capacity of multicast networks, which is the upper bound of the rate at which information can be send from one source node to one or more destination nodes through a lossless or lossy network.

Other than increased rate, network coding also brings other benefits such as robustness and adapatability against packet losses or link failures. In particular, when using random linear network coding (RLNC) [22], which is a distributed network coding approach achieving the max-flow capacity where information packets transmitting in the network are random linear combinations of the original source packets, network coding brings a useful property that coded packets can be made to be "equally important" and therefore destination nodes only need to receive a sufficient number of coded packets to decode, regardless of which packets are received. This property makes RLNC a *rateless code* in networks where packet losses may happen. The rateless code term refers to a class of erasure codes with the property that a potentially unlimited number of coded packets can be generated from a given set of source packets such that the original source packets can be successfully recovered from any subset of the coded packets of size equal to or only slightly larger than the number of source packets. The concept of rateless coding (also known the digital fountain property) is first proposed in [5]. LT codes [33] and Raptor codes [56] are examples of high performance rateless codes for point-to-point channels. Compared to ARQ, FEC and hybrid FEC, RLNC as a rateless code is able to achieve the max-flow capacity of a lossy network without the need of acknowledgements between nodes (indicating whether a packet is received or not) and estimation of packet loss rates of links.

Given the benefits, the application of network coding has drawn much attention over the past decade. For example, Microsoft has announced a prototype application called *Avalanche* for large file distribution on peer-to-peer networks based on RLNC [14]. However, one major problem that has been preventing network coding from being widely employed in practice is its complexity. When using RLNC, the decoder

has to solve a dense system of linear equations, resulting in cubic complexity using Gaussian elimination. The computational cost would be prohibitively high if network coding is performed across a large number of packets. The problem is severe if network coding is used in networks where destination nodes are mobile devices that have limited computing capability or battery power, e.g. cellphones.

In order to apply network coding in practice, it is important to devise efficient coding methods that possess the benefits of network coding but reduce the expensive computational coding and decoding cost. This is the theme of the thesis. The thesis shows that in many networks the benefits of network coding can be utilized at a low computational cost through careful code design.

1.2 Thesis Overview

In the thesis, low-complexity network coding is addressed for information dissemination in several network scenarios. Throughout the work, lossy networks are considered where links are modeled as memoryless binary erasure channels (BEC). The information unit is a *packet*, where each contains a number of information symbols from a finite field $\mathbb{F}_q$ of size q. Network coding is performed across packets. BEC refers to channels where packets are either received by nodes successfully and correctly or else lost.

The key idea of the work is to design *sparse* linear network codes, in which coded packets are in the forms of linear combinations of a small subset (rather than all) of the source packets. The thesis is organized as follows:

In Chapter 2, the background of network coding is first reviewed. A brief literature review on key theoretic and experimental results in the area is presented. Existing

strategies of sparse code designs are reviewed.

In Chapter 3, a systematic network coding method for two-hop lossy links is presented. The method sends a mixture of uncoded and coded packets to reduce the need of coding at the source and relay nodes. The method is shown to achieve a rate no smaller than RLNC at a much lower computational cost in this network. A Markov chain based technique is provided to calculate the expected completion time of the transmission in the finite regime.

In Chapter 4, the systematic network coding method in Chapter 3 is extended to a network which consists of parallel two-hop lossy links. This network topology is useful in modeling many transmission scenarios in practice, for example, WiFi offloading mode of cellular networks. The method is again shown to outperform RLNC in terms of both rate and computational cost. A packet allocation problem is formulated to allocate uncoded packets transmissions among paths such that the number of uncoded packets that can be received by the destination is maximized; the computational cost is then minimized. It is also shown that maximizing the number of received uncoded packets also results in a maximized rate.

From Chapter 5, we shift our focus from networks with simple and known topologies to networks with complex or unknown topologies. In these networks, the design of sparse network codes is more challenging because it is difficult to precisely define the behavior of network nodes to forward sparse coded packets as proposed in Chapter 3 and 4. Therefore, we turn to a generation-based strategy in which source packets are grouped into subsets called *generations* whereby network coding is only allowed to be performed among packets belonging to the same generation. This strategy ensures that coded packets received by the destination are always sparse even though

intermediate nodes may not be aware of the network topology. However, other issues are introduced such as the scheduling of generations and the problem of reception *overhead* due to the sparseness of codes and/or non-optimal scheduling, where the reception overhead is defined as the extra received packets required beyond the number of source packets to decode all the source data.

In Chapter 5, we design generation-based network codes (GNC codes) for the generation-by-generation (G-by-G) decoder, in which source packets are decoded from each generation *separately* until all generations are decoded. The *GNC code* specifies how generations are constructed from the given source packets and how coded packets are generated from each generation. The G-by-G decoder on GNC codes has decoding complexity upper bounded by the largest generation size and therefore the overall decoding complexity can be made linear if generation sizes are properly chosen. The decoder has low decoding cost even if the number of source packets is very large. We first propose in this chapter a new scheduling strategy to reduce the overhead that may be caused by scheduling at intermediate nodes. We then analyze the G-by-G decoding of GNC using a modified and-or tree analysis technique, based on which we design GNC codes with reduced overhead.

Chapter 6 considers the design of GNC codes for the overhead-optimized decoding. The overhead-optimized decoding decodes generations jointly rather than separately as in the case of G-by-G decoder. Given a GNC code, overhead-optimized decoding has the minimum overhead among all possible decoders. However, the decoding cost is increased. The aim of Chapter 6 is to design a low-complexity GNC decoder for scenarios where the number of source packets may be moderate rather than asymptotically large, based on which GNC codes with very low overhead can be designed. It

turns out that, by exploiting the structure and sparseness among GNC coded packets, we can achieve overhead-optimized decoding of GNC codes with low decoding cost, while achieving reduced overhead. This property is of interest in applications where low overhead is a strict requirement, for example where network services are charged by the amount of transmitted data.

1.3 Contributions

The main contributions of the thesis are briefly summarized as follows:

- A systematic network coding method for two-hop lossy links that achieves end-to-end rates no smaller than RLNC at a much reduced computational cost.
- An accurate completion time analysis of the above systematic coding method based on Markov chain model is provided.
- An extension of the systematic network coding method to parallel path two-hop lossy links is proposed and an uncoded packet allocation problem is solved and shown to provide optimal end-to-end rates and decoding costs.
- A general design method for generation-based network coding using the G-by-G decoder is developed based on a modification to the and-or tree analysis technique.
- The optimal amount of overlap among generations for GNC codes using G-by-G decoder is analytically determined using the above design method.
- A new GNC code with unequal-size generations is proposed and is demonstrated to have both lower overhead and G-by-G decoding cost compared to existing

GNC codes based on equal-size generations.

- A low-complexity overhead-optimized GNC decoder which has optimal overhead is designed and shown to have low decoding cost.

- A new GNC code based on existing techniques is designed for the above overhead-optimized decoder and is shown to have lower code overhead and sparser structure than existing codes.

Chapter 2

Background

2.1 Basics of Network Coding

The literature on network coding dates back to the 1990s. In their seminal work [2], Ahlswede et al. show that the achievable rate of multicasting information from one source node to multiple destination nodes through a network is bounded by the *max-flow capacity* of the network. The results utilize the max-flow min-cut theorem [27] where a directed graph is used to model the network and the flow is the information to be multicast. It is shown that the max-flow capacity can be achieved by allowing intermediate nodes of the network to perform coding on the information; this intermediary coding operation is referred to as *network coding*. Later in [29] and [26], it is shown that the max-flow capacity can be achieved by linear network codes, in which encoding and decoding are based on linear operations on information packets. In [23], a deterministic algorithm is provided to find a valid linear network code for a directed acyclic network in polynomial-time.

The benefits of network coding can be illustrated in the well-known butterfly network shown in Figure 2.1 (given by Ahlswede et al. in [2]). In this example, two

packets **a** and **b** are sent from S to E and F; each link of the network is lossless with capacity one packet per unit time. The max-flow capacity of the network is 2 packets per unit time. On the link C-D, forwarding either **a** or **b** from C would not allow E and F to receive both **a** and **b**. Instead, if network coding is performed at C by sending $\mathbf{a} + \mathbf{b}$ on C-D, E and F can simultaneously receive the two packets: E will receive **a** and $\mathbf{a} + \mathbf{b}$ while F will receive **b** and $\mathbf{a} + \mathbf{b}$.

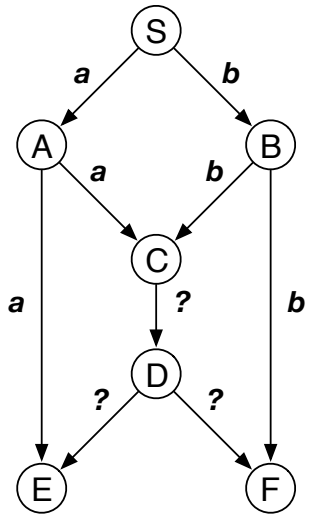

Figure 2.1: The butterfly network example where network coding achieves the max-flow capacity.

In [22], Ho et al. prove that it is sufficient to use random linear network coding (RLNC) to achieve the max-flow capacity of multicasting. In RLNC, coded packets are random linear combinations of source packets throughout the network, and the max-flow capacity can be asymptotically achieved if the finite field from which the coding coefficients of the linear combinations are chosen is sufficiently large. The idea of RLNC opens the door for the application of network coding in practice because it allows network coding to be applied in a distributed manner and in networks whose

topologies may be unknown.

Suppose that M source packets $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_M$ are sent from a source node to one or more destination nodes; each packet consists of K symbols from a finite field $\mathbb{F}_q$ of size q. The received coded packets at a destination node of a network where RLNC is used are in the forms of

$$\mathbf{r} = \sum_{i=1}^{M} g_i \mathbf{s}_i, \tag{2.1}$$

where g_i's are the encoding coefficients that may be chosen uniformly randomly from $\mathbb{F}_q$. The vector $[g_1, g_2, \ldots, g_M]$ is referred to as the *coding vector* of the coded packet $\mathbf{r}$. In [8], it is proposed that the coding vector can be delivered in the header of the coded packet. The cost of delivering the coding vector is M extra $\mathbb{F}_q$ symbols in each packet, which is negligible if $M \ll K$.

At each destination node of a lossy network in which links are memoryless BEC, received packets are all error-free. Suppose that N packets $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N$ are received. The decoding of the source packets is equivalent to solving the following linear system of equations with multiple right-hand sides

$$\begin{bmatrix} g_{1,1} & \cdots & g_{1,M} \\ \vdots & \ddots & \vdots \\ g_{N,1} & \cdots & g_{N,M} \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_M \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_N \end{bmatrix}, \tag{2.2}$$

where the left-most matrix is referred to as the *decoding matrix*, whose rows are coding vectors extracted from the received packets and the $N \times K$ right-hand side matrix contains information symbols of each received coded packet. In practice, packets arrive at the destination node in a sequential manner. A received packet whose coding vector is not in the span of coding vectors of all previous received packets of

a node is referred to as an *innovative packet* to the node. Solving (2.2) is possible if and only if M innovative packets are obtained among the N received packets, i.e., when the decoding matrix is full-rank. Suppose that N received packets are needed for successful decoding. The reception *overhead*

$$\varepsilon = \frac{N - M}{M}. \tag{2.3}$$

When RLNC is applied across all the M source packets, encoding coefficients are uniformly randomly chosen from $\mathbb{F}_q$ and therefore the decoding matrix is uniformly random and dense. The decoding of RLNC can be performed using Gaussian elimination (GE), which results in unique solution as soon as the decoding matrix becomes full-rank. Solving (2.2) requires $\mathcal{O}(M^3 + M^2K)$ arithmetic operations over the finite field using GE, where an *operation* refers to either a divide or multiply-and-add operation between two elements of $\mathbb{F}_q$. The computational cost may be prohibitively high if M is larger than several hundreds as reported in [64], [55].

To reduce the decoding cost, the key idea, which is also the central theme of the thesis, is to design sparser coding methods that may not perform RLNC all the time and/or may not perform RLNC across all the M source packets. In these ways, sparser and/or structured instead of uniformly dense decoding matrices may be obtained. We can then exploit the properties of the decoding matrix to achieve low computational cost.

According to when the decoder can succeed, we can classify decoders into two general categories. Decoders that succeed as soon as M linearly independent packets are received are referred to as *overhead-optimized* decoders. Given a network code, the achieved overhead of using overhead-optimized decoders is the smallest among

all possible decoders. GE is one example of such decoders. On the other hand, some decoders may not be guaranteed to succeed when M linearly independent packets are received, which mostly occurs when the decoder is set to exploit certain properties of the decoding matrix. The generation-by-generation decoder in Chapter 5 is one such example, where the decoding is always performed on sub-matrices of the decoding matrix. For these decoders, the overhead would be larger than that of using overhead-optimized decoders. However, as a trade-off, the decoding cost would be reduced.

2.2 Sparse Code Design: Fountain Codes

Fountain coding, as first proposed in [5], is a concept of scalable coding method for dissemination of data over BEC. The motivation of fountain coding is that the source node can send a potentially unlimited number of coded packets out of a given set of source packets such that the original source packets can be recovered (hopefully at a low computational cost) from any subset of coded packets of size equal to or only slightly larger than the number of source packets. During the transmission process, the source node does not need to know the channel erasure rate to perform coding and no feedback channel is required to ensure reliable delivery. In this sense, RLNC can be viewed as a type of fountain codes with cubic decoding complexity.

Unlike network codes, fountain codes are originally end-to-end codes, meaning that received packets are always subsets of coded packets generated at the source node; no coding is allowed in the middle of the network. They are designed for one-hop networks, e.g. point-to-point, point-to-multipoint or multipoint-to-point transmissions. However, good designs of fountain codes involve inspiring sparse coding ideas that may be useful in design of sparse network codes.

LT codes [33] are the first example of practically implementable fountain code over $\mathbb{F}_2$; it is a sparse code, meaning that each coded packet is coded across only a small subset of source packets. In LT codes, each coded packet is an exclusive-or (XOR) combination of a small subset of source packets of size d, where d is the *degree* of the coded packet whose value is drawn from a *degree distribution* at random. When d is sampled from the distribution, d source packets are uniformly randomly chosen from the M source packets without replacement to form a coded packet. The coding process can be viewed as building a bipartite graph with M left nodes corresponding to source packets and a potentially unlimited number of right nodes corresponding to coded packets, where the right node degrees are randomly drawn from the degree distribution. The chosen source packets are referred to as the *neighbors* of the coded packet. In [33], Luby introduces two forms of degree distributions called the ideal soliton distribution and the robust soliton distribution. The former one is theoretically optimal in terms of overhead and the latter one performs better in practice. Both of the distributions result in the average degree $d_{\text{avg}} = \log(M)$ among coded packets.

Given a number of received packets, the decoding of LT codes is performed by the so-called *Luby-Transform process*. The process is also known under the name of *belief-propagation* decoder over BEC [36], [52]. In each step of the process the decoder needs to find at least one coded packet among received packets that has degree one. The set of degree-1 packets among received packets is called the *ripple*. As long as the ripple is not empty, the decoder is able to decode one packet because each degree-1 packet must be identical to some source packet. When one source packet is decoded, it is removed (by XOR) from the remaining received packets which have it as a neighbor. The removal may increase the size of ripple as it may reduce some

received packets to degree one. The process ends when the ripple is empty. The decoding process succeeds if all source packets are decoded in the end or else fails if the ripple is empty when not all source packets are decoded.

Essentially, the LT decoding process is no more than solving a linear system of sparse equations in the form of (2.2). Each equation corresponds to a received coded packet of symbols; the coordinates of nonzero elements (i.e., the 1's because $\mathbb{F}_2$ is used) in the equation are the indices of XORed source packets in the coded packet. However, due to the non-uniform sparseness introduced in the encoding using the soliton distributions, solving (2.2) can be done quite efficiently instead of using GE. To see this, we rephrase the decoding process in the context of solving the linear system of equation through *pivoting* the decoding matrix. This viewpoint will be shown to be useful in some of the designs of sparse network codes proposed in this thesis.

Pivoting is a process of selecting a sequence of elements from the matrix and re-ordering rows and columns of the matrix accordingly. In each step, one nonzero element of the matrix is selected as a *pivot* according to a specific pivoting algorithm. After M pivots are successfully obtained, the matrix is reordered according to the sequence of selected pivots by exchanging rows and columns such that the i-th selected pivot is now the i-th diagonal element of the new matrix. Note that the rows of the right-side matrix of the linear system of equations need to be reordered accordingly such that the solution to the linear system remains the unchanged. The pivoting, if properly done, would ensure that the sparseness of the matrix can be maintained during row reductions. In Figure 2.2 [10] we show a typical example where pivoting and reordering are necessary, where if forward row reductions are performed on the

original matrix on the left directly, all zero elements will be changed to nonzero after eliminating elements of the first column. However, if the matrix is pivoted and reordered to the form on the right, the sparse structure would not be ruined.

Figure 2.2: Reorder matrix before row reductions to preserve the sparsity.

In effect, LT decoding corresponds to the following simple algorithm: in each step of pivoting, the algorithm always chooses the nonzero element from a singleton row as the pivot, where singleton rows are rows with only one nonzero element; if there are multiple singleton rows, choose the pivot from one of them at random; if else no singleton rows can be found before M pivots are found, the decoding declares a failure.

The reordered decoding matrix of LT codes after successful pivoting is in a lower-triangular form, and the average number of nonzero elements in each row below the diagonal is $\log(M)$. Solving such a linear system of equations only requires forward row reductions to eliminate nonzero elements below the diagonal, and therefore the number of required operations is linearly proportional to the number of nonzero elements and hence results in $\mathcal{O}(M \log(M))$ arithmetic operations rather than $\mathcal{O}(M^3)$ using GE.

A remarkable feature of LT codes is that the robust soliton distribution designed

by Luby enables the described pivoting algorithm to succeed with high probability with only slightly larger than M received coded packets, resulting in a very small overhead. It is noted that $\log(M)$ is an information-theoretic lower bound of the average degree that is needed such that M source packets can be recovered from about M coded packets, regardless of what decoding methods are used. A proof of it using balls-in-bins arguments can be found in [56].

Raptor codes [56] are improved fountain codes based on LT codes. Raptor codes have $\mathcal{O}(M)$ rather than $\mathcal{O}(M\log(M))$ decoding complexity. The basic idea of it is to use a degree distribution that results in a constant (rather than $\log(M)$) average degree among coded packets and use a fixed-rate erasure-correction precode to recover some of the source packets that are not recoverable by the LT decoding process.

Raptor codes have been selected as the standard coding method for reliable multimedia broadcast in 3GPP systems [1] for its linear-time decoding complexity. In [57], the standardized Raptor codes, Raptor10 (R10) and RaptorQ (RQ), are described in details. One of the key improvements of standardized Raptor codes over the original designed Raptor codes is the pivoting algorithm used in decoding. In standardized Raptor codes, a so-called *inactivation decoding* is used, which essentially is a pivoting algorithm that when the pivoting fails to find a singleton row, some columns of the matrix are declared as inactive such that singleton rows can be found in the remaining sub-matrix that consists of the rest of active columns. The pivoting algorithm is also seen in [46], [51] and referred to as the *structured Gaussian elimination*. The reordered matrix after pivoting is in the form of the left matrix of Figure 2.3, where the right part of the matrix are the inactivated columns and $\mathbf{0}$ refers to the area where elements are zero. The left part of the reordered matrix is in a lower triangular form

and therefore can be diagonalized by forward row reductions. The average degree of each row below the diagonal is a constant. The matrix is transformed to the right one of Figure 2.3 after the diagonalization of the left half. The decoding is then to first reduce the sub-matrix $\mathbf{T}_I$ which is now dense because of row operations during the diagonalization to identity matrix using Gaussian elimination and then eliminate $\mathbf{U}_I$ to convert the whole matrix to identity.

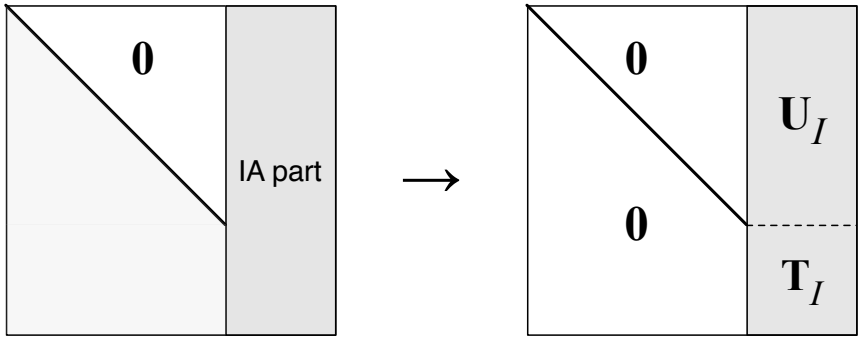

Figure 2.3: Reordered matrix after inactivation decoding.

Note that unlike the LT decoding process, the pivoting method above ensures that the decoding is successful when the matrix is full-rank and therefore the decoding is overhead optimized. It is important to note that a well-designed degree distribution of Raptor codes would result in only a small number of columns being inactivated, normally in the order of $\sqrt{M}$. Therefore, the Gaussian elimination on $\mathbf{T}_I$ is linear in M and the overall decoding complexity is $\mathcal{O}(M)$.

The lesson we can learn from the design of fountain codes is two-fold: 1) the computational cost of solving linear systems relies on not only the sparsity of the matrix (i.e., the average degree of all equations) but also the "structure" of the sparsity, which is determined by the degree distributions. Generally, it would be beneficial to have different numbers of nonzero coefficients in each equation, i.e., *non-uniform*

sparsity; 2) using a hybrid process of Gaussian elimination and belief-propagation decoding may obtain a better trade-off between the overhead and computational cost performances than using one of them exclusively. Specifically, if the sparsity "structure" is properly designed, the decoder would achieve the overhead close to that of Gaussian elimination and the computational cost close to that of belief-propagation.

It is a natural thought to directly apply LT or Raptor codes in networks, while aiming at preserving its sparsity when performing network coding at intermediate nodes. Examples of efforts in this direction are [47], [16], [7] for line networks, whose topology is relatively easy to handle when trying to preserve the sparsity of codes because there is only one downstream neighboring node of each node. In networks with more complex topology, however, the task to maintain the required degree distribution of codes gets challenging. To circumvent the issue, some works such as [6] consider to decode fountain codes completely at intermediate nodes, i.e., use fountain codes in a hop-by-hop manner. This way, however, loses much of the benefit of network coding as it incurs much long delay.

Chapter 3

Systematic Network Coding for Known Network Topologies: Two-Hop Links

3.1 Introduction

Packet transmission over two-hop lossy links is increasingly important in communication networks. For example, in current cellular networks, WiFi access points (AP) may be deployed in cells to offload traffic from the base-station (BS). In this scenario, transmissions from the P-GW (Packet Data Network Gateway) to a destination via BS and AP can each be modeled as packet transmissions over two-hop lossy links.

Network coding can be beneficial in these types of networks. When packets pass through a lossy two-hop link, if intermediate nodes re-encode packets with previously received ones, coding can achieve the max-flow capacity of the link [67], and therefore improve the end-to-end rate of the transmission compared to uncoded schemes. More importantly, this improvement is achieved without the need of either acknowledgements between nodes (as required by ARQ) or estimation of link qualities (as required by FEC). Random linear network coding (RLNC) [22], in which the re-encoded packets are random linear combinations of packets received at the intermediate node, can

achieve this in a distributed manner. RLNC is known to be asymptotically *rate-optimal* as it achieves the max-flow capacity when the size of the arithmetic field of the RLNC goes to infinity.

As mentioned in Chapter 2, an important drawback of RLNC is complexity. Decoding of RLNC, performed on M source packets each with K symbols from a finite field of size q, $\mathbb{F}_q$, requires the solution of a linear system of equations in M unknowns, which requires $\mathcal{O}(M^3 + M^2K)$ operations for unstructured matrices, e.g., via Gaussian elimination (GE). For handheld devices with limited computing capability and battery power, this computational cost may be prohibitively high. Moreover, the rate-optimality of RLNC relies on a sufficiently large field size, which, as a consequence, requires a larger packet header to carry coding vectors and also requires more resources when performing finite field arithmetic (for example, large look up tables may be needed to perform efficient multiplication). When the field size is small, for example $q = 2$ (i.e., binary), the achievable rate of RLNC can be much lower than the max-flow capacity, especially when the number of source packets is also small.

In this chapter, we address network coding for two-hop lossy links. We propose a network coding method for this network topology that possesses similar rate performance as RLNC but at significantly lower encoding/decoding computational cost. The proposed scheme first sends uncoded packets in their original order, and then sends a potentially unlimited number of coded packets using RLNC. Combined with the proposed re-encoding strategy at the intermediate node, which forwards received uncoded packets or else performs the same coding as RLNC, the scheme will be shown to achieve higher rate than RLNC but with less computation. In applications where the field size and the number of source packets are not very large, for example in

WiFi where we may often have frequent transmissions of small amount of data, the improvement of the proposed scheme over using RLNC is considerable.

The transmission scheme where a source node sends uncoded packets first and then sends coded packets has recently appeared in the literature for one-hop links. For example, in [20], [58] and [38], the benefits of reducing decoding complexity and completion time are explored. For two-hop links, a scheme is presented in [47] (Section III.B) in which fountain-coded packets (e.g. using LT codes [33]) are transmitted in the first hop; the intermediate node forwards the received packets from the first hop and sends some re-encoded packets from the buffer after forwarding. The method is studied from a coding theory point of view in the asymptotic regime. In [13], a scheme with a similar transmission strategy in the first hop as ours is proposed. However, the method in [13] assumes a direct link between the source and destination nodes and only sends coded packets from the buffer on the second hop after all uncoded packets are transmitted, and does not make use of possible transmission opportunities on the second hop before uncoded packets are finished. Therefore, the achievable rate of the scheme in [13] is strictly lower than the max-flow capacity of the link.

The rest of the chapter is organized as follows: in Section 3.2, we present the system model and describe the proposed scheme. Section 3.3 compares the rate and complexity performance of the proposed scheme with RLNC, where Section 3.3.1 presents a method to analytically determine the expected completion time of the proposed scheme using a Markov chain model and in Sections 3.3.2 and 3.3.3 we establish the advantages of S2HNC compared to RLNC. In Section 3.4, we consider the effect of using limited size buffer at the intermediate node. We present numerical and simulation results in Section 3.5 and offer conclusions in Section 3.6.

3.2 System Model and Coding Schemes

3.2.1 Transmission Model

Suppose a source node, S, has M data packets with K symbols from $\mathbb{F}_q$ per packet, denoted as $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_M$, to send to a destination node, D. A relay node, R, connects S and D to reduce transmission distance. There is assumed to be no direct link between S and D. Figure 3.1 shows the network topology. Let S-R and R-D denote the two links between the nodes. The links are modeled as delay-free memoryless erasure channels. The erasure probabilities of S-R and R-D are denoted as ϵ_1 and ϵ_2, respectively. The capacities of the two links are therefore $1-\epsilon_1$ and $1-\epsilon_2$, respectively. Node R operates in half-duplex mode, meaning that it is only able to either receive from node S or transmit to node D at a given time but not simultaneously. For convenience, we use S-R-D to refer to the overall two-hop link.

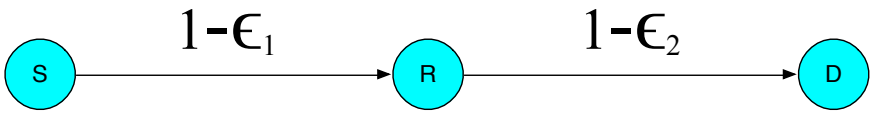

Figure 3.1: Two-hop lossy link.

We assume that transmission opportunities are presented to S and R successively. We refer to two successive transmissions on S-R and R-D as a *network use*. When R receives a packet from S, it stores the received packet in its buffer. We denote the buffer size at R as U. In most of this chapter, we assume an unlimited buffer size, i.e., $U = \infty$ except for Section 3.4. Throughout the transmission, we assume no feedback

between nodes except that D informs S and R when it has finished recovering all the source packets.

3.2.2 Coding Methods

Random Linear Network Coding

As the baseline scheme, RLNC refers to the case in which S keeps sending random linear combinations of packets, and R transmits random linear combinations of packets that have been received thus far on S-R. This packet-level RLNC scheme was first proposed in [39].

Packets sent from S are in the form of $\mathbf{p} = \sum_{i=1}^{M} g_i' \mathbf{s}_i$, where g_i' is a randomly chosen encoding coefficient from $\mathbb{F}_q$ at the source node. In RLNC, we assume that coefficients g_i' are chosen at random with uniform probability from $\mathbb{F}_q$. If we denote packets received at R as $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{B_n}$, where B_n is the number of buffered packets after at time n, the packet sent on R-D in the next transmission is a linear combination $\sum_{i=1}^{B_n} h_i' \mathbf{p}_i$, where h_i' are uniformly randomly chosen re-encoding coefficients at R. Coding coefficients are delivered in the header of encoded packets. The coding vector of the received packets at the destination node is denoted as $[g_{i,1}, g_{i,2}, \ldots, g_{i,M}]$ for the i-th received packets, $i = 1, 2, \ldots$.

Suppose that N packets $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N$ are received by D. The recovery of source packets corresponds to solving the linear system of equations

$$\begin{bmatrix} g_{1,1} & \cdots & g_{1,M} \\ \vdots & \ddots & \vdots \\ g_{N,1} & \cdots & g_{N,M} \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_M \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_N \end{bmatrix}, \tag{3.1}$$

where the matrix on the left-hand side is referred to as the *decoding matrix*, whose rows are coding vectors of received packets and the $N \times K$ right-hand side matrix consists of symbols in received coded packets. An *innovative packet* refers to a received packet whose coding vector is not in the span of coding vectors of all previously received packets. Decoding is only possible if M innovative packets are received, i.e., when the decoding matrix is full-rank. As the decoding matrix in (3.1) is unstructured, $\mathcal{O}(M^3 + M^2K)$ operations are required for GE over the finite field, where an *operation* refers to either a divide or multiply-and-add finite field operation. If the finite field is $\mathbb{F}_2$, an operation refers to the XOR of any two bits.

Systematic Two-Hop NC

When M is large, it is clear that the decoding complexity of RLNC is high. Here, we propose a coding method with lower complexity: S sends M uncoded packets in their original order for the first M transmission opportunities and sends coded packets using RLNC after that. Over the S-R-D link, if R receives an uncoded packet from S, it forwards the packet on R-D in its next transmission and also saves a copy of the packet in its buffer. If R does not receive a packet or receives a coded packet on S-R, in its next transmission it generates a packet from the buffer using RLNC which it then transmits to D. We refer to this overall scheme as *systematic two-hop network coding* (S2HNC). The decoding of S2HNC is similar to that of RLNC, i.e., solving the linear system of equations (3.1). However, since received packets may be uncoded, some rows of the decoding matrix are singleton, i.e., contain only one nonzero element.

3.2.3 Performance Metrics

We are interested in two performance metrics: end-to-end rate and decoding complexity. The end-to-end rate is defined as $R_e = \frac{M}{T}$, where T is the expected completion time, which is defined as the expected number of network uses that are needed for D to recover M source packets. R_e is upper bounded by the max-flow capacity of the S-R-D link, which is $R_c = (1 - \max\{\epsilon_1, \epsilon_2\})$ packets per network use. It is clear that R_e is inversely proportional to the expected completion time T as M is a constant.

The decoding complexity of schemes are measured by the average number of operations that are needed per packet in recovering all the source data.

3.3 Analysis of S2HNC

3.3.1 Expected Completion Time of S2HNC

It is known that when M is large, RLNC achieves the max-flow capacity of the network. In practice, however, where M may not be large, we would experience some loss in end-to-end rate. In this section, we employ a Markov chain model to calculate the expected completion time of S2HNC in the small M regime given erasure rates. In [54], a Markov chain model is proposed to analyze the two-hop network coded transmission, where RLNC of infinite field size is assumed. In the following we apply a similar technique to finite field sizes.

We use two Markov chains to characterize the process of D collecting innovative packets. One denoted as C1 is for the period of time when S is sending uncoded packets and the other one C2 is for when S is sending coded packets. The state spaces of the two Markov chains are the same and can be expressed in a two-tuple state, (k, r), where k and r represent the numbers of innovative packets received

by R and D at the beginning of each S-R transmission. The state transitions that may occur at each step is shown in Figure 3.2. It is noted that $r \leq k$ because the information received by D cannot exceed that received by R at any time. We have a total of $n = \frac{(M+1)(M+2)}{2}$ two-tuple states.

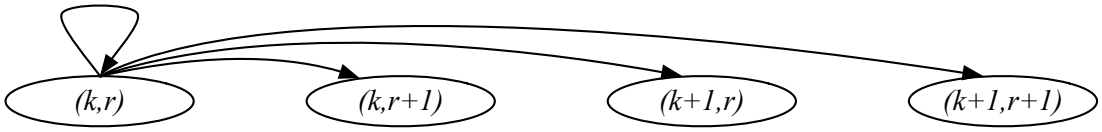

Figure 3.2: State transitions of the considered Markov chains.

When S transmits uncoded packets, the probability that k increases by 1 after a network use is $p_{k\to k+1} = 1 - \epsilon_1$ for $k < M$, i.e., a successful transmission on S-R and 0 for $k = M$. The increment of r is dependent on k's evolution. For $k < M$, we have $p_{r\to r+1|k\to k+1} = 1-\epsilon_2$ for a successful forward by R or $p_{r\to r+1|k\to k} = (1-\epsilon_2)\left(1 - \frac{1}{q^{k-r}}\right)$ for a successful transmission of an innovative packet coded from R's buffer[1]. For $k = M$, we have $p_{r\to r+1|k\to k} = (1-\epsilon_2)(1-\frac{1}{q^{M-r}})$. The evolution of (k, r) for C1 can then be characterized. For example, denoting $p_{(k,r)\to(k+1,r+1)}$ as the probability that k and r are each increased by 1 after a network use, it is seen that it is equal to $p_{k\to k+1}p_{r\to r+1|k\to k+1}$ for $r \leq k < M$. Similarly, we can obtain $p_{(k,r)\to(k,r)}$, $p_{(k,r)\to(k+1,r)}$, and $p_{(k,r)\to(k,r+1)}$.

When S starts sending coded packets, the transition probabilities are different. In this stage, k is increased with probability $p_{k\to k+1} = (1-\epsilon_1)(1-\frac{1}{q^{M-k}})$ for $k \leq M$. When $k < M$, the probability that r increases is $p_{r\to r+1|k\to k} = (1-\epsilon_2)(1-\frac{1}{q^{k-r}})$ and $p_{r\to r+1|k\to k+1} = (1-\epsilon_2)(1-\frac{1}{q^{k+1-r}})$, where the term $\frac{1}{q^{k+1-r}}$ is because one more

[1]The item $\frac{1}{q^{k-r}}$ is because the coded packet is a random linear combination of buffered packets [28].

innovative packet is received at R before it generates a coded packet. When $k = M$, $p_{r\to r+1|k\to k} = (1-\epsilon_2)(1-\frac{1}{q^{M-r}})$. With these, we can obtain $p_{(k,r)\to(k,r)}$, $p_{(k,r)\to(k+1,r)}$, $p_{(k,r)\to(k,r+1)}$ and $p_{(k,r)\to(k+1,r+1)}$ for C2.

The detailed expressions of transition probabilities of C1 and C2 are provided in Appendix A. The calculation of the expected completion time of S2HNC consists of determining the state of C1 after exactly M steps (i.e., M network uses) starting from $(0,0)$ and the expected steps that are further needed for C2 to transit from that state to (M,M). To simplify the computation, we can label each two-tuple state with an index. Our choice here is to list states in a sequence as $(0,0)$, $(1,0)$, $(1,1)$, $(2,0)$, $(2,1)$, ..., $(M,M-1)$, (M,M), and then we label state (k,r) using index $i = \frac{k(k+1)}{2} + r$, i.e., using a new sequence of states $0,1,2,3,\ldots$ to refer to $(0,0)$, $(1,0)$, $(1,1)$, $(2,0),\ldots$. Below we will interchangeably use i or (k,r) to refer to a two-tuple state. A consequence of this labeling choice is that state i cannot transit to a state $j < i$. With the notation, we can write the transition matrix of C1 as Π_1, which is an upper-triangular matrix with elements π_{ij}^1 denoting the probability of transition from state i to j. Similarly, the transition probabilities of C2 are put into a matrix denoted as Π_2.

After M steps starting from $(0,0)$ with transition matrix Π_1, the probability that the system is in state i is equal to the i-th element of $\mathbf{1}_0\Pi_1^M$, where $\mathbf{1}_0$ is a length-n ($n = \frac{(M+1)(M+2)}{2}$) row vector with only the first item being non-zero, 1. The probability vector $\mathbf{1}_0\Pi_1^M$ is the "input" of C2, which is an absorbing Markov chain in which (M,M) is the only absorbing state. It is easy to verify that Π_1 and Π_2 can be written in the forms of $\begin{bmatrix} Q_i & R_i \\ 0 & 1 \end{bmatrix}$, $i = 1,2$, respectively. Q_1 and Q_2 are of the size $t \times t$; $t = n-1$ is the number of transient states. Then the expected number

of steps, Δ_M, that is needed by C2 to enter the absorbing state given $\mathbf{1}_0\Pi_1^M$ as the initial probability vector of states is equal to the first entry of the vector [25]

$$Q_1^M(I_t - Q_2)^{-1}\mathbf{1}_t, \tag{3.2}$$

where I_t is a $t \times t$ identity matrix and $\mathbf{1}_t$ is a length-t column vector whose entries are all 1. Then the expected completion time of S2HNC is

$$\mathrm{E}\{T\} = M + \Delta_M. \tag{3.3}$$

We note that the load of computing the expected completion time from (3.2) is high for large M, even though that Q_1 and Q_2 are very sparse. For example, the computation of Δ_M for $M = 500$ requires matrix multiplications between matrices Q_1 and Q_2, each of order $125,750$. However, since S2HNC asymptotically achieves the max-flow capacity for large M, we only need the presented technique to calculate the expected completion time for small M, which would not be a problem.

The method presented includes the calculation of expected time for RLNC scheme as a special case, in which no uncoded packets are sent and C2 with initial state $(0, 0)$ characterizes the whole transmission process. The expected number of steps before the process entering the absorbing state is then equal to the first entry of the vector $(I_t - Q_2)^{-1}\mathbf{1}_t$.

The variances of the completion time can also be obtained. Note that the initial state of S2HNC for Π_2 is $\mathbf{1}_0\Pi_1^M$ while that of RLNC is $\mathbf{1}_0$. Let $N = (I_t - Q_2)^{-1}$; we have [25]

$$\mathrm{Var}\{T\} = \mathbf{v}(2N - I_t)N\mathbf{1}_t - (\mathbf{v}N\mathbf{1}_t)^2, \tag{3.4}$$

where $\mathbf{v}$ is the first row of Q_1^M for S2HNC and is a length-t vector with only the first item being non-zero, 1, for RLNC.

3.3.2 Advantage in End-to-End Rate

Based on the above analysis, we can establish that S2HNC has shorter expected completion time, and therefore achieves higher end-to-end rate than RLNC. We achieve this by examining the probabilities that a received packet is innovative for D when using S2HNC and RLNC, respectively. S2HNC will be shown to have larger such probability at any time than RLNC.

Since S2HNC has the same behavior (which is characterized by the Markov chain C2) as RLNC after M transmissions from S, we only need to compare the two schemes during the first M transmissions from S. We consider S2HNC first. Recall that k and r denote the numbers of innovative packets that have been received by R and D at the beginning of each S-R transmission, $r \leq k$. During the first M transmissions from S, the probability that D receives an innovative packet after two successive transmissions on S-R and R-D is equal to

$$p_{\text{S2HNC}}(k,r) = (1-\epsilon_2)\left[(1-\epsilon_1) + \epsilon_1(1-\frac{q^r}{q^k})\right], \tag{3.5}$$

which corresponds to that an uncoded packet is successfully forwarded (which is innovative for D with probability 1) or the coded packet from the k innovative packets at R is innovative for D.

Now consider the RLNC scheme. For the same k and r, the probability that a new innovative packet is received by D after two successive transmissions on S-R and

R-D is equal to

$$\begin{aligned} p_{\text{RLNC}}(k,r) &= (1-\epsilon_2)\left\{(1-\epsilon_1)\left[\left(1-\frac{q^k}{q^M}\right)\left(1-\frac{q^r}{q^{k+1}}\right)\right.\right. \\ &\quad \left.\left.+\frac{q^k}{q^M}\left(1-\frac{q^r}{q^k}\right)\right]+\epsilon_1\left(1-\frac{q^r}{q^k}\right)\right\}, \end{aligned} \tag{3.6}$$

where the bracketed term $\left(1-\frac{q^k}{q^M}\right)\left(1-\frac{q^r}{q^{k+1}}\right)+\frac{q^k}{q^M}\left(1-\frac{q^r}{q^k}\right)$ correspond to that the coded packet sent from R is innovative for D if a packet was successfully received by R.

We have $p_{\text{S2HNC}}(k,r) > p_{\text{RLNC}}(k,r)$ because

$$\left(1-\frac{q^k}{q^M}\right)\left(1-\frac{q^r}{q^{k+1}}\right)+\frac{q^k}{q^M}\left(1-\frac{q^r}{q^k}\right)<1.$$

3.3.3 Advantage in Decoding Complexity

For M source packets, it is expected that $M_u=(1-\epsilon_1)(1-\epsilon_2)M$ uncoded packets are received by D using S2HNC. Therefore, S2HNC only needs to decode the remaining $M-M_u$ packets from RLNC coded packets using GE. For $K \gg M$, the number of operations that are needed using GE is approximately M^2K for RLNC and is $(M-M_u)^2K$ for S2HNC. This corresponds to a savings of a factor of $\frac{M^2K}{(M-M_u)^2K}=\frac{1}{(\epsilon_1+\epsilon_2-\epsilon_1\epsilon_2)^2}$ computations. This is a considerable saving compared to RLNC when ϵ_1 and ϵ_2 are small as in many system scenarios.

3.4 Impact of Using Limited Size Buffer

We have assumed unlimited buffer size at the relay node thus far. In practice, however, it is unrealistic to have unlimited buffer size. Therefore, we need to examine the

impact of using limited buffer size at R, where at most U packets, $U \ll M$, can be stored. With this setting, we assume that the following buffering scheme [40] is used when storing packets (either uncoded or coded): if the buffer is not full, the received packet, $\mathbf{r}$, is stored in its original form; otherwise, it is multiplied by U random coefficients $g_1, g_2, \ldots, g_U$ chosen independently uniformly from $\mathbb{F}_q$, and the products $g_1\mathbf{r}, g_2\mathbf{r}, \ldots, g_U\mathbf{r}$ are added to the U buffered packets, respectively. As a result, all store packets in the buffer is actually coded.

We may use rate loss to measure the influence of using limited size buffer, which is defined as $1-R_e/R_c$. We conduct a similar analysis as in [40] under unlimited field size assumption, where the maximum achievable end-to-end rate under the assumption is shown to be equal to

$$R_e^* = (1-\epsilon_2)(1-\epsilon_1\pi_0), \tag{3.7}$$

where π_0 is the steady-state probability of state 0 of the Markov chain shown in Figure 3.3, in which the states denote the numbers of innovative packets that the U-size buffer at R has for sending to D. The steady-state probabilities are denoted as $\pi_0, \pi_1, \ldots, \pi_U$. The equation comes from the fact that, under unlimited field size assumption, every successful transmission on R-D conveys an innovative packet to D if there is at least one innovative packet buffered in R, and that if the number of innovative packet at R is zero, then only a successful overall transmission on S-R-D would convey an innovative packet. The transition probabilities showing in Figure 3.3 are $\alpha = (1-\epsilon_1)\epsilon_2$, $\beta = \epsilon_1(1-\epsilon_2)$, $\gamma = \epsilon_1\epsilon_2 + (1-\epsilon_1)(1-\epsilon_2)$, $a = \epsilon_1 + (1-\epsilon_1)(1-\epsilon_2)$, $b = \epsilon_1\epsilon_2 + (1-\epsilon_1)$, and $c = \epsilon_1(1-\epsilon_2)$.

We note that (3.7) holds regardless of how packets are buffered, i.e., it is the performance upper bound for any buffering scheme. We also note that the system

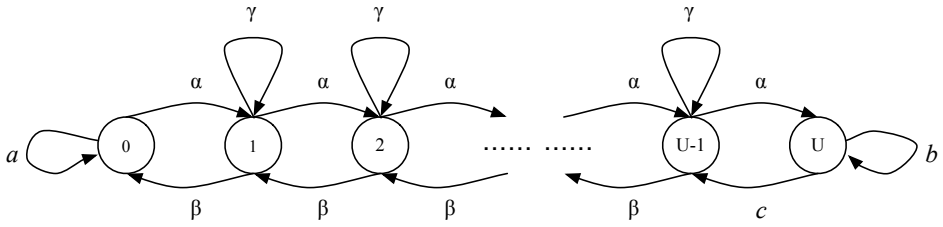

Figure 3.3: Evolution of number of innovative packets in the buffer.

model in this chapter is based on half-duplex R, and therefore the transition probabilities above are slightly different from those in [40] for b and c. Using the fact that $\pi_j = \sum_i \pi_i p_{ij}$ for all j and that $\sum_i \pi_i = 1$, where p_{ij} is the transition probability from state i to j, it is easy to verify that $\alpha\pi_i = \beta\pi_{i+1}, i = 0, U-2$ and $\alpha\pi_{U-1} = c\pi_U$, and therefore

$$\pi_0 = \frac{1}{1 + \rho + \rho^2 + \ldots + \rho^{U-1} + \frac{\beta}{c}\rho^U}, \tag{3.8}$$

where $\rho = \alpha/\beta$.

A key point that we have observed relevant to (3.7) is that the maximum achievable rate fraction using a fixed buffer size varies given different pairs of ϵ_1 and ϵ_2. Specifically, the rate loss performance is comparatively worse at $\epsilon_1 = \epsilon_2$. To see this, for a given U, without loss of generality, we fix ϵ_2 and consider different values of ϵ_1. The fraction of the capacity of the link that can be achieved can then be expressed as a function of ϵ_1 as

$$f(\epsilon_1) \triangleq \frac{R_e^*(\epsilon_1)}{R_c(\epsilon_1)} = \begin{cases} 1 - \epsilon_1\pi_0, & \epsilon_1 < \epsilon_2; \\ \frac{U}{U+\epsilon_2}, & \epsilon_1 = \epsilon_2; \\ \frac{1-\epsilon_2}{1-\epsilon_1}(1 - \epsilon_1\pi_0), & \epsilon_1 > \epsilon_2, \end{cases} \tag{3.9}$$

where we obtain the expression for the case $\epsilon_1 = \epsilon_2$ by setting $\rho = 1$ to (3.8). By substituting (3.8), it is easy to see that $f(\epsilon_1)$ is monotonically decreasing with ϵ_1 on $(0, \epsilon_2)$ and monotonically increasing with ϵ_1 on $(\epsilon_2, 1)$.

The phenomenon is illustrated in Figure 3.4 for fixed $\epsilon_2 = 0.2$ and $U = 5$. We see that using limited size buffer would affect the end-to-end rate the most at $\epsilon_1 = 0.2$, i.e., given a fixed buffer size, the system would perform comparatively worse if two hops are of about the same quality.

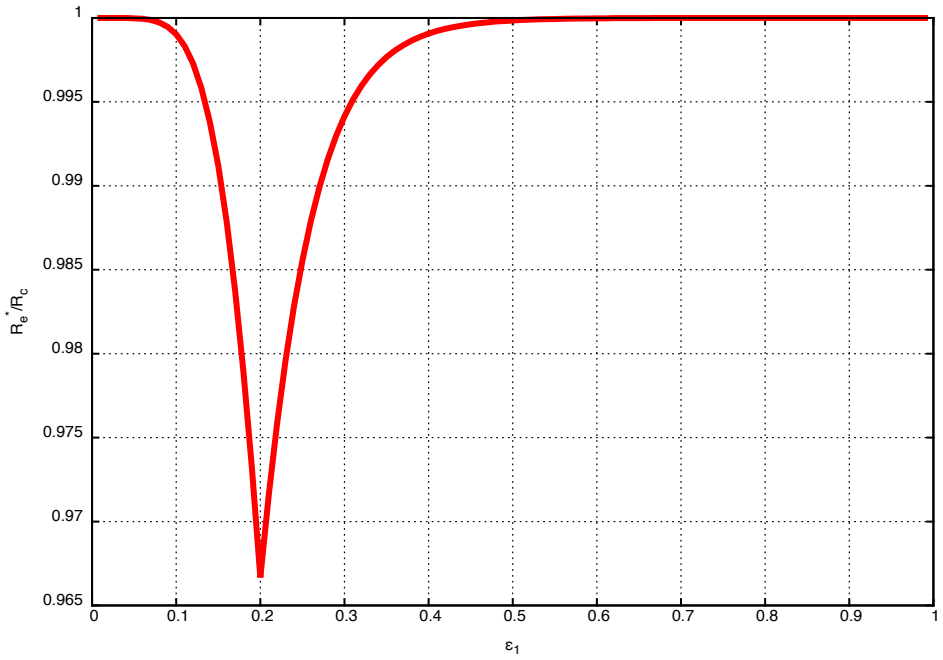

Figure 3.4: Normalized maximum achievable rates for various ϵ_1 under $\epsilon_2 = 0.2$ and $U = 5$.

Note that this analysis does not mean that the absolute achievable rate R_e^* at $\epsilon_1 = \epsilon_2$ is lower than that at $\epsilon_1 > \epsilon_2$, but that around $\epsilon_1 = \epsilon_2$ a much larger buffer is needed to achieve the same level of R_e^*/R_c compared to other scenarios.

In Section 3.5, we observe by simulations that S2HNC performs better than RLNC

under the same buffer and finite field size conditions.

3.5 Numerical and Simulation Results

In this section we simulate the performance of S2HNC and compare it with RLNC. We also include a simulation of the LT-code scheme proposed in [47]. We first compare S2HNC with RLNC in different scenarios. Specifically, we show the following three cases: $\epsilon_1 = 0.05$, $\epsilon_2 = 0.2$, $\epsilon_1 = 0.2$, $\epsilon_2 = 0.2$, and $\epsilon_1 = 0.2$, $\epsilon_2 = 0.05$, which correspond to cases where the quality of the first hop is better, equal to, and worse than that of the second hop, respectively. In all cases, the max-flow capacity of the two-hop link is 0.8 packets per network use. Coding is performed in $\mathbb{F}_2 = \{0, 1\}$.

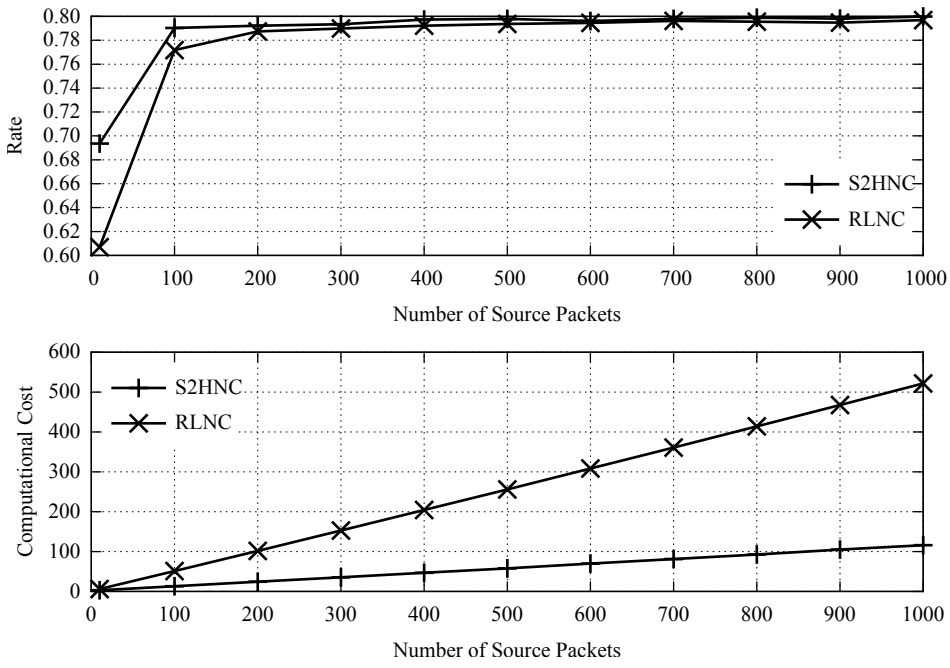

Figure 3.5: End-to-end rate and decoding complexity for $\epsilon_1 = 0.05$, $\epsilon_2 = 0.2$.

The simulations use $K = 8192$ bits per packet; the numbers of packets vary from $M = 10$ to 1000. We count the number of operations, N_{ops}, that are needed in

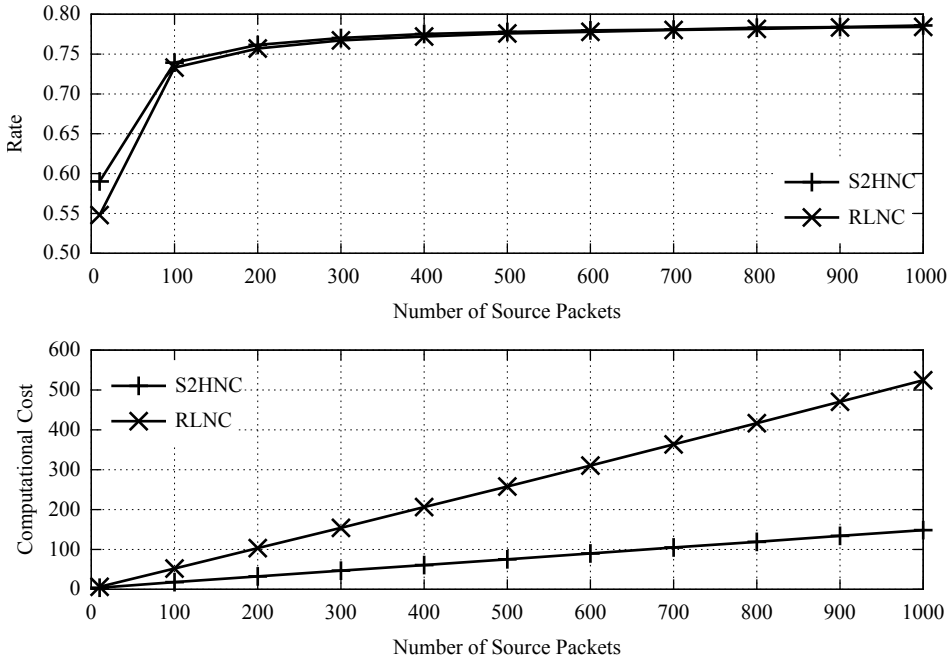

Figure 3.6: End-to-end rate and decoding complexity for $\epsilon_1 = 0.2$, $\epsilon_2 = 0.2$.

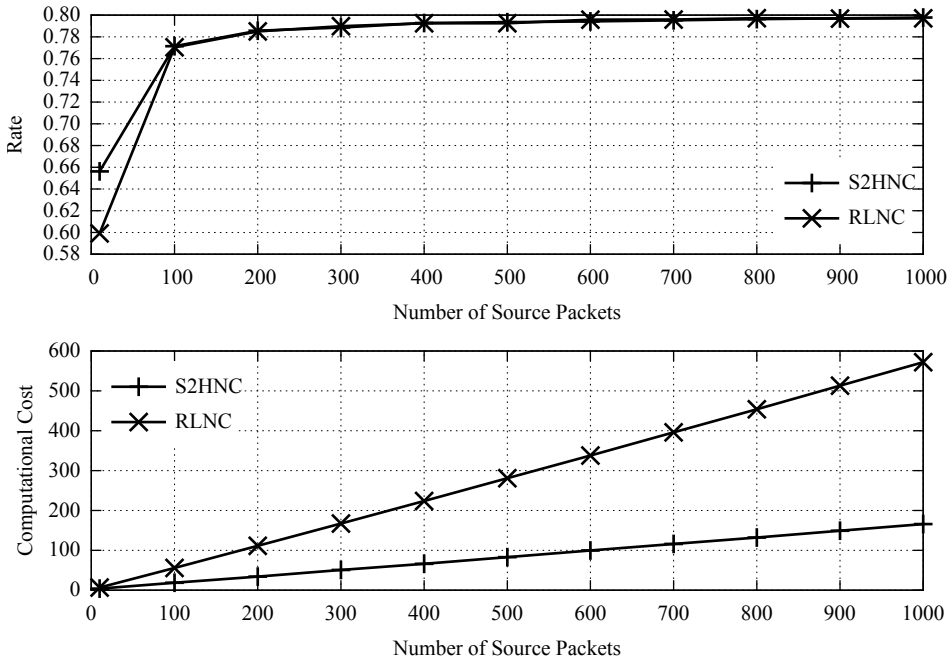

Figure 3.7: End-to-end rate and decoding complexity for $\epsilon_1 = 0.2$, $\epsilon_2 = 0.05$.

recovering all the data in the implementation. The computational cost measurement we use is the average number of operations per packet symbol (more specifically, bit as in $\mathbb{F}_2$) for successful decoding, which is equal to $\frac{N_{ops}}{MK}$. Each curve of the following simulation is averaged over $10,000$ trials. The performances for the three cases are shown in Figures 3.5-3.7, respectively.

Clearly, the rate of S2HNC is always higher, even though the gap is not large and all the schemes achieve capacity quickly as M increases. The lower decoding complexity of S2HNC, however, is quite obvious. It is seen that the improvement of S2HNC in terms of the rate is more considerable in Figure 3.5, where the first hop is more reliable than the second but is little if the second hop is better as shown in Figure 3.7. This is because in the former case more uncoded packets would be received at R, which are always innovative for D when using S2HNC; if the second hop is more reliable, then S2HNC appears to resemble RLNC at R because more coded packets would be sent. In practice, the former case is of particular interest for networks that have a more reliable first hop. For example in cellular networks where a base-station or WiFi access point is used to relay information from the data gateway to users, the first hop is much more reliable than the second hop.

We note that when the link quality is poor, i.e., the erasure rates of S-R and R-D are high, the performance gap between S2HNC and RLNC tends to shrink. An example of this is shown in Figure 3.8, where the two hops have equal erasure probability 0.8 and the performances of S2HNC and RLNC are almost identical. This is anticipated because the advantage of S2HNC comes from its use of uncoded packets, which are innovative for downstream nodes with probability one while require no decoding. When links are highly lossy, very few uncoded packets would be received

and S2HNC reduces to RLNC.

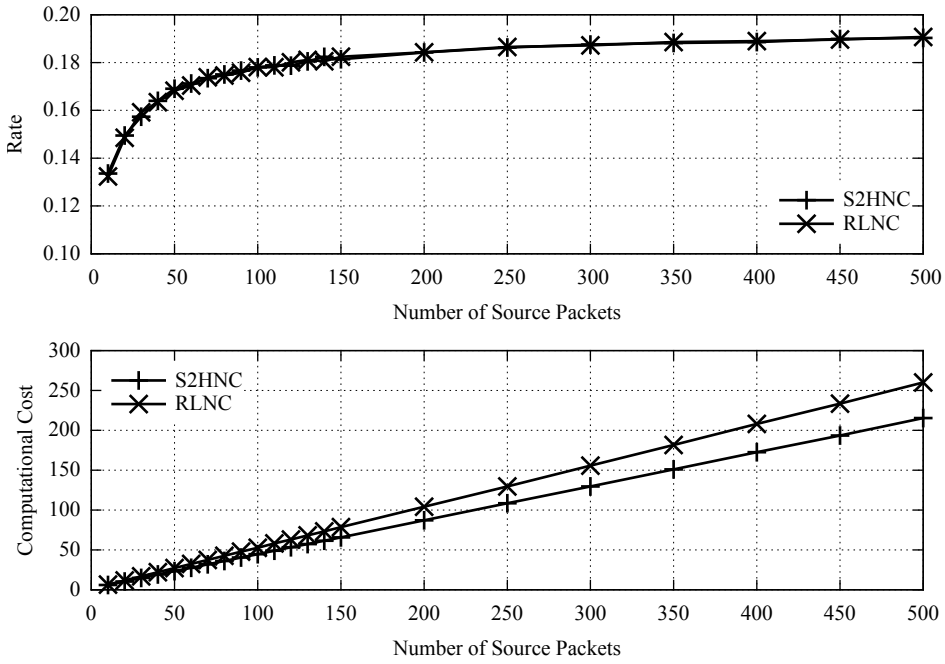

Figure 3.8: End-to-end rate and decoding complexity for $\epsilon_1 = \epsilon_2 = 0.8$.

We have seen that both S2HNC and RLNC approach link capacity as M increases. However, when M is small, performance is far below capacity. We show this in Figure 3.9, where the rates and computational costs of the two schemes at small values of $M = 10$ to 100 are compared. A simulation of the LT-code scheme proposed in [47] is included, where LT-coded packets are transmitted from S and R either forwards received packets if receive any or generates RLNC re-encoded packets from its buffer. Note that here the decoding of LT-coded packets is done by the inactivation decoding [57] as described in Chapter 2, which finishes at the first instance a full-rank decoding matrix is received, i.e., the achievable rate of LT-coded scheme is maximized in the plot. In the low M and q regime, the rate improvement of S2HNC over the other two schemes is quite obvious. The LT-coded scheme has the lowest complexity due to its

sparse nature but has the lowest rate.

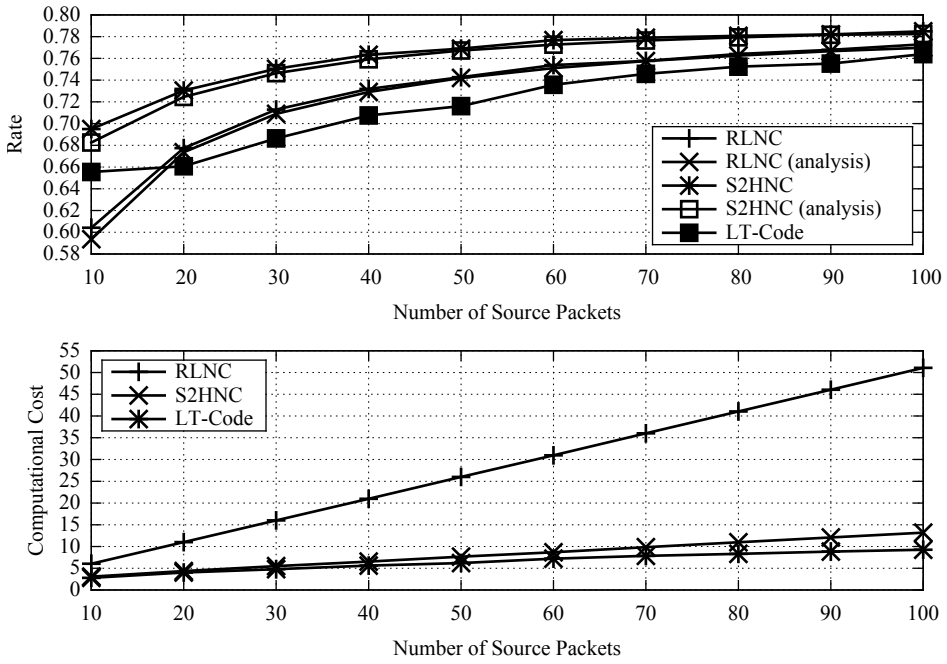

Figure 3.9: Comparison of S2HNC and RLNC for small M, $\epsilon_1 = 0.05$, $\epsilon_2 = 0.2$.

The expected rates, $M/\mathrm{E}\{T\}$, of S2HNC and RLNC are also provided in Figure 3.9, where $\mathrm{E}\{T\}$'s are calculated using techniques in Section 3.3.1, respectively. The analysis results match simulations closely. In Figure 3.10, we show the standard deviations (i.e., square root of variance) of the completion times of S2HNC and RLNC, which are calculated using (3.4). It is seen that the deviations are small, suggesting that using expected completion time to characterize the actual completion time is sound.

In Figures. 3.11-3.13, we show the rate losses for different buffer sizes at R. We fix $\epsilon_2 = 0.2$ in the three plots and simulate $\epsilon_1 = 0.05, 0.2$ and 0.3, respectively for $M = 100$. In each figure we have included the best achievable rate loss performance calculated from (3.7).

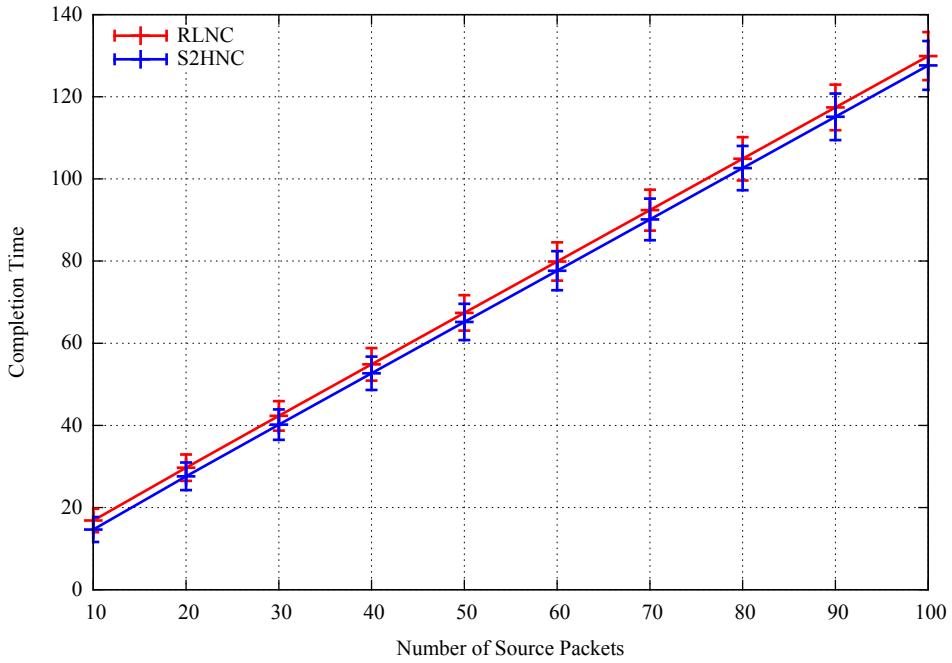

Figure 3.10: Expected completion times and standard deviations of S2HNC and RLNC for small M, $\epsilon_1 = 0.05$, $\epsilon_2 = 0.2$.

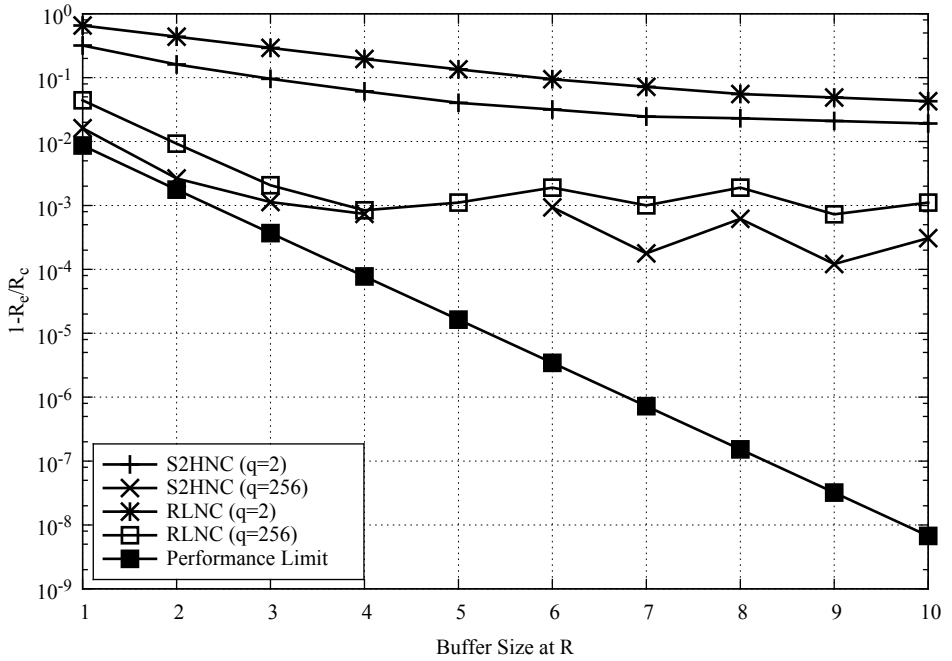

Figure 3.11: Rate loss due to limited buffer size. $M = 100$, $\epsilon_1 = 0.05$ and $\epsilon_2 = 0.2$.

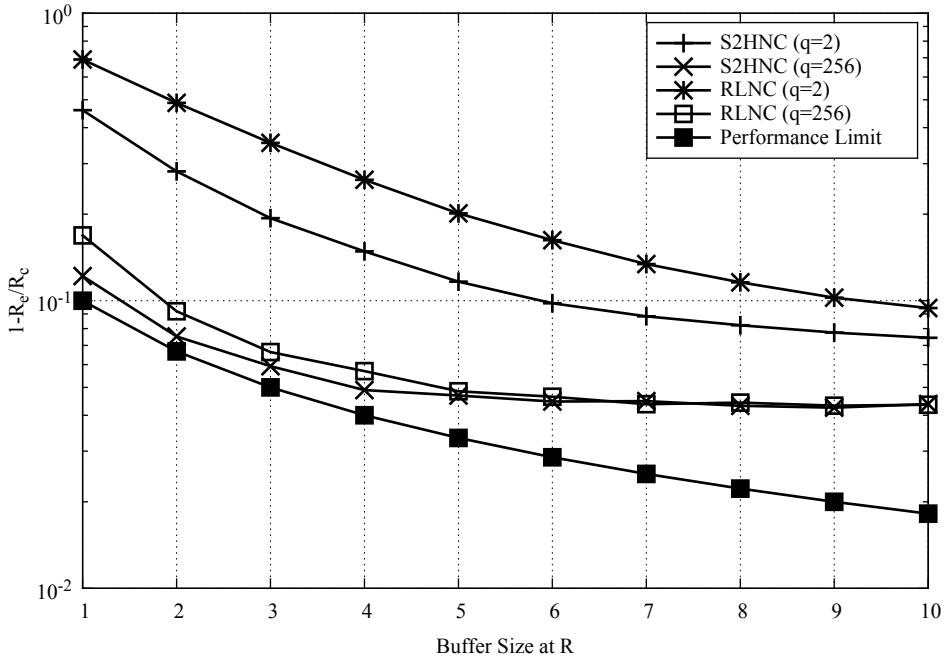

Figure 3.12: Rate loss due to limited buffer size. $M = 100$, $\epsilon_1 = 0.2$ and $\epsilon_2 = 0.2$.

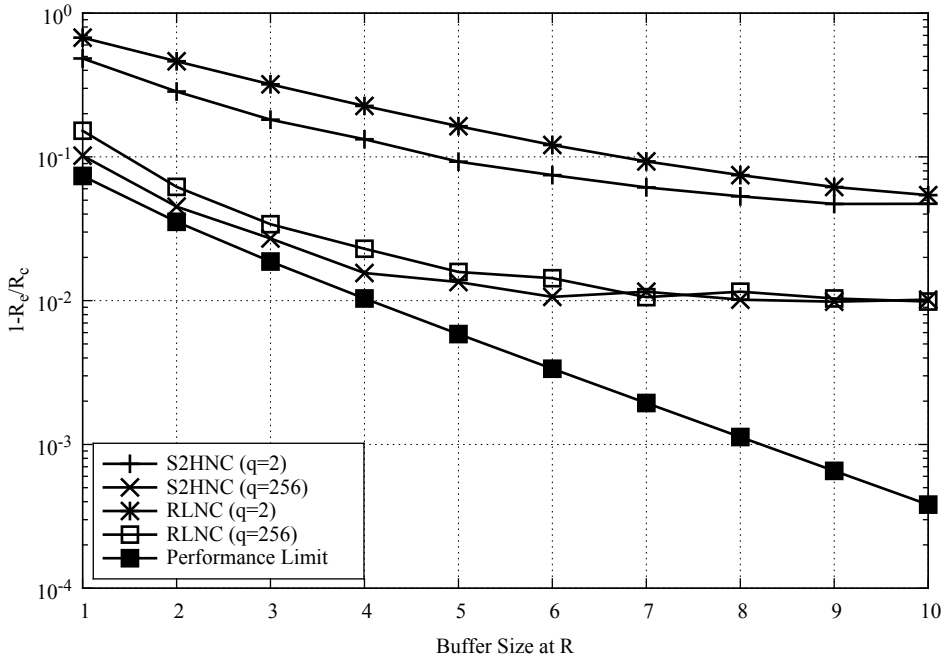

Figure 3.13: Rate loss due to limited buffer size. $M = 100$, $\epsilon_1 = 0.3$ and $\epsilon_2 = 0.2$.

We see that both S2HNC and RLNC would not approach the performance limit for the small M even when high order finite fields are used. However, in all cases S2HNC has better performance than RLNC for the same fixed field size.

3.6 Discussion and Summary

In this chapter, we proposed to use systematic network coding for two-hop lossy link where transmissions occur successively on the two hops. For this specific network topology and traffic model, we showed that we do not have to always perform the high-complexity random linear network coding to achieve the max-flow capacity. Sending some uncoded packets is shown to be advantageous compared to sending randomly coded ones all the time.

Compared to random linear network coding, which is capacity achieving for large numbers of source packets and sufficiently large finite field sizes, our proposed scheme is observed to have better performance in terms of both end-to-end rate and decoding complexity in the finite regime where fixed M and q are given, and even if limited size buffer is used at the relay node. The improvement is especially appreciable when the number of packets is small and binary field is used. The scheme may have potential use in many applications that consist of two-hop lossy links. One particular scenario is WiFi offloading, where a WiFi AP is employed to help base station to deliver data packets from a gateway to users. The transmission in the WiFi offloading scenario mostly involves with only a small number of packets.

Chapter 4

Systematic Network Coding for Known Network Topologies: Parallel Two-Hop Links

4.1 Introduction

In Chapter 3, we saw that using systematic coding in two-hop lossy links may improve end-to-end rates and reduce computational cost of decoding compared to using RLNC. In this chapter, we extend the results in Chapter 3 to a network where parallel two-hop lossy links are used simultaneously for transmission.

An example scenario where the topology of the parallel two-hop lossy links found is in the WiFi offloading mode of cellular networks. The rapidly increasing data traffic with the advent of smart-phones and tablets has been posing great challenges to current cellular networks. To meet the explosively growing demand, many cellular operators start to utilize WiFi networks to transmit part of data. WiFi offloading is attractive financially because WiFi networks have lower infrastructure cost and more flexibility. Users also benefit because WiFi access may be cheaper and the simultaneous use of cellular and WiFi connections increases throughput and reliability.

In WiFi offloading, data packets may be transmitted from P-GW (Packet Data

Network Gateway) to user equipment (UE) on two parallel links simultaneously to speed up the transmission; one is through the base station (BS) and the other one is through the WiFi access point (AP), as shown in Figure 4.1. The multi-path TCP (MPTCP) [12] protocol can be used to seamlessly establish/close the two two-hop connections. The fountain code properties of network coding are potentially useful in this network, in which the intermediate nodes of the two two-hop links may re-encode previously received packets to achieve the max-flow capacity of the link. In [9], it is shown that incorporating RLNC with MPTCP may achieve higher throughput than MPTCP without network coding when links are lossy, in which the concept "seen packet" proposed in [61] is used to address the compatibility issue of network coding with TCP.

In this chapter, we show that systematic network coding is beneficial when parallel two-hop lossy links are available. While we present the chapter in the context of WiFi offloading, these techniques may be extended to other parallel multi-hop lossy link topologies. Following the methodology from Chapter 3, we show that using systematic network coding outperforms RLNC in terms of end-to-end rates and decoding costs. In the following, since parallel paths are available, we formulate a packet allocation problem to schedule the transmission of uncoded (which will be innovative) packets among the two paths such that the number of received uncoded packets at the receiver is maximized, so as to minimize the expected completion time of the transmission and the decoding cost.

The rest of the chapter is organized as follows. In Section 4.2 we present the system model of the problem and describe the coding methods for the system. In Section 4.3 we establish the advantages of the proposed coding method compared to RLNC,

and we formulate an uncoded packet allocation problem to optimize transmission. Simulation results are presented in Section 4.4. We supply the proof for the theorem of Section 4.3 in Section 4.6.

4.2 System Model

4.2.1 Offloading Transmission Model

Suppose that a packet data network gateway, denoted as GW, has M data packets $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_M\}$ to send to a UE. Each packet consists of K symbols from $\mathbb{F}_q$. Two parallel paths exist for the transmission as shown in Figure 4.1; each is a two-hop lossy link. We use GW-BS-UE and GW-AP-UE to refer to the two two-hop links through a cellular basestation (BS) and WiFi access point (AP), respectively.

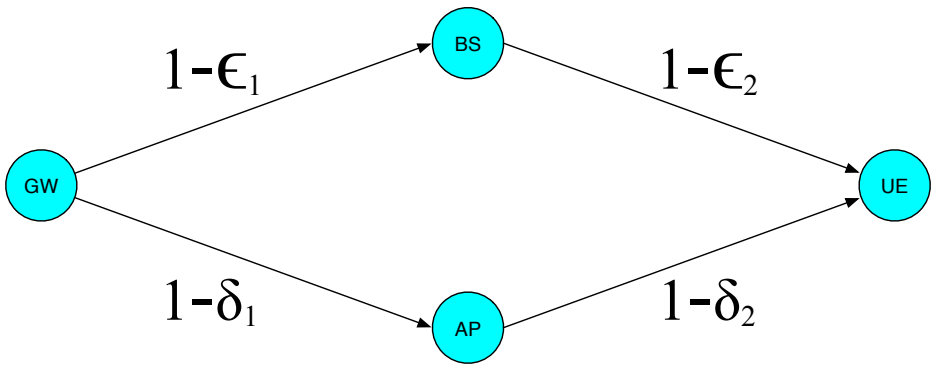

Figure 4.1: Network topology of WiFi offloading.

We assume that BS and AP operate in the half-duplex mode. Transmission occurs successively on the two hops of each path and the two successive transmissions is referred to as a *use* of the path. Let rate R_1 define the number of packets that are sent on GW-BS or BS-UE per transmission, and let R_2 define the number of packets that are sent on GW-AP or AP-UE per transmission. Links are assumed

delay-free, meaning that transmitted packets will be received immediately by the downstream node if not erased. Given the rates R_1 and R_2, let ϵ_1, ϵ_2, δ_1 and δ_2 denote the corresponding packet loss probabilities on links GW-BS, BS-UE, GW-AP and AP-UE, respectively. The max-flow capacities of the parallel two paths can then be expressed as $C_1 = R_1(1 - \max\{\epsilon_1, \epsilon_2\})$ and $C_2 = R_2(1 - \max\{\delta_1, \delta_2\})$ packets per path use, respectively. A *network use* refers to using each path once. The max-flow capacity of the network is then $C_1 + C_2$ packets per network use.

Throughout the transmission, we assume that no feedback is available between nodes to indicate loss/reception of packets except that the UE can inform GW, BS and AP when it finishes recovering all the M source packets. We denote the expected number of network uses that are needed by UE to complete decoding of all packets as the expected *completion time* of the transmission, T.

4.2.2 Random Linear Network Coding

In the RLNC scheme, GW always sends random linear combinations of source packets while BS and AP send random linear combinations of packets in their buffers, respectively.

Received packets at UE are in the form of $\mathbf{r} = \sum_{i=1}^{M} g_i \mathbf{s}_i$, where g_i is a random coding coefficient chosen from $\mathbb{F}_q$. The vector $[g_1, \ldots, g_M]$ is referred to as the *coding vector* of $\mathbf{r}$. UE does not need to distinguish whether the received packets are from BS or AP.

Suppose that N packets $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N$ are received. The recovery of source packets

corresponds to solving the linear system of equations

$$\begin{bmatrix} g_{1,1} & \cdots & g_{1,M} \\ \vdots & \ddots & \vdots \\ g_{N,1} & \cdots & g_{N,M} \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_M \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_N \end{bmatrix}, \tag{4.1}$$

where the matrix of the left-hand side is referred to as the *decoding matrix*, whose rows are coding vectors of received packets and the $N \times K$ right-hand side matrix consists of information symbols of the received coded packets. An *innovative packet* refers to a received packet whose coding vector is not in the span of coding vectors of all previously received packets. Decoding is only possible if M innovative packets are received, i.e., when the decoding matrix is full-rank. As the decoding matrix in (4.1) is unstructured for RLNC, $\mathcal{O}(M^3 + M^2K)$ operations are required for GE over the finite field, where an *operation* refers to either a divide or multiply-and-add finite field operation. Operations reduce to XOR if $\mathbb{F}_2$ is used.

RLNC is capacity-achieving in the asymptotic regime; it achieves the minimum expected completion time $T_{\min} = \frac{M}{C_1+C_2}$ when M and q are sufficiently large [22].

4.2.3 Systematic 2-Hop Network Coding

A systematic 2-hop network coding (S2HNC) scheme for the network topology of Figure 4.1 refers to where the first M_1 and M_2 packets sent from GW to BS and AP, respectively, are uncoded source packets. After that, RLNC coded packets (i.e., random linear combinations of packets) are sent until UE successfully decodes all M source packets. At both BS and AP, every received packet is stored in its buffer. If BS or AP receives an uncoded packet, the packet is forwarded to UE; if no packet

is received or the received packet is RLNC coded, a RLNC packet is generated from the buffer for transmission to UE.

The set (M_1, M_2) is referred to as an *uncoded packet allocation* of the S2HNC scheme; different choices of (M_1, M_2) result in different performances as we will show.

The decoding at UE is similar to that of the RLNC except that some received packets may be uncoded, which means that some rows in the decoding matrix are singleton, i.e., contain only one nonzero element.

In the following parts of the chapter, we quantify the S2HNC scheme's reduction in the expected completion time and computation over that of RLNC, as a function of given packet loss probabilities.

4.3 S2HNC Advantages and Design

4.3.1 Completion Time

Lemma 4.1. *The M_1 and M_2 transmitted uncoded packets on the two paths being distinct is a necessary condition for S2HNC to achieve shorter expected completion time than RLNC.*

Proof. For any M received RLNC packets at UE, packets are linearly independent of one another as $q \to \infty$. Therefore, the RLNC scheme finishes at $T_{\min} = \frac{M}{C_1+C_2}$ with $(R_1 + R_2)T_{\min}$ packets sent from GW with probability approaching one as $q \to \infty$ [22]. Now assume that the same number of packets are sent from GW using S2HNC, among which there are $M_1 + M_2$ uncoded packets. At UE, this results in $M_u = (1-\epsilon_1)(1-\epsilon_2)M_1 + (1-\delta_1)(1-\delta_2)M_2$ uncoded packets and $M - M_u$ RLNC coded packets. As $q \to \infty$, decoding is successful if and only if the M_u uncoded packets are distinct. However, if there is any uncoded packet sent more than once, then there

is a constant probability (depends on packet loss probabilities but not q) that would not decrease to zero as $q \to \infty$ that the uncoded packet is received for more than once at UE, i.e., the probability that M_u packets are distinct does not approach zero as $q \to \infty$, resulting in a constant decoding failure probability at $T_{\min}$. The expected completion time of such a S2HNC scheme is not shorter than that of RLNC. □

Theorem 4.1. *Any S2HNC uncoded packet allocation with the M_1 and M_2 uncoded packets being distinct has shorter expected completion time than the RLNC scheme.*

Proof. See Section 4.6. □

In the rest of this chapter, when we refer to S2HNC, we mean that uncoded packets sent on the two paths are distinct. Note that Theorem 4.1 holds regardless of the values of M, q and packet loss probabilities of links. Therefore, S2HNC achieves RLNC's minimum completion time asymptotically, as $M \to \infty$ and $q \to \infty$, and in the finite regime where M and q may be small, S2HNC has shorter expected completion time.

Corollary 4.1. *The allocation (M_1, M_2) that results in the most uncoded packets received at UE has the shortest expected completion time among all S2HNC uncoded packet allocations.*

Proof. The proof is straightforward from the proof of Theorem 4.1, in which it is shown that any received uncoded packet at UE is innovative with a higher probability compared to if a coded packet were received. □

4.3.2 Computational Cost and Uncoded Packet Allocation

Depending on how many uncoded packets are received by UE, S2HNC exhibits different computational cost compared to that of RLNC. Assume that M_u uncoded packets are received, then UE needs to decode only the remaining $M - M_u$ RLNC coded packets. This corresponds to a savings of a factor of $\frac{(M-M_u)^2}{M^2}$ computations in solving the linear system of equations.

To reduce computational cost, we can maximize M_u. Based on Corollary 4.1, maximizing M_u results in the uncoded packet allocation that achieves the minimum expected completion time. The maximization can be done by solving an optimization problem to determine M_1 and M_2. The optimization, however, requires to know the link parameters, R_1, R_2, ϵ_1, ϵ_2, δ_1 and δ_2. The allocation problem can be formulated as follows:

$$\begin{aligned} \underset{M_1,M_2}{\text{maximize}} \quad & (1-\epsilon_1)(1-\epsilon_2)M_1 + (1-\delta_1)(1-\delta_2)M_2 \\ \text{subject to} \quad & M_1 + M_2 \leq M \\ & \frac{M_1}{R_1} \leq \frac{M}{C_1 + C_2} \\ & \frac{M_2}{R_2} \leq \frac{M}{C_1 + C_2} \\ & M_1, M_2 \in \mathbb{Z}^*, \end{aligned} \tag{4.2}$$

where the first constraint is that at most M distinct uncoded packets can be sent and the next two constraints ensure that uncoded packets on either path should be transmitted within the first $T_{\min}$ network uses. These constraints arise to avoid uncoded packets being "over-allocated" to one of the paths. If over-allocated on one of the paths, the allocated uncoded packets may not actually be all transmitted before UE finishes decoding; the remaining uncoded packets could have been transmitted

on the other path.

We observe that $M_1 + M_2 = M$ always holds according to Corollary 4.1, because GW has to send at least M packets and that distinct uncoded packets are always better than coded ones. We can therefore solve (4.2) and obtain M_1 in closed form as

$$M_1 = \begin{cases} \lfloor \frac{MR_1}{C_1 + C_2} \rfloor & (1-\epsilon_1)(1-\epsilon_2) > (1-\delta_1)(1-\delta_2) \\ M - \lfloor \frac{MR_2}{C_1 + C_2} \rfloor & \text{otherwise,} \end{cases} \tag{4.3}$$

and $M_2 = M - M_1$, where $\lfloor \cdot \rfloor$ maps a real number to the largest previous integer.

With optimized (M_1, M_2), GW can choose any disjoint subsets of source packets of sizes M_1 and M_2 as the first transmitted packets on GW-BS and GW-AP, respectively.

4.4 Simulation Results

We compare the performances of RLNC and S2HNC by simulations where coding is performed in $\mathbb{F}_2$. The simulations use $K = 8192$ bits per packet and $\mathbb{F}_2$ is used when performing coding. In measuring the computational cost, we count the number of operations, N_{ops}, that are needed in recovering all the source packets in software implementations. The measurement we use is the average number of operations per packet symbol, which is equal to $\frac{N_{ops}}{MK}$. Each curve of the following simulation is averaged over $10,000$ trials.

The first set of network parameters of simulations are $R_1 = 1$, $R_2 = 2$, $\epsilon_1 = 0.01$, $\epsilon_2 = 0.05$, $\delta_1 = 0.01$ and $\delta_2 = 0.1$. This combination of parameters corresponds to where the WiFi offloading path may have higher transmission rate but also higher packet loss rate (e.g. due to congestion) and where the first hops of each of the two paths are more reliable than the second.

In Figures. 4.2 and 4.3, we show the averaged completion times and computational costs of using RLNC and S2HNC schemes, respectively. The solutions of the (4.2) for some M are given in Table 4.1.

M	20	40	60	80	100	120	140	160	180	200
M_1	7	14	21	29	36	43	50	58	65	72
M_2	13	26	39	51	64	77	90	102	115	128

Table 4.1: Optimized uncoded packet allocations for $R_1 = 1, R_2 = 2, \epsilon_1 = 0.01, \epsilon_2 = 0.05, \delta_1 = 0.01, \delta_2 = 0.1$, as a function of number of source packets, M.

In addition to the optimized allocation obtained by solving (4.2), for comparison purposes we also include three other allocations for S2HNC, namely all-via-BS and all-via-AP in which uncoded packets are only sent to BS and AP, respectively, and *proportional* where uncoded packets are allocated as $M_1 \approx MR_1/(R_1 + R_2)$ and $M_2 \approx MR_2/(R_1 + R_2)$. To clearly illustrate the comparison, we plot the average end-to-end rates of schemes, $\frac{M}{T}$, which are inversely proportional to the averaged completion times. Achieving the minimum completion time is equivalent to achieving the max-flow capacity between GW and UE, which is equal to $C_1 + C_2 = 2.75$ packets per network use for the parameters used.

It is clear that all S2HNC allocations achieve higher end-to-end rates than RLNC. Although the gap decreases as M increases, the rate improvement is considerable for small M. As expected, RLNC requires the most computations in decoding. Different allocations of uncoded packets result in different computational costs. The optimized allocation results in the least computational cost and a significant saving in computational cost compared to RLNC is obtained. It is noted that the proportional allocation achieves almost the same rate and computational cost as the optimized

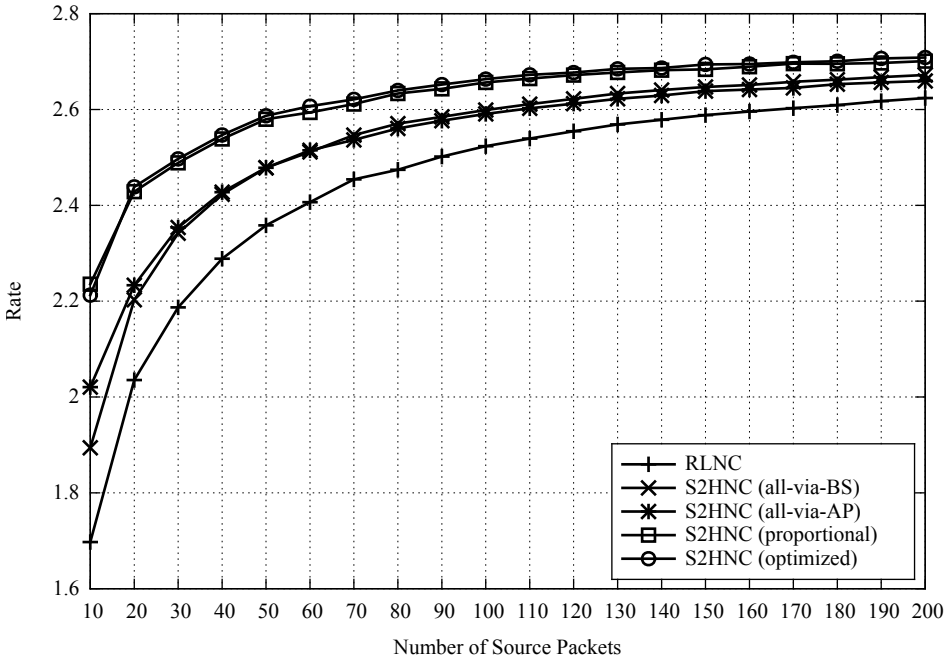

Figure 4.2: Rates of using RLNC and S2HNC as a function of number of source packets, M, with uncoded packet allocation schemes and $R_1 = 1$, $R_2 = 2$, $\epsilon_1 = 0.01$, $\epsilon_2 = 0.05$, $\delta_1 = 0.01$, $\delta_2 = 0.1$.

allocation. This is anticipated because the allocation in Table 4.1 is very close to being proportional to R_1/R_2.

We present simulation results for another set of parameters $R_1 = 1$, $R_2 = 4$, $\epsilon_1 = 0.01$, $\epsilon_2 = 0.5$, $\delta_1 = 0.01$ and $\delta_2 = 0.05$, which corresponds to that the cellular path has lower quality in terms of both transmission rate and packet loss rate. The max-flow capacity of the scenario is $C_1 + C_2 = 4.3$ packets per network use. The achieved rates and required computational costs of using RLNC and S2HNC with different uncoded packet allocations are shown in Figures 4.4 and 4.5, respectively. Again, all the S2HNC schemes achieve higher rates and lower computational costs than the RLNC scheme.

Since the WiFi offloading path has much higher rate and better quality than

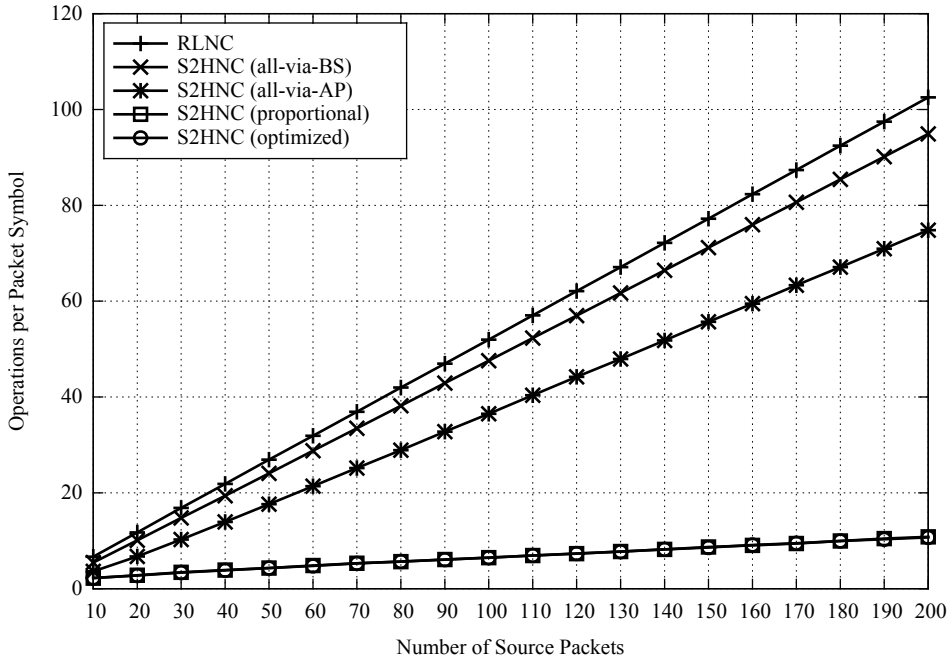

Figure 4.3: Computational costs of using RLNC and S2HNC as a function of number of source packets, M, with various uncoded packet allocation scehmes and $R_1 = 1$, $R_2 = 2$, $\epsilon_1 = 0.01$, $\epsilon_2 = 0.05$, $\delta_1 = 0.01$, $\delta_2 = 0.1$.

the cellular path in this scenario, we expect that the proportional and the all-via-AP allocations have performances close to that of the optimized allocation. The optimal allocation solution is provided in Table 4.2, where most of uncoded packets are allocated to the WiFi path. In Figure 4.4, the achieved rates of proportional, all-via-AP and optimized allocations are almost the same. However, it is seen that the three allocations have considerable difference in computational costs at the receiver as shown in Figure 4.5. In particular, we note that the optimal allocation in Table 4.2 is less close to being proportional to R_1/R_2 as in the case of Table 4.1, which explains the gap of computational cost between the optimal and the proportional allocations, indicating the importance of judicious use (allocation) of every uncoded packet.

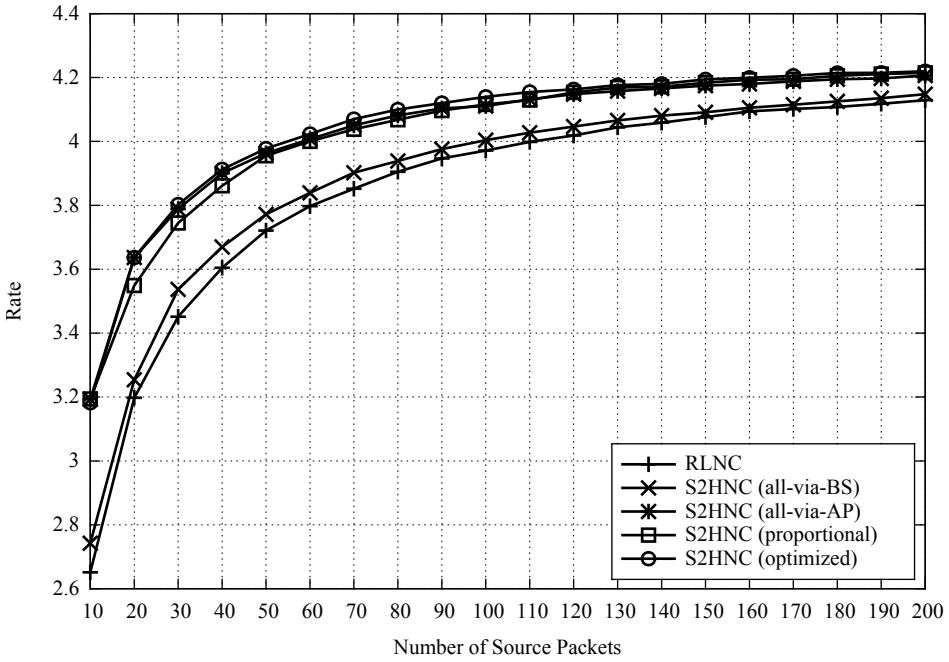

Figure 4.4: Rates of using RLNC and S2HNC as a function of number of source packets, M, with uncoded packet allocation schemes and $R_1 = 1$, $R_2 = 4$, $\epsilon_1 = 0.01$, $\epsilon_2 = 0.5$, $\delta_1 = 0.01$, $\delta_2 = 0.01$.

M	20	40	60	80	100	120	140	160	180	200
M_1	2	3	5	6	7	9	10	12	13	14
M_2	18	37	55	74	93	111	130	148	167	186

Table 4.2: Optimized uncoded packet allocations for $R_1 = 1, R_2 = 4, \epsilon_1 = 0.01, \epsilon_2 = 0.5, \delta_1 = 0.01, \delta_2 = 0.05$, as a function of number of source packets, M.

4.5 Conclusion

In this chapter we presented a systematic 2-hop network coding method for parallel two-hop lossy link transmission. The coding method can be used in WiFi offloading transmission. The computational cost and completion time are quantified and compared to random linear network coding. The improvements are considerable when the number of source packets of the transmission is small, which may often occur

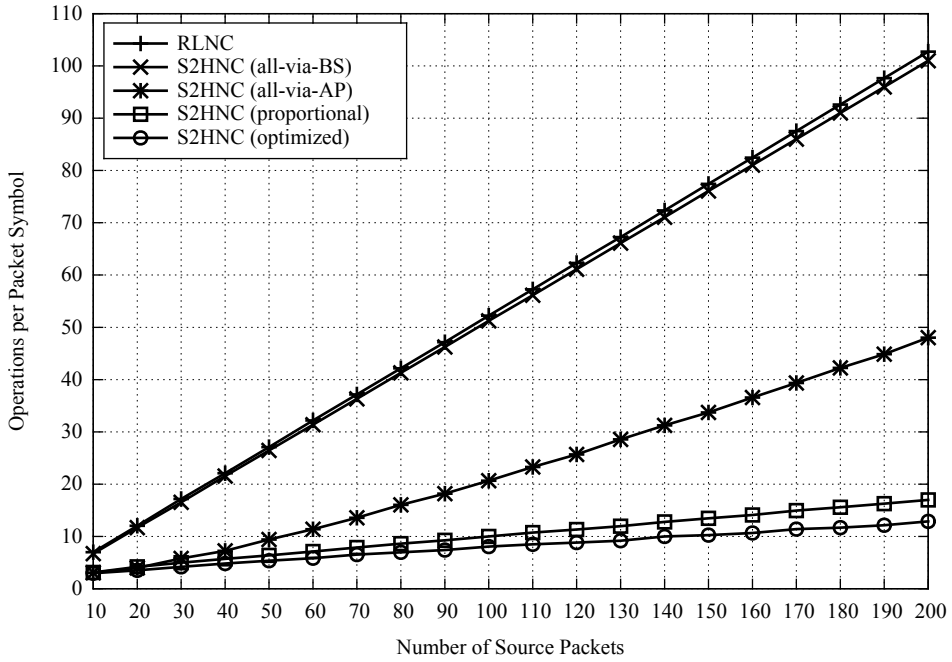

Figure 4.5: Computational costs of using RLNC and S2HNC as a function of number of source packets, M, with various uncoded packet allocation scehmes and $R_1 = 1$, $R_2 = 4$, $\epsilon_1 = 0.01$, $\epsilon_2 = 0.5$, $\delta_1 = 0.01$, $\delta_2 = 0.05$.

in practice. A key point observed from the proposed method is that the use of uncoded packets (systematic coding) provides a benefit in WiFi offloading transmission network topology. We have formulated an uncoded packet allocation problem for determining the optimal allocation of uncoded packets on the parallel paths, resulting in maximized end-to-end rate and minimized decoding cost.

4.6 Proof of Theorem 4.1

We prove the theorem by examining the probability that a received packet at UE is innovative with respect to (w.r.t.) previously received packets. S2HNC will be shown to have a larger such probability at any time than RLNC. The completion time is inversely proportional to this probability because the process of UE collecting

innovative packets is Markovian, i.e., the probability that whether a received packet is innovative only depends on the number of innovative packets that UE has received so far.

Consider a S2HNC with fixed M_1 and M_2. In the first $T_1 = \min(\frac{M_1}{R_1}, \frac{M_2}{R_2})$ network uses, packets transmitted on paths GW-BS and GW-AP are all distinct uncoded packets. The information flows on the two paths are therefore independent. Using the results from Section 3.3, it is clear that on either path using S2HNC will result in more innovative packets at UE than using RLNC after T_1 network uses.

During the time period from T_1 to $T_2 = \max(\frac{M_1}{R_1}, \frac{M_2}{R_2})$, one of the paths will be transmitting RLNC coded packets while the other one is transmitting uncoded packets in S2HNC. Without loss of generality, we assume that GW-BS is the one transmitting RLNC coded packets, i.e., $\frac{M_1}{R_1} < \frac{M_2}{R_2}$. Now the proof reduces to showing that transmitting uncoded packets on GW-AP while transmitting RLNC coded packets on GW-BS is indeed beneficial compared to transmitting RLNC coded packets on both GW-AP and GW-BS.

Let $r_1^{(T_1)}$ and $r_2^{(T_1)}$ denote the numbers of innovative packets that have been received by UE through GW-BS-UE and GW-AP-UE after T_1 network uses, respectively. Let $r_1^{(t)}$ and $r_2^{(t)}$ denote the numbers of innovative packets that are received through GW-BS-UE and GW-AP-UE during (T_1, t), $T_1 \leq t \leq T_2$. A new received packet after t network uses is innovative if and only if it is linearly independent w.r.t. all the $r_1^{(T_1)} + r_2^{(T_1)} + r_1^{(t)} + r_2^{(t)}$ packets. For a given number of innovative packets at AP, if GW-AP was unsuccessful, the probability that UE receives an innovative packet from AP in the next transmission is then the same using S2HNC and RLNC schemes. However, if GW-AP was successful when using S2HNC, a new uncoded

packet is received by AP, which is innovative with probability 1 w.r.t. previously received innovative packets at AP and the $r_1^{(T_1)}$, $r_2^{(T_2)}$ and $r_2^{(t)}$ innovative packets at UE because these $r_1^{(T_1)} + r_2^{(T_2)} + r_2^{(t)}$ packets are in the span of other distinct uncoded packets. The forwarded uncoded packet is non-innovative if and only if it is in the span of the $r_1^{(t)}$ RLNC packets. The probability of such event is no larger than that when using RLNC because a RLNC coded packet may also be in the span of the other $r_1^{(T_1)} + r_2^{(T_2)} + r_2^{(t)}$ packets. Again, uncoded packets are innovative for AP with probability 1 and therefore AP collects more innovative packets than the RLNC scheme. Therefore, transmitting uncoded packets on GW-AP is advantageous.

Therefore, in the first T_2 network uses, S2HNC always results in more innovative packets at BS, AP and UE compared to RLNC. After T_2, S2HNC and RLNC have the same behavior. Altogether, S2HNC achieves shorter completion time.

Chapter 5

Generation-Based Network Coding for Unknown Network Topologies: Generation-by-Generation Decoding

5.1 Introduction

In Chapters 3 and 4, we see that in networks with simple topologies such as line or parallel multi-hop networks, the benefits of network coding may be obtained without the need for random linear network coding (RLNC). Sending some uncoded packets or sparsely coded packets [47] may achieve end-to-end rates close to or even higher than RLNC at a much lower decoding cost.

However, in networks whose topology is complex or unknown during the transmission, the design and maintenance of the sparse structure of codes will be more challenging. A particular scenario of interest that has such network topology is peer-to-peer file sharing, in which an overlay network is formed on the fly and nodes may exit the network at random times. When coded packets pass through multiple intermediate nodes who are not aware of the network topology and therefore have to

perform RLNC, the resulting decoding matrix inevitably gets dense [41]. The sparseness of codes is lost and the decoding complexity increases.

In networks with unknown topologies, a possible solution is to group source packets into subsets called *generations* and network coding is restricted to be performed among packets belonging to the same generation [8]. Each received packet is then guaranteed to be coded across only a small number of source packets; the sparsity is therefore maintained throughout the network even if nodes are not aware of the neighboring topology. In [44], it is proposed that intermediate nodes may randomly *schedule* generations to perform coding and hence nodes do not require downstream acknowledgement. The random scheduling of generations enables the overall coding method to have the same fountain property as RLNC but be much sparser. This type of acknowledgement-free scheme is referred to as generation-based network coding (GNC) and will be the focus of this and the following chapter [1].

A GNC scheme consists of three major components: a *GNC code* at the source node defining the mapping from source packets to coded packets, a *scheduling strategy* at intermediate nodes determining from which generation to re-encode a packet, and a *GNC decoder* at the destination. The GNC code specifies how generations are formed and how coded packets are generated from each generation. Generations may be disjoint [44] or overlapping [59], [17], [63], [32], and may be of equal size [44], [17], [63], [32] or of unequal size [59], [31]. Scheduling strategies such as random scheduling [44], random push [65], local-rarest-first scheduling [66] and transmitted-innovation-maximizer scheduling [19] can be used at intermediate nodes, where the random scheduling requires no feedback between nodes while the latter three can

[1]It is noted that nothing prevents GNC schemes from being employed in networks considered in Chapters 3 and 4, but in this chapter we assume an unknown network topology.

make use of feedback to improve performance.

At the destination, a GNC decoder is employed to recover source packets from coded packets that may belong to different generations. In this chapter, we consider the *generation-by-generation* (G-by-G) decoder as in [44], [59], [63], [32], [62], [42], [31]: the decoder solves linear systems of equations within each generation. A generation that can be decoded from coded packets belonging to itself is referred to as *separately decodable*. If there is overlap between generations, decoded packets of a generation will be substituted into the remaining generations that also contain the decoded source packets. The decoding continues as long as new separately decodable generations can be found and fails otherwise. Compared to jointly decoding generations, which will be considered in Chapter 6, the advantage of G-by-G decoding is that the complexity is upper bounded by the largest generation size and therefore the decoding cost can be maintained low.

A key metric of GNC schemes is the reception *overhead*, which is defined as the extra received packets required beyond the number of source packets to decode all the source data. Generally, the causes of overhead can be classified into three types: *encoder-induced*, *decoder-induced*, and *network-induced*. Encoder-induced overhead (or code overhead) is from GNC encoding, which is introduced if among a random subset of M generated packets there are fewer than M linearly independent ones; decoder-induced overhead is incurred if a decoder is unable to decode as soon as M linearly independent packets are received; network-induced overhead may be incurred if scheduling or re-encoding at intermediate nodes introduces linear dependency.

The G-by-G decoder has nonzero decoder-induced overhead: in each step of decoding, a separately decodable generation has to be found; otherwise the decoder is

stuck until more packets are received. This process may cause overhead because even if sufficient number of linearly independent packets have been received it is still not guaranteed that separately decodable generation can be found among them.

The aim of this chapter is to design GNC codes for the G-by-G decoder to have low reception overhead. We will focus mainly on code and decoder-induced overheads and design GNC codes that have low overhead when applying the G-by-G decoding directly on the GNC coded packets generated at the source node. Therefore, we do not distinguish between code and the decoder-induced overheads and we simply use *code overhead* to refer to either code and decoder-induced overheads in the rest of the chapter.

Although intermediate nodes may also incur reception overhead via scheduling and re-encoding, we will only discuss network-induced overhead very briefly. Here, we propose a new scheduling strategy to replace the existing random scheduling strategy of [44] and show that it reduces network-induced overhead.

In [59], [17] and [63], it is shown that when constructing GNC generations, overlapping generations may have reduced overhead than disjoint generations if the amount of overlap is properly chosen. We extend this line of work in this chapter to design GNC codes based on overlapping generations. To facilitate design, we first use an and-or tree analysis based technique [34] to analyze the G-by-G decoding of GNC codes and then design GNC codes based on the analysis.

The rest of the chapter is organized as follows: Section 5.2 presents network and buffer models, and describes encoding and decoding operations of generation-based network coding. Section 5.3 briefly examines the overhead that may be caused by local scheduling and proposes the maximum local potential innovativeness (MaLPI)

scheduling strategy to replace the random scheduling proposed in [44]. Section 5.4 models the GNC codes as graph codes and uses the and-or tree analysis technique to analyze the decoding of GNC codes. In Sections 5.5 and 5.6, we present two GNC code designs based on the obtained analysis results from Section 5.4 and evaluate performance through simulations. Section 5.7 provides simulation results of codes and the proposed MaLPI scheduling applied to some example network environments. The chapter is summarized in Section 5.8.

5.2 System Model

5.2.1 Network Model

The network is modeled as a directed acyclic graph, denoted as $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of $|\mathcal{V}|$ vertices that represents nodes in the network and $\mathcal{E}$ is the set of $|\mathcal{E}|$ edges that represents links between nodes. The sets of incoming and outgoing links of a node $v \in \mathcal{V}$ are denoted as $\text{In}(v)$ and $\text{Out}(v)$, respectively. Links are modeled as memoryless erasure channels. We use (i, j) to denote a link connecting nodes i and j, and denote its erasure probability as $p_{e_{ij}}$, which is fixed throughout the transmission.

We consider that M source packets, $\mathbf{s}_1, \ldots, \mathbf{s}_M$, are to be sent from a source node $s \in \mathcal{V}$ to a destination node $t \in \mathcal{V}$ over the network. Each packet consists of K symbols from a finite field $\mathbb{F}_q$, where q is the field size.

For each transmission opportunity at a node, a packet is sent to each of the downstream nodes of the node. If no erasure occurs, the packet is received immediately by the neighboring node. We assume that nodes have no knowledge of global network topology and are not able to exchange their buffer states information with each other

to coordinate transmissions. We assume that the destination node can only acknowledge to the source node when it successfully recovers all the M source packets.

5.2.2 Generation-Based Network Coding

We now describe the detailed generation-based network coding (GNC) method. In the GNC scheme, source packets may be first *precoded* using a traditional fixed-rate erasure correction code. A total of $(1+\theta)M$ *intermediate packets*, $\mathcal{S} = \{\mathbf{s}_i, 1 \leq i \leq (1+\theta)M\}$, are generated from the M source packets if a precode of rate $1/(1+\theta), \theta > 0$, is applied. We assume that $(1+\theta)M$ is an integer. Without loss of generality, we assume a systematic precode, i.e., the first M intermediate packets are identical to the original M source packets, and intermediate packets of indices $M+1, M+2, \ldots, (1+\theta)M$ are parity-check packets. The source is said to be *not-precoded* if $\theta = 0$. The reason of precoding will be clear when we analyze the decoding of GNC codes.

The intermediate packets are then grouped into generations. For convenience, below we refer to packets in generations as *intermediate packets* even if the source packets are not-precoded. Each generation is a subset of $\mathcal{S}$. A GNC code is defined on a set of L generations $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_L\}$, where $\mathcal{G}_l = \{\mathbf{s}_1^{(l)}, \mathbf{s}_2^{(l)}, \ldots, \mathbf{s}_{G_l}^{(l)}\}$, $1 \leq l \leq L$ are groupings of the $(1+\theta)M$ intermediate packets, in which $\mathbf{s}_i^{(l)} = \mathbf{s}_j$ for some j. Size of generation $\mathcal{G}_l$ is denoted as G_l. We assume that $\cup_{l=1}^{L} \mathcal{G}_l = \mathcal{S}$, i.e., each intermediate packet is present in at least one generation. We define $d_{\min} \triangleq \min_l G_l$, $d_{\max} \triangleq \max_l G_l$, and $d_{\text{avg}} \triangleq (1/L)\sum_{l=1}^{L} G_l$, where d_{avg} is the average generation size. We assume d_{avg} to be an integer. The generations are said to be of *equal-size* if $G_i = G_j, \forall i, j$, or of *unequal-size* if $G_i \neq G_j$ for some i, j. Generations are called *disjoint* if $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset, \forall i \neq j$, or *overlapping* if there exists $\mathcal{G}_i \cap \mathcal{G}_j \neq \emptyset$ for

some $i \neq j$. For overlapping generations we have $\sum_{l=1}^{L} G_l > (1+\theta)M$ and refer to $(\sum_{l=1}^{L} G_l - (1+\theta)M)$ as the *amount of overlap*. In this chapter, we consider GNC codes based on overlapping generations and treat GNC codes based on disjoint generations as special cases. When not specified, we assume that intermediate packets in each generation could be formed randomly from $\mathcal{S}$.

An overview of the considered system is shown in Figure 5.1. A GNC code is defined on $\mathcal{G}$ at the source node as follows. For each transmission from the source node, a coded packet is generated as a random linear combination of source packets in a randomly chosen generation; coefficients are chosen from $\mathbb{F}_q$. Let $\mathcal{P} = \{p_1, p_2, \ldots, p_L\}$ be the distribution where p_i denotes the probability that $\mathcal{G}_i$ is chosen when generating a packet. The GNC code is then characterized by $\{\mathcal{G}, \mathcal{P}, q\}$. GNC codes are rateless, meaning that a potentially unlimited number of packets can be generated by the source node. An encoded packet of $\mathcal{G}_l, l = 1, \ldots, L$ is written as $\mathbf{p}^{(l)} = \sum_{j=1}^{G_l} g_j^{(l)} \mathbf{s}_j^{(l)}$, where $g_j^{(l)}$ is an encoding coefficient chosen uniformly randomly from $\mathbb{F}_q$. Row vector $\mathbf{g}^{(l)} = [g_1^{(l)}, \ldots, g_{G_l}^{(l)}]$, referred to as the *encoding vector*, is delivered in the header of $\mathbf{p}^{(l)}$.

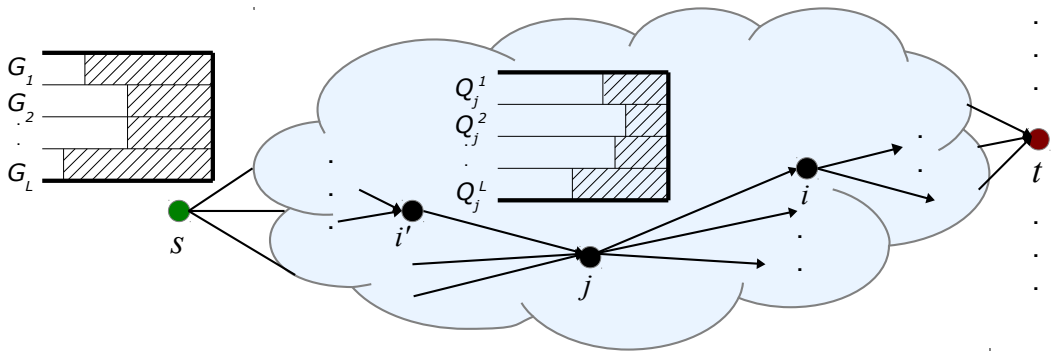

Figure 5.1: An overview of the system.

At each intermediate node $j \in \mathcal{V} \setminus \{s,t\}$, L queues Q_j^l, $1 \leq l \leq L$ are maintained to buffer received packets, one queue for each generation. A received packet belonging to $\mathcal{G}_l$ is said to be *innovative* for node j if its encoding vector is not in the span of encoding vectors of packets in Q_j^l. We assume that received packets are processed such that non-innovative packets are discarded. In reality this is not necessary and may require intermediate-node computations, but the assumption simplifies the model.

Let $|Q_j^l(n)|$ be the number of *buffered packets* in queue l at time n. When a transmission opportunity is presented on an outgoing link (j,i) at time n, a generation is chosen according to a *scheduling strategy*. We denote the index of the scheduled generation as $l_{ji}^*(n)$. A packet is then encoded from the buffered packets in $Q_j^{l_{ji}^*(n)}$ using RLNC [39] and sent on (j,i). The transmitted packet is in the form $\bar{\mathbf{p}}^{(l_{ji}^*(n))} = \sum_{m=1}^{G_{l_{ji}^*(n)}} \bar{g}_m^{(l_{ji}^*(n))} \mathbf{s}_m^{(l_{ji}^*(n))}$.

We maintain an array $[S_{ji}^1(n), S_{ji}^2(n), \ldots, S_{ji}^L(n)]$ for each link $(j,i) \in \mathrm{Out}(j)$, where $S_{ji}^l(n)$ records the numbers of times queue l has been scheduled for transmission on link (j,i) up to time n and can be interpreted as a measure for tracking the reduction in innovativeness of Q_j^l due to transmissions on the link. We define $P_{ji}^l(n) = |Q_j^l(n)| - S_{ji}^l(n)$ as the *local potential innovativeness* of Q_j^l on (j,i). Here terms "local" and "potential" are used because the innovativeness is only from the sending-node's perspective and does not incorporate knowledge of packet losses or receptions of downstream nodes. We refer to arrays $\mathbf{P}_{ji}(n) = [P_{ji}^1(n), P_{ji}^2(n), \ldots, P_{ji}^L(n)]$, $\forall (j,i) \in \mathrm{Out}(j)$ as the *buffer states* of node j at time n. If queue l is chosen, the value of $S_{ji}^l(n)$ is increased by one.

In this chapter, we propose to use the *maximum local potential innovativeness*

(MaLPI) scheduling strategy at intermediate nodes: choose the generation

$$l_{ji}^{*}(n) = \arg\max_{l} P_{ji}^{l}(n) \tag{5.1}$$

on link (j, i) for transmission at time n. If more than one queue attains the maximum, one of them is arbitrarily chosen.

5.2.3 Decoding of GNC Codes: Generation-by-Generation

At the destination, random linear combinations of intermediate packets are received due to RLNC performed in the network. The decoder needs to successively recover intermediate packets from the received packets (which are network coded) and recover of source packets from intermediate packets (if source packets are precoded). The latter is done by the precode decoder. In this chapter, we focus on recovering intermediate packets from the received network-coded packets.

We use the following generation-by-generation (G-by-G) decoder: the decoder decodes intermediate packets of each generation by solving $\mathbf{A}_l\mathbf{X}_l = \mathbf{B}_l$ using GE, where successive rows of the $G_l \times G_l$ $\mathbf{A}_l$ and the $G_l \times K$ $\mathbf{B}_l$ are encoding vectors and message symbols of received packets belonging to $\mathcal{G}_l, l = 1, \ldots, L$, respectively; $\mathbf{X}_l$ contains unknown intermediate packets in $\mathcal{G}_l$. $\mathbf{A}_l$ is called the *decoding matrix* of $\mathcal{G}_l$. To distribute the computations over time, GE is progressively performed upon each packet reception using Algorithm 1, where $\mathbf{A}_l[i]$ and $\mathbf{A}_l[i][i]$ refer to the i-th row and diagonal element of $\mathbf{A}_l$, respectively. The generation is decodable if all the G_l diagonal elements of $\mathbf{A}_l$ are nonzero, i.e., $\mathbf{A}_l$ is a full-rank upper triangular matrix. Compared to performing GE only after a sufficient number of packets are received, the computational cost of the above progressive GE is the same but the decoder can

know immediately when a generation is decodable.

Algorithm 1 Progressive Forward Gaussian Elimination (GE)

Initialize $G_l \times G_l$ zero matrix $\mathbf{A}_l, l = 1, \ldots, L$
Initialize $z(\mathbf{A}_l) = G_l$ ▷ number of zero rows of $\mathbf{A}$
while $z(\mathbf{A}_l) > 0$ **do**
    Receive a packet whose encoding vector is $\mathbf{g}$
    $stored \leftarrow 0$
    while $stored = 0$ and $\mathbf{g} \neq \mathbf{0}$ **do**
        $i \leftarrow i_{\mathrm{LM}}(\mathbf{g})$
        if $\mathbf{A}_l[i][i] = 0$ **then**
            $\mathbf{A}_l[i] \leftarrow \mathbf{g}$
            $stored \leftarrow 1$
            $z(\mathbf{A}_l) \leftarrow z(\mathbf{A}_l) - 1$
        else
            $t \leftarrow \mathbf{g}[i]/\mathbf{A}_l[i][i]$
            $\mathbf{g} \leftarrow \mathbf{g} + \mathbf{A}_l[i] \times t$
        end if
    end while
end while

Suppose that generation $\mathcal{G}_l$ is decodable; standard backward substitution is performed on the matrix resulting from Algorithm 1, and $\mathbf{A}_l\mathbf{X}_l = \mathbf{B}_l$ is completely solved. Since there is overlap between generations, decoded packets can be subtracted from received packets of the remaining generations which also contain the decoded packets [2]. The subtraction may result in newly decodable generations because it reduces the number of unknown packets in other generations. If no decodable generations can be found after the subtraction, the decoder continues to collect packets until another decodable generation is found. The subtraction is essentially a binary message-passing process (see e.g. [52]) and will be investigated in more detail in Section 5.4. The following is an example of G-by-G decoding.

[2]The subtraction corresponds to putting a '1' as the corresponding diagonal element of the decoding matrix of the generation, and eliminating the rest of the elements of the corresponding column.

Example 5.1 (Example of G-by-G decoding process)**.** *Assume that* 5 *intermediate packets are grouped into two generations* $\mathcal{G}_1 = \{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3\}$ *and* $\mathcal{G}_2 = \{\mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5\}$. *Suppose that* 3 *and* 2 *packets are received for* $\mathcal{G}_1$ *and* $\mathcal{G}_2$*, respectively, written as* $\mathbf{r}_1^{(1)} = \mathbf{s}_1 + \mathbf{s}_2$, $\mathbf{r}_2^{(1)} = \mathbf{s}_2 + \mathbf{s}_3$, $\mathbf{r}_3^{(1)} = \mathbf{s}_1 + \mathbf{s}_2 + \mathbf{s}_3$ *and* $\mathbf{r}_1^{(2)} = \mathbf{s}_3 + \mathbf{s}_5$, $\mathbf{r}_2^{(2)} = \mathbf{s}_3 + \mathbf{s}_4$, *respectively. In this case,* $\mathcal{G}_1$ *is decoded first. Since* $\mathbf{s}_3$ *is also present in* $\mathcal{G}_2$*, it is then subtracted from* $\mathbf{r}_1^{(2)}$ *and* $\mathbf{r}_2^{(2)}$. $\mathcal{G}_2$ *is then also decodable because the decoding matrix from the resultant packets* $\mathbf{r}_1^{(2)'} = \mathbf{s}_5$ *and* $\mathbf{r}_2^{(2)'} = \mathbf{s}_4$ *is now full-rank.*

When the number of decoded intermediate packets reaches a threshold, which depends on the rate of precoding, the precode decoder kicks in and all source packets are recovered using conventional erasure correction techniques.

Given a GNC code, suppose that the expected number of randomly generated packets at the source node with which the G-by-G decoding is successful is N. We define the *code overhead* of the GNC code as $\varepsilon = (N - M)/M$. The goal of this chapter is to minimize ε by optimizing the GNC code parameters.

5.3 MaLPI Scheduling and Local Scheduling Delay

In this section we show that our proposed MaLPI scheduling strategy outperforms random scheduling (RS) as used in [44], [32] and minimizes the probability of the event that a queue lacking innovative packets would be scheduled. The event is referred to as the *local scheduling delay* because a transmission opportunity is wasted, additional overhead is incurred, and the completion of transmission may be delayed. For ease of exposition, we omit all subscripts in the sequel and consider an arbitrary intermediate node and its outgoing links.

We define an indicator random variable $I(n)$, where $I(n) = 1$ when a non-innovative packet is sent and $I(n) = 0$ otherwise. For any random scheduling, a generation index l^* is randomly chosen with probability $\Pr\{l^*_{ji} = l\} = w_l$, where $w_l \in (0,1)$ and $\sum_l w_l = 1$. We have

$$p_{\mathrm{RS}} \triangleq \Pr\{I_{\mathrm{RS}}(n) = 1\} = \sum_l w_l \Pr\{P^l(n) = 0\}, \tag{5.2}$$

while for MaLPI scheduling,

$$p_{\mathrm{MaLPI}} \triangleq \Pr\{I_{\mathrm{MaLPI}}(n) = 1\} = \prod_l \Pr\{P^l(n) = 0\}. \tag{5.3}$$

Lemma 5.1. *For arbitrary distribution $w_1, w_2, \ldots, w_L$ and any buffer state $\mathbf{P}(n)$, we have $p_{\mathrm{RS}} > p_{\mathrm{MaLPI}}$.*

Proof. This is equivalent to showing that $\sum_{l=1}^{L} w_l x_l > \prod_{l=1}^{L} x_l$ for any w_l and x_l satisfying $\sum_l w_l = 1$, $w_l \in (0,1)$, and $x_l \in (0,1)$, $1 \le l \le L$.

Using concavity of the $\log(\cdot)$ function and Jensen's inequality, we have

$$\begin{aligned} \log(\sum_{l=1}^{L} w_l x_l) &\geq \sum_{l=1}^{L} w_l \log x_l \\ &> \sum_{l=1}^{L} \log x_l = \log(\prod_{l=1}^{L} x_l), \end{aligned} \tag{5.4}$$

where the second inequality follows from $w_l \log x_l > \log x_l$ given that $w_l \in (0,1)$ and $x_l \in (0,1)$ for all l. □

The above result is intuitively not surprising because RS does not take the buffer state into account unlike MaLPI scheduling.

It is noted that the MaLPI scheduling is similar to the local-rarest-first scheduling of [66], which makes use of feedback from downstream nodes to determine which generation has the most innovative packets to send. The MaLPI scheduling, however, estimates such information based on its own buffer state and therefore requires no feedback.

5.4 Decoding Analysis of GNC Codes as Graph Codes

In this section, we first model the GNC code as a graph code, and then analyze its G-by-G decoding using graphs.

5.4.1 Graph Representation of GNC code

A GNC code (with L generations) defined on $(1+\theta)M$ intermediate packets can be modeled with bipartite graphs. The packets and generations correspond to two separate sets of vertices on the graph, referred to as *packet nodes* and *generation nodes*, respectively. An edge is created to connect a packet and generation node pair if the packet is contained in the generation, so the total number of edges $E = \sum_{l=1}^{L} G_l$. A node is said to be of degree i if i edges are connected to the node. We say an edge is of packet-side degree i if its connected packet node is of degree i and of generation-side degree i if its connected generation node is of degree i, respectively. We denote the fractions of edges that are of the packet-side and generation-side degrees i as $\lambda_i, 1 \leq i \leq L$ and $\rho_i, d_{\min} \leq i \leq d_{\max}$, respectively.

Since content of generations can be chosen at random from $\mathcal{S}$, a GNC code can then be viewed as a sampled instance from an ensemble of bipartite graphs. The ensemble is characterized by the packet-side edge degree distribution $\lambda(x) = \sum_{i=1}^{L} \lambda_i x^{i-1}$ and the

generation-side edge degree distribution $\rho(x) = \sum_{i=d_{\min}}^{d_{\max}} \rho_i x^{i-1}$. Let $\Psi(x) = \sum_{i=1}^{L} \Psi_i x^i$ and $\Omega(x) = \sum_{i=d_{\min}}^{d_{\max}} \Omega_i x^i$ such that $\lambda(x) = \Psi'(x)/\Psi'(1)$ and $\rho(x) = \Omega'(x)/\Omega'(1)$, where $\Psi'(x)$ and $\Omega'(x)$ denote derivatives of $\Psi(x)$ and $\Omega(x)$ with respect to x, respectively; $\Psi'(1) = \sum_{i=1}^{L} i\Psi_i$ and $\Omega'(1) = \sum_{i=d_{\min}}^{d_{\max}} i\Omega_i$ are the average degree of packet and generation nodes, respectively. Note that Ψ_i is the probability that one intermediate packet is present in i generations and Ω_i is the probability that one generation contains i intermediate packets. We refer to $\Omega(x)$ as the *generation-size* distribution.

According to [34], instead of analyzing the performance of decoding on a particular instance of bipartite graph, it is reasonable and also easier to analyze the expected decoding performance on the ensemble of graphs.

5.4.2 Message Passing and Analysis of Decoding Process

We now model the G-by-G decoding of GNC codes as a message passing process on the ensemble of bipartite graphs. The G-by-G decoding includes two types of operations: GE decoding of a generation and subtraction of decoded packets from other generations. With the graph code representation, we use the and-or tree evaluation technique [34] to analyze the message-passing decoding. This requires some modification to the original technique [34] to incorporate the GE decoding.

At the decoder, each generation node is associated with a random number of received packets. In the sequel, we assume that the first d received packets of a generation node of degree d are innovative. This assumption corresponds to the case where a sufficiently large finite field is used to perform coding. We refer to the number of innovative packets of the generation node, k, as the *received rank* of the node; $k \in \mathcal{R} = \{0, 1, 2 \ldots, d\}$ where d is the degree of the node.

We define a binary message alphabet $\mathcal{M} = \{0, 1\}$ that is used by both of packet and generation nodes, where 0 and 1 indicates that the node is *unknown* (not decoded) and *known* (decoded) nodes, respectively. At the beginning of the decoding, all the packet and generation nodes send *unknown* messages to each of their neighbors. Each generation node is associated with a received rank $k \in \mathcal{R}$. The number of adjacent edges of a node carrying incoming unknown messages is referred to as the *unknown degree* of the node, denoted as ς_p and ς_g for packet nodes and generation nodes, respectively. Corresponding to the decoding process in Section 5.2.3, the message mapping rules on the graph are as follows: A generation node sends a *known* message on all its adjacent edges if and only if its received rank k is larger than $\varsigma_g - 1$, which means that the generation can be decoded by GE because there are k innovative packets while there are only $\varsigma_g \leq k$ unknown packets therein. A packet node sends known messages on its adjacent edges if ς_p is smaller than its node degree, which means that at least one generation containing the packet has been decoded.

The decoding may be more easily explained and analyzed from an and-or tree perspective [34]. By randomly choosing one edge of the bipartite graph that is uniformly sampled from the ensemble of graphs that are characterized by $\lambda(x)$ and $\rho(x)$, and expanding the graph starting from its connected generation node, we can obtain a sub-graph being a tree with high probability [34], [52], [37]. An example is shown in Figure 5.2, where the sub-graph is obtained by expanding from a generation node to within a depth $2h$. We denote this sub-graph as $\mathcal{P}_h$. In this representation, packet and generation nodes occur at depths $0, 2, \ldots, 2h-2$ and $1, 3, \ldots, 2h-1$, respectively.

Let us consider the decoding of the root node of the tree, corresponding to subgraph $\mathcal{P}_h$. Suppose that the subgraph was obtained by exapnsion from a generation

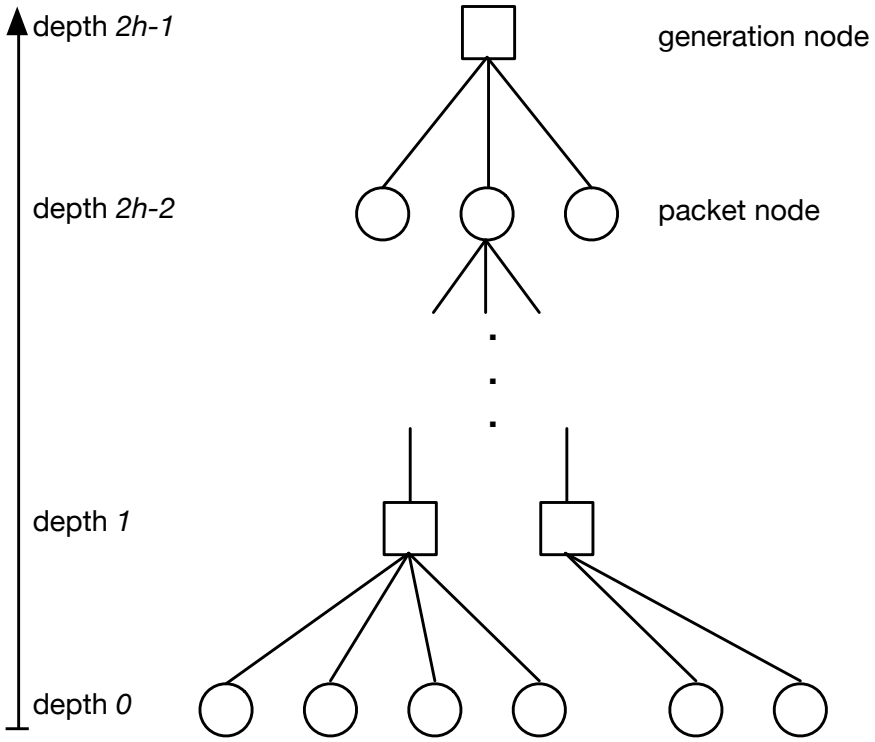

Figure 5.2: Expanding the graph as a tree.

node of degree m that has received μ packets. Let $u_h(m, \mu)$ denote the probability that it is not decodable, and assume it to be independent of any other generation node. For $d_{\min} \leq m \leq \mu$, we have $u_h(m, \mu) = 0$ because generations can be decoded immediately if the numbers of their received innovative packets are larger than their degrees. We refer to these generations as *self-decodable*. Let z_h denote the probability that an arbitrary packet node contained in the generation is sending an unknown message, and assume it to be independent of any other packet node. Then we can

obtain $u_h(m,\mu)$ for $m \geq \mu + 1$:

$$u_h(m,\mu) = \sum_{k=\mu}^{m-1} \left(\begin{array}{c} m-1 \\ k \end{array} \right) (z_h)^k (1-z_h)^{m-1-k}. \tag{5.5}$$

The right-hand side of (5.5) is the probability that the number of received packets is smaller than the unknown degree of the generation node.

Take different values of μ into account. Let $\eta_{m,\mu}$ denote the probability that the chosen root node is of degree m and associated with μ received packets. Note that $\eta_{m,\mu}$ depends on $\rho(x)$ and on the number of received packets for each generation. Let y_h denote the probability that an arbitrarily chosen generation node is not decodable by evaluating to within distance $2h$ on the bipartite graph. We have

$$\begin{aligned} y_h &= \sum_{(m,\mu):m\geq\mu+1} \eta_{m,\mu} \sum_{k=\mu}^{m-1} \left(\begin{array}{c} m-1 \\ k \end{array} \right) (z_h)^k (1-z_h)^{m-1-k} \\ &\triangleq f(z_h, A), \end{aligned} \tag{5.6}$$

where the summation is over all possible (m,μ) pairs and A is a placeholder matrix consisting of probabilities $\eta_{m,\mu}$ as its elements. The exact form of A will be specified in later sections on code design.

Now we need to determine the probability z_h. For $h > 0$, since the subgraph $\mathcal{P}_h$ is a tree, as explained in [34] we can evaluate z_h based on subgraphs of $\mathcal{P}_h$, i.e., $\mathcal{P}_{h-1}$. The probability that a d-degree packet node beneath the root of $\mathcal{P}_h$ sends an

unknown message is as follows:

$$v_h^{(d)} = \begin{cases} 1 & d = 1, \\ (y_{h-1})^{d-1} & d = 2, \ldots, L, \end{cases} \tag{5.7}$$

where y_{h-1} is the probability that the root node in a subgraph $\mathcal{P}_{h-1}$ is not decodable. The two cases in (5.7) correspond to 1) the packet node connects to only one generation node (i.e., the root node of $\mathcal{P}_h$), which is definitely not decoded, and 2) all other generation nodes connecting this packet node are not decoded, respectively. By simply counting nodes in the subgraph,

$$z_h = \lambda_1 + \sum_{d=2}^{L} \lambda_d (y_{h-1})^{d-1} = \lambda(y_{h-1}), \tag{5.8}$$

for $h \geq 1$.

Substituting (5.8) into (5.6), we have the recursion

$$y_h = f(\lambda(y_{h-1}), A). \tag{5.9}$$

Eqn. (5.9) shows that, given fixed $\lambda(x)$, $\rho(x)$, and the number of received packets of each generation, the evolution of y_h, i.e., the decodability of each generation, can be predicted. For $h = 0$, the subgraph $\mathcal{P}_0$ only contains the root generation node and its packet nodes; $z_0 = 1$, and $y_0 \leq 1$ corresponds to the probability that a randomly chosen generation is not self-decodable. The final value of y_h, denoted as $\delta \triangleq \lim_{h\to\infty} y_h$, corresponds to the smallest probability that the G-by-G decoder can reach after going through all generations, or in other words, the expected fraction of generations that are not recoverable at the end of decoding given the received packets.

For precoded GNC, the value of δ can be straightforwardly determined. If the G-by-G is only able to recover a fraction of $(1-\delta)$ intermediate packets from received packets, the precode decoder should be able to recover the rest. Given a precode of rate $1/(1+\theta)$, we then have $\delta < \theta/(1+\theta)$.

For the sake of simplicity, we now omit index h and denote the probability that a generation node is not decodable at any time as y, $y \in [\delta, 1]$. To ensure that the decoding process continues, it is not hard to see that (5.9) should satisfy

$$f(\lambda(y), A) < y, \quad y \in [\delta, 1], \tag{5.10}$$

which means that the probability that a generation node is not decodable should be strictly decreasing until a fraction of $(1-\delta)$ generations are decoded. This inequality will be used in the rest of the chapter.

5.5 Design for Equal-size GNC codes: Optimal Amount of Overlap

In this section, we design a GNC code based on the *random annex code* (RAC) [32] using the analysis results obtained above. RAC is an equal-size GNC code, and the construction of an RAC that has generation size G is as follows (we have modified the definition using the language of this chapter).

Definition 5.1 (Random Annex Code [32])**.** *The $(1+\theta)M$ intermediate packets are first partitioned into L disjoint subsets $\mathcal{B}_l = \{\mathbf{s}_{(l-1)B+1}, \ldots, \mathbf{s}_{lB}\}, l = 1, \ldots, L$ of equal size B, one per generation. We assume L to be a divisor of $(1+\theta)M$, $(1+\theta)M = LB$; if it is not the case, some null packets can be padded. $\mathcal{B}_l$ is referred to as the base part of the generation. After that, each generation $\mathcal{G}_l$ is equipped with a random annex of*

size H, denoted as $\mathcal{H}_l$, which consists of a random selection of H packets from $\mathcal{S}-\mathcal{B}_l$. The annex causes overlap between generations. The overall generation $\mathcal{G}_l = \mathcal{B}_l \cup \mathcal{H}_l$ and $G = G_l = B + H, l = 1, \ldots, L$.

When generating a coded packet, the source node chooses a generation uniformly at random. In [32], it is shown that RAC is helpful in reducing overhead when G-by-G decoding is used, because random annexes introduce overlap among generations and the decoding of one generation would provide help to other generations to decode. A key parameter, the optimal amount of overlap that should be used, remains to be determined. Through simulations, it is observed that different amounts of overlap may result in different performances, and too much overlap has severe negative effects. The amount of overlap is known to have an upper bound (see e.g. [42], [24]) which unfortunately is loose and does not provide insight into a desirable or optimal amount of overlap to use. In this section, we apply results from the previous section to estimate the optimal amount of overlap for RAC.

5.5.1 Derivation of $\Psi(x)$ and $\lambda(x)$ of RAC

We first determine $\Psi(x)$ and $\lambda(x)$ for RAC. According to the RAC construction method, we observe that the corresponding $\Psi(x)$ and $\lambda(x)$ of RAC may only depend on B and G when the number of generations is large. It is easy to verify that, for given B, G and L as defined in Definition 5.1, the probability that a packet node connects to i generations given the generation construction method is

$$\Psi_i = \begin{pmatrix} L-1 \\ i-1 \end{pmatrix} \left(\frac{G-B}{LB}\right)^{i-1} \left(1-\frac{G-B}{LB}\right)^{L-i}.$$

Therefore, we have

$$\begin{aligned}
\Psi(x) &= \sum_{i=1}^{L} \Psi_i x^i \\
&= \sum_{i=1}^{L} \binom{L-1}{i-1} \left(\frac{G-B}{LB}\right)^{i-1} \left(1-\frac{G-B}{LB}\right)^{L-i} x^i \\
&= x \sum_{i=0}^{L-1} \binom{L-1}{i} \left(\frac{G-B}{LB}x\right)^{i} \left(1-\frac{G-B}{LB}\right)^{L-1-i} \\
&= x\left[1-\frac{(G/B-1)(1-x)}{L}\right]^{L-1}, \qquad (5.11)
\end{aligned}$$

where the last equality comes from the binomial theorem. Using $\lambda(x) = \Psi'(x)/\Psi'(1)$, we have

$$\lambda(x) \approx \left(\frac{B}{G} + \left(1-\frac{B}{G}\right)x\right) \mathrm{e}^{-(G/B-1)(1-x)}, \qquad (5.12)$$

where the approximation comes from using the fact that $\lim_{m\to\infty}\left(1+\frac{1}{m}\right)^m = \mathrm{e}$, which is accurate even when L is not very large (e.g., 100).

5.5.2 Estimation of Optimal Amount of Overlap

Suppose that the G-by-G decoder is directly applied on the coded packets of RAC and the decoder succeeds with $(1+\varepsilon)M$ generated packets. Our goal is to design RAC parameters such that the G-by-G decoder can succeed with the minimum required overhead, ε.

Since packets are generated from each generation with equal probability, the number of generated packets belonging to each generation, μ, follows the binomial distribution

$$\Pr\{\mu = j\} = \begin{pmatrix} (1+\varepsilon)M \\ j \end{pmatrix} \left(\frac{1}{L}\right)^{j} \left(1 - \frac{1}{L}\right)^{(1+\varepsilon)M-j}. \tag{5.13}$$

There are difficulties that 1) μ's of L generations are not independent of one another and hence the calculation of the $\eta_{m,\mu}$ in (5.6) will be difficult and 2) the computation of (5.13) is difficult for large M. To resolve the issues, we can use Poisson approximation [45]. The Poisson approximation models the numbers of packets belonging to each generation as independent and identically distributed (IID) Poisson random variables with rate parameter $\lambda = \frac{(1+\varepsilon)M}{L} = \frac{(1+\varepsilon)B}{1+\theta}$. The approximation is accurate when M is large, $1/L$ is small and M/L is fixed.

For a RAC code with generation size G, all the generation nodes of the bipartite graph have degree G, so we can write $\eta_{m,\mu}$ in (5.6) as

$$\eta_{m,\mu} = \eta_{G,\mu} = \frac{\lambda^{\mu} \mathrm{e}^{-\lambda}}{\mu!}. \tag{5.14}$$

We can therefore rewrite (5.10) as

$$\sum_{u=0}^{G-1} \frac{(\frac{(1+\varepsilon)B}{1+\theta})^{\mu} \mathrm{e}^{-\frac{(1+\varepsilon)B}{1+\theta}}}{\mu!} \sum_{k=u}^{G-1} \begin{pmatrix} G-1 \\ k \end{pmatrix} (\lambda(y))^{k} (1-\lambda(y))^{G-1-k} < y, \tag{5.15}$$

where $y \in [\delta, 1]$ and $\delta = \frac{\theta}{1+\theta}$. We denote the left-hand side of (5.15) as $f_{\mathrm{RAC}}(B, G, \varepsilon, y)$.

For RAC with fixed parameters B and δ, let

$$\varepsilon^{*}(G) = \inf \{\varepsilon : f_{\mathrm{RAC}}(B, G, \varepsilon, y) < y, y \in [\delta, 1]\} \tag{5.16}$$

denote the minimum overhead that is required for a given G such that (5.10) is satisfied. We can therefore design G for RAC by solving the problem:

$$\begin{aligned} &\underset{G}{\text{minimize}} \quad \varepsilon^*(G) \\ &\text{subject to} \quad f_{\text{RAC}}(B, G, \varepsilon^*(G), y) < y, \quad y \in [\delta, 1], \end{aligned} \tag{5.17}$$

which finds the optimal G with which the G-by-G decoder can succeed with the minimum number of coded packets. Unfortunately, it is not possible to obtain a closed-form solution due to the complicated form of (5.15) and therefore we resort to search.

To understand how to perform the search, we examine an example where $\varepsilon^*(G)$ is plotted in Figure 5.3 as a function of G when fixing $B = 32$ and $\delta = 0.015$. For any given value of G, $\varepsilon^*(G)$ is found by evaluation of ε from 0 up with desired precision step $\Delta\varepsilon$ until the first ε is found such that (5.15) is satisfied. Inequality (5.15) is tested at J uniformly discretized points $y_1, y_2, \ldots, y_J \in [\delta, 1]$ for some integer J.

Figure 5.3 shows that $\varepsilon^*(G)$ first decreases as G increases, indicating the benefit of introducing overlap. Note that $G = 32$ corresponds to the disjoint case where no overlap is introduced among generations. When increasing G to a specific level (in Figure 5.3 at $G = 46$), a local minimum can be attained. Experimentally we have observed that $\varepsilon^*(G)$ increases monotonically with G afterwards, suggesting that this local minimum may actually be a global minimum. This suggests a heuristic that the exhaustive search may be reduced. This phenomenon also matches with the simulation results of [32] that there exists an optimal amount of overlap and if increasing the amount of overlap to beyond the optimal value, there is only negative effect (i.e., overhead increases).

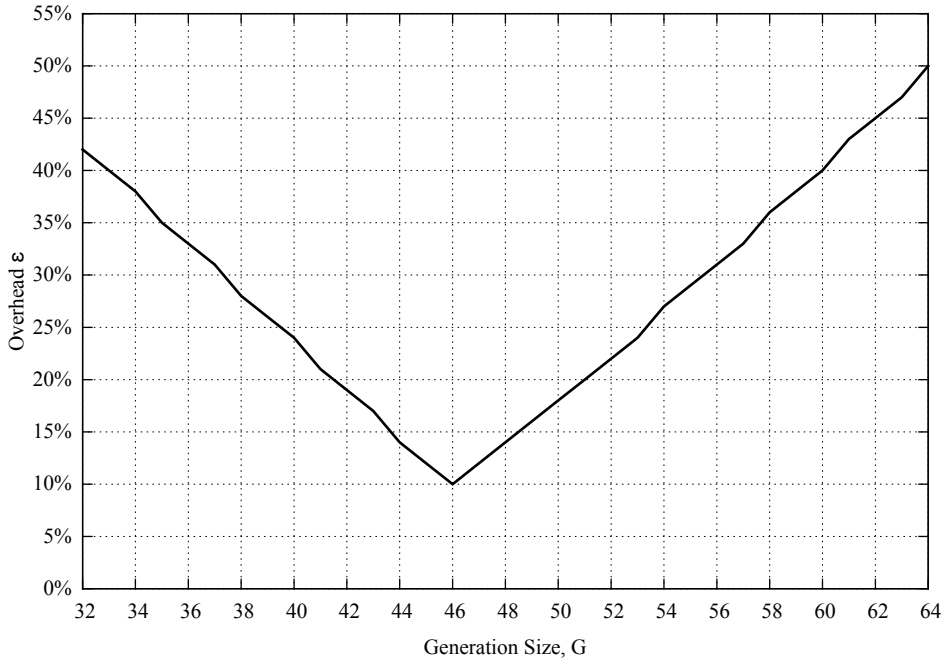

Figure 5.3: Values of $\varepsilon^*(G)$ for various G; $B = 32$, $\delta = 0.015$.

Our reduced-complexity search heuristic is then to find the first local minimum. We can perform the search as listed in Algorithm 2 to solve (5.17). The obtained estimation of the optimal G is denoted as G^* and the corresponding overhead is denoted as $\varepsilon_{\min}$.

5.5.3 Overlap Upper Bound

As a comparison to the estimated amount of overlap, we include the derivation of a known upper bound of the amount of overlap, which corresponds to *at least having one decodable generation among slightly more than M received packets* (see e.g. [24]). Suppose generations of equal size a are used and we require no larger than ε overhead. To have at least one decodable generation among the first $(1+\varepsilon)M$ received packets, we need $\Pr\{X \geq a\} \geq 1/L$. Here, X denotes the number of received packets of a

Algorithm 2 Search for the optimal G

```
Input: B, δ, Δε
Output: G*, ε_min
Initialize G ← B, G* ← G, ε_min ← ∞
found ← 0
while found = 0 do
    ε* ← 0
    inequality_sat ← 0
    while inequality_sat = 0 do
        if f_RAC(B, G, ε*, y) < y,   y ∈ [δ, 1] then
            inequality_sat ← 1
        else
            ε* ← ε* + Δε
        end if
    end while
    if ε* < ε_min then
        ε_min ← ε*
        G* ← G
        G ← G + 1
    else
        found ← 1
    end if
end while
```

specific generation, which is a random variable following the binomial distribution with parameters $(1+\varepsilon)M$ and $1/L$.

The generation size G is then upper bounded by

$$\begin{aligned} G &\leq \sup\left\{a \in \mathbb{Z} : \sum_{k=0}^{a-1} \Pr\{X=k\} \leq 1 - \frac{1}{L}\right\} \\ &\triangleq G_{\mathrm{MAX}}, \end{aligned} \tag{5.18}$$

where $\Pr\{X=k\} = \begin{pmatrix} (1+\varepsilon)M \\ k \end{pmatrix} \left(\frac{1}{L}\right)^k \left(1-\frac{1}{L}\right)^{(1+\varepsilon)M-k}$. To simplify the calculation, we can again use the Poisson approximation with rate parameter $\lambda = (1+\varepsilon)M/L$

to approximate X.

It is noted that a critical issue with determining this bound, besides its looseness, is that we have to first specify the value of ε. This is awkward because we do not know the minimum achievable ε *a priori*. If we specify a small value of ε, the resultant G_{MAX} might be infeasible while if we specify a large one the resultant G_{MAX} might be too loose.

5.5.4 Numerical and Simulation Results

We now examine the effect of overlap and verify our design of RAC through numerical and simulation results. Throughout we use $B = 32$ in RAC constructions. The first simulation uses $M = 65536$ source packets, each with $K = 1024$ symbols from $\mathbb{F}_{2^8}$, i.e., each packet is 1KB. RLNC is also performed in $\mathbb{F}_{2^8}$. We precode the source using the same systematic LDPC precode as standard raptor codes [35] (Section 5.4.2.3) because of its fixed yet efficient structure. For $M = 65536$, the LDPC precode of [35] adds $S = 1019$ check packets and hence $\delta \approx 0.015$. The resulting 66555 intermediate packets are grouped into $L = 2080$ generations. Each curve of the following simulation is averaged over 1000 trials.

We first demonstrate the impact of estimating the optimal amount of overlap. In Figure 5.4, we plot the overheads of RAC codes as a function of G. We show both of the simulated results and the estimated minimum ε from solving (5.17). Different G result in different overhead performance. When $G = B = 32$, the code reduces to the disjoint case. The simulation shows that overhead is decreasing when increasing the amount of overlap from $G = 32$ up until $G = 44$. However, after that, such improvement starts vanishing. The analytical result, i.e., $\varepsilon^*(G)$, matches with the

simulation results closely. However, we note that there is a gap between the two curves for smaller G whereas the curves are perfectly match for $G \geq 46$. The gap suggests that the analysis may not be applicable to scenarios where the amount of overlap between generations is small and therefore the decoding of one generation helps very little to the decoding of others.

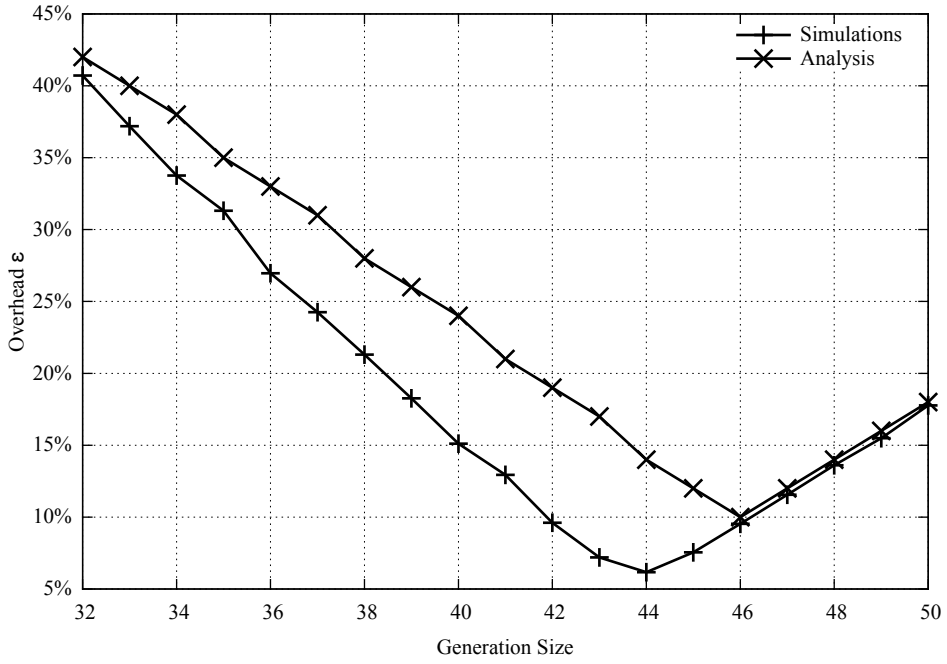

Figure 5.4: Simulated overheads at various G; $M = 65536$, $B = 32$, $\delta = 0.015$.

In Figure 5.5, we compare the minimum achievable overheads for various M with the overheads that are achieved by using the G_{MAX} calculated using (5.18) and the estimated G^* with Algorithm 2, respectively. The precode rate for each M is determined according to Section 5.4.2.3 of [35]. The minimum achievable overheads are obtained by extensive simulations. In the calculation of (5.18), we set ε to be the minimum achievable overheads that are obtained by the extensive simulations. The estimated $G^* = 46$ for all the M's.

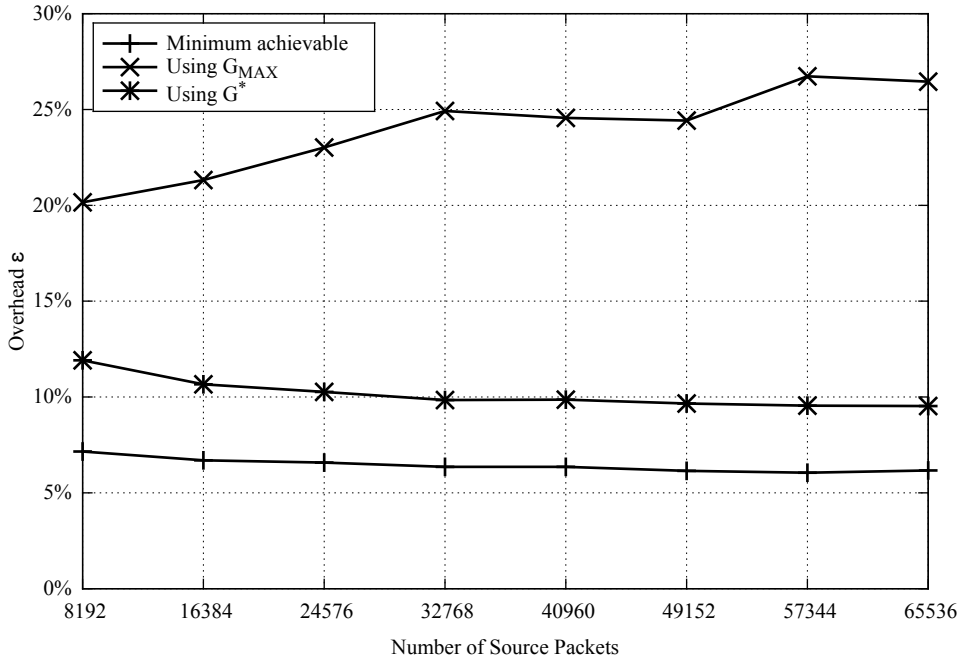

Figure 5.5: Comparison of the minimum achievable overheads for various M with overheads that are achieved when using generation sizes G_{MAX} and G^*; $B = 32$.

From Figure 5.5, it is seen that the overheads of G^* is much lower than that of G_{MAX} and is only about 3% higher than the minimum achievable overheads. Even though we have used the values of the minimum achievable overheads in the calculation of G_{MAX}, its achieved overheads are far from satisfactory, which are above 20%.

The role of overlap can be better understood by examining (5.15). In Figure 5.6, we plot $y - f_{\mathrm{RAC}}(B, G, \varepsilon, y)$ as a function of $1 - y$ for various G on $[0, 1 - \delta]$. From the previous analysis, we know that $y - f_{\mathrm{RAC}}(B, G, \varepsilon, y)$ corresponds to the expected fraction of newly decodable generations when a fraction of $(1 - y)$ generations have been decoded. In Figure 5.6 we set $\varepsilon = 10\%$ and $\delta = 0.015$ for all G. For each G, the curve depicts the fraction of newly decodable generations throughout the decoding

process of the code when given a fixed amount, 10%, overhead. The point when the curve goes below 0 corresponds to when the decoding process stalls because no decodable generations can be found, meaning that the given amount of overhead is insufficient.

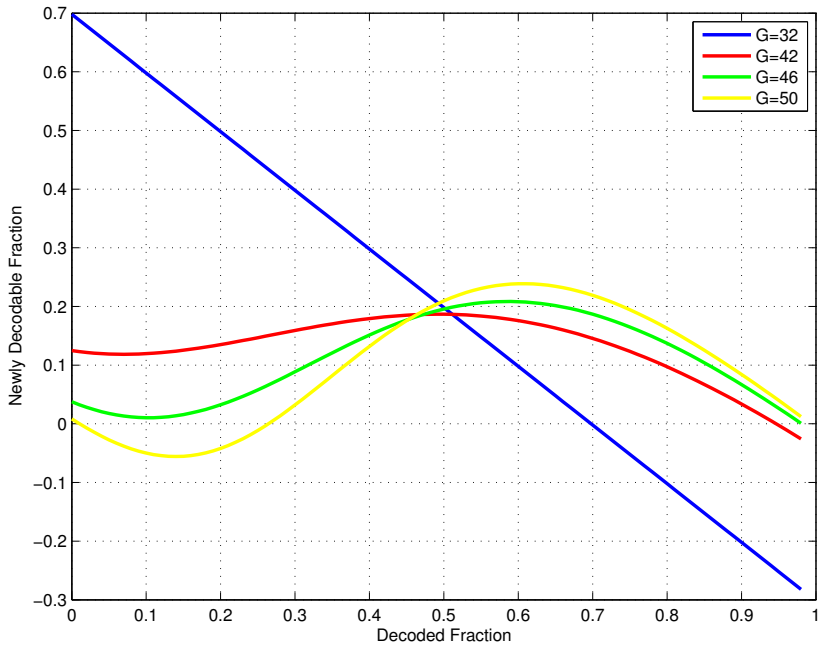

Figure 5.6: Newly decodable fractions for various G; fixed $\varepsilon = 10\%$, $B = 32$, $\delta = 0.015$.

Figure 5.6 reveals that choosing G is essential to balance the difficulty of decoding between the beginning and the end. On the one hand, when no overlap is used, i.e., $G = 32$, decoding proceeds quite easily at the beginning but gets easily stuck at the end (after 70% generations are decoded). On the other hand, if we introduce a large amount of overlap, e.g. $G = 50$, decoding would not even be able to begin with the given ε.

5.6 Unequal-size GNC codes

5.6.1 Motivation

Inspired by the fact that LDPC codes based on irregular bipartite graphs [37], [52] have superior performance compared to regular ones, we are interested in whether the advantage persists for GNC codes. This corresponds to the question of whether using unequal-size generations would have advantages over that of equal-size. In [31], it is shown that the answer is positive. Unequal-size GNC code is shown to incur lower overhead and decoding cost at the same time compared to the equal-size counterpart, achieving better overhead-complexity tradeoffs. In this section, we design unequal-size GNC codes under the framework of the proposed analysis in Section 5.4.

Intuitively, the motivation of using unequal-size generations is as follows: suppose that L generations of equal-size G are used, where $GL > (1+\theta)M$. Then decoding will not begin until at least one generation has received G packets, and the decoder has to solve at least one system of linear equations with G unknowns. If unequal-size generations are used, with generation sizes drawn from a distribution $\Omega(x)$ of average size $d_{\text{avg}} = G$, exactly the same amount of overlap is introduced as the equal-size case, but decoding could begin earlier because now there are generations of sizes smaller than G, which might be decodable with fewer than G received packets. Also, this may help to reduce the unknown degree of other generation nodes earlier. If generation sizes and amount of overlap are chosen carefully, there is a chance that there are always generations with unknown degrees smaller than G throughout the decoding process. As a result, both overhead and complexity may be reduced.

In Figures 5.7 and 5.8, we illustrate the idea using 8 packets, which are grouped into 4 equal-size and unequal-size generations, respectively. The same amount of

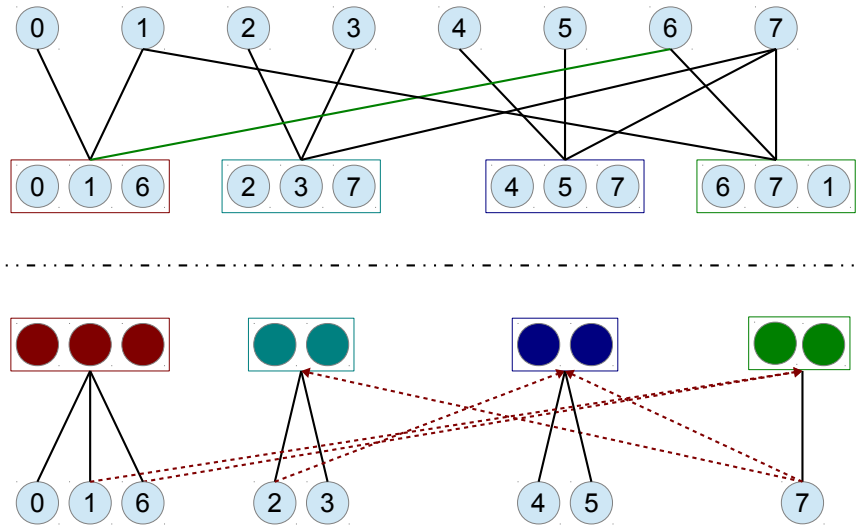

Figure 5.7: Decoding example of an equal-size GNC code.

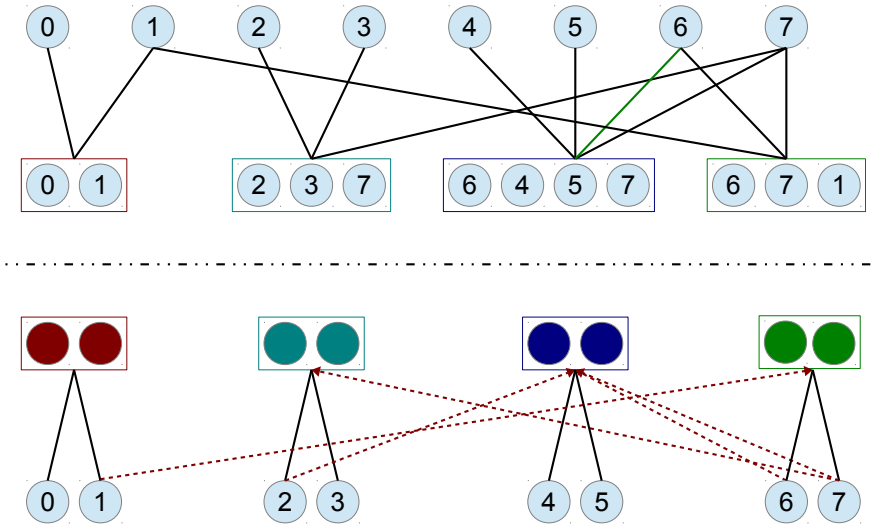

Figure 5.8: Decoding example of an unequal-size GNC code.

overlap is used; $G = 3$. In each figure, the upper graph above the dash line shows how packets are connected to generations while the lower graph depicts the decoding process. Red dashed lines in lower graphs of Figure 5.7 and Figure 5.8 correspond to back-substitutions due to the overlap between generations. In Figure 5.7, consider when two packets are received for each generation, then the decoder has to wait for the third packet for the first generation to begin decoding, so the overhead is 1. The number of operations needed in decoding generations is $3^3 + 2^3 + 2^3 = 43$ by GE. However, if we move packet 6 from the first generation to the third, resulting in unequal-size generations as shown in Figure 5.8, decoding successfully begins after two packets are received for each generation, and the overhead is reduced to 0. In this case, we only need to decode generations with $2^3 \times 4 = 32$ GE operations. We refer to such improvement on both overhead and computational cost as an *overhead-complexity tradeoff* benefit.

5.6.2 Generation-size Distribution Design for Unequal-Size RAC

In this section, we design an unequal-size GNC code which we refer to as unequal-size RAC (uRAC). The construction of the code is similar to that of equal-size RAC except that random annexes of unequal-size RAC are of unequal-sizes; the size of base parts of generations remains equal. The minimum generation size is set to $d_{\min} = B = (1 + \theta)M/L$ and the maximum generation size is denoted as $d_{\max}$.

A key component of uRAC is the generation-size distribution, from which generation sizes are drawn when constructing generations. Using analysis results from Section 5.4, it is equivalent to designing $\Omega(x)$ or $\rho(x)$ such that (5.10) is satisfied. Note that $\rho_m, d_{\min} \leq m \leq d_{\max}$ are encapsulated in A of (5.10). For convenience, in the

sequel we denote the vector $\boldsymbol{\rho} \triangleq [\rho_{d_{\min}}, \rho_{d_{\min}+1}, \ldots, \rho_{d_{\max}}]$. To determine $f(\lambda(y), A)$ of (5.10), we notice that in order to obtain the aforementioned overhead-complexity tradeoff benefit, the decoding should be able to succeed with $\mu < G$ received packets per generation. Therefore, supposing that each generation has $\mu < G$ packets received, (5.10) can be re-written as

$$\sum_{m=\mu+1}^{d_{\max}} \rho_m \sum_{k=\mu}^{m-1} \binom{m-1}{k} (\lambda(y))^k (1-\lambda(y))^{m-1-k} < y, \quad , y \in [\delta, 1]. \tag{5.19}$$

We denote the left-hand side of (5.19) as $f_{\text{uRAC}}(B, G, \boldsymbol{\rho}, \mu, y)$.

For a specified G, we can then design $\boldsymbol{\rho}$ through solving the following problem:

$$\begin{aligned} \underset{\boldsymbol{\rho}}{\text{minimize}} \quad & \mu \\ \text{subject to} \quad & \sum_{m=d_{\min}}^{d_{\max}} \rho_m = 1, \\ & \sum_{m=d_{\min}}^{d_{\max}} \frac{\rho_m}{m} = \frac{1}{G}, \text{ and} \\ & f_{\text{uRAC}}(B, G, \boldsymbol{\rho}, \mu, y) < y, \quad y \in [\delta, 1], \end{aligned} \tag{5.20}$$

which corresponds to minimizing the average number of received packets per generation that are needed for successful G-by-G decoding. Note that the expected overhead is $\varepsilon = \frac{\mu L}{M} = \frac{\mu(1+\theta)}{B}$. The problem (5.20) therefore minimizes the expected overhead of uRAC.

To solve (5.20), we can conduct an exhaustive search to first find the minimum μ such that the constraints are satisfied. In doing so, we can test from $\mu = B$ up until the first feasible μ is found. Testing each μ corresponds to solving the following

feasibility problem:

$$\begin{aligned} \underset{\boldsymbol{\rho}}{\text{minimize}} \quad & 1 \\ \text{subject to} \quad & \sum_{m=d_{\min}}^{d_{\max}} \rho_m = 1, \\ & \sum_{m=d_{\min}}^{d_{\max}} \frac{\rho_m}{m} = \frac{1}{G}, \text{ and} \\ & f_{\text{uRAC}}(B, G, \boldsymbol{\rho}, \mu, y) < y, \quad y \in [\delta, 1], \end{aligned} \tag{5.21}$$

where the inequality constraint can be tested at J uniformly discretized points as we did in solving (5.17). Each of the J inequalities is a linear function in $\boldsymbol{\rho}$, and therefore testing the constraints is a linear programming feasibility problem and can be easily solved using standard tools.

Let $\hat{\mu}$ denote the minimum μ with which the problem (5.21) is feasible. Any feasible solution $\boldsymbol{\rho}$ corresponding to the $\hat{\mu}$ is supposed to be sufficient to make the decoding succeed. However, further refinements can still be made to ensure that the distribution works well in practice. The first refinement, similar to the design of ripple size in raptor codes [56], is to generalize the inequality constraint by including a parameter $c_{\hat{\mu}} > 0$, which represents the increment of decodabilities of other generations when a generation is decoded. Again, we can search for the largest $c_{\hat{\mu}}$ from the initial value $c_{\hat{\mu}} = 0$ such that (5.20) is feasible with known $\hat{\mu}$, i.e., enforce the probability increase as quickly as possible. Note that now the last inequality constraint is $f_{\text{uRAC}}(B, G, \boldsymbol{\rho}, \hat{\mu}, y) < y - c_{\hat{\mu}}$, and is still linear in $\boldsymbol{\rho}$. Therefore, the optimal $c_{\hat{\mu}}$, which denoted as $\hat{c}_{\hat{\mu}}$, is also the solution to a linear programming problem.

After obtaining $\hat{c}_{\hat{\mu}}$, an objective function can also be chosen to find a better

$\boldsymbol{\rho}$. A function that works well is $\sum_{j=1}^{J} f_{\text{uRAC}}(B, G, \boldsymbol{\rho}, \hat{\mu}, y_j)$. On one hand, from a performance point of view, minimizing $\sum_{j=1}^{J} f_{\text{uRAC}}(B, G, \boldsymbol{\rho}, \hat{\mu}, y_j)$ corresponds to maximizing the gap area between $f(\cdot)$ and $y - \hat{c}_{\hat{\mu}}$, the latter is the upper-bound probability that a generation is not decodable at each stage of decoding. The larger the area is, the larger the portion of newly decodable generations we would have. On the other hand, the minimization is a least l_1-norm problem on $\boldsymbol{\rho}$, which produces a $\boldsymbol{\rho}$ with a large number of zero components [3]. This is a very important property because it simplifies generation construction significantly in the way that only several generation sizes are possible even when the degree spread (i.e., $d_{\max} - d_{\min}$) is large.

The generation-size distribution $\Omega(x)$ is then expressed in terms of $\boldsymbol{\rho}$ using the fact that $\Omega_m = G\rho_m/m, m = d_{\min}, \ldots, d_{\max}$.

5.6.3 Simulation Results

An example of unequal-size RAC is outlined here. For $d_{\min} = B = 32$, $d_{\max} = 64$, $G = 46$, and $\delta = 0.015$, we solve (5.20) and then apply the second refinement (we omit the first refinement as the obtained $\hat{c}_{\hat{\mu}}$ is very close to zero in this case). The resulting $\Omega(x)$ is

$$\begin{aligned}\Omega(x) &= 0.0179x^{33} + 0.1332x^{35} + 0.1229x^{38} \\ &\quad + 0.0507x^{39} + 0.1313x^{43} + 0.0782x^{44} + 0.1396x^{53} + 0.3262x^{54}.\end{aligned}$$

In Figure 5.9, we show the performance of the example code and compare it with the equal-size counterpart for various M. We show their overheads and computational costs measured as the average number of required operations per packet symbol in decoding. It is seen that uRAC has both lower overhead and decoding cost, even

though the two codes have the same G.

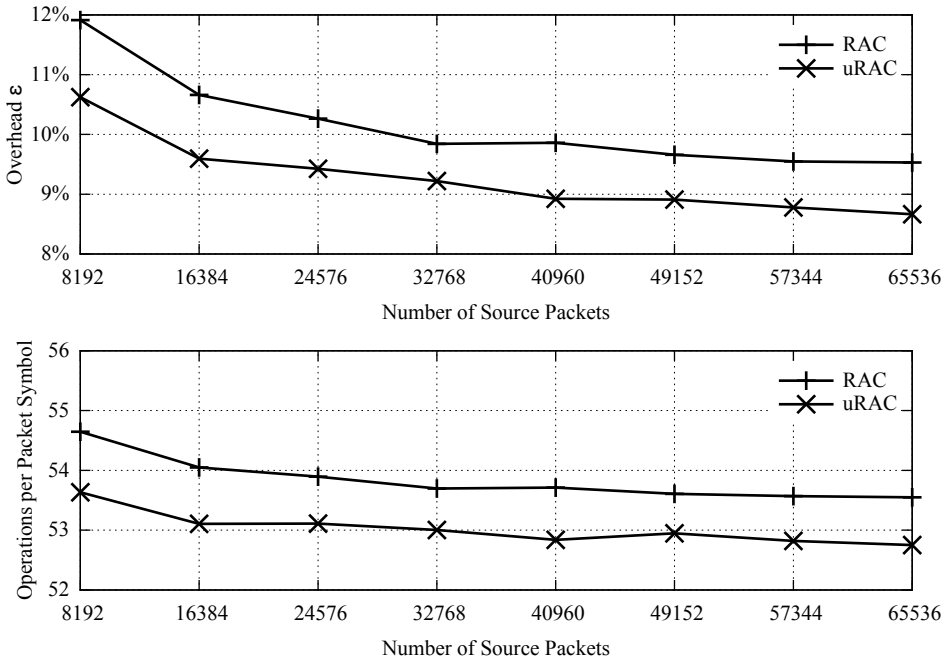

Figure 5.9: Performance of RAC and uRAC for various M with fixed $G = 46$; $B = 32$, $\delta = 0.015$.

Figure 5.10 shows the performance comparison between RAC and uRAC for the same $M = 65536$ with various G. We have not listed the generation-size distributions for all the G's to save space.

It is seen that for most of G, uRAC would achieve the same performance as RAC in both overhead and decoding cost. However, if G is increased, uRAC always has no higher overhead and decoding cost than RAC and the gap between the two widens as G increases. This suggests that uRAC would provide benefit when the optimal G might not be precisely estimated. As we will show in the next section, the optimal amounts of overlap for transmissions in more complex networks deviate somewhat from that in the end-to-end scenario. In these cases, using uRAC would be more

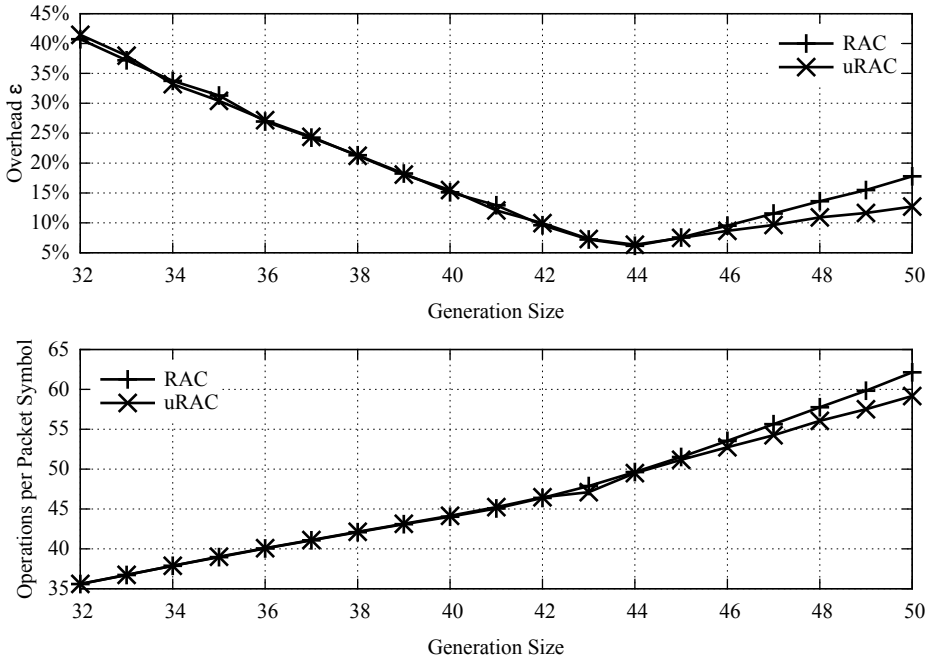

Figure 5.10: Performance of RAC and uRAC with various G; $M = 65536$, $B = 32$, $\delta = 0.015$.

compelling.

5.7 Network Performance of uRAC with MaLPI Scheduling

In this section, we evaluate our code designs in network scenarios where intermediate nodes may re-encode packets. We incorporate our proposed MaLPI scheduling strategy at intermediate nodes and compare the performance with the existing random scheduling strategy. Throughout the source node sends coded packets from each generation uniformly randomly.

We present simulation results for two networks: the two-hop line network of Figure 5.11 and the butterfly network of Figure 5.12. Each link of the networks is modeled as a lossy channel.

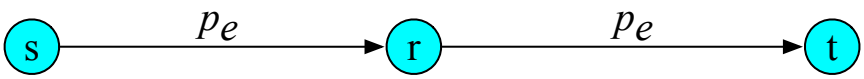

Figure 5.11: Two-hop lossy line network.

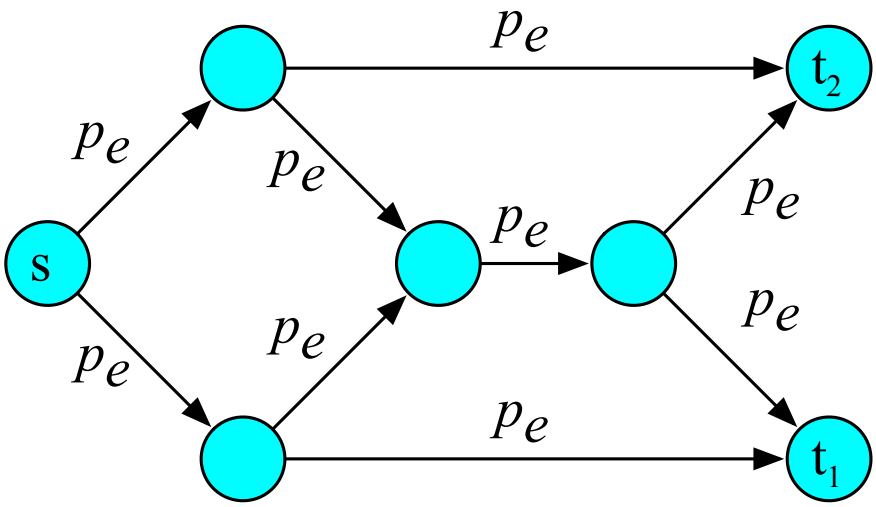

Figure 5.12: Butterfly lossy network.

In Figure 5.13, we show the overheads and decoding costs of transmitting a variety of RAC and uRAC codes over a two-hop lossy links, where the erasure probability of each hop is equally $p_e = 0.2$. The random scheduling is used at the relay node. It is seen that, compared to Figure 5.10 where codes are evaluated in the point-to-point scenario, the overheads in Figure 5.13 are much higher (about 20% higher), which means that much network-induced overhead is introduced by the added intermediate node. It is interesting to note that the minimum overheads of the two codes are achieved at $G = 43$ rather than the $G = 44$ in Figure 5.10. This suggests that the optimal amount of overlap estimated using the techniques from Section 5.5 may become less precise in network transmissions. However, it is noted that the advantage

of using unequal-size generations persists in the network environment.

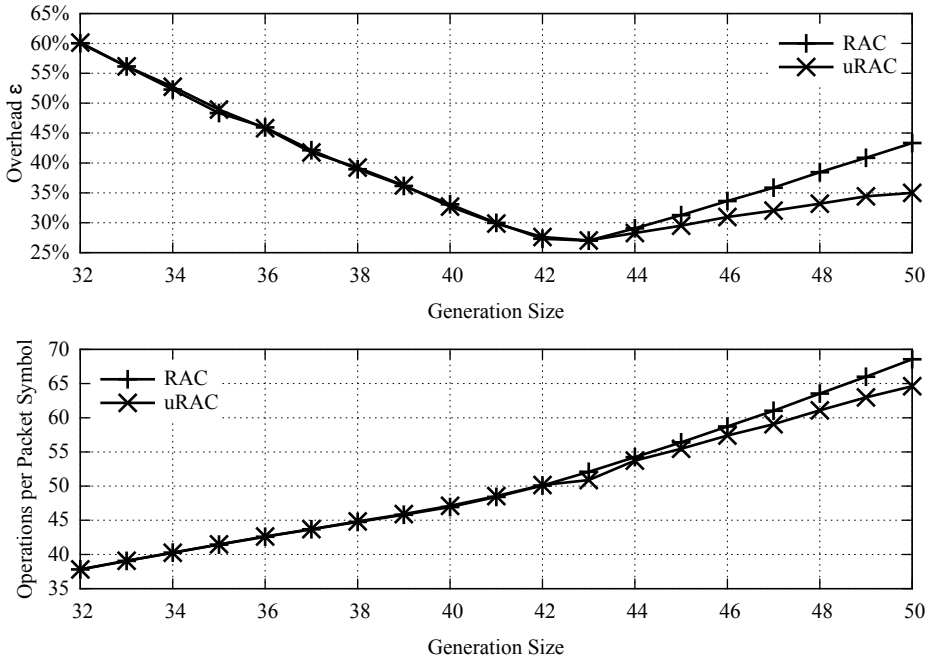

Figure 5.13: Performance of RAC and uRAC with various G over two-hop lossy link where the relay node performs random scheduling as in [44]; $M = 65536$, $B = 32$, $\delta = 0.015$, $p_e = 0.2$.

In Figure 5.14, we show performances of RAC and uRAC for different M and the fixed $G = 46$ estimated by the techniques from Section 5.5. Figure 5.14 shows that uRAC always achieves lower overhead and decoding cost than RAC.

We now replace the random scheduling strategy at the relay node with the proposed MaLPI scheduling. It is observed from Figure 5.14 that the overheads of RAC and uRAC are much reduced over the case when random scheduling is used. The decoding cost is also improved because the decoder needs to process fewer redundant packets. The overhead of uRAC is still lower. The achieved overhead of using uRAC with MaLPI scheduling is about 16% when M is sufficiently large; this corresponds to $R = \frac{p_e}{1+\varepsilon} \approx 0.69$ packets per network use, where a *network use* refers to where the

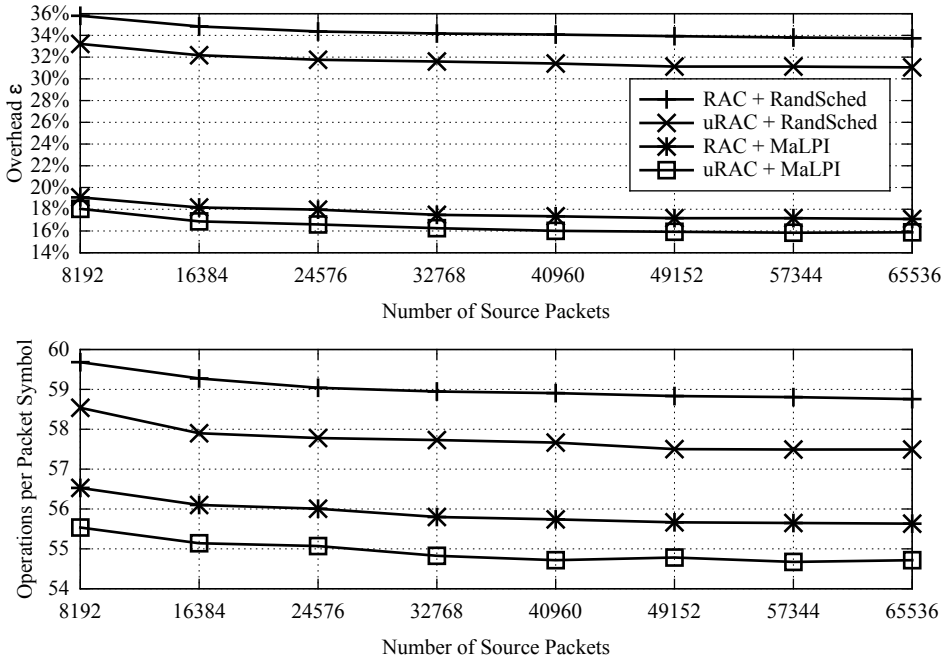

Figure 5.14: Performance of RAC and uRAC for various M over two-hop lossy link; $G = 46$, $B = 32$, $\delta = 0.015$, $p_e = 0.2$.

source and the relay node each transmit a packet. Note that the max-flow capacity of the network is 0.8 while the end-to-end capacity when no coding is allowed at the relay is $0.8 \times 0.8 = 0.64$. Therefore, the benefits of network coding are obtained.

In Figures 5.15 and 5.16, the same sets of RAC and uRAC codes as in Figures 5.13 and 5.14, respectively, are applied to the butterfly lossy network, where $p_e = 0.1$ is used for each link. We see that the overhead increases as the number of hops increases. Moreover, the lowest achieved overhead is now around $G = 42$, suggesting that the optimal amount of overlap has changed even more compared to that in the end-to-end and the two-hop lossy scenarios. Therefore, we note that our estimation of the optimal amount of overlap may be less precise in more complex networks. But again, the uRAC is shown to achieve lower overhead and decoding cost than RAC,

and using MaLPI at intermediate nodes reduce the overhead significantly compared to using random scheduling. The achieved overhead of using uRAC with MaLPI scheduling, 24%, corresponds to the rate of about 1.45 packets per network use in the network, whose max-flow capacity is 1.8 for $p_e = 0.1$.

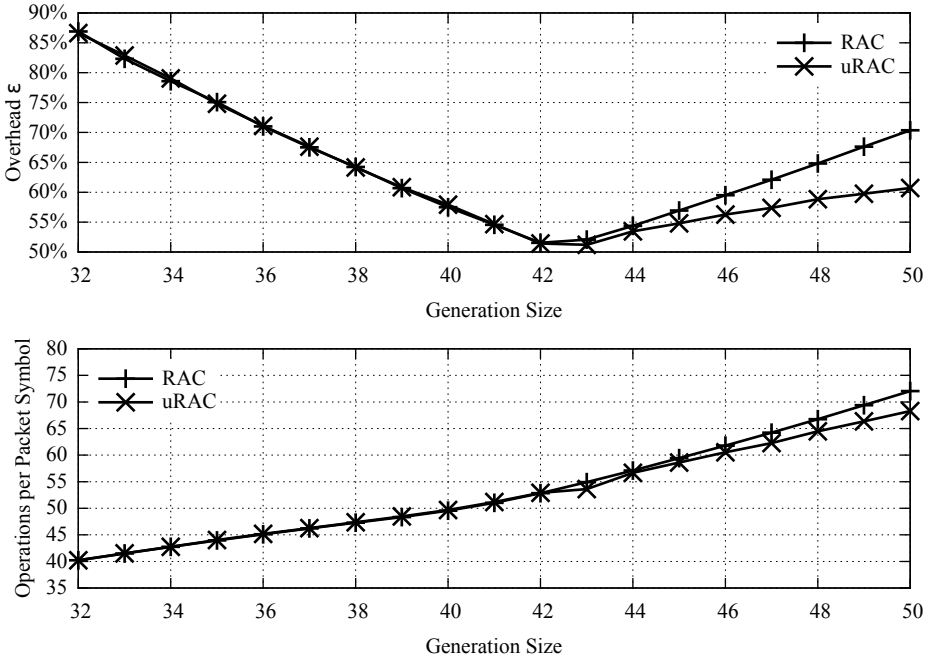

Figure 5.15: Performance of RAC and uRAC with various G over butterfly lossy link where the relay node performs random scheduling as in [44]; $M = 65536$, $B = 32$, $\delta = 0.015$, $p_e = 0.1$.

5.8 Summary and Conclusions

This chapter addresses sparse random linear network coding for general network topologies. We use generation-based network coding method to maintain the sparseness of code throughout the network. To improve the performance, we propose a maximum local potential innovativeness scheduling algorithm at intermediate nodes to improve upon previously used random scheduling and show that it reduces local

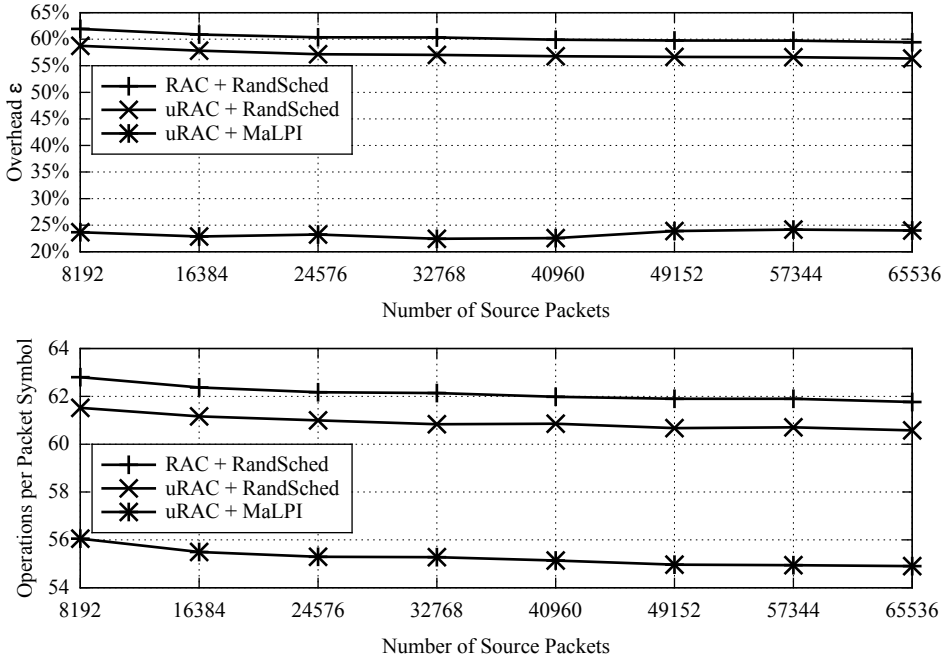

Figure 5.16: Performance of RAC and uRAC for various M over butterfly lossy link; $G = 46$, $B = 32$, $\delta = 0.015$, $p_e = 0.1$.

scheduling delay, which accounts for a large portion of the overhead at the destination. To design GNC codes, the code is modeled as a graph code and then we use techniques from graph codes to analyze the decoding process of GNC codes and design more efficient codes.

The proposed analysis technique is shown to be useful in understanding and designing important parameters of GNC codes, namely the optimal amount of overlap and generation sizes. It shows that introducing overlap among generation is essential to balance the difficulty of decoding between the beginning and the end. We use the analysis result to estimate the optimal amount of overlap that results in the lowest code overhead. In the end-to-end scenario, we can significantly reduce the overhead. When the code is used in more complex networks, it is shown that the

optimal amount of overlap may change due to the network-induced overhead, which is however not considered in our analysis. Therefore, the estimated amount of overlap gets less useful. In this case, it is shown that using unequal-size GNC code may be useful, because it has no higher overhead than the equal-size counterparts. The benefit is especially compelling when the optimal amount of overlap to be used cannot be precisely estimated.

The improvements achieved by the proposed scheme have significant potential in file distribution applications. Given the inherent ratelessness and the erasure-correcting capability of network coding, using GNC in cloud storage and peer-to-peer communications has the potential to reduce protocol overhead and improve network throughput significantly, especially when the UDP protocol is applied.

Chapter 6

Generation-Based Network Coding for Unknown Network Topologies: Overhead-Optimized Decoding

6.1 Introduction

In Chapter 5 we have discussed the design of generation-based network codes (GNC) for the generation-by-generation (G-by-G) decoder. It is seen that GNC codes with the G-by-G decoder can reduce the computational cost of RLNC in general network topologies while retaining the throughput benefits of RLNC.

As mentioned in the introduction of Chapter 5, a key performance metric of the GNC coding method is the reception overhead, whose components can be classified into three categories: encoder-induced, decoder-induced and network-induced. Although the G-by-G decoder considered in Chapter 5 has fast decoding, it may have nonzero decoder-induced overhead, which means that the decoder may not succeed as soon as M linearly independent packets are received.

In some application scenarios, low overhead may be a critical requirement. For

example, if packets are requested by users who are charged by the amount of transmitted data, reception overhead means a waste of money for users and should be avoided as much as possible. In this chapter, we aim to design GNC codes that have lower overheads than that of Chapter 5. Unlike in Chapter 5, we consider code overhead and decoder-induced overhead separately in this chapter. We design GNC codes for moderate number of source packets, M, to have negligible ($< 1\%$) code overhead and a decoder with zero decoder-induced overhead. By moderate we mean the value of M varies between hundreds and thousands, which is commonly seen in applications, for example the streaming media, where the supported number of packets of the playback buffer is moderate.

We first propose a general low-complexity overhead-optimized decoder for GNC codes. The decoder succeeds as soon as M linearly independent packets are received and hence has zero decoder-induced overhead. Generally, overhead-optimized decoding has high decoding cost and may not be practical for very large M. However, we show that for moderate M, the decoding may bring considerable advantages. The proposed decoder is termed *overlap-aware* (OA). It is able to exploit the sparseness of GNC codes whose decoding matrices may have deterministic or random structure. The decoder also provides flexibility to control the trade-off between overhead and decoding cost: it can achieve either zero decoder-induced overhead with slightly higher decoding cost than that of the G-by-G decoder or the same overhead as the G-by-G decoder with the same decoding cost, or some operating point in between.

The basic idea of OA decoding is to first process received packets *locally* within generations to obtain sparser coding vectors and then pivot the *global* sparse linear system to lower computational cost. The pivoting preserves matrix sparseness

during row reductions [10]. While the benefits of pivoting have been exploited in the maximum-likelihood (ML) decoding of sparse erasure-correction codes, such as LDPC codes [4], [50], [48] and standardized Raptor codes [57], its application to decode GNC codes has not been investigated to date. A crucial difference between GNC and erasure-correction codes is that GNC-coded packets mix with others from the same generation throughout the network, whereas erasure-correction coded packets are independent of one another after being generated. Therefore, while one round of pivoting suffices in [4], [50], [48], [57] to exploit the sparseness, for GNC codes local processing and multiple rounds of pivoting are necessary.

Given the above decoder, we then propose a GNC code that combines binary precoding, using random overlapping generations, with binary RLNC. We show that the proposed code can achieve close to zero code overhead and has efficient OA decoding. Precoding [56], [43] and random generation overlap [32] together ensure that the probability of receiving an innovative (i.e., linearly independent) packet does not decrease significantly after a large fraction of innovative packets has been received, avoiding overhead when collecting the last several innovative packets. The use of binary field $\mathbb{F}_2$ for RLNC within generations allows for simpler encoding and sparser coding vectors because operations are XOR and each coding coefficient has a probability of $\frac{1}{2}$ being zero. On the other hand, using $\mathbb{F}_2$ results in a smaller probability that a coded packet is innovative, which may incur overhead. However, we will show that through precoding and overlapping generations, we can achieve almost the same code overhead as codes in which higher order finite fields are used.

Similar overhead-optimized decoding of GNC codes has been considered in some prior work. In [17], [18], [11] and [21], GNC based on banded (or close-to banded)

matrices are designed. Each band of the decoding matrix corresponds to a generation. The decoding therein uses straightforward GE, which preserves the banded structure during row reductions and therefore has low decoding cost [15].

Our work differs from and improves upon [17], [11], [21] in several major ways. First, the proposed OA decoder has low complexity for GNC codes with more general overlapping patterns while the straightforward GE decoder in [17], [11], [21] only applies to GNC codes with banded decoding matrices. To achieve close-to-zero overhead, GNC codes with random overlap may be more desirable as we will show. Second, in achieving close-to-zero code overhead, the proposed design creates sparser codes due to the combined use of precoding and random overlap, with lowered computational cost, while no precoding is used in [17], [11], [21].

The rest of the chapter is organized as follows: Section 6.2 presents the model of generation-based network coding and describes the relevant encoding and decoding operations. The decoding problem of GNC codes is considered in Section 6.3, where we review existing decoders and propose the OA decoder. We analyze the OA decoder in Section 6.4. In Section 6.5, we present the code design and its OA decoding. We analyze its performance and propose a method to determine key parameters of codes. In Section 6.6, we evaluate our design by simulation. The chapter is summarized in Section 6.7.

6.2 Model: Generation-based Network Coding

We consider that M source packets, $\mathcal{S} = \{\mathbf{s}_i, 1 \leq i \leq M\}$, are presented to the source node to be sent to destination nodes over a network that contains intermediate nodes. Links between nodes are lossy and are modeled as erasure channels. Each

source packet consists of K message symbols from a finite field $\mathbb{F}_q$, where q is the finite field size. Each $\mathbf{s}_i$ is then a K-length row vector with elements from $\mathbf{F}_q$. A set of L generations $\mathcal{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_L\}$ are constructed from the M source packets. Each generation $\mathcal{G}_l = \left\{\mathbf{s}_1^{(l)}, \ldots, \mathbf{s}_{G_l}^{(l)}\right\}, l = 1, \ldots, L$, is a subset of source packets of size $G_l = |\mathcal{G}_l|$, where $\mathbf{s}_j^{(l)} = \mathbf{s}_i$ for some $1 \leq i \leq M$ for each $j = 1, \ldots, G_l$. We assume that $\cup_{l=1}^{L} \mathcal{G}_l = \mathcal{S}$, i.e., each source packet is present in at least one generation. A one-to-one index mapping $f_l(\cdot) : j \rightarrow i$ indicates that the i-th source packet is selected as the j-th packet in $\mathcal{G}_l$. The source packet indices in $\mathcal{G}_l$ are stored in the set $\mathcal{I}_l = \{f_l(1), f_l(2), \ldots, f_l(G_l)\}$. $\mathcal{G}$ is said to be *disjoint* if $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset, \forall i \neq j$ or else *overlapping*, and is of *equal-size* if $G_i = G_j, \forall i, j$ or else of *unequal-size*. For overlapping $\mathcal{G}$, $\sum_{l=1}^{L} G_l > M$.

A GNC code is defined on $\mathcal{G}$ as follows. For each transmission from the source node, a coded packet is generated as a random linear combination of source packets in a randomly chosen generation; coefficients are chosen from $\mathbb{F}_q$. Let $\mathcal{P} = \{p_1, p_2 \ldots, p_L\}$ be the set of probabilities where p_l denotes the probability that $\mathcal{G}_l$ is chosen when generating a packet, $1 \leq l \leq L$. The GNC code is then characterized by $(\mathcal{G}, \mathcal{P}, q)$. The GNC codes are *rateless* codes, meaning that a potentially unlimited number of coded packets may be generated. The source node is informed to stop transmission only after the destinations have recovered all the source packets. Let N' be the expected number of randomly generated packets among which there are M linearly independent ones. We define $\varepsilon_c = (N' - M)/M$ as the *code overhead* of the GNC code.

At intermediate nodes, packets are re-encoded from previously received packets of

a chosen generation. The re-encoding is assumed to be RLNC, i.e., re-encoded packets are random linear combinations of received packets at intermediate nodes. The procedure of choosing generations is called *scheduling*. Different scheduling strategies may be used, such as random scheduling [44], [32] and the maximum local potential innovativeness scheduling (MaLPI) proposed in Chapter 5.

At a destination, received packets of $\mathcal{G}_l, l = 1, \ldots, L$ are in the form of $\mathbf{r}^{(l)} = \sum_{i=1}^{G_l} g_i^{(l)} \mathbf{s}_i^{(l)}$, where $g_i^{(l)}$ is an encoding coefficient from a finite field. We refer to $\mathbf{g}^{(l)} = [g_1^{(l)}, g_2^{(l)}, \ldots, g_{G_l}^{(l)}]$ as a *generation encoding vector* (GEV) belonging to generation $\mathcal{G}_l$. The GEV is delivered in the header of each coded packet. Each GEV can be transformed to a length-M *encoding vector* (EV), denoted as $\mathbf{g}$, in which elements $g_{f_l(j)} = g_j^{(l)}$ for $j = 1, \ldots, G_l$ and the rest of the $M - G_l$ elements are zero.

Decoding of GNC codes is performed by solving linear systems of equations since the received packets are linear combinations of source packets. We refer to a received packet whose EV is not in the span of EVs of the previously received packets of the destination node as an *innovative packet* and the EV is referred to as an *innovative EV*. The receiver has to receive M innovative EVs to recover all the source packets.

Let N'' be a random variable denoting the number of received packets among which M innovative packets can be obtained in a transmission session, we refer to $\varepsilon_{cn} = (N'' - M)/M$ as the *code-and-network-induced overhead*. This overhead may be caused by either the encoding at the source node or scheduling and re-encoding at intermediate nodes, or both, but the receiver cannot distinguish the cause of the overhead. Supposing that the decoder successfully decodes M source packets after receiving $N \geq N''$ packets, $\varepsilon_d = (N - N'')/M$ is referred to as the *decoder-induced*

overhead. Zero decoder-induced overhead is achieved if the decoder succeeds immediately when M innovative packets are received. The overall reception overhead of the session is $\varepsilon = (N - M)/M = \varepsilon_{cn} + \varepsilon_d$.

The *decoding cost* of GNC codes is measured as the number of finite field arithmetic operations required in decoding all source packets, where an *operation* refers to either a divide or a multiply-and-add between two elements of a finite field.

6.3 Decoding Algorithms of GNC Codes

In this section, we present the GNC decoding algorithms. We first review the generation-by-generation (G-by-G) decoder and identify its decoder-induced overhead. Two approaches are used to improve efficiency: the first approach serves as a baseline and the second is the proposed OA decoder.

6.3.1 G-by-G Decoder

Definition 6.1 (G-by-G Decoder)**.** *At the beginning of each step, the decoder attempts to select a generation $\mathcal{G}_l, 1 \leq l \leq L$ and decode it by solving $\mathbf{A}_l\mathbf{X}_l = \mathbf{B}_l$ using GE. Successive rows of $\mathbf{A}_l$ and $\mathbf{B}_l$ are the GEVs and information symbols of received coded packets belonging to $\mathcal{G}_l$, respectively. Rows of $\mathbf{X}_l$ are the undecoded source packets in $\mathcal{G}_l$. A generation $\mathcal{G}_l$ whose $\mathbf{A}_l$ is full-rank is called separately decodable. When one generation is decoded, decoded packets are subtracted from received packets of remaining undecoded generations which also contain the decoded packets. This marks the end of the current step. Since the subtraction may reduce the numbers of unknown packets of other generations, new separately decodable generations may be found. If this is the case, the next decoding step can begin; if no such generations can be found,*

decoding waits until further packets are collected such that a new separately decodable generation is found. The decoder proceeds until all generations are decoded.

The following example shows that the above G-by-G decoder has nonzero decoder-induced overhead.

Example 6.1 (Inefficiency of G-by-G decoding)**.** *Assume that* 4 *source packets are grouped into two generations* $\mathcal{G}_1 = \{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3\}$ *and* $\mathcal{G}_2 = \{\mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4\}$*. Suppose that* 2 *packets have been received for each generation, written as* $\mathbf{r}_1^{(1)} = \mathbf{s}_1 + \mathbf{s}_2$*,* $\mathbf{r}_2^{(1)} = \mathbf{s}_2 + \mathbf{s}_3$ *and* $\mathbf{r}_1^{(2)} = \mathbf{s}_2 + \mathbf{s}_4$*,* $\mathbf{r}_2^{(2)} = \mathbf{s}_2 + \mathbf{s}_3 + \mathbf{s}_4$*, respectively. In this case, the* 4 *packets are linearly independent. However, neither generation is separately decodable and we have to receive more packets if using the G-by-G decoder.*

6.3.2 Straightforward Overhead-Optimized Decoding

The inefficiency of G-by-G decoding arises from decoding each generation separately and possible overlaps among generations are only checked after a generation is decoded. To resolve the issue straightforward overhead-optimized approach may be used, i.e., directly solving $\mathbf{AX} = \mathbf{B}$ using GE where successive rows of the $M \times M$ $\mathbf{A}$ and the $M \times K$ $\mathbf{B}$ are the EVs and information symbols of received coded packets, respectively, and $\mathbf{X}$ contains all the M source packets. In this case, decoding is successful as soon as M innovative packets are received, resulting in zero decoder-induced overhead. For illustration purposes, in the following we focus on the operations on $\mathbf{A}$ and omit the corresponding operations on $\mathbf{B}$.

The approach straightforwardly performs GE on each received packet, whose GEV is transformed to a length-M EV. We refer to this decoder as the *naive decoder*. The key component is the forward elimination that results in a full-rank upper triangular

$\mathbf{A}$ when M innovative packets are received. The detailed procedure is presented as Algorithm 3, where $\mathbf{A}[i]$ and $\mathbf{A}[i][i]$ denote the i-th row and the i-th diagonal element of $\mathbf{A}$, respectively, and $i_{\mathrm{LM}}(\mathbf{g})$ denotes the coordinate of the left-most nonzero element of vector $\mathbf{g}$. The upper triangular matrix is converted to an identity matrix using standard backward substitution of GE.

Algorithm 3 Progressive Forward Elimination of GE

```
1: Initialize M × M zero matrix A
2: Initialize z(A) = M                          ▷ number of zero rows of A
3: while z(A) > 0 do
4:     Receive a packet whose EV is g
5:     stored ← 0
6:     while stored = 0 and g ≠ 0 do
7:         i ← i_LM(g)
8:         if A[i][i] = 0 then
9:             A[i] ← g
10:            stored ← 1
11:            z(A) ← z(A) − 1
12:        else
13:            t ← g[i]/A[i][i]
14:            g ← g + A[i] × t
15:        end if
16:    end while
17: end while
```

Since the naive decoder does not take into account the fact that the EVs may be sparse, the decoding cost may be high. Note that the naive decoder is efficient for GNC codes with banded decoding matrices, because GE preserves the band structure [15]. However, this is not the case for GNC codes with more general structures. Nonetheless, the above decoder does guarantee zero decoder-induced overhead and therefore can be used as a reference decoder for overhead comparison.

6.3.3 Overlap-Aware Decoder

We now present a new *overlap-aware* (OA) overhead-optimized decoder for GNC codes with the same overhead as the naive decoder but at a lower computational cost. The decoder is overlap aware because even though some generations may also be separately decodable, the OA decoder is better able to exploit possible overlaps among generations. We first define the following:

Definition 6.2 (Solo packets)**.** *Solo packets refer to source packets that are only present in one generation.*

Definition 6.3 (Overlapping packets)**.** *Overlapping packets refer to source packets that are present in at least two generations.*

Definition 6.4. *$\mathcal{D}_l$ and $\mathcal{O}_l$ denote the subsets of indices of solo and overlapping packets in each generation $\mathcal{G}_l$, $l = 1, \ldots, L$, respectively; $D_l = |\mathcal{D}_l|$, $O_l = |\mathcal{O}_l|$ and $D_l + O_l = G_l$. Note that $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset, \forall i \neq j$. The total numbers of solo and overlapping packets are denoted by $M_D = \sum_l D_l$ and $M_O = M - M_D$, respectively.*

In the following, we assume that coding coefficients in received GEVs and EVs are sorted in increasing order of the number of times that the corresponding source packets are included in all generations. Therefore, coding coefficients corresponding to solo packets are positioned at the first coordinates of GEVs and EVs.

The OA decoding is first performed *locally* in each generation. Let $\mathbf{A}_l$ and $\mathbf{B}_l$ denote the decoding and message matrices of $\mathcal{G}_l, l = 1, \ldots, L$, respectively. We refer to the $G_l \times G_l$ $\mathbf{A}_l$ as the *local decoding matrix* (LDM) of $\mathcal{G}_l$. The local decoding performs a similar progressive forward elimination as Algorithm 3 on the GEVs to populate the $\mathbf{A}_l$'s. Each stored GEV (step 9 of Algorithm 3) is referred to as an

innovative GEV of the current generation. Note that an innovative GEV may not be an innovative EV. We declare the decoder as *OA ready* when 1) the first D_l rows of each $\mathbf{A}_l$ are not zero and 2) a total of M innovative GEVs have been received.

When the decoder is OA ready, we attempt to *jointly* decode generations. For joint decoding, $\mathbf{A}_l, l = 1, \ldots, L$ are first *partially diagonalized* by eliminating elements above nonzero diagonal elements, which makes $\mathbf{A}_l$ even sparser. The M innovative GEVs in the resulting $\mathbf{A}_l$'s are then converted to EVs and put into the *global decoding matrix* (GDM) $\mathbf{A}$. The detailed GDM construction is summarized as Algorithm 4. The resulting GDM can be expressed in the form:

$$\mathbf{A} = \begin{bmatrix} \mathbf{D} & \mathbf{U} \\ \mathbf{0} & \mathbf{T} \end{bmatrix}, \tag{6.1}$$

where $\mathbf{D}$ is an $M_D \times M_D$ diagonal matrix whose diagonal elements are from the $D_l \times D_l$ diagonal sub-matrix of each $\mathbf{A}_l$ and $\mathbf{T}$ is an $M_O \times M_O$ matrix.

Lemma 6.1. *OA readiness is a necessary condition for the successful decoding of GNC codes.*

Proof. Only one or zero innovative EV may be generated from an innovative GEV. Therefore, if fewer than M innovative GEVs are received, no decoder can succeed as $\mathbf{A}$ is rank deficient. Consider that M innovative GEVs are received but the LDM $\mathbf{A}_l$ of generation $\mathcal{G}_l$ has one of the first D_l rows being zero; this means that there is one zero row among the first M_D rows of $\mathbf{A}$. Even though M innovative GEVs are available, there are more than M_O innovative GEVs whose left-most nonzero elements are at coordinate $D_l + 1$ or greater, i.e., these GEVs are only coded across overlapping packets; there are at most M_O independent vectors among them. $\mathbf{A}$ is therefore not

Algorithm 4 Construction of GDM

Input: $\mathbf{A}_l, l = 1, \ldots, L$
Initialize $M \times M$ zero matrix $\mathbf{A}$
for $l = 1$ to L **do**
    for $k = G_l$ to 1 **do**
        if $\mathbf{A}_l[k][k] \neq 0$ **then**
            for $j = 1$ to k **do**
                $t \leftarrow \mathbf{A}_l[j][k]/\mathbf{A}_l[k][k]$
                for $m = k + 1$ to G_l **do**
                    if $\mathbf{A}_l[m][m] = 0$ **then**
                        $\mathbf{A}_l[j][m] \leftarrow \mathbf{A}_l[j][m] + \mathbf{A}_l[k][m] \times t$
                    end if
                end for
                $\mathbf{B}_l[j] \leftarrow \mathbf{B}_l[j] + \mathbf{B}_l[k] \times t$
                $\mathbf{A}_l[j][k] \leftarrow 0$
            end for
        end if
    end for
    $n_d \leftarrow 0$, $n_o \leftarrow 0$
    for $k = 1$ to D_l **do**
        Convert GEV $\mathbf{A}_l[k]$ to EV
        $n_d \leftarrow n_d + 1$
        $\mathbf{A}[n_d] \leftarrow \mathbf{g}$
    end for
    for $k = D_l + 1$ to G_l **do**
        if $\mathbf{A}_l[k][k] \neq 0$ **then**
            Convert GEV $\mathbf{A}_l[k]$ to EV $\mathbf{g}$
            $n_o \leftarrow n_o + 1$
            $\mathbf{A}[M_D + n_o] \leftarrow \mathbf{g}$
        end if
    end for
end for

full-rank and decoding cannot be successful. □

Some properties of matrix $\mathbf{T}$ should be noted. First, in each length-M_O row of $\mathbf{T}$ there are at most $O_M = \max\{O_l, l = 1, \ldots, L\}$ nonzero elements. In GNC codes, $O_M \ll M_O$ and therefore $\mathbf{T}$ tends to be sparse. Second, due to the assumed orderings of GEV and EV, the last few columns of $\mathbf{T}$ contain more nonzero elements because the corresponding source packets are covered for more times in all generations. Third, if any generation is already separately decodable, its LDM has been fully diagonalized, resulting in some rows in $\mathbf{T}$ that are *singleton*, i.e., contain only one nonzero element.

The joint decoding needs to transform $\mathbf{T}$ to an $M_O \times M_O$ identity matrix. Due to $\mathbf{T}$'s sparse structure, $\mathbf{T}$ may be pivoted as described in next subsection such that the transformation has low computational cost. Once $\mathbf{T}$ is transformed, M_O source packets are decoded. By back-substituting these packets, we can eliminate $\mathbf{U}$ in (6.1) and completely decode the remaining source packets.

6.3.4 Pivoting T in OA Decoder

The purpose of pivoting is to reorder rows and columns of a sparse matrix such that *fill-ins* can be avoided, where a fill-in refers to a zero element in the matrix that is changed to nonzero during row reductions, thereby increasing computational cost thereafter. Unfortunately, finding the globally optimal pivoting sequence that minimizes the number of fill-ins of a matrix is known to be an *NP-complete* problem [53]. Therefore, we only consider local heuristic methods. We propose the following method that employs two rounds of pivoting:

First Round

We first pivot $\mathbf{T}$ using an *inactivation* approach. Inactivation [46], [51], employed in decoding of standardized raptor codes [57], is a special type of pivoting in which only a nonzero element of a singleton row can be chosen as the pivot in each step. If no singleton rows can be found, *one or more columns that contain the largest number of nonzero elements are declared as inactive* such that singleton rows can be found in the resultant sub-matrix that comprises the *active* columns. The reordered matrix after pivoting consists of a number of sparse active columns (*active part*) and some other dense inactive columns (*inactive part*). Since pivots of the active part are from singleton rows, after reordering the active part will be lower triangular and therefore can be diagonalized without incurring any fill-ins. This is shown in Figure 6.1, where $\mathbf{0}$ refers to areas consisting of only zero elements, and $\mathbf{U}_I$ and $\mathbf{T}_I$ refer to two sub-matrices of the inactive part after diagonalizing the active part of $\mathbf{T}$. The complexity of inactivation pivoting is $\mathcal{O}(n)$ for an $n \times n$ matrix.

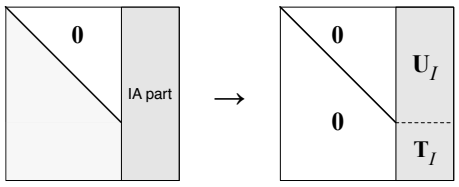

Figure 6.1: Reordering and partial diagonalization of $\mathbf{T}$ using inactivation pivoting.

Second Round

After the first round of pivoting, $\mathbf{T}_I$ of Figure 6.1 becomes dense due to row reductions to diagonalize the active part of $\mathbf{T}$. However, due to the structure of GNC codes (nonzero elements of the inactive part might not be uniformly located), $\mathbf{T}_I$ may still

be sparse. We perform another round of pivoting on $\mathbf{T}_I$ to exploit this sparsity.

The second round pivots $\mathbf{T}_I$ using a modified *Markowitz criterion* due to Zlatev [68]. The Markowitz criterion (see e.g. [10]) selects a nonzero element $\mathbf{T}_I[i][j]$ as the pivot if it has the smallest *Markowitz count* of the matrix, defined as $(r_i - 1)(c_j - 1)$, where r_i and c_j are the numbers of nonzero elements on the corresponding row and column, respectively. After finding each pivot, the numbers of nonzero elements on the rows and columns of the remaining sub-matrix are updated. Note that the criterion ensures that if there is a singleton row (which rarely occurs because $\mathbf{T}_I$ is much denser than the original $\mathbf{T}$), its nonzero element will be chosen as the pivot. $\mathbf{T}_I$ is reordered after pivoting such that the i-th selected pivot is the i-th diagonal element of the reordered matrix. The modification used in Zlatev pivoting is that each pivot is searched only from a constant number (normally ≤ 3) of rows with the least number of nonzero entries. In contrast, the original Markowitz criterion searches $n - i + 1$ rows for the i's pivot. Therefore, the complexity of Zlatev pivoting is also $O(n)$ for an $n \times n$ matrix while that of the original Markowitz criterion is $\mathcal{O}(n^2)$.

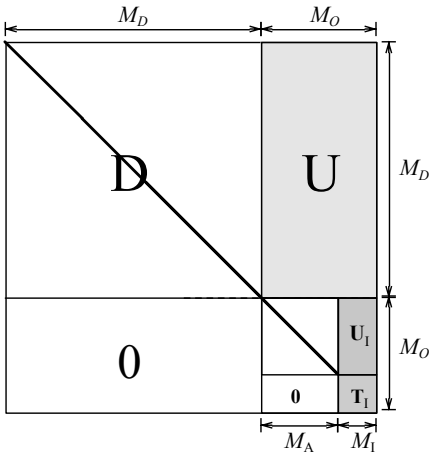

Figure 6.2: The final form of GDM A using inactivation pivoting.

After two rounds of pivoting, the final form of $\mathbf{A}$ is shown in Figure 6.2, where M_I is the number of inactivated columns and $M_A = M_O - M_I$. Regular row reductions are then performed on the smaller dimension matrix $\mathbf{T}_I$ to reduce it to an identity matrix. Eliminating $\mathbf{U}_I$ and $\mathbf{U}$ will recover all the source packets.

6.4 Analysis of OA Decoder

6.4.1 Zero Decoder-Induced Overhead

Given M innovative EVs, from Lemma 6.1 the decoder must be OA ready. Since M innovative EVs are received, the constructed $\mathbf{A}$ in (6.1) is full-rank. Since pivoting does not change the matrix rank, OA decoding is guaranteed to succeed and therefore has zero decoder-induced overhead. On the other hand, given M innovative EVs, the G-by-G decoding may have nonzero decoder-induced overhead, as illustrated in Example 6.1.

6.4.2 OA Decoding Cost

Let us consider the OA decoding cost when M innovative EVs are received. For simplicity, we consider generations of equal-size, G. From the previous section, OA decoding includes the following steps: (a) process received packets in each generation and partially diagonalize each LDM when constructing the GDM; (b) diagonalize the active part of $\mathbf{T}$; (c) reduce $\mathbf{T}_I$ to the unit matrix to decode M_I packets after the second round of pivoting; (d) back-substitute M_I decoded packets to eliminate $\mathbf{U}_I$, which decodes M_A packets; (e) back-substitute $M_O = M_A + M_I$ packets to eliminate $\mathbf{U}$, which decodes the M_D solo packets.

Parts (a) and (e) are two sub-parts of full GE in each generation and can be

	Number of Operations	
Parts	LHS	RHS
(a)+(e)	$\frac{2G^3+3G^2-5G}{6}L$	LG^2K
(b)	$\rho(\frac{M_A^2}{2}+M_AM_I)(M_I+1)$	$\rho(\frac{M_A^2}{2}+M_AM_I)K$
(c)	$\frac{2M_I^3+3M_I^2-5M_I}{6}$	M_I^2K
(d)	0	M_IM_AK

Table 6.1: Number of operations required in OA decoding.

combined. Let ρ denote the fraction of nonzero elements in $\mathbf{T}$ when the GDM is constructed. Since $\mathbf{T}$ is sparse, $\rho \ll 1$. The numbers of required operations involved in all parts on the left-hand side (LHS) and right-hand side (RHS) can be explicitly counted using [15] and are listed in Table 6.1. Note that although we have pivoted $\mathbf{T}_I$ using Zlatev pivoting, we still count computations in part (c) as full GE for simplicity, which is an upper bound of the actual cost.

We are more interested in the case that K is large and hence operations on RHS dominate the decoding cost. It is clear that decoding cost decreases as ρ and M_I decrease, whose values are determined by the specific GNC code design.

An important observation here is that if we have $M_I = 0$, i.e., inactivated no columns when pivoting $\mathbf{T}$, then $M_A = M_O$ and the total number of computations is $LG^2K+\rho\frac{M_A^2}{2}K$, which is exactly the decoding cost of the G-by-G decoder where LG^2K and $\rho\frac{M_A^2}{2}$ correspond to the cost of solving each generation and subtracting decoded packets from other generations, respectively. Therefore, the extra computational cost of OA decoding compared to G-by-G decoding only comes from the inactivated columns in $\mathbf{T}$.

Note that OA decoder has the flexibility to control the trade-off between overhead and decoding cost by postponing its joint decoding. If the joint decoding begins

after more than M innovative GEVs are received (i.e., postpones the declaration of OA ready), the code-and-network-induced overhead may increase and the decoder-induced overhead may no longer be zero. However, the decoding cost would decrease. This is because later joint decoding tends to increase the number of separately-decodable generations, and hence the number of singleton rows in $\mathbf{T}$, resulting in fewer columns that need to be inactivated. The naive decoder, however, does not have this property because it decodes generations jointly from the very beginning.

We notice that OA decoding would have the same decoding cost as that of G-by-G decoding if OA decoder were to be allowed the same number of received packets with which G-by-G decoding is successful. To sketch a proof, it is equivalent to showing that the number of inactivated columns using the OA decoder is zero. Without loss of generality, we may assume that the generations would be successfully decoded in the order $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_L$ with $N > M$ received packets using the G-by-G decoder and that in each step only one generation is separately decodable. In OA decoding, this corresponds to where only the LDM of $\mathcal{G}_1$ is fully diagonalized when constructing the GDM. This results in a number of singleton rows in $\mathbf{T}$, whose nonzero entries are chosen as pivots. Nonzero elements of the corresponding columns of the chosen pivots in $\mathbf{T}$ are then eliminated. The rows belonging to the eliminated elements are the EVs transformed from GEVs of generations that are overlapping with generation $\mathcal{G}_1$. If the G-by-G decoder would proceed, $\mathcal{G}_2$ should now be found to be separately decodable. In OA decoding, this corresponds to the case where the LDM of $\mathcal{G}_2$ had been partially diagonalized except for some columns, whose nonzero elements are exactly among those just eliminated elements. Therefore, after eliminating these elements, the LDM of $\mathcal{G}_2$ should now be diagonal, which results in new singleton rows

in $\mathbf{T}$; no columns need to be inactivated. The claim can then be proved by induction.

6.5 Code Design and OA Decoding Performance

We next design a GNC code that is able to achieve close to zero code overhead and can be decoded using the proposed decoder at low decoding cost.

6.5.1 Code Description

The code is based on the *random annex code* (RAC), which is proposed in [32] originally for the G-by-G decoder. The design of code parameters for the OA decoder, however, differs significantly from that for the G-by-G decoder. A RAC that contains L generations of equal size G is defined as follows:

Definition 6.5 (Random Annex Code [32])**.** *The M source packets, $\mathcal{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_M\}$, are first partitioned into L disjoint subsets $\mathcal{B}_l = \{\mathbf{s}_{(l-1)B+1}, \ldots, \mathbf{s}_{lB}\}, l = 1, \ldots, L$ of equal size B (we assume L to be a divisor of M, $M = LB$; otherwise null packets can be used for padding), one per generation. $\mathcal{B}_l$ is referred to as the base part of the generation. After that, each generation $\mathcal{G}_l$ is equipped with a random annex of size H, denoted as $\mathcal{H}_l$, which consists of a random selection of H packets from $\mathcal{S} - \mathcal{B}_l$. The annex code introduces overlap between generations. The overall generation $\mathcal{G}_l = \mathcal{B}_l \cup \mathcal{H}_l$ and $G = |\mathcal{G}_l| = B + H$. When generating a coded packet, one generation is chosen uniformly at random.*

The RAC is chosen to base our design for its following properties: first, its code overhead approaches zero quickly when increasing the amount of overlap (equivalently the generation size), and therefore a full rank yet sparse decoding matrix can be obtained; second, its GDM is close to being uniformly random and sparse and is

therefore suited to OA decoding; third, its random structure enables performance analysis.

The proposed design has two customizations relative to the original RAC: 1) *precoded* source packets using a binary systematic erasure-correction code and 2) only binary field is used when coding a packet. We therefore refer to it as the *precoded binary RAC* (PB-RAC). The detailed construction is as follows:

Definition 6.6 (Precoded Binary Random Annex Code)**.** *The M source packets are first precoded using a binary systematic code to obtain $(1+\theta)M$ intermediate packets, $\{\mathbf{s}_1, \ldots, \mathbf{s}_M, \mathbf{c}_1, \ldots, \mathbf{c}_S\}$, where $\mathbf{c}_i = \sum_{j=1}^{M} w_{i,j}\mathbf{s}_j, i = 1, \ldots, S$ are parity-check packets, the $w_{i,j}$ are coding coefficients chosen from $\mathbb{F}_2$ and $S = \theta M$. RAC is then applied to the intermediate packets, i.e., $L' = \lceil \frac{(1+\theta)M}{B} \rceil$ generations are formed from the intermediate packets with the same base part size B and generation size G.*

Similar to Raptor codes [56] where precoding is also used, we assume that the precoding coefficients of PB-RAC are known *a priori* to all the receivers of the transmission. This can be achieved by setting up a protocol between source and destination nodes.

6.5.2 OA Decoding of PB-RAC

The PB-RAC can be decoded very effectively using the OA decoder. Unlike successive decoding network codes and precodes as seen in previous works, e.g. [62], where the decoder needs to recover a pre-defined fraction of intermediate packets before the precode decoder can begin, the structure of OA decoder enables joint decoding of the two parts of PB-RAC from the beginning. The joint decoding ensures that decoding is successful if M innovative EVs are received. Note that the successive decoding

approach does not guarantee that a pre-defined fraction of intermediate packets to be recoverable from M innovative network coded packets, i.e., successive decoding has nonzero decoder-induced overhead.

The key idea of OA decoding of PB-RAC is to treat the S parity-check constraint equations of the precode as encoding vectors (whose corresponding message packets are all-zero) belonging to a special *precoding generation*. This is because $\sum_{j=1}^{M} w_{i,j}\mathbf{s}_j + \mathbf{c}_i = 0$ for $i = 1, 2, \ldots, S$. The precoding generation covers all the $M+S$ intermediate packets and its corresponding decoding matrix can be expressed in the form

$$\mathbf{H} = \begin{bmatrix} \mathbf{W} & \mathbf{I}_{S\times S} \end{bmatrix}, \tag{6.2}$$

where elements $w_{i,j}$ of $\mathbf{W}$, $1 \le i \le S$, $1 \le j \le M$ are coding coefficients of the systematic precode and $\mathbf{I}_{S\times S}$ is the size-S identity matrix. It is noted that

$$\begin{bmatrix} \mathbf{W} & \mathbf{I}_{S\times S} \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_M \\ \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_S \end{bmatrix} = \mathbf{0}_{S\times K}, \tag{6.3}$$

where on the right-hand side $\mathbf{0}_{S\times K}$ is the $S \times K$ zero matrix.

The OA decoding of PB-RAC is almost identical to that of normal GNC codes. The only differences are first that there are $(L' + 1)$ generations where the extra generation (i.e., the precoding generation) covers all the intermediate packets, so *no intermediate packets are solo* in PB-RAC. The GDM is constructed as soon as M

innovative GEVs are received and the resulting GDM $\mathbf{A}$ of (6.1) only consists of $\mathbf{T}$ (i.e., $\mathbf{D}$ is empty). Second, since RAC is applied to $M+S$ intermediate packets, $\mathbf{A}$ has $M+S$ columns rather than M as in the non-precoded case.

Pivoting will then be performed on the linear system of equations:

$$\begin{bmatrix} \mathbf{A} \\ \mathbf{H} \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_M \\ \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_S \end{bmatrix} = \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix}, \tag{6.4}$$

where the zero matrix on the right-hand side is of size $S \times K$. We denote the binary $(M+S)\times(M+S)$ matrix $\mathbf{A}_{\text{eff}} = \begin{bmatrix} \mathbf{A} \\ \mathbf{H} \end{bmatrix}$, where $\mathbf{A}$ is of size $M \times (M+S)$ and $\mathbf{H}$ is of size $S \times (M+S)$. A good precode ensures that $\mathbf{A}_{\text{eff}}$ is full-rank with high probability when a rank-M $\mathbf{A}$ is received.

6.5.3 Analysis of RAC with Precoding

In this subsection we examine how the parameters (i.e., M, θ, B and G) affect code overhead and decoding cost by analyzing a precoded RAC where a sufficiently large finite field size is used. Intuitively, PB-RAC would have higher code overhead but lower decoding cost. In Section 6.6, we will show that the difference in the overhead between using binary and larger finite field is however very small while that in decoding cost is indeed considerable, suggesting that PB-RAC is more desirable.

First we are interested in the code overhead, i.e., how innovative EVs are received

by the decoder. The analysis in the following assumes that a naive decoder with unlimited computation power is used. This does not affect the overhead results because OA decoding has the same overhead as the naive decoder. We need to receive M innovative length-$(M+S)$ EVs. Consider that $k < M$ innovative packets are received. The on-the-fly $k \times (M+S)$ GDM can always be transformed to the form

$$\mathbf{A}_{\text{otf}} = \begin{bmatrix} \mathbf{I}_{k\times k} & \mathbf{U} \end{bmatrix} \tag{6.5}$$

through row/column exchanges and row reductions, where $\mathbf{U}$ is a sub-matrix consisting of some random elements. We refer to the intermediate packets corresponding to the first k columns of $\mathbf{A}_{\text{otf}}$ as *received pivots*. With a sufficiently large finite field size, if the next received packet is from a generation that contains at least one intermediate packet other than the received pivots, its EV is innovative, i.e., after performing row reductions against the k rows of $\mathbf{A}_{\text{otf}}$, the resulting EV is not zero. Assuming that packets of different generations arrive with equal probability $1/L'$, the probability that a new packet is innovative when k innovative packets have been received is

$$p_k = \frac{a_k}{L'}, \quad 0 \le k \le M-1, \tag{6.6}$$

where a_k denotes the number of generations covering at least one of the remaining $((1+\theta)M - k)$ not-received pivots. This is equivalently the expected number of generations that an arbitrary subset of $((1+\theta)M - k)$ packets may be connected to. Given G and B, each packet is present in about G/B generations, and therefore a_k is approximately

$$a_k = L' - \left(1 - \frac{G}{BL'}\right)^{(1+\theta)M-k} L', \tag{6.7}$$

and hence

$$p_k = 1 - \left(1 - \frac{G}{(1+\theta)M}\right)^{(1+\theta)M-k}. \tag{6.8}$$

It is noted that (6.7) is only an approximation because it assumes that each packet independently randomly selects its G/B generations.

Let X_k denote the number of required received packets to obtain the $(k+1)$-th innovative packet when k innovative packets have been received; X_k is a geometric random variable with parameter p_k so $\mathrm{E}\{X_k\} = \frac{1}{p_k}$. The expected number of packets that are needed to obtain M innovative packets is then

$$\begin{aligned} \mathrm{E}\{C_M\} &= \mathrm{E}\left\{\sum_{k=0}^{M-1} X_k\right\} \\ &= \sum_{k=0}^{M-1} \frac{1}{p_k} \\ &= M + \sum_{k=0}^{M-1} \frac{1}{\left(1 - \frac{G}{(1+\theta)M}\right)^{k-(1+\theta)M} - 1}. \end{aligned} \tag{6.9}$$

We see from (6.9) that the code overhead $\mathrm{E}\{C_M\}/M - 1$ can be reduced to be arbitrarily close to 0 by increasing either G (i.e., the amount of overlap; note that the amount of overlap cannot be increased unboundedly in designing RAC for the G-by-G decoder [32], [30]) or θ (the precode rate). However, from the decoding cost point of view, a large θ should be avoided, because we may need to inactivate at least θM columns when pivoting $\mathbf{A}_{\text{eff}}$ (since $\mathbf{H}$ is normally dense). Similarly, we cannot increase G too much either, because that would result in a dense $\mathbf{A}$.

To understand the OA decoding cost, in the following we estimate the fraction of columns that would be inactivated when pivoting $\mathbf{A}_{\text{eff}}$. For the analysis, we make

the following further assumptions: 1) a set of parameters $\{M, S, G, B\}$ are chosen so that the first M received packets are linearly independent; 2) after uniformly randomly inactivating αM columns in $\mathbf{A}$, using G-by-G decoding can successfully decode $(1-\delta)L'$ generations (or equivalently $(1-\delta)(M+S)$ intermediate packets) with M received packets, where $\delta = \frac{\theta}{1+\theta}$.

According to the analysis in Section 6.4.2, the above assumptions ensure that $\mathbf{A}_{\text{eff}}$ can be pivoted successfully with a total of $(\alpha+\theta)M$ columns being inactivated. Now we can determine the expected proportion of inactivated columns, α, using the asymptotic analysis of the G-by-G RAC decoding from Chapter 5 (Section 5.5). In order to recover $(1-\delta)L'$ generations using the G-by-G decoder when no inactivation is introduced, the following inequality has to be satisfied for all $x \in [\delta, 1]$:

$$\sum_{u=0}^{G-1} \eta_u \sum_{k=u}^{G-1} \begin{pmatrix} G-1 \\ k \end{pmatrix} (\lambda(x))^k (1-\lambda(x))^{G-1-k} < x, \tag{6.10}$$

where η_u is the probability that one generation has u innovative packets received. The definition of $\lambda(x)$ is defined in (5.12).

Now if we inactivate αM packets in L' generations, approximately $\alpha M/L'$ per generation, then the G-by-G decoding only needs to "decode" out $G' = G - \alpha M/L'$ packets from each generation (the "decoded" packets will be linear combinations of the inactivated packets of the generation). We therefore need to modify (6.10) to

$$\sum_{u=0}^{G'-1} \eta_u \sum_{k=u}^{G'-1} \begin{pmatrix} G'-1 \\ k \end{pmatrix} (\lambda(x))^k (1-\lambda(x))^{G'-1-k} < x, \tag{6.11}$$

for $x \in [\delta, 1]$.

Similar as in Section 5.5, we model the number of received packets of each generation as independent and identically distributed (IID) Poisson variables with rate parameter $\lambda = M/L'$ when M packets are received[1]. Therefore $\eta_u = \frac{\lambda^u \mathrm{e}^{-\lambda}}{u!}$ in (6.11). Let $f(M, S, G, B, \alpha)$ denote the left-hand side of (6.11). The required value of α is

$$\alpha^* = \inf \{\alpha : f(M, S, G, B, \alpha) < x,\ x \in [\delta, 1]\}. \tag{6.12}$$

We find α^* using exhaustive search up from $\alpha = 0$ with desired precision increment $\Delta\alpha$. In each search step, the above inequality is tested by discretizing x in $[\delta, 1]$.

6.5.4 Parameter Choices of PB-RAC

Throughout we fix $B = 32$ as it is demonstrated in [49] that generations of size 32 yield good decoding speed when performing RLNC in $\mathbb{F}_2$. For a given M, we choose to use the same systematic LDPC precode as standard raptor codes [35] (Section 5.4.2.3), which has a fixed yet efficient structure and is suitable for inactivation pivoting. The value of S therein is the smallest prime number larger than or equal to $\lceil 0.01M \rceil + X$ where X is the smallest integer such that $X(X-1) \geq 2M$.

Given B and S, G should be chosen to satisfy the following criteria: when a total of M packets are received where each packet may belong to any generation with equal probability, no generation receives more than G packets. This is because any extra packet exceeding G of a generation would be redundant. We again model the numbers of received packets of each generation as IID Poisson random variables with rate parameter $\lambda = M/L'$. Let Y denote the number of received packets of a

[1]Note that here we strictly require that M packets are received while the analysis in Section 5.5 assumes $(1+\varepsilon)M$ received packets.

generation. We propose to find the smallest G (in favor of sparseness) satisfying the criteria, which can be expressed as:

$$G_{\min} = \inf \left\{ G \in \mathbb{Z} : \Pr\{Y > G\} < \frac{1}{L'} \right\}, \tag{6.13}$$

where the inequality ensures that the expected number of generations that violate the criteria is less than one. $G_{\min}$ can be found by integer search starting from $G = B$.

6.6 Performance Evaluation

We now evaluate the performance of our design. Throughout we use packets each containing $K = 1600$ bytes consisting of 1600 1-byte message symbols. Coding coefficients used for precoding and RLNC at the source or intermediate nodes, if not explicitly stated, are from $\mathbb{F}_2$.

We count the total number of operations in the software implementations (written in `C`), denoted as N_{ops}. We use the *average number of operations per packet symbol* (or operations per symbol for brevity), $\frac{N_{\text{ops}}}{MK}$, to compare decoding costs. As a reference point, GE for decoding RLNC across M source packets requires about M operations per symbol. Each curve of the following simulation is averaged over 1000 trials.

6.6.1 Point-to-Point Performance

As a baseline, we first evaluate the design in the point-to-point scenario, i.e., the decoders are directly applied on GNC-coded packets and therefore only code and decoder-induced overheads may be incurred.

In Figure 6.3, we compare the proposed OA decoder with the G-by-G and the naive decoders. We use non-precoded RAC for comparison. We show the overheads

and decoding costs of RAC for different generation sizes. It is seen that the OA decoder has the same overhead as the naive decoder. The inefficiency of G-by-G decoding in terms of overhead is clearly demonstrated. The overheads of the naive and OA decoders only come from code overhead while the G-by-G decoder incurs significant decoder-induced overhead. The decoding cost of G-by-G decoder is the lowest and it increases only slightly with generation size. Since the OA decoder exploits the sparseness of codes, the OA decoding cost is much lower than that of the naive decoder and is close to that of the G-by-G decoder.

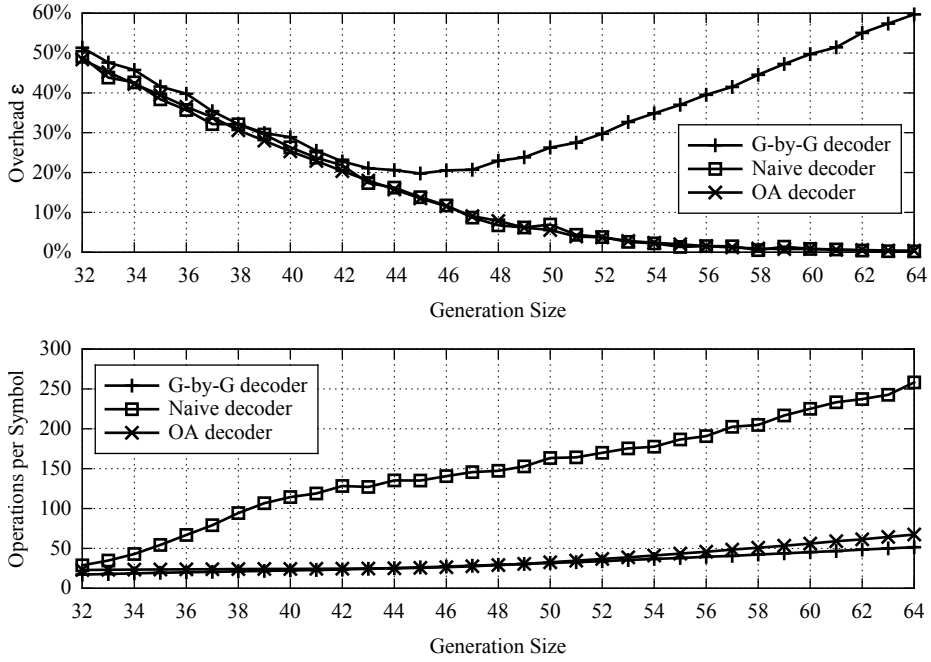

Figure 6.3: Comparison of G-by-G, naive and OA decoders; $M = 1024$, $B = 32$ and $L = 32$.

In Figure 6.4, we show the numbers of inactivated columns in OA decoding of the RAC with $G = 64$ as a function of reception overhead, in which OA decoder postpones its start of joint decoding. As explained in Section 6.4.2, the number

of inactivated columns gradually approaches zero as the allowed overhead increases. From Figure 6.3, the overhead of G-by-G decoding of $G = 64$ RAC is 60%, which matches Figure 6.4, where the number of inactivated columns of OA decoding is 0 at $\varepsilon = 60\%$, meaning that the performances of OA and G-by-G decoder are identical at $\varepsilon = 60\%$.

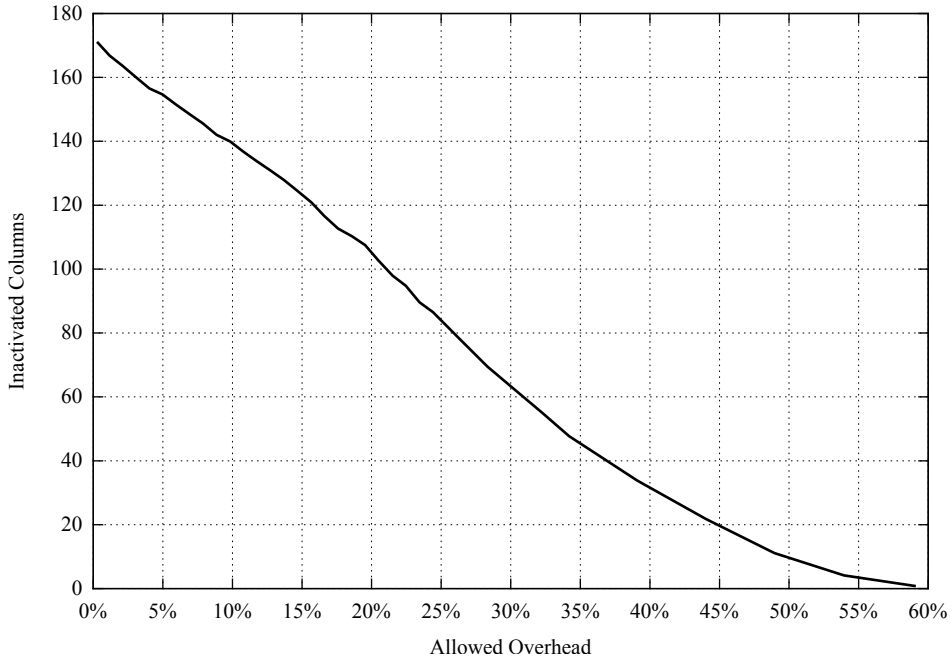

Figure 6.4: Performance of OA decoding when allowing for more reception overhead; RAC code with $M = 1024$, $B = 32$, $G = 64$, $L = 32$ is used.

Next we compare the proposed PB-RAC with the original RAC of [32], *head-to-toe* (H2T) codes of [17], [18], *windowed* codes of [21], and *banded* codes of [11]. The H2T has the same number of $L = M/B$ generations as RAC. Generations of H2T are overlapped consecutively rather than randomly. The windowed code can be viewed as a H2T code with M generations. The banded code is similar to the windowed code except that it does not allow for wrap around and therefore has only $M - G + 1$

generations. The decoding matrices of banded codes are strictly banded while those of H2T and windowed codes are close to banded (except for the last few rows). In Figure 6.5, we show overheads and decoding costs of the four codes for different generation sizes. Note that the probability of sending coded packets from each generation of the banded code is not uniform as in [11] while that of the other three codes are uniform. The decoding of H2T, windowed codes and banded codes utilize straightforward GE whereas the decoding of RAC and PB-RAC use the proposed OA decoder.

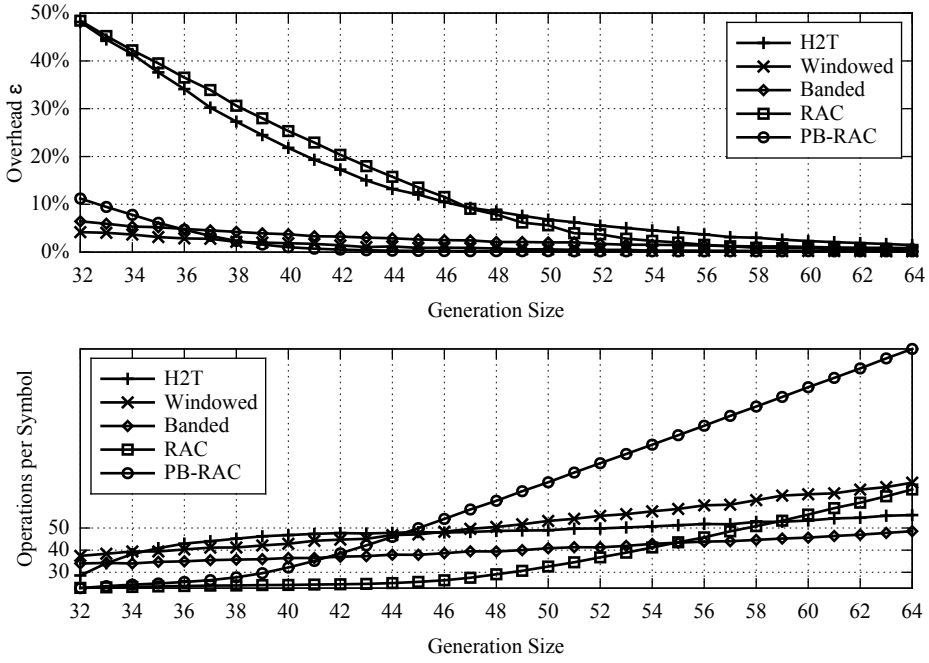

Figure 6.5: Comparison of H2T from [17], [18], windowed codes from [21], banded codes from [11], RAC, and PB-RAC; $M = 1024$ and $B = 32$, $S = 59$ for PB-RAC.

Figure 6.5 shows that, as generation size grows, overheads of all codes decrease. However, PB-RAC has much lower overhead than that of H2T and RAC, achieving less than 1% at $G = 41$ while RAC achieves this at $G = 58$ and H2T at $G > 64$. It is important to note that overheads of H2T decrease more slowly than that of

RAC as G increases when approaching zero, which is a major justification of RAC for our code design. The decoding costs of achieving 1% overheads for PB-RAC, RAC and H2T are 35, 50, and 60 operations per symbol, respectively. A significant improvement in both overhead and decoding cost is obtained using PB-RAC. It is interesting to note that windowed and banded codes have very similar code overheads to that of PB-RAC even though no precoding is used. The windowed and banded code achieve 1% overhead at $G = 45$ and $G = 61$, respectively, and both require about 47 operations per symbol to decode. However, subsequent results show that the two codes might not be suitable for use in networks where intermediate nodes need to re-encode packets on-the-fly during the transmission.

We have noted that OA decoding costs of RAC and PB-RAC increase quickly with generation size, while those of H2T, windowed and banded codes are much flatter. This is because the decoding matrices of RAC and PB-RAC have less structure, which require more decoding operations to decode even though OA decoding has been used to exploit sparseness. Nevertheless, since the overheads of H2T, windowed and banded codes decrease more slowly as the generation size increases, to achieve the same level of overhead PB-RAC with OA decoding requires fewer operations because the required code can be much sparser.

In Table 6.2 we show decoding performances of four PB-RAC codes with different S that each results in about 1% overhead, where S is the number of parity-check packets added by the precode and $S = 0$ corresponds to the non-precoded case (i.e., original RAC). The trade-off between S and G is clear and a balanced choice of S and G yields the least number of inactivated columns and, as a consequence, the least cost.

S	G	ε	$\frac{N_{\text{ops}}}{MK}$	M_I
0	58	0.92%	50	127
59	**41**	**0.74%**	**35**	**80**
101	39	0.70%	38	100
149	37	0.95%	39	112

Table 6.2: PB-RAC achieving $\varepsilon = 1\%$ with different S; $M = 1024$, $B = 32$.

We show performances of PB-RAC for various M in Figure 6.6, where code parameters are determined according to Section 6.5.4. We compare with two other schemes. One applies G-by-G decoding on another set of PB-RAC codes with the same B and S but G are chosen such that the minimum overhead is attained. The other one is P256-RAC whose coding parameters are the same as the designed PB-RAC but precoding and RLNC are all performed in $\mathbb{F}_{256}$. We also perform OA decoding on P256-RAC.

We see that both PB-RAC and P256-RAC achieve almost zero overhead whereas PB-RAC with G-by-G decoding has more than 10% overhead. The overhead of using $\mathbb{F}_2$ is slightly higher than that of using $\mathbb{F}_{256}$ but the difference is very little. We plot the decoding speeds of the three schemes of our implementations. The comparison (compiled with `-O3` option using `gcc`) is conducted on a 2.66GHz quad-core Intel Core 2 CPU with 4 GB RAM under the same implementation framework. The implementation is not optimized and the speeds are for comparison purposes only. The G-by-G decoding has the highest decoding speed and the speed does not decrease much as M grows, indicating that it has constant decoding cost. The OA decoding speed is much lower than G-by-G decoding and decreases faster as M grows, suggesting that OA decoding may not be practically feasible for larger M. The OA decoding speed of PB-RAC is considerably higher than that of P256-RAC. The OA

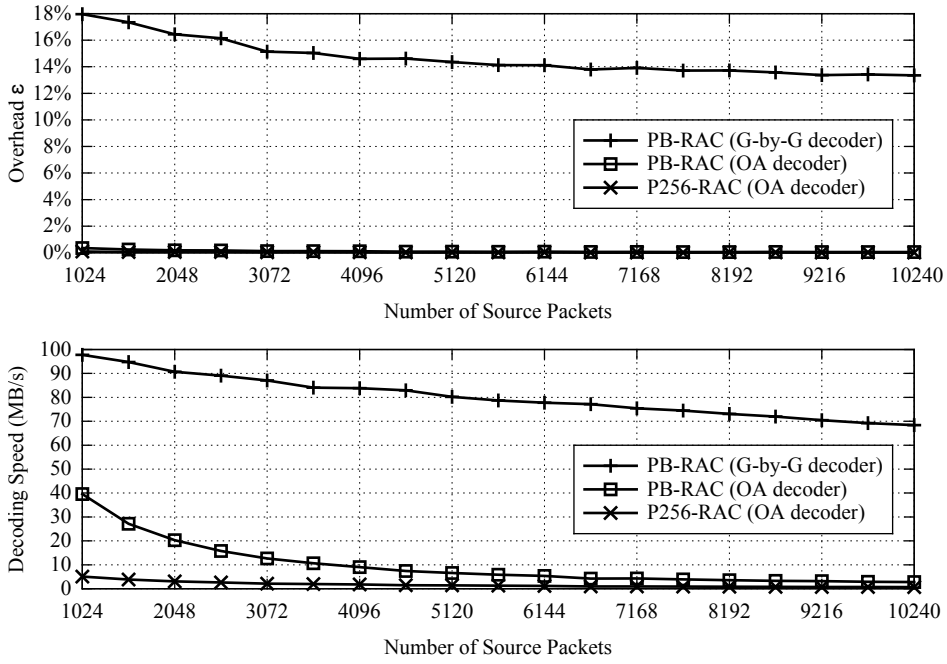

Figure 6.6: Performances of PB-RAC codes for various numbers of source packets, M.

decoding speed of PB-RAC at $M = 10240$ is 3MB/s.

In Figure 6.7 we show the fraction of inactivated columns in OA decoding, $\frac{M_I}{M}$. We present the simulation results of PB-RAC, P256-RAC and the analysis results of (6.12). The analysis closely matches the simulation of P256-RAC as shown. The benefit of using $\mathbb{F}_2$ is more clear: inactivated columns are reduced by half.

6.6.2 Network Performance

In this subsection we evaluate performance of PB-RAC in the well-known lossy butterfly network [2]. Erasure rates of links are set equally to $p_e = 0.1$. The max-flow capacity of the network is 1.8 packets per network use, where each *network use* corresponds to the source and intermediate nodes each send a packet.

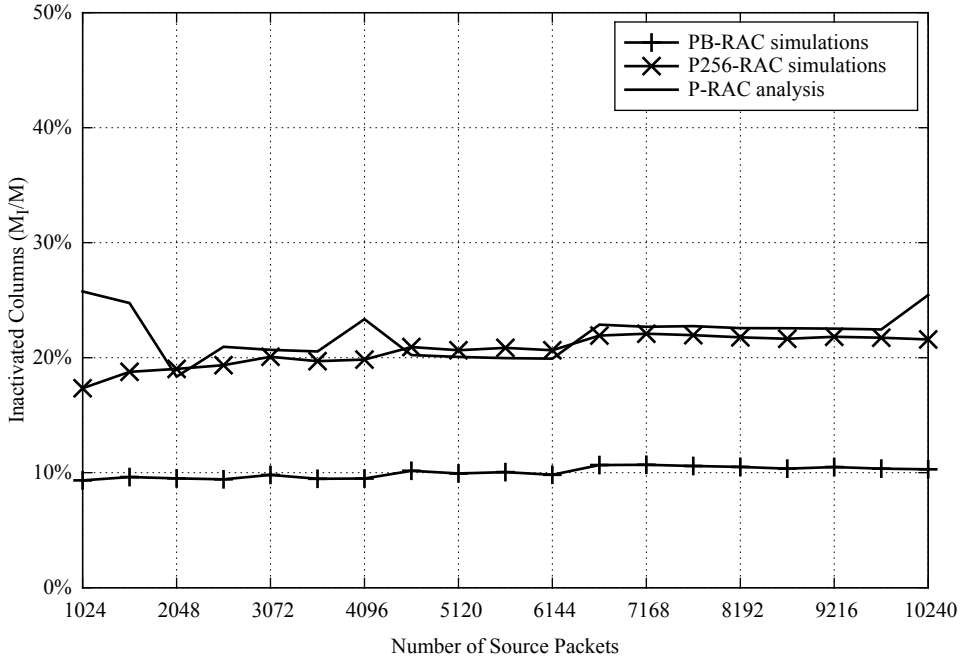

Figure 6.7: Inactivated columns in OA decoding of PB-RAC.

Figure 6.8 shows the overheads and decoding costs when using H2T, windowd code, banded code and PB-RAC. We set $G = 2\sqrt{M}$ for H2T, windowed and banded codes, as it is shown empirically in [60] that $2\sqrt{M}$ is the required width for a banded random matrix such that it has similar probabilistic rank properties to behave as a dense random matrix. For example, $G = 64$ for $M = 1024$. Parameters of PB-RAC are the same as in Figure 6.6. When it needs to send a packet, each intermediate node performs random scheduling [44] (i.e., uniformly randomly choose a generation for H2T, windowed and PB-RAC and non-uniformly randomly for banded according to [11]) and re-encodes packets using random coefficients from $\mathbb{F}_2$. Straightforward GE is used for decoding H2T, windowed and banded codes. PB-RAC uses OA decoding.

Figure 6.8 shows that PB-RAC and H2T have the same overhead even though PB-RAC is much sparser. For example, $G = 43$ is used for $M = 1024$ in PB-RAC and all

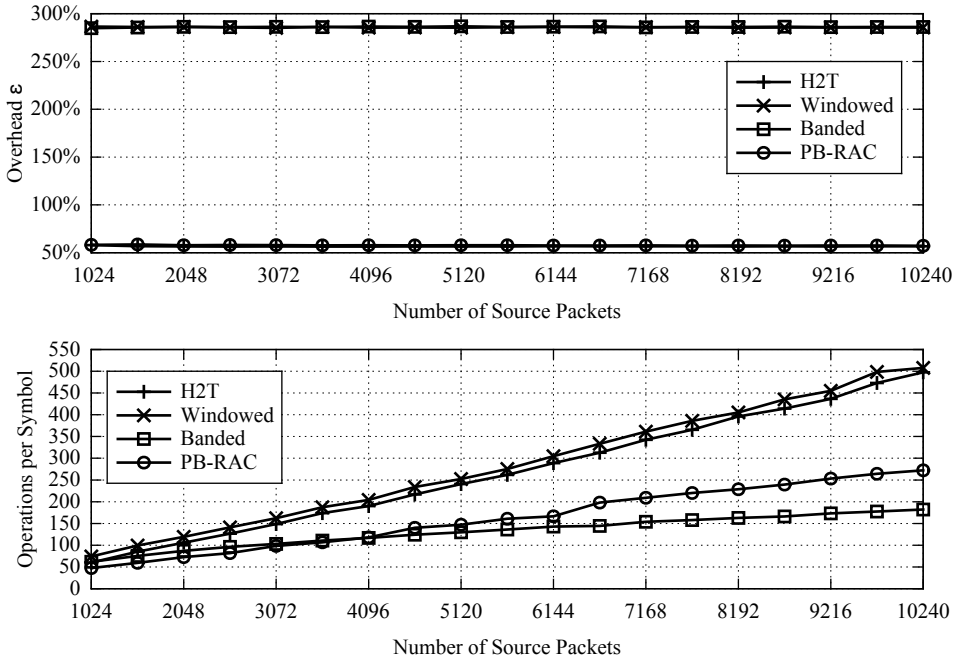

Figure 6.8: Comparison of H2T ($B = 32$), windowed code, banded code and PB-RAC over a lossy butterfly network of [2] where all relay nodes perform random scheduling and re-encoding in $\mathbb{F}_2$; erasure rates of links are equally $p_e = 0.1$.

G's of PB-RAC in the figure are actually smaller than 50. The overheads of windowed and banded codes, however, are poor. The reason is due to limited re-encoding opportunity: the two codes each have almost M generations and therefore the number of buffered packets of each generation at the relay is much fewer than that of H2T or PB-RAC, each of which have $L \ll M$ generations. We note that some measures can be taken at intermediate nodes as discussed in [11] and [21] to alleviate the issue. For example, one can search for buffered packets from different generations that can be re-encoded so that the re-encoded packet is confined to a desired window/band; or one can perform (partial) decoding at intermediate nodes to *passively* obtain a re-encoded packet. The measures, however, add significant complication to intermediate

nodes and may not be suitable for scenarios where intermediate nodes only serve as relays. Since PB-RAC is much sparser, its decoding cost is lower than those of H2T and windowed codes. The decoding cost of banded code is the lowest due to its strict banded structure.

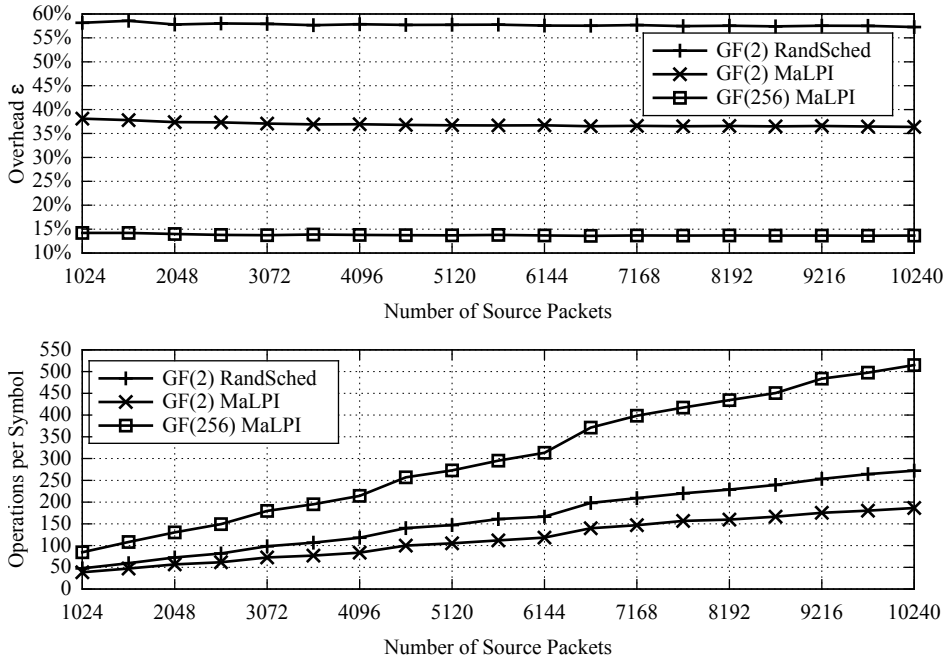

Figure 6.9: Performance of PB-RAC with different scheduling strategies at relay nodes in butterfly network; $p_e = 0.1$ for all links.

It is seen from Figure 6.8 that although the codes may have close-to zero code overhead and zero decoder-induced overhead (see Figures 6.5 and 6.6), the reception overhead over a network is nonzero because of scheduling and re-encoding at intermediate nodes. We remark that the performance may be improved by optimizing scheduling. In Figure 6.9, we show the performances of the same PB-RAC codes over the network where the MaLPI scheduling proposed in Chapter 5 is used at intermediate nodes. Unlike random scheduling, MaLPI chooses a generation that is *least*

scheduled with respect to the numbers of received packets of each generation. The overhead is reduced from 58% to 38%. If we use $\mathbb{F}_{256}$ for re-encoding at intermediate nodes, the overheads can be further reduced to 14%, which correspond to the rate of about 1.57 packets per network use (i.e., close to 90% of the max-flow capacity). Note that re-encoding with $\mathbb{F}_{256}$ will change all coding coefficients to $\mathbb{F}_{256}$, so the decoding costs will increase accordingly, as shown in the figure.

6.7 Summary

In this chapter we proposed an efficient overhead-optimized decoder that has zero decoder-induced overhead for generation-based network codes. We showed that the decoding cost can be maintained low for moderate number of source packets by exploiting the structure of GNC codes through local processing and multiple rounds of pivoting of the decoding matrix. Based on the decoder, we have designed a new sparse network code that combines binary precoding, using random overlapping generations and binary RLNC. Compared to previous works, the code is able to achieve close-to zero code overhead at decoding speed of several megabytes per second using the proposed decoder. Our overall design has potential in applications where low overhead is critical and the number of source packets is moderate.

Chapter 7

Summary and Future Work

In this chapter, conclusions are summarized based on the results obtained in previous chapters. Several topics for future research are also suggested.

7.1 Summary

Chapter 3 studies a systematic network coding method for transmission over a two-hop lossy link. In Section 3.3 we proved that transmitting uncoded packets first, followed by coded packets requires much less decoding cost and at the same time achieves a higher end-to-end rate than RLNC. A Markov chain model is used in Section 3.3.1 to analytically determine the completion time of the proposed scheme and the model also includes the random linear network coding scheme as a special case. In Section 3.5, simulation results show that the benefit of systematic network coding is significant when the number of source packets is small. For example at $M = 50$, as shown in Figure 3.9, the rate of the proposed scheme is increased by 4% from 0.74 of RLNC to 0.77 while its decoding cost is only 1/5 of that of the RLNC method.

Chapter 4 extends the scheme of Chapter 3 to the network that consists of two

parallel two-hop lossy links. In Section 4.3 we proved that the benefits of systematic network coding persist in this network. We also proved that if the number of uncoded packets that are received by the destination is maximized, then the rate and decoding cost benefits would both be maximized. We then formulated and solved an uncoded packet allocation problem to obtain such maximization. Simulation results in Section 4.4 showed that every uncoded packet received by the destination would strictly improve the performance of RLNC. Taking $M = 100$ as an example, Figure 4.2 shows that the proposed scheme with the optimized uncoded packet allocation would improve the rate from 2.5 to 2.66 (a 6.4% improvement) while reducing decoding cost from 56 to 8 operations per packet symbol (a 93% reduction).

In Chapter 5, we study the generation-based network coding scheme, a sparse network coding scheme that can be used in networks where the topology is more complex or changing during the transmission. In this chapter, the generation-by-generation (G-by-G) decoding is considered. We first modeled generation-based network codes as graph codes and then analyze its G-by-G decoding on the graph. We designed two codes based on the analysis results. First, we determined a key parameter (i.e., the optimal amount of overlap among generations) for the existing random annex codes and showed that the estimation is much more accurate than before. As shown in Figure 5.5, the achieved overhead of our design is only 3% higher than the minimum achievable overheads, while that of previous design is more than 10% higher. Second, we proposed a new code based on unequal-size generations and provided a method to design the generation-size distribution for the code. We established that using unequal-size generations for the G-by-G decoding achieves both lower overhead and decoding cost than its equal-size counterpart, if the generation-size distribution is

properly designed. In this chapter, we also proposed a new scheduling strategy, maximum local potential innovativeness scheduling, to be used at intermediate nodes. We showed that the new scheduling strategy outperforms the previous random scheduling in terms of overhead.

In Chapter 6, we continue the design of generation-based network codes. Unlike Chapter 5, Chapter 6 considers overhead-optimized decoding. This decoding results in the minimum overhead and therefore is useful in applications where low-overhead is critical, e.g. where users may be charged according to the amount of data received. However, overhead-optimized decoding may require higher decoding cost. In Chapter 6, we first proposed a low-complexity decoder that combines local processing and pivoting algorithms when solving linear system of equations. This decoder has low decoding cost for generation-based network codes that may have general overlap structure. The decoder is also showed to be able to flexibly control the trade-off between overhead and decoding cost. We then designed a generation-based network code that has very low code overhead and decoding cost with the proposed decoder. Although the decoding speed using the proposed decoder is lower than that of the G-by-G decoder (which is as expected), the achieved speed is acceptable for targeted applications. For example, 3 MB/s decoding speed can be achieved for the case where $M = 10240$ packets each consisting of 1600 bytes are coded together. This speed should be sufficient to support smooth playback in streaming media applications. Most importantly, the overall scheme has close-to zero overhead when evaluated in the point-to-point scenario whereas that of using the G-by-G decoder is more than 10%.

7.2 Future Work

There are several topics that are suggested for future research:

- In Chapters 3 and 4, when the relay node needs to re-encode a packet, buffered packets are combined using RLNC. Therefore, packets sent from the relay are either uncoded or fully coded. It is of interest to examine the cases in-between, i.e., whether sparsely re-encoding packets would further lower the decoding cost while retaining the rate benefits.

- In Chapter 4, solving the uncoded packet allocation problem requires precise information about link erasure rates. In practice, however, this kind of information may not be easily obtained. It is therefore of interest to study the allocation problem for the case when link erasure rates are unknown or only statistically known.

- We have seen that the benefits of systematic network coding method in Chapter 3 and 4 come mostly from the uncoded packets that are received by the destination node. It is of interest to investigate how the method scales to line networks with more hops and under different traffic models, where the number of uncoded packets that can be received by the destination node may change significantly.

- Throughout the thesis, we have considered that no acknowledgements/feedback are available between nodes during the transmission. It would be of interest to study the designs for when (partial) feedback between nodes are available. In particular, it is of interest to investigate how much improvement feedback may

bring in when feedback is available for scheduling algorithms at intermediate nodes in Chapters 5 and 6.

- It is of great interest to systematically evaluate the code designs of Chapters 5 and 6 in real-world network environments.

Bibliography

[1] 3GPP TS 26.346: Multimedia Broadcast/Multicast Services (MBMS); Protocols and Codecs (Release 6). 3GPP Technical Specification 26.346, 2005.

[2] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[4] D. Burshtein and G. Miller. An efficient maximum-likelihood decoding of LDPC codes over the binary erasure channel. *IEEE Transactions on Information Theory*, 50(11):2837–2844, 2004.

[5] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *Proc. of the ACM SIGCOMM '98*, pages 56–67, Sept. 1998.

[6] J. Castura and Y. Mao. Rateless coding for wireless relay channels. *IEEE Transactions on Wireless Communications*, 6(5):1638–1642, 2007.

[7] M.-L. Champel, K. Huguenin, A.-M. Kermarrec, and N. Le Scouarnec. LT network codes. In *Proc. IEEE 30th International Conference on Distributed Computing Systems (ICDCS)*, pages 536–546, June 2010.

[8] P. A. Chou, Y. Wu, and K. Jain. Practical network coding. In *Proc. 41st Allerton Conference on Communication, Control, and Computing*, pages 40–49, Oct. 2003.

[9] J. Cloud, F. du Pin Calmon, W. Zeng, G. Pau, L.M. Zeger, and M. Medard. Multi-path TCP with network coding for mobile devices in heterogeneous networks. In *Proc. IEEE 78th Vehicular Technology Conference (VTC Fall)*, pages 1–5, Sept. 2013.

[10] I. S. Duff, A. M. Erisman, and J. K Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, New York, 1986.

[11] A. Fiandrotti, V. Bioglio, M. Grangetto, R. Gaeta, and E. Magli. Band codes for energy-efficient network coding with application to P2P mobile streaming. *IEEE Transactions on Multimedia*, 16(2):521–532, 2014.

[12] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP extensions for multipath operation with multiple addresses, January 2013. IETF, RFC6824.

[13] G. Giacaglia, X. Shi, M. Kim, D. Lucani, and M. Médard. Systematic network coding with the aid of a full-duplex relay. *CoRR*, abs/1204.0034, 2012.

[14] C. Gkantsidis and P. R. Rodriguez. Network coding for large scale content distribution. In *Proc. IEEE 24th Annual Joint Conf. of the IEEE Computer and Communications Societies (Infocom)*, volume 4, pages 2235–2245, 2005.

[15] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 3rd ed. edition, 1996.

[16] R. Gummadi and R. S. Sreenivas. Relaying a fountain code across multiple nodes. In *Proc. IEEE Information Theory Workshop (ITW)*, pages 149–153, May 2008.

[17] A. Heidarzadeh and A. H. Banihashemi. Overlapped chunked network coding. In *Proc. IEEE Information Theory Workshop (ITW)*, pages 1–5, 2010.

[18] A. Heidarzadeh and A. H. Banihashemi. Network codes with overlapping chunks over line networks: A case for linear-time codes. *CoRR*, abs/1105.5736, 2011.

[19] A. Heidarzadeh and A. H. Banihashemi. How much can knowledge of delay model help chunked coding over networks with perfect feedback? In *IEEE International Symposium on Information Theory (ISIT)*, pages 456–460, June 2014.

[20] 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 Workshops*, pages 1–6, 2009.

[21] J. Heide, M. V. Pedersen, F. H. P. Fitzek, and M. Medard. A perpetual code for network coding. In *Proc. IEEE Vehicular Technology Conference (VTC) - Wireless Networks and Security Symposium*, 2014.

[22] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.

[23] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen. Polynomial time algorithms for multicast network code construction. *IEEE Transactions on Information Theory*, 51(6):1973–1982, 2005.

[24] G. Joshi and E. Soljanin. Round-robin overlapping generations coding for fast content download. In *Prof. IEEE International Symposium on Information Theory (ISIT)*, pages 2740–2744, 2013.

[25] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. D. Van Nostrand Company, Inc., Princeton, New Jersey, 1960.

[26] R. Koetter and M. Medard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11(5):782–795, 2003.

[27] Jr. L. K. Ford and D. K. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, New Jersey, 1962.

[28] G. Landsburg. Uber eine anzahlbestimmung und eine damit zusammenhangende. In *Reihe, J. Reine Angew. Math.*, volume 111, pages 87–88, 1893.

[29] S.-Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, 2003.

[30] Y. Li, S. D. Blostein, and W.-Y. Chan. Large file distribution using efficient generation-based network coding. In *Proc. IEEE Globecom International Workshop on Cloud Computing Systems, Networks, and Applications*, 2013.

[31] Y. Li, W.-Y. Chan, and S. D. Blostein. Network coding with unequal size overlapping generations. In *Proc. International Symposium on Network Coding (NetCod)*, pages 161–166, 2012.

[32] Y. Li, E. Soljanin, and P. Spasojevic. Effects of the generation size and overlap on throughput and complexity in randomized linear network coding. *IEEE Transactions on Information Theory*, 57(2):1111–1123, 2011.

[33] M. Luby. LT codes. In *Proc. 43rd Annual IEEE Symp. Foundations of Computer Science*, pages 271–280, 2002.

[34] M. Luby, M. Mitzenmacher, and A. Shokrollahi. Analysis of random processes via and-or tree evaluation. In *Proc. 9th Annu. ACM-SIAM Symp. Discrete Algorithms*, pages 364–373, January 1998.

[35] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer. Raptor forward error correction scheme for object delivery, September 2007. Internet Engineering Task Force, RFC5053.

[36] M. G. Luby, M. Mitzenmacher, A. Shokrollahi, and D. A. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569–584, 2001.

[37] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Improved low-density parity-check codes using irregular graphs. *IEEE Transactions on Information Theory*, 47(2):585–598, 2001.

[38] D. E. Lucani, M. Medard, and M. Stojanovic. Systematic network coding for time-division duplexing. In *Prof. IEEE International Symposium on Information Theory (ISIT)*, pages 2403–2407, 2010.

[39] D. S. Lun, M. Médard, and M. Effros. On coding for reliable communication over packet networks. In *Proc. 42nd Annual Allerton Conference on Communication, Control, and Computing*, 2004.

[40] D. S. Lun, P. Pakzad, C. Fragouli, M. Medard, and R. Koetter. An analysis of finite-memory random linear coding on packet streams. In *Prof. IEEE 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 1–6, 2006.

[41] G. Ma, Y. Yu, M. Lin, and Y. Xuan. A content distribution system based on sparse linear network coding. In *Proc. NetCod'07*, 2007.

[42] K. Mahdaviani, M. Ardakani, H. Bagheri, and C. Tellambura. Gamma codes: A low-overhead linear-complexity network coding solution. In *Prof. International Symposium on Network Coding (NetCod)*, pages 125–130, June 2012.

[43] P. Maymounkov. Online codes. Technical report, New York University, 2002.

[44] P. Maymounkov, N. J. A. Harvey, and D. S. Lun. Methods for efficient network coding. In *Proc. of Allerton Conference on Communication, Control, and Computing*, pages 482–491, Sep. 2006.

[45] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, 2005.

[46] A. M. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. In *Proc. Of the EUROCRYPT 84 Workshop on Advances in Cryptology:*

Theory and Application of Cryptographic Techniques, pages 224–314, New York, 1985. Springer-Verlag.

[47] P. Pakzad, C. Fragouli, and A. Shokrollahi. Coding schemes for line networks. In *Proc. IEEE Int. Symp. Information Theory*, pages 1853–1857, 2005.

[48] E. Paolini, G. Liva, B. Matuz, and M. Chiani. Maximum likelihood erasure decoding of LDPC codes: Pivoting algorithms and code design. *IEEE Transactions on Communications*, 60(11):3209–3220, 2012.

[49] A. Paramanathan, M. V. Pedersen, D. E. Lucani, F.H.P. Fitzek, and M. Katz. Lean and mean: network coding for commercial devices. *IEEE Wireless Communications Magazine*, 20(5):54–61, 2013.

[50] H. Pishro-Nik and F. Fekri. On decoding of low-density parity-check codes over the binary erasure channel. *IEEE Transactions on Information Theory*, 50(3):439–454, 2004.

[51] C. Pomerance and J. W. Smith. Reduction of huge, sparse matrices over finite fields via created catastrophes. *Experiment. Math*, 1:89–94, 1992.

[52] T. J. Richardson and R. L. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2):599–618, 2001.

[53] D. Rose and R. Tarjan. Algorithmic aspects of vertex elimination on directed graphs. *SIAM Journal on Applied Mathematics*, 34(1):176–197, 1978.

[54] X. Shi, M. Medard, and D. E. Lucani. Whether and where to code in the wireless packet erasure relay channel. *IEEE Journal on Selected Areas in Communications*, 31(8):1379–1389, 2013.

[55] H. Shojania and B. Li. Random network coding on the iPhone: Fact or fiction? In *Proc. 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 37–42, 2009.

[56] A. Shokrollahi. Raptor codes. *IEEE Transactions on Information Theory*, 52(6):2551–2567, 2006.

[57] A. Shokrollahi and M. Luby. Raptor codes. *Foundations and Trends in Communications and Information Theory*, 6(3-4):213–322, 2009.

[58] B. Shrader and N. M. Jones. Systematic wireless network coding. In *Proc. IEEE Military Communications Conf. MILCOM*, pages 1–7, 2009.

[59] D. Silva, W. Zeng, and F. R. Kschischang. Sparse network coding with overlapping classes. In *Proc. Workshop Network Coding, Theory, and Applications (NetCod)*, pages 74–79, 2009.

[60] C. Studholme and I. F. Blake. Random matrices and codes for the erasure channel. *Algorithmica*, 56(4):605–620, 2010.

[61] J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros. Network coding meets TCP. In *Proc. IEEE INFOCOM 2009*, pages 280–288, 2009.

[62] B. Tang, S. Yang, Y. Yin, B. Ye, and S. Lu. Expander graph based overlapped chunked codes. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 2451–2455, July 2012.

[63] J.-P. Thibault, W.-Y. Chan, and S. Yousefi. A family of concatenated network codes for improved performance with generations. *Journal of Communication and Networks, special issue on network coding*, 10:384–395, 2008.

[64] M. Wang and B. Li. How practical is network coding? In *Proc. 14th IEEE International Workshop on Quality of Service (IWQoS).*, pages 274–278, June 2006.

[65] M. Wang and B. Li. R2: Random push with random network coding in live peer-to-peer streaming. *IEEE Journal on Selected Areas in Communications*, 25(9):1655–1666, 2007.

[66] Jinbiao Xu, Jin Zhao, Xin Wang, and Xiangyang Xue. Swifter: Chunked network coding for peer-to-peer content distribution. In *Proc. IEEE Int. Conf. Communications (ICC)*, pages 5603–5608, 2008.

[67] R. W. Yeung. *Information Theory and Network Coding.* Springer, 2008.

[68] Z. Zlatev. On some pivotal strategies in gaussian elimination by sparse technique. *SIAM Journal on Numerical Analysis*, 17(1):18–30, 1980.

Appendix A

Transition Probabilities of the Markov Chains Modeling S2HNC

For the period of time when S is sending uncoded packets, we have the following transition probabilities for different combinations of (k, r) in Markov Chain C1. We have

$$p_{(0,0)\to(0,0)} = \epsilon_1, \tag{A.1}$$

$$p_{(0,0)\to(0,1)} = 0, \tag{A.2}$$

$$p_{(0,0)\to(1,0)} = (1-\epsilon_1)\epsilon_2, \tag{A.3}$$

$$p_{(0,0)\to(1,1)} = (1-\epsilon_1)(1-\epsilon_2), \tag{A.4}$$

for $k = r = 0$.

For $k \geq 1$ and $r < k < M$,

$$p_{(k,r)\to(k,r)} = \epsilon_1 \left(\epsilon_2 + (1-\epsilon_2)\frac{q^r}{q^k} \right), \tag{A.5}$$

$$p_{(k,r)\to(k,r+1)} = \epsilon_1(1-\epsilon_2)\left(1 - \frac{q^r}{q^k}\right), \tag{A.6}$$

$$p_{(k,r)\to(k+1,r)} = (1-\epsilon_1)\epsilon_2, \tag{A.7}$$

$$p_{(k,r)\to(k+1,r+1)} = (1-\epsilon_1)(1-\epsilon_2). \tag{A.8}$$

For $k \geq 1$ and $r = k < M$,

$$p_{(k,r)\to(k,r)} = \epsilon_1, \tag{A.9}$$

$$p_{(k,r)\to(k,r+1)} = 0, \tag{A.10}$$

$$p_{(k,r)\to(k+1,r)} = (1-\epsilon_1)\epsilon_2, \tag{A.11}$$

$$p_{(k,r)\to(k+1,r+1)} = (1-\epsilon_1)(1-\epsilon_2). \tag{A.12}$$

and for $r < k = M$,

$$p_{(k,r)\to(k,r)} = \epsilon_2 + (1-\epsilon_2)\frac{q^r}{q^k}, \tag{A.13}$$

$$p_{(k,r)\to(k,r+1)} = (1-\epsilon_2)(1-\frac{q^r}{q^k}), \tag{A.14}$$

$$p_{(k,r)\to(k+1,r)} = 0, \tag{A.15}$$

$$p_{(k,r)\to(k+1,r+1)} = 0. \tag{A.16}$$

For the period of time when S is sending coded packets, i.e., for Markov chain C2,

we have

$$p_{(0,0)\to(0,0)} = \epsilon_1 + (1-\epsilon_1)\frac{1}{q^M}, \tag{A.17}$$

$$p_{(0,0)\to(0,1)} = 0, \tag{A.18}$$

$$p_{(0,0)\to(1,0)} = (1-\epsilon_1)\left(1-\frac{1}{q^M}\right)\left(\epsilon_2 + (1-\epsilon_2)\frac{1}{q}\right) \tag{A.19}$$

$$p_{(0,0)\to(1,1)} = (1-\epsilon_1)\left(1-\frac{1}{q^M}\right)(1-\epsilon_2)\left(1-\frac{1}{q}\right), \tag{A.20}$$

for $k = r = 0$.

For $k \geq 1$ and $r < k < M$,

$$p_{(k,r)\to(k,r)} = \left(\epsilon_1 + (1-\epsilon_1)\frac{q^k}{q^M}\right)\left(\epsilon_2 + (1-\epsilon_2)\frac{q^r}{q^k}\right), \tag{A.21}$$

$$p_{(k,r)\to(k,r+1)} = \left(\epsilon_1 + (1-\epsilon_1)\frac{q^k}{q^M}\right)(1-\epsilon_2)\left(1-\frac{q^r}{q^k}\right), \tag{A.22}$$

$$p_{(k,r)\to(k+1,r)} = (1-\epsilon_1)\left(1-\frac{q^k}{q^M}\right)\left(\epsilon_2 + (1-\epsilon_2)\frac{q^r}{q^{k+1}}\right), \tag{A.23}$$

$$p_{(k,r)\to(k+1,r+1)} = (1-\epsilon_1)\left(1-\frac{q^k}{q^M}\right)(1-\epsilon_2)\left(1-\frac{q^r}{q^{k+1}}\right). \tag{A.24}$$

For $k \geq 1$ and $r = k < M$,

$$p_{(k,r)\to(k,r)} = \epsilon_1 + (1-\epsilon_1)\frac{q^k}{q^M}, \tag{A.25}$$

$$p_{(k,r)\to(k,r+1)} = 0, \tag{A.26}$$

$$p_{(k,r)\to(k+1,r)} = (1-\epsilon_1)\left(1-\frac{q^k}{q^M}\right)\left(\epsilon_2 + (1-\epsilon_2)\frac{q^r}{q^{k+1}}\right), \tag{A.27}$$

$$p_{(k,r)\to(k+1,r+1)} = (1-\epsilon_1)\left(1-\frac{q^k}{q^M}\right)(1-\epsilon_2)\left(1-\frac{q^r}{q^{k+1}}\right). \tag{A.28}$$

and for $r < k = M$,

$$p_{(k,r)\to(k,r)} = \epsilon_2 + (1-\epsilon_2)\frac{q^r}{q^k}, \tag{A.29}$$

$$p_{(k,r)\to(k,r+1)} = (1-\epsilon_2)(1-\frac{q^r}{q^k}), \tag{A.30}$$

$$p_{(k,r)\to(k+1,r)} = 0, \tag{A.31}$$

$$p_{(k,r)\to(k+1,r+1)} = 0. \tag{A.32}$$