Design and Analysis of Random Linear Network Coding Schemes: Dense Codes, Chunked Codes and Overlapped Chunked Codes

by

Anoosheh Heidarzadeh, M.A.Sc.

A dissertation submitted to the
Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering (OCIECE)
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada, K1S 5B6

December 2012

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-94221-5

Our file Notre référence
ISBN: 978-0-494-94221-5

Abstract

In this thesis, the problem of designing and analyzing network codes that are both communicationally and computationally efficient over unicast (one-source one-sink) networks (particularly, line networks) is considered.

At first, we revise the existing analysis of *chunked codes* (CC), which are a low-complexity version of random linear network codes (a.k.a. dense codes), over worst-case (arbitrary deterministic) traffics, and derive tighter bounds on the performance of CC. Next, to improve the speed of convergence of CC (with or without precoding), while maintaining their advantage in reducing the computational complexity, we propose and analyze a new CC scheme with overlapping chunks, called *overlapped chunked codes* (OCC). We prove that for smaller chunks, which are advantageous due to lower computational complexity, OCC with larger overlaps provide a better tradeoff between the computational complexity and the speed of convergence or the message/packet error rate. We also prove that a linear-time OCC (i.e., with a chunk size constant in the message size) has a superior performance compared to a CC with the same computational complexity.

Thereafter, we analyze the coding delay and the average coding delay of dense codes and CC over some well-known probabilistic traffics (delay-free deterministic regular or Poisson transmissions and Bernoulli losses). Our results are (i) for dense codes, in some cases more general, and in some other cases tighter, than the existing bounds, and provide a more clear picture of the speed of convergence of dense codes to the capacity of line networks; and (ii) the first of their kind for CC (with or without precoding) over networks with such probabilistic traffics.

Finally, the problem of designing and analyzing efficient feedback-based scheduling policies for CC is considered. We propose two new scheduling policies, referred to as *minimum-distance-first* (MDF) and *minimum-current-metric-first* (MCMF). Unlike the existing policies, the MDF and MCMF policies incorporate transmission, loss and delay models of the link in the selection process of the chunk to be transmitted. Our simulations show that the MDF and MCMF policies significantly reduce the expected decoding time compared to the existing policies over line networks.

To Life

Acknowledgments

I am heartily thankful to my supervisor, Professor Amir Banihashemi, for his tremendous time and efforts spent in leading, supporting and encouraging me in the course of my thesis and study. This dissertation would not have been possible without his constant help and invaluable guidance throughout my study. I would also like to sincerely thank my thesis committee members for accepting the invitation to be part of the committee.

Many thanks to my colleagues for their understanding and friendship during the time we spent together.

Last but not least, I would like to express my special thanks to my loving family for their unending inspiration, continuous understanding, support and encouragement in my life.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CC	Chunked Code
OCC	Overlapped Chunked Code
LILE	(packets with) Linearly Independent Local Encoding (vectors)
CCP	Chunked Code with Precoding
RB	Random Banded
ISRB	Irregular Symmetric Random Banded
IARB	Irregular Asymmetric Random Banded
RSRB	Regular Symmetric Random Banded
RARB	Regular Asymmetric Random Banded
LDPC	Low-Density Parity-Check
RBLT	Random Banded Lower-Triangular
MDF	Minimum-Distance-First
MCMF	Minimum-Current-Metric-First
RP	Random Push
LRF	Local-Rarest-First
MER	Message Error Rate
PER	Packet Error Rate

Chapter 1

Introduction

1.1 Efficient Network Coding Schemes

There has recently been a surge of interest in the application of network coding [1–3] over packet networks, e.g., for large-scale file sharing [4–7]. Network coding has been shown to generally reduce the time for message delivery (called *delivery time*, e.g., the file-downloading time in the application of file sharing) in comparison to routing protocols for various probabilistic and deterministic models of flow transmission (called *traffic*) [8–12]. Considering real-world scenarios, however, one of the issues, which has been the focus of many researchers in the past decade, is the problem of designing efficient network coding schemes for various network scenarios [13–18]. This itself leads to the problem of modeling the traffic properly for different cases in practice. From a practical point of view, however, the accurate modeling of the traffic might be too complex and/or infeasible [19–22]. Examples of such cases are networks with time-varying or adversarial traffic schedules. The difficulty of the code design problem also increases when (some or all) network nodes are blind to the traffic. In particular, in various applications, the use of feedback or channel-state information is not practical or even feasible. Examples of such applications are information transmission over networks with large packet transfer delays. In a practical setting, an interesting problem is thus to achieve or approach the capacity of the network with high probability (w.h.p.) under any arbitrary traffic unknown at the network nodes

(i.e., when the network nodes have no a priori or a posteriori knowledge about the traffic).[1]

Presently, one of the most promising solutions for this problem is to utilize the random linear network coding scheme. To be more specific, random linear network codes (a.k.a. dense codes), originally proposed by Ho *et al.* in [17], are known to asymptotically (i.e., when the message size tends to infinity) achieve the capacity of unicast (one-source one-sink) networks with arbitrary network topologies. This however comes at the expense of having a linear coding cost (the encoding/decoding algorithms at each node require a number of packet operations per message packet linear in the message size) [23,24]. The rather high coding cost, on the other hand, impedes the application of dense codes for the transmission of large files. One would thus be interested in devising coding schemes with relatively low complexity. To overcome the computational inefficiency of dense codes, Maymounkov *et al.* proposed *chunked codes* (CC) in [24]. These codes operate by dividing the original message into non-overlapping chunks. Each node then randomly chooses a chunk at any time instant and transmits it by using a dense code. Thus, CC require less complex coding operations as they apply coding on smaller chunks rather than the original message. (The coding cost of such codes is linear in the size of the chunks.) This advantage of CC, however, comes at the cost of lower speed of convergence or higher message or packet error rate compared to dense codes. However, the problem of designing network codes over arbitrary traffics in this setting which are more computationally efficient than dense codes, on the one hand, and are more communicationally efficient than CC, on the other hand, has been still left open in the literature.

It is noteworthy that although the speed of convergence of dense codes and CC to the capacity of unicast networks with arbitrary (deterministic) traffics (with arbitrary deterministic transmission schedules, loss and delay models) was studied in [24], it is not straightforward to apply those results to the networks with probabilistic traffics. From another view point, however, such results for the case of the probabilistic traffics are very useful in practice to estimate the delivery time. In the case of probabilistic

[1]Hereafter, in this thesis, we assume that any type of traffic, unless explicitly mentioned otherwise, is unknown at the network nodes.

traffics, Lun *et al.* [18], for the first time, showed that dense codes achieve the capacity of unicast networks with probabilistic traffics specified by stochastic processes with bounded average rate. They however did not discuss the speed of convergence of such codes to the capacity. Later, in [10,25], the authors studied the speed of convergence of such codes to the capacity of unicast line networks with some probabilistic traffics. In contrast to a few works on dense codes, to the best of our knowledge, there is no such result on CC over the networks with probabilistic traffics in the literature.

Besides the existing works, mentioned earlier, on the analysis of the speed of convergence of CC over unicast networks (particularly, those with a line topology) with various (arbitrary deterministic or some probabilistic) traffics in the absence of feedback, there are also some results in the literature focusing on the problem of increasing the speed of convergence of CC to the capacity (i.e., reducing the delivery time) over networks with feedback [26,27]. The general approach in these works is to use the feedback information in order to devise more efficient policies for scheduling the chunks (for transmissions over the network) compared to the random scheduling policy, which was originally proposed in [24] to perform efficiently over networks without feedback. In some real-world scenarios, the feedback is often available. In such cases, one thus expects to reduce the (average) delivery time when the feedback is properly used. In fact, the scheduling of chunks uniformly at random might result in wasting a large number of transmission opportunities. The reason is that, such a scheme treats those chunks which are already decodable or are short of only a few more packets to be decodable, similar to those chunks which need a much larger number of packets to be decodable. In the design of efficient CC for networks in the presence of feedback, the problem is therefore how to use feedback and devise a scheduling policy which outperforms the random scheduling policy. The design of optimal scheduling policies for different network scenarios is still an open problem, and more specifically, it is still not well-understood how the existing policies perform over networks with loss and delay.

1.2 Thesis Contributions and Related Work

Chunked codes (CC) [24], as introduced earlier, are an attractive alternative to random linear network codes (dense codes) due to their lower computational complexity. In addition to their advantage in terms of the computational efficiency, in [24], it was also shown that CC asymptotically achieve the capacity of line networks with arbitrary deterministic traffics so long as the size of chunks is bounded from below. This lower bound has been shown to be an increasing function of the message size. Thus the chunk size cannot be reduced down to a constant in the message size. The coding algorithms, therefore, cannot be performed in linear time (with constant costs in the message size). This may hamper the use of such codes in practical applications with severe computational resource limitations. The need for coding schemes with smaller coding cost further motivated the authors in [24] to also analyze CC with smaller chunk sizes. They showed that CC (with precoding) with chunks of smaller sizes (down to some constant in the message size) approach the capacity with an arbitrarily small but nonzero constant gap. In both cases of (sufficiently large or smaller) chunk sizes, however, the higher is the computational efficiency, the lower is the speed of convergence of such codes to the capacity (with or without a gap), or the larger is the message or packet error rate.

In Chapter 3, motivated by the work of [24], we focus on the problem of designing (communicationally and computationally) efficient network codes for unicast over line networks with arbitrary deterministic traffics. In particular, we are interested in studying the complexity-performance tradeoff, i.e., the tradeoff between the computational complexity, on the one hand, and the speed of convergence to the capacity (with or without a gap) and the message or packet error rate, on the other hand. Targeting the design of (chunk-based) codes with better complexity-performance tradeoff compared to the state-of-the-art (i.e., CC of [24]), the main contributions in Chapter 3 are as follows:

- We provide a more detailed analysis of dense codes, compared to that of [24]. The results of this part show that for arbitrary deterministic traffics, a dense code always achieves the capacity. The analysis of dense codes will then serve

as the basic framework for the analysis of the other coding schemes discussed in the rest of the chapter.

- We revise the analysis of CC in [24], and prove that a CC with any chunk size provides a better tradeoff between the computational complexity, on the one hand, and the speed of convergence and the message or packet error rate, on the other hand, in comparison with what was previously thought, based on the results of [24]. In particular, we show that for arbitrary deterministic traffics (i) a CC achieves the capacity, so long as the size of chunks is lower bounded by a function super-logarithmic in the message size and super-log-cubic in the network length, and (ii) a CC, preceded by a capacity-achieving erasure code, approaches the capacity with an arbitrarily small but nonzero constant gap, so long as the size of chunks is lower bounded by a function constant in the message size and log-cubic in the network length.

- We propose chunked codes with overlapping chunks, referred to as *overlapped chunked codes* (OCC), and show that (i) for sufficiently large chunk sizes, an OCC (an OCC with larger overlap size) achieves the capacity, but with a lower speed of convergence compared to a CC (an OCC with smaller overlap size), for any given message error rate; (ii) for smaller chunk sizes, an OCC (an OCC with larger overlap size) approaches the capacity with an arbitrarily small but nonzero constant gap with a larger speed of convergence when compared to a CC (an OCC with smaller overlap size), for sufficiently small given message or packet error rate.

- The result of (ii) leads to the design of linear-time network codes (with very small chunks of constant size with respect to the message size), which perform better than the existing linear-time codes in the literature with the same computational complexity over networks with arbitrary deterministic traffics.

- As part of the machinery used in our analysis, we generalize a recently proposed conjecture in [28] on the rank property of a special class of random matrices

with overlapping bands to four classes of more general random banded matrices.[2]

- We also demonstrate the advantage of finite-length OCC (OCC with larger overlap sizes) over CC (OCC with smaller overlap sizes) through extensive simulation results. For example, our results show that when compared to a CC with a similar coding cost, the application of OCC can decrease the downloading time of a 1MB file from a file server 4 hops away by about 17% – 30%.

Parts of the work in Chapter 3 have appeared in [29,30]. A more detailed version is available in [31].

It is worth noting that considering the problem of designing computationally efficient codes, there are also a number of variations of chunk-based codes, different from those in [24], in the literature of efficient network codes as well as efficient erasure codes over a single erasure channel (e.g., see [26–28,32–41]). To the best of our knowledge, however, none of these codes has been provably shown to perform better than that in [24] for arbitrary deterministic traffics. In particular, in [26, 27, 35–41], the authors considered problem settings that are different from ours (i.e., either the traffic model is not arbitrary, or it is known at the network nodes, or some feedback or channel-state information is available at the network nodes), and hence their results will not be discussed in this thesis. However, the settings considered in [28,32–34] are similar to ours to some extent, i.e., they consider the problem of design and/or analysis of chunk-based codes for arbitrary deterministic traffics, but over an erasure channel, unlike our work that is concerned with an arbitrarily long network of erasure channels. To be more specific, focusing on the design of computationally efficient codes over an erasure channel with an arbitrary deterministic traffic, in [28], Studholme and Blake propose "windowed erasure codes." These codes have a similar structure to the CC of [24], except that every two contiguous chunks overlap in all but one packet. From the results of [28], it can be concluded that, compared to non-overlapping chunks, the application of overlapping chunks provides a significant

[2]The validity of our conjecture is supported via simulations, but a formal proof is still unknown.

improvement in the speed of convergence to the capacity and/or in the probability of decoding failure. This advantage arises from the large size of the overlap between the chunks. However, the larger is the overlap size, the larger will be the number of chunks. Having a large number of chunks, regardless of whether the chunks overlap or not, hampers the application of such codes over a longer network of erasure links (as shown in [24]). This is simply a result of the fact that the more are the chunks and/or the network links, the longer it would take for the last chunk to be decodable (i.e., to receive a sufficient number of informative packets) at the end of the network (this phenomenon is also known as "the curse of the coupon collector" [24]). To remedy this situation, we reduce the overlap size in our version of CC with overlapping chunks. We in fact demonstrate that with the right balance between the overlap size and the number of chunks, OCC can be a viable choice for information transmission over packet networks.

The idea of overlapping chunks has also been proposed by Silva *et al.* in [32], independently. Unlike our contiguous overlapping scheme, the overlapping scheme of [32] has a grid structure. However, in [32], no theoretical result is presented and the simulation results are on the application of such codes over a single erasure channel, not a (longer) line network. In [34], Li *et al.* propose a randomized overlapping scheme and provide a finite-length analysis of such codes over an erasure channel. Our preliminary simulations, however, demonstrate that these codes do not perform well over longer line networks. Moreover, the analysis of [34] does not seem to be generalizable to line networks. In Chapter 3, however, we provide an asymptotic analysis of OCC (with contiguous overlaps) over line networks in general. The proposed codes are superior to those of [32] and [34] in the scenario of arbitrary deterministic traffics. This arises from the fact that due to the structure of overlapping schemes in [32] and [34], for a low-complexity decoding algorithm, the chunks need to be decoded one at a time. In our work, however, the decoding can be performed on the set of all the chunks together while preserving the low complexity of the decoding algorithm. This thus results in a better tradeoff between the speed of convergence to the capacity and the message or packet error rate. In fact, by performing the decoding algorithm on the set of all the chunks simultaneously, for

successful decoding, no chunk needs to be recoverable in isolation. Thus a smaller number of successful packet transmissions is sufficient to ensure successful decoding for a given message/packet error rate. This, however, may come at the cost of increasing memory requirements.

After studying the complexity-performance tradeoffs of the dense codes and CC (with or without overlapping chunks) over line networks with arbitrary deterministic traffics in the first part of the thesis (in Chapter 3), in Chapter 4, we pursue our analytical work and consider the problem of studying the speed of convergence of such codes over line networks with some probabilistic traffics. In the literature, there are two metrics, referred to as "coding delay" and "average coding delay," which have been used for measuring the speed of convergence of a given code to the capacity (with or without a gap) of a unicast network with a given traffic. The *coding delay* of a given code over a network with a given traffic (schedule of transmissions, losses and delays) is the minimum time that the code takes to transmit the message from the source node to the sink node. The coding delay of a class of codes over a class of traffics is therefore a random variable due to the randomness in both the code and the traffic. The *average coding delay* of a class of codes with respect to a class of traffics is the coding delay of the underlying class of codes averaged out over all the traffics (but not the codes), and hence is a random variable due to the randomness in the code.

Pakzad *et al.* [10], for the first time, studied the average coding delay of dense codes (operating in the field of size two) over line networks with (delay-free) deterministic regular transmissions and Bernoulli losses, where the special case of two identical links in tandem was considered. The analysis however did not provide any insight about how the coding delay can deviate from the average coding delay.

More recently, Dikaliotis *et al.* [25] studied both the average coding delay and the coding delay of dense codes (operating in a finite field of infinitely large size) over the line networks of arbitrary length with traffics similar to those in [10], but under the assumption that there exists a unique worst link (i.e., a unique link with the minimum probability of transmission success) in the network.

Motivated by the lack of generality in the results of [10, 25], in Chapter 4, we

extend the analysis technique used in Chapter 3 and study the coding delay and the average coding delay of dense codes, and for the first time, CC for different ranges of the chunk sizes, operating in the field of size two, over line networks with traffics similar to those in [10, 18, 25]. In Chapter 4, our focus is on CC with disjoint (non-overlapping) chunks. The extension of the analysis to OCC is not straightforward and is beyond the scope of this work. Unlike the works of [10, 25], our study in Chapter 4 has no limiting assumptions on the traffic parameters or the length of the network. It is worth noting that any upper bound on the coding delay or on the average coding delay of any coding scheme over the field of size two serves as an upper bound for the underlying code over any finite field of larger size. The method of analysis in this work is itself, however, generalizable to finite fields of larger size, but the generalization is not trivial and is beyond the scope of this thesis. The main contributions of Chapter 4 are:

- We derive upper bounds on the coding delay and the average coding delay of a dense code, or a CC alone, or a CC with precoding, in the asymptotic setting, i.e., as the message size tends to infinity, over line networks (with traffics) with (delay-free) deterministic regular transmissions or Poisson transmissions and Bernoulli losses with arbitrary parameters.[3] The upper bounds are functions of the message size, the length of the network, and the parameters of the traffic and the code. In order to investigate how the upper bounds are related to the actual values of all the traffic parameters (not only the minimum traffic parameter), we also consider a special case with unequal traffic parameters, where no two links have equal traffic parameters. The upper bounds, in this case, particularly indicate how the coding delay or the average coding delay change as a function of the minimum of the (absolute value of the) difference between the traffic parameters of any two consecutive links in the network.

- We show that: (i) our upper bounds on the average coding delay of dense codes are in some cases more general, and in some other cases tighter, than

[3]The scenario of (delay-free) deterministic regular transmissions and Bernoulli losses has been used in [10,25], and the scenario of (delay-free) Poisson transmissions with Bernoulli losses has been used in [18] as a special case of the probabilistic traffics over line networks.

what were presented in [10, 25], and (ii) the coding delay of dense codes may have a large deviation from the average coding delay in both cases of identical and non-identical links; for non-identical links, our upper bound on such a deviation is smaller than what was previously shown in [25]. It is noteworthy that, for identical links, upper bounding such a deviation has been an open problem (see [25]).

- We also show that: (i) a CC achieves the capacity, so long as the size of the chunks is bounded from below by a function super-logarithmic in the message size and super-log-linear in the network length, and (ii) the combination of a CC and a capacity-achieving erasure code approaches the capacity with an arbitrarily small nonzero constant gap, so long as the size of the chunks is bounded from below by a function constant in the message size and log-linear in the network length. The lower bounds in both cases are smaller than those over the networks with arbitrary deterministic traffics. Thus both coding schemes (i.e., stand-alone CC and CC with precoding) are less computationally complex (require smaller chunks), for the same speed of convergence (with or without a gap) to the capacity, over such probabilistic traffics, compared to arbitrary deterministic traffics.

- In a capacity-achieving scenario, for such probabilistic traffics, we show that for CC: (i) the upper bound on the overhead (the difference between the coding delay and the capacity[4]) grows sub-log-linearly with the message size and the network length, and decays sub-linearly with the size of the chunks, and (ii) the upper bound on the average overhead (the difference between the average coding delay and the capacity) grows sub-log-linearly (or poly-log-linearly) with the message size, and sub-log-linearly (or log-linearly) with the network length, and decays sub-linearly (or linearly) with the size of the chunks, in the case with arbitrary (or unequal) traffic parameters. For arbitrary deterministic traffics, the upper bound on the overhead or that on the average overhead (as shown

[4]For the definition of capacity used here, see Section 4.1.1

in Chapter 3) is similar to the case (i), mentioned above, but with a larger (super-linear) growth rate with the network length.

The results of Chapter 4 have been presented in part in [42,43]. A more detailed version is available in [44].

In the last part of the thesis, focusing on the design of efficient network codes over networks in the presence of feedback (unlike the cases with the absence of feedback, considered in Chapters 3 and 4), in Chapter 5, we consider the problem of devising (more) efficient feedback-based scheduling policies for chunk-based codes (compared to the random scheduling policy used in Chapters 3 and 4) over unicast networks, particularly line networks.

In this context, in earlier works [26, 27], two general policies, which utilize the feedback information to schedule the chunks, were proposed. These scheduling policies were referred to as *random push* (RP) and *local-rarest-first* (LRF), respectively. Both RP and LRF scheduling policies, employed by the transmitting node over a link, use the number of *innovative packets*[5] which have been received by the receiving node of the link till the current transmission time. In RP [26], the node transmitting over a link chooses a chunk uniformly at random from the set of chunks that still need more innovative packets to be decodable at the receiving node of the link. In LRF [27], however, the transmitting node chooses a chunk which needs the largest number of innovative packets at the receiving node.

In both RP and LRF policies, at each time instant, a transmitting node makes a decision based on the set of received packets at the receiving node up to that point in time. In the presence of delay, however, such a decision fails to take into account the contribution of the (successful) packets that were transmitted earlier to the receiving node (over the same link or the other links with different transmitting nodes but with the same receiving node as the underlying one) but have not still been received due to the delay. One thus expects to be able to improve these scheduling

[5]A packet is said to be "innovative" at a node if its global encoding vector (i.e., the vector of the coefficients which represent the mapping between the packet and the message packets at the source node) is linearly independent of the global encoding vectors of the packets previously received by the node.

polices over the networks with delay. Related to this, one should note that both RP and LRF policies utilize the feedback information in order to count the number of innovative packets delivered to the receiving node. This, however, disregards the packets which have been (successfully) transmitted but still have not been received. Nevertheless, the more are such transmissions corresponding to a chunk, the larger is the probability of delivering more useful information about the underlying chunk. In addition, thanks to the literature on modeling the packet loss and the packet delay over networks with feedback (e.g., see [45, 46] and references therein), such probabilities can be computed with a reasonably high accuracy. This however comes at the cost of more computation at the network nodes. (In this thesis, we do not focus on the problem of modeling the loss and the delay of the network links, the estimation of the model parameters, and the tradeoff between the accuracy of such parameters and the computational complexity.) In Chapter 5, we assume that the models of the packet transmission, packet loss and the packet delay of each link are known at the transmitting/receiving nodes of the link. The question then is how to properly use (i) the knowledge about the sets of transmitted and received packets over a link, (ii) the knowledge about the sets of received packets over the rest of the links with the same receiving node (as that of the underlying link), and (iii) the knowledge about the link model parameters, in order to decrease the average delivery time. In an attempt to answer this question in Chapter 5, the main contributions are as follows:

- We propose a new scheduling policy for CC,[6] referred to as *minimum-distance-first* (MDF), devised based on a new metric, i.e., the expected number of innovative packets transmitted prior to the *next* transmission time.

- Aiming at the design of a low-complexity version of MDF, we also propose another scheduling policy for CC, referred to as *minimum-current-metric-first* (MCMF), which works based on the expected number of innovative packets transmitted prior to the *current* transmission time.

[6]Similar to Chapter 4, CC are the focus of Chapter 5, and in the case of OCC, the generalization of the proposed scheduling policies is not trivial, and is beyond the scope of this thesis.

- We show through extensive simulations over line networks (as the simplest non-trivial network topology for the unicast problem[7]) that (i) the MDF scheduling policy performs (near) optimal in the sense of minimizing the average delivery time; (ii) both MDF and MCMF are always superior to LRF and RP with respect to the average delivery time, and that the improvements are particularly large for smaller chunks and larger networks as well as delays with smaller mean and variance; (iii) MCMF is always inferior to MDF, but the performance loss becomes smaller for larger chunks, smaller networks and delays with smaller mean and variance; (iv) the relative performance of MDF or MCMF compared to the random scheduling policy depends on the delay distribution. In particular, the advantage of the proposed scheduling policies becomes more profound for smaller chunks, larger networks and delays with smaller mean and variance; (v) for sufficiently small chunks and sufficiently large networks, RP is superior to LRF, and the advantage is more evident for smaller chunks, larger networks, and for delays with larger mean and variance.

1.3 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, the basic model is described, and some preliminary definitions and notations are defined. Chapter 3 provides the analytical results for dense codes, chunked codes (CC) and overlapped chunked codes (OCC), in an asymptotic regime, over line networks with arbitrary deterministic traffics in the absence of feedback. In Chapter 3, we also present finite length simulation results for such codes in the described setting. Chapter 4 generalizes the analysis framework of Chapter 3 to the case of line networks (with some probabilistic traffics) with (delay-free) deterministic regular or Poisson transmissions and Bernoulli losses in the absence of feedback, and gives an asymptotic analysis of coding delay and average coding delay of dense codes and CC in such scenarios. The

[7]In a practical scenario, the line network topology would be the right model for an overlay network where the sequence of nodes are determined by an underlying routing protocol.

proof of the results in Chapters 3 and 4 are given in Appendices A and B, respectively. In Chapter 5, we propose (and analyze) new scheduling policies for CC (via extensive simulations) over networks with loss and delay in the presence of feedback. Chapter 6 concludes the thesis and highlights directions for future work.

Chapter 2

Basic Model and Definitions

In this chapter, we describe the basic model used in this thesis, unless explicitly mentioned otherwise, and give some preliminary definitions and some basic notations that will be frequently used in the rest of the thesis.

2.1 Network Scenario

In this thesis, we focus on a unicast information flow problem. In Chapters 3 and 4, we consider the problem of unicast over line networks. In Chapter 5, the problem of unicast over networks with arbitrary topologies is considered, although similar to Chapters 3 and 4, in Chapter 5, our focus will be on line networks. It is noteworthy that our results and analysis are generalizable to more general network information flow problems, e.g., (multiple) unicast or multicast, over more general network topologies. The generalizations however are not straightforward, and are beyond the scope of this thesis.

2.1.1 Unicast over Line Networks

The collection of L links connecting $L+1$ nodes $\{v_i : 0 \leq i \leq L\}$ in tandem is called a *line network* of *length* L, i.e., for every $0 \leq i < L$, there is a directed link (v_i, v_{i+1}) between two network nodes v_i and v_{i+1} (see Figure 2.1). The network nodes v_i and v_{i+1} are called the *transmitting* and *receiving* nodes of the link (v_i, v_{i+1}), respectively.

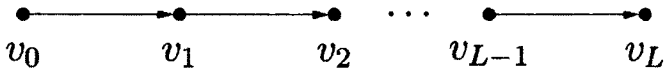

Figure 2.1: A line network of length L.

Over a line network of length L, a *unicast* (one-source one-sink) *problem* is defined as follows: The node v_0, called the *source node*, generates a vector of messages, and the node v_L, called the *sink node*, demands the vector of messages. The rest of the nodes $\{v_i : 0 < i < L\}$ in the network, called *internal nodes*, are responsible for relaying the vector of messages from the source node to the sink node.

2.1.2 Finite Field and Vector Space

We assume that the source node has a set of k message vectors $\{\boldsymbol{x}_j : 1 \leq j \leq k\}$. The message vectors (also called the *message packets*) are each drawn from a vector space $\mathcal{F}(\mathbb{F})$ over a finite field $\mathbb{F}$. The index j is called the *label* of the message vector $\boldsymbol{x}_j$. We denote the set of labels of the message vectors by $\mathcal{M}$.[1]

2.1.3 Traffic over Network

Suppose that following a certain timing schedule, for every $0 \leq i < L$, the i^{th} node transmits a packet, i.e., a vector in $\mathcal{F}(\mathbb{F})$, over the $(i+1)^{\text{th}}$ link in any opportunity that it gets. (The time instants at which the packet transmissions occur are assumed to follow a model referred to as the *transmission model*.) The links may be lossy, i.e., a packet which is transmitted by the i^{th} node may not reach the $(i+1)^{\text{th}}$ node. In this case, the packet is called an *erased packet*, otherwise, it is referred to as a *successful packet*. (The transmission erasures are assumed to follow a model referred

[1]The finite field $\mathbb{F}$ is defined based on two operations "addition" and "multiplication," respectively represented by symbols "+" and "·". We assume that both operations have the same cost, as one field operation. We also define two types of operations in a vector space $\mathcal{F}$: (i) $\boldsymbol{y} + \boldsymbol{z}$, for $\boldsymbol{y}, \boldsymbol{z} \in \mathcal{F}$, is taken to be symbol-wise with respect to operator + in $\mathbb{F}$, and requires one packet operation (where a vector in $\mathcal{F}(\mathbb{F})$ is called a *packet*), and (ii) $x\boldsymbol{y}$, for $x \in \mathbb{F}$ and $\boldsymbol{y} \in \mathcal{F}$, symbol-wise with respect to operator . in $\mathbb{F}$, which also requires one packet operation.

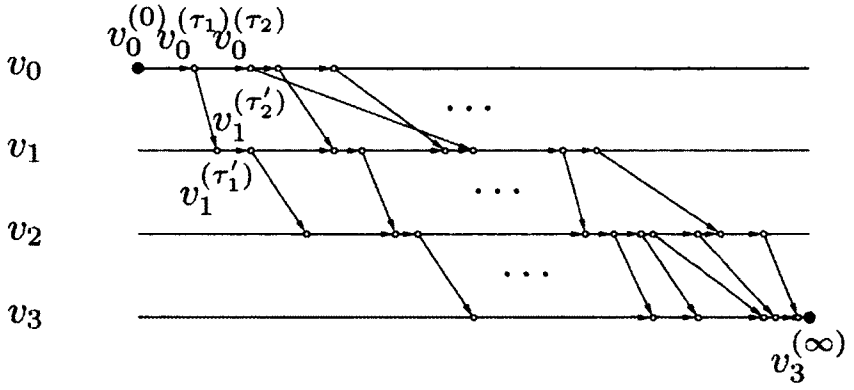

Figure 2.2: A trellis for a traffic over a line network of length 3.

to as the *loss model.*) Each successful packet may also experience some delay. (The delays that the successful packets experience are assumed to follow a model referred to as the *delay model.*) The flow of transmission of successful packets is called traffic. (The traffic model over a network can be specified by the transmission, loss and delay models.)

2.1.4 Graphical Representation of Traffic

We use a digraph, called *trellis connectivity graph* (or *trellis* for brevity), to represent a given traffic (see Figure 2.2). The trellis only represents the successful transmissions through the network. For every $0 \leq i < L$, the node set of the trellis includes the nodes $v_i^{(\tau)}$ such that a successful packet either departs from or arrives at the i^{th} node at time τ, and two extra nodes $v_0^{(0)}$ and $v_L^{(\infty)}$.[2] The edge set of the trellis consists of two groups of directed edges specified as follows; (i) ***traffic edges***: every 2-tuple $(v_i^{(\tau)}, v_{i+1}^{(\tau')})$, $\tau \leq \tau'$, representing an *in-edge* (*out-edge*) of the $(i+1)^{\text{th}}$ (i^{th}) node, for every pair of nodes $v_i^{(\tau)}$ and $v_{i+1}^{(\tau')}$, if there is at least one packet transmitted by the i^{th} node at time τ and received by the $(i+1)^{\text{th}}$ node at time τ', and (ii) *memory edges*:

[2]The node $v_0^{(0)}$ represents the source node at time zero when all the message vectors are available, and the node $v_L^{(\infty)}$ represents the sink node at a time after which there is no more coded packet to arrive.

every 2-tuple $(v_i^{(\tau)}, v_i^{(\tau')})$, $\tau \leq \tau'$, if, for all $\tau'' : \tau < \tau'' < \tau'$, there is no node $v_i^{(\tau'')}$ in the trellis. For instance, in the trellis shown in Figure 2.2, the edges $(v_0^{(0)}, v_0^{(\tau_1)})$, $(v_0^{(\tau_1)}, v_0^{(\tau_2)})$, and $(v_1^{(\tau_1')}, v_1^{(\tau_2')})$ are examples of memory edges, and the edge $(v_0^{(\tau_1)}, v_1^{(\tau_1')})$ is an example of a traffic edge.

Without loss of generality, the traffic edges are assumed to have unit capacity, i.e., only one packet is successfully sent over a traffic edge, as parallel traffic edges are allowed. Also, the memory edges are assumed to have infinite capacity, i.e., the size of the memory at the network nodes is assumed to be unbounded (all the packets received by any given node will remain the memory of that node till the end of the period of transmissions, and at any given time, the node has access to all its previously received packets).

For every $0 \leq i \leq L$, let us label the in-edges (out-edges) of the i^{th} node prior to time τ such that the first in-edge (out-edge) has label 1, the second has label 2 and so forth. Let $\mathcal{I}_i^{(\tau)}$ ($\mathcal{O}_i^{(\tau)}$) be the set of labels of the in-edges (out-edges) of the i^{th} node prior to time τ. Clearly $\mathcal{I}_0^{(\tau)} = \mathcal{O}_L^{(\tau)} = \emptyset$, for all τ. The set $\mathcal{I}_i^{(\tau)}$ ($\mathcal{O}_i^{(\tau)}$) represents the ordering of the successful packets received (transmitted) by the i^{th} node prior to time τ. For simplifying the notation, hereafter, for every $0 \leq i \leq L$, let us denote the set $\mathcal{I}_i^{(\infty)}$ ($\mathcal{O}_i^{(\infty)}$) by $\mathcal{I}_i$ ($\mathcal{O}_i$).

2.1.5 Capacity of Traffic

The maximum number of message vectors that can be successfully transmitted through a network with a given traffic is called the *capacity of network under the traffic* or simply the *capacity of the traffic.*

Modeling a traffic as a flow network, the capacity of the traffic equals the maximum flow between the two nodes $v_0^{(0)}$ and $v_L^{(\infty)}$ in the trellis graph of the traffic. By the max-flow min-cut theorem, the capacity of a traffic thus equals the minimum of the sum of the capacities of the cutset edges among all the cutsets in the trellis. This quantity is called the *min-cut capacity.*

The min-cut capacity is achievable if the network nodes are able to (properly) process their received packets and generate new packets to be transmitted. When

processing is allowed at the nodes, the information flow scheme is called *network coding*, and the min-cut capacity is therefore often referred to as the *network coding capacity*.

When the network nodes are only allowed to generate coded packets by linearly combining their received packets, the network coding scheme is called *linear*. When network codes are restricted to be linear, the maximum number of message vectors that can be transmitted through a network with a given traffic is called the *linear coding capacity*. We focus on linear network codes in this thesis, and for brevity, hereafter, we refer to the linear coding capacity as the *capacity*. The capacity of a unicast scenario is itself equal to the number of (traffic) edge-disjoint paths between the two nodes $v_0^{(0)}$ and $v_L^{(\infty)}$ in the trellis.

2.1.6 Coding over Traffic

In this thesis, we restrict the codes to be *linear*. In a linear coding scheme, a coded packet $\boldsymbol{y}_e$ to be transmitted over the out-edge (with label) e of i^{th} node at time τ (or the source node at time τ) is generated by $\sum_{\ell \in \mathcal{I}_i^{(\tau)}} \lambda_{e,\ell} \boldsymbol{y}_\ell$ (or $\sum_{j \in \mathcal{M}} \mu_{e,j} \boldsymbol{x}_j$), where $\lambda_{e,\ell}, \mu_{e,j} \in \mathbb{F}$, $\forall \ell \in \mathcal{I}_i^{(\tau)}$, and $\forall j \in \mathcal{M}$; and $\forall \ell \in \mathcal{I}_i^{(\tau)}$, $\boldsymbol{y}_\ell \in \mathcal{F}(\mathbb{F})$ is the packet received along the in-edge ℓ of the i^{th} node at some time prior to time τ. For every message vector $\boldsymbol{x}_j$, $j \in \mathcal{M}$, the vector $\hat{\boldsymbol{x}}_j$ is estimated by $\sum_{e \in \mathcal{I}_L} \nu_{e,j} \boldsymbol{y}_e$, where $\nu_{e,j} \in \mathbb{F}$, $\forall e \in \mathcal{I}_L, \forall j \in \mathcal{M}$, and $\boldsymbol{y}_e \in \mathcal{F}(\mathbb{F})$ is the packet received along the in-edge e of the sink node.

We say that a given code fails to transmit k message vectors over a given traffic of capacity n if, by applying the underlying code over the underlying traffic, the sink node fails to recover all the k message vectors; otherwise, the code is said to succeed.

Let $k_{n,\epsilon}$ be the largest function of n, such that for a given $0 < \epsilon < 1$ (in general, ϵ itself can depend on k), with probability of failure no larger than ϵ, a randomly chosen code (from a given class of codes) fails to transmit $k_{n,\epsilon}$ message vectors over a randomly chosen traffic (from a given class of traffics) with capacity n. For a given $0 < \epsilon < 1$, if the ratio $(n - k_{n,\epsilon})/n$ goes to 0, or γ, for some constant $0 < \gamma < 1$, as n goes to infinity, the coding scheme (of the underlying class of codes) is said to

achieve the capacity, or approach the capacity with a gap γ, respectively. For any given n, the closer is the ratio $(n - k_{n,\epsilon})/n$ to 0, the higher is said to be the speed of convergence of a coding scheme.

2.1.7 Coding Complexity

The number of packet operations for applying the encoding functions divided by the product of k and L is called the *encoding cost* (i.e., to calculate the encoding cost of a coding scheme, one can add up the number of packet operations needed to generate all the coded packets at all the non-sink nodes, and normalize it by the number of message vectors multiplied by the number of links). The number of packet operations for applying the decoding functions divided by k is called the *decoding cost*. We often refer to the encoding cost and the decoding cost simply as the coding cost, unless there is a danger of confusion.[3]

2.2 Some Notations

We here define some of notations that will be frequently used in this thesis. The rest of the notations will be defined throughout the thesis.

- We use the asymptotic notations $O(.)$, $o(.)$, $\Omega(.)$ and $\omega(.)$ defined as follows. For non-negative functions $f(n)$ and $g(n)$, we write: (i) $f(n) = O(g(n))$ if and only if $\limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty$; (ii) $f(n) = o(g(n))$ if and only if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$; (iii) $f(n) = \Omega(g(n))$ if and only if $\limsup_{n\to\infty} \frac{f(n)}{g(n)} > 0$; (iv) $f(n) = \omega(g(n))$ if and only if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$; and (v) $f(n) \sim g(n)$ if and only if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$.
- For the ease of exposition, for any real number x, we will often denote $\frac{x}{2}$ by $\dot{x}$.
- For any real number z, we write $[z]^+ = z$, if $z \geq 0$, and $[z]^+ = 0$, otherwise.

[3]The coding cost excludes field operations that are independent of the size of the packet. The reason is that the packet size is usually very large in practice compared to the number of message vectors, and the computations dealing with packet operations dominate the computational cost.

- For any random variable Y, we denote its expected value and variance by $\mathbf{E}(Y)$ (or $\mathbf{E}[Y]$) and $\mathbf{Var}(Y)$, respectively.

Chapter 3

Network Coding over Networks with Arbitrary Deterministic Traffics

3.1 Network Model and Assumptions

3.1.1 Transmission, Loss and Delay Models

In this chapter, we focus on a class of traffics, referred to as *arbitrary deterministic traffics* with arbitrary deterministic transmission, loss and delay models, defined as follows.

The transmissions are assumed to occur at arbitrary times over each link. The links are assumed to be erasure channels (as described in Section 2.1.3), and we assume that the erasure rates of different links might be different and might also be time-varying. In addition, the links are assumed to have arbitrary time-varying delays. Thus, the successful packets over each link might be re-ordered at the receiving end of that link.

We consider the class of arbitrary deterministic traffics of a given capacity, and study the performance of coding schemes of our interest in this thesis over such traffics. The class of such traffics of a given capacity n is the set of all traffics whose trellis graphs consist of n (and only n) edge-disjoint paths starting from $v_0^{(0)}$ and ending at $v_L^{(\infty)}$. Due to the (arbitrary) nature of such traffics, the analysis of an arbitrarily chosen traffic from the class of the underlying traffics however does not

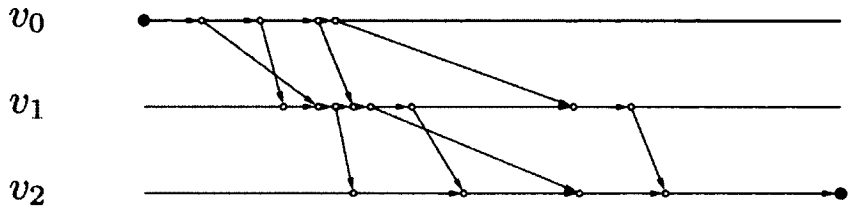

Figure 3.1: An example of a network of length 2 with a traffic of capacity 4 with exactly 4 transmitted and/or received packets at any network node.

yield any insight into the performance of different coding schemes over the class of traffics. We, instead, analyze the worst-case traffics in the class of arbitrary deterministic traffics of a given capacity, and compare different coding schemes for such traffics. The worst case traffics considered here are defined as follows. The class of worst-case traffics of a given capacity n is the set of all traffics of capacity n whose trellis graphs satisfy two conditions as follows: for every $0 < i < L$, (i) $|\mathcal{I}_i| = |\mathcal{O}_i| = n$, i.e., precisely n packets are (successfully) transmitted and received by the i^{th} node, and (ii) for any time τ, $|\mathcal{O}_i^{(\tau)}| \leq |\mathcal{I}_i^{(\tau)}|$, i.e., the number of successful transmissions at the i^{th} node prior to time τ does not exceed the number of successful receptions at that node prior to time τ. Condition (i) corresponds to having the minimum number of successful transmissions that can support a traffic of capacity n. Condition (ii) ensures that under the constraint of Condition (i), the trellis has n edge-disjoint paths. An example of a traffic satisfying both conditions (i) and (ii) is shown in Figure 3.1.

3.1.2 Assumptions

We assume that the size of the memory at the network nodes is unbounded, i.e., all the packets, received by a node, will remain in the memory of that node till the end of the transmission time.

We also assume that network nodes are blind to the traffic, i.e., there is no a priori or a posteriori knowledge about the traffic at the network nodes. To be more

specific, we consider a scenario where (i) the transmission, loss and delay models are "unknown" at the network nodes, and (ii) no (instantaneous or non-instantaneous) feedback or (short- or long-term) channel-state information is available in the network in the course of transmission (i.e., before the time that the sink node recovers all the message vectors). Whenever the decoding process is successful, the sink node sends an acknowledge message to the second last node. The second last node then stops transmitting new packets to the sink node, and relays the acknowledge message to the third last node. The feedback relaying process continues over the links till the time that the source node receives the acknowledge message, and stops transmitting new packets. The feedback transmissions are assumed to be error/erasure-free and with zero delay.

3.2 Problem Setup

Consider a line network of length L with an arbitrary deterministic traffic of capacity n. For any given coding scheme over a vector space $\mathcal{F}(\mathbb{F}_2)$,[1] our goal is to derive tight upper bounds on (i) the capacity of the traffic (n) as a function of k, L and ϵ (and maybe other code or traffic parameters), in the asymptotic regime (i.e., as k goes to infinity), so that the coding scheme fails to transmit k message vectors drawn from $\mathcal{F}(\mathbb{F}_2)$, with probability (w.p.) bounded above by (b.a.b.) ϵ, for a given $0 < \epsilon < 1$, and (ii) the encoding and decoding costs.

[1]We restrict the finite field $\mathbb{F}$ to the binary field $\mathbb{F}_2$ (i.e., each message vector is a stream of bits). The reason for this restriction is two-fold: (i) to have the lowest computational complexity, and (ii) to consider the case with the lowest speed of convergence (the larger is the field, the larger would be the speed of convergence of any coding scheme considered in this thesis, but at the cost of increasing the computational complexity).

3.3 Capacity-Achieving Codes

3.3.1 Dense Codes

We start with the analysis of random linear codes (a.k.a. dense codes),[2] and demonstrate that (i) dense codes are capacity-achieving over line networks of length L, with arbitrary deterministic traffics and k message vectors, so long as

$$L \log \frac{kL}{\epsilon} = o(k),$$

as k goes to infinity; (ii) a dense coding scheme fails w.p. b.a.b. ϵ, so long as the capacity of the traffic, n, is at least

$$k + L \log \frac{kL}{\epsilon} + \log \frac{1}{\epsilon} + L + 1,$$

and (iii) the encoding and decoding costs of dense codes are each $O(k)$.

Encoding:

Consider an arbitrary traffic in the class of worst-case traffics of capacity n defined in Section 3.1.1. For every out-edge (with label) e of the i^{th} node at time τ in the trellis of the traffic, $\boldsymbol{y}_e = \sum_{\ell \in \mathcal{I}_i} \lambda_{e,\ell} \boldsymbol{y}_\ell$ if the i^{th} node is an internal node, and $\boldsymbol{y}_e = \sum_{j \in \mathcal{M}} \mu_{e,j} \boldsymbol{x}_j$ if the i^{th} node is the source node, where, for every $\ell \in \mathcal{I}_i \backslash \mathcal{I}_i^{(\tau)}$, $\lambda_{e,\ell}$ is 0, and $\forall \ell \in \mathcal{I}_i^{(\tau)}$, $\forall j \in \mathcal{M}$, $\lambda_{e,\ell}$ and $\mu_{e,j}$ are symbols independently and uniformly drawn from $\mathbb{F}_2$.

Since every coded packet is a result of linearly combining the message vectors, every coded packet $\boldsymbol{y}_e$ over an out-edge e of the i^{th} node at time τ, for every $0 < i < L$,

[2] In a random linear coding scheme as explained later in detail, each packet transmitted by a node is a random linear combination of *all* previously received packets. Thus the number of non-zero coefficients in linear combinations is rather large, resulting in *dense* linear combinations.

can be written as

$$\begin{aligned} \boldsymbol{y}_e &= \sum_{\ell \in \mathcal{I}_i} \lambda_{e,\ell} \sum_{j \in \mathcal{M}} \mu_{\ell,j} \boldsymbol{x}_j \\ &= \sum_{j \in \mathcal{M}} \sum_{\ell \in \mathcal{I}_i} \lambda_{e,\ell} \mu_{\ell,j} \boldsymbol{x}_j \\ &= \sum_{j \in \mathcal{M}} \mu_{e,j} \boldsymbol{x}_j, \end{aligned}$$

where $\mu_{e,j} \doteq \sum_{\ell \in \mathcal{I}_i} \lambda_{e,\ell} \mu_{\ell,j}$, and $\mu_{\ell,j}$'s can be defined recursively.

The vector of coefficients of the linear combination associated with a packet is called the *local encoding vector* of the packet, and the vector of the coefficients representing the mapping between the message vectors and a coded packet is called the *global encoding vector* of the packet. In particular, the vector $\boldsymbol{\lambda}_e$ of n elements $\{\lambda_{e,\ell} : \ell \in \mathcal{I}_i\}$, and the vector $\boldsymbol{\mu}_e$ of k elements $\{\mu_{e,j} : j \in \mathcal{M}\}$ are the local and global encoding vectors of packet $\boldsymbol{y}_e$ defined earlier. The global encoding vector of a packet is assumed to be transmitted along with the packet in the packet header.[3]

Decoding:

For every message vector $\boldsymbol{x}_j$, $j \in \mathcal{M}$, the sink node provides an estimate $\hat{\boldsymbol{x}}_j = \sum_{e \in \mathcal{I}_L} \nu_{e,j} \boldsymbol{y}_e$, for some $\nu_{e,j}$ in $\mathbb{F}_2$, $\forall e \in \mathcal{I}_L$, by solving the system of linear equations $\{\boldsymbol{y}_e = \sum_{j \in \mathcal{M}} \mu_{e,j} \hat{\boldsymbol{x}}_j : e \in \mathcal{I}_L\}$ with respect to the unknown packets $\{\hat{\boldsymbol{x}}_j : j \in \mathcal{M}\}$.

This system is uniquely solvable and for every $j \in \mathcal{M}$, $\hat{\boldsymbol{x}}_j$ equals $\boldsymbol{x}_j$ if there exists an innovative collection[4] (of packets) of size k at the sink node. Thus, a dense code succeeds if the sink node receives a collection of k packets which form an innovative collection. Then to recover the message vectors, the sink node has to solve a system of linear equations whose unknowns, known constants and coefficient matrix's rows

[3] A dense code is set up to never transmit a coded packet with all-zero global encoding vector. Whenever such a packet is generated by a node, it will be discarded and a new packet will be generated till the resulting packet has a global encoding vector with at least one non-zero entry.

[4] We say that a collection of packets at a given node is *innovative* if their global encoding vectors are *linearly independent*. We refer to such a collection of packets by the set of (the labels of) in-edges over which these packets are received.

are the message vectors, the received packets at the sink node and the global encoding vectors of the packets at the sink node, respectively. This can be done, e.g., by using Gaussian or Gauss-Jordan elimination [47].

Analysis [Outline]:

The packets over the first link are random linear combinations of the message vectors and their global encoding vectors' entries are all independently uniformly distributed (i.u.d.) Bernoulli random variables. Suppose that the receiving node of the first link in the network is to recover the k message vectors after receiving n packets. This can be done so long as there exist k packets at this node with linearly independent global encoding vectors. Now, a question is to find a lower bound on n such that a collection of k such packets exists at this node. This lower bounding problem itself translates into the problem of lower bounding the probability that a set of n vectors of length k whose entries are i.u.d. Bernoulli random variables includes k linearly independent vectors. This can be done by applying a well-known result in the literature, see, e.g., [48, Proposition 2].

For every $1 \leq i < L$, the packets over the $(i+1)^{\text{th}}$ link are also each a random linear combination of the packets previously received over the i^{th} link. The entries of the global encoding vectors of these packets are uniformly distributed Bernoulli random variables but not necessarily independent. To the best of our knowledge, however, there is no non-trivial lower bound on the probability that a set of n such vectors of length k includes k linearly independent vectors.

In the following, we show that the entries of the global encoding vectors of any collection of packets at any given node are i.u.d. Bernoulli random variables so long as the packets' local encoding vectors are linearly independent. This implies that the global encoding vector of any packet over the $(i+1)^{\text{th}}$ link can be written as a linear combination of those global encoding vectors over the i^{th} link whose entries are all i.u.d.. For every $1 \leq i < L$, the main idea in our analysis is to lower bound the size of a maximal collection of packets with linearly independent local encoding vectors over the $(i+1)^{\text{th}}$ link.

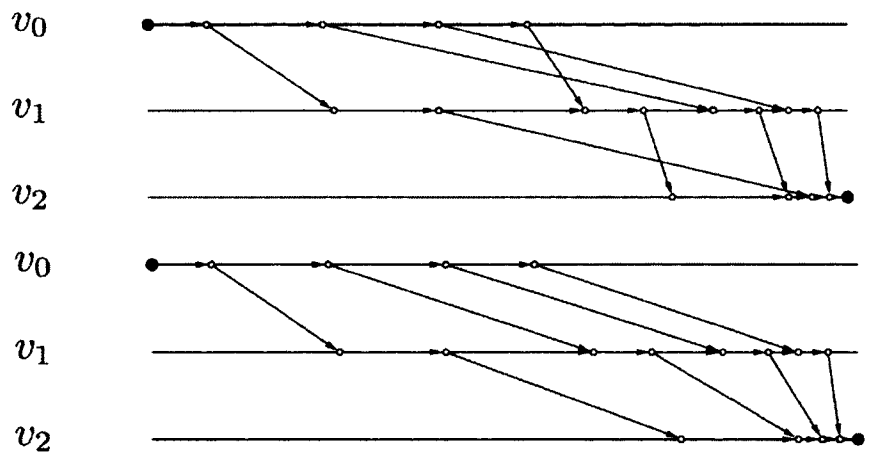

Figure 3.2: Two examples of a network of length 2 with one-in-one-out traffics of capacity 4 (with and without re-ordering of received packets).

The number of packets with linearly independent local encoding vectors in general depends on the traffic. In this chapter, however, the traffic is arbitrary. We hence study two extremal categories of worst-case traffics in which, for every $1 \leq i \leq L$, the probability that the local encoding vector of any packet transmitted over the i^{th} link is linearly independent to those of the packets transmitted earlier over that link is the "smallest" or the "largest" possible.

The first category of worst-case traffics consists of those in which, for every $1 \leq i < L$, the i^{th} node successfully transmits one and only one packet over the $(i+1)^{\text{th}}$ link between any two contiguous arrivals over the i^{th} link. In this case, the smallest number of received packets are available to contribute in generating any packet to be transmitted (see Figure 3.2); and thus the local encoding vector of any given packet is linearly dependent on those of the rest of the packets with the largest probability. We refer to such traffics as *one-in-one-out*.

The second category of worst-case traffics consists of those in which, for every $1 \leq i < L$, the i^{th} node transmits its first successful packet over the $(i+1)^{\text{th}}$ link

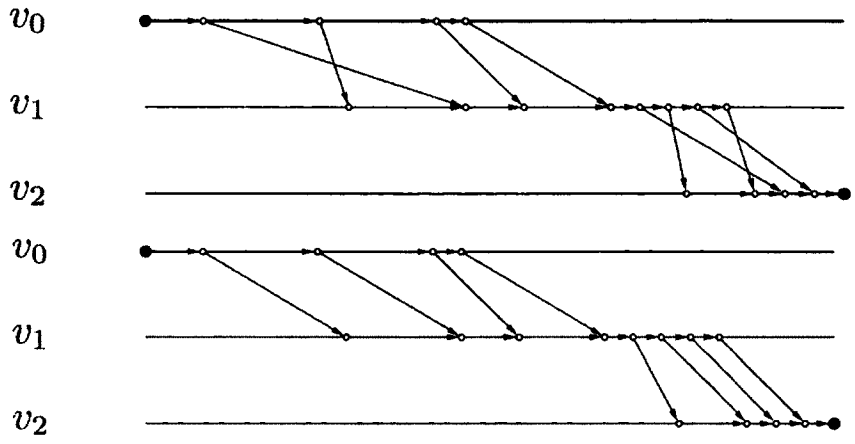

Figure 3.3: Two examples of a network of length 2 with all-in-all-out traffics of capacity 4 (with and without re-ordering of received packets).

after receiving all the n packets over the i^{th} link (see Figure 3.3). We refer to such traffics as *all-in-all-out.*

The local encoding vectors not only depend on the traffic but also are random. The number of linearly independent local encoding vectors over any given link is therefore a random variable. In the following, we derive probabilistic lower bounds on the value of this random variable for each type of worst-case traffics of interest. Taking a union bound over the number of links, we then provide a lower bound on the number of packets with i.u.d. Bernoulli entries over the last link. Finally, by applying a lower bound on the probability of existence of k linearly independent vectors in a set of vectors with i.u.d. Bernoulli entries, we derive a lower bound on the probability that the decoding is successful at the sink node.

Analysis [Details]:

Every node combines its received packets in a random fashion. Intuitively, the likelihood of linear dependence among the global encoding vectors of the packets increases

as they travel down the links from the source node to the sink node.

In the following, we derive a lower bound on the number of received packets by any receiving node such that the node fails to receive k innovative packets w.p. b.a.b. ϵ, for a given $0 < \epsilon < 1$.

For every $1 \leq i \leq L$, let Q_i be a matrix of size $n \times k$ whose rows are the global encoding vectors of the packets received by the i^{th} node, i.e., $\forall \ell \in \mathcal{I}_i$, and $\forall j \in \mathcal{M}$, $(Q_i)_{\ell,j} = \mu_{\ell,j}$. We call Q_i the *decoding matrix* at the i^{th} node. The i^{th} node can recover the set of all the k message vectors so long as Q_i includes k linearly independent rows. Let T be a matrix over $\mathbb{F}_2$. The *rank* of T, denoted by $\text{rank}(T)$, is the size of a maximal collection of linearly independent rows in T over $\mathbb{F}_2$. Thus, the i^{th} node succeeds to recover the k message vectors so long as $\text{rank}(Q_i) = k$. Clearly, $\text{rank}(Q_i)$ depends on the structure of matrix Q_i. We study the structure of matrix Q_i below.

Clearly, the entries of the matrix Q_1 are all i.u.d. (by the structure of a dense coding scheme, all the packets over the first link have (local) global encoding vectors with i.u.d. entries over $\mathbb{F}_2$). However, the scenario is different for the entries of the matrix Q_i, for every $1 < i \leq L$. In particular, for every time τ, and every $e \in \mathcal{I}_i$, the global encoding vector of packet $\boldsymbol{y}_e$, over the in-edge (with label) e of the i^{th} node at time τ, is a function of $\{\mu_{e,j} : j \in \mathcal{M}\}$; and for every $j \in \mathcal{M}$, $\mu_{e,j}$ is itself a function of $\lambda_{e,\ell}\mu_{\ell,j}$'s, for all in-edges ℓ of the $(i-1)^{\text{th}}$ node prior to time τ. Clearly, for all such in-edges ℓ, $\lambda_{e,\ell}$'s are i.u.d. over $\mathbb{F}_2$, and $\forall j \in \mathcal{M}$, $\mu_{\ell,j}$'s are known at the $(i-1)^{\text{th}}$ node, and no longer random. Thus, for all $j \in \mathcal{M}$, $\mu_{e,j}$'s (i.e., the entries of the global encoding vector of the packet over the in-edge e), and thus the entries of Q_i (i.e., the global encoding vectors of all the packets over the i^{th} link) are uniformly but not necessarily independently distributed.

Clearly, for every $1 \leq i \leq L$, $\text{rank}(Q_i)$ is a random variable (due to the randomness in the code and the traffic). We thus need to lower bound the probability that $\text{rank}(Q_i) = k$, i.e., the probability of receiving an innovative collection of size k of packets by the i^{th} node, when n packets are received at the node. While finding such lower bounds is rather simple where Q_i's entries are all i.u.d. (the case in matrix Q_1), the same cannot be said about the cases where Q_i's entries are not necessarily i.u.d.

(the case in matrix Q_i, for every $1 < i \leq L$). Our approach is to derive tight lower bounds by focusing on a subset of packets at the i^{th} node, for every $1 \leq i \leq L$, for which the entries of the global encoding vectors are i.u.d. (Such packets are called the *globally dense packets* at the i^{th} node). To perform this, we remove a minimal collection of rows in Q_i, so that the remaining rows all have i.u.d. entries over $\mathbb{F}_2$. We denote the set of remaining rows by Q'_i; a sub-matrix of Q_i (Then, Q'_i is Q_i restricted to its rows pertaining to the global encoding vectors of the globally dense packets at the i^{th} node.). Clearly, $\text{rank}(Q'_i) \leq \text{rank}(Q_i)$.

Let Q be a matrix over $\mathbb{F}_2$. The *density* of Q, denoted by $\mathcal{D}(Q)$, is the size of a maximal *dense* collection of rows in Q, where a collection of rows is dense if the rows have all i.u.d. entries over $\mathbb{F}_2$. Further, Q is called a *dense matrix* if all its rows form a dense collection.

It is not hard to see that the packets transmitted by the source node are all globally dense, i.e., $\mathcal{D}(Q_1) = n$. The density of the decoding matrices at the other nodes further down in the sequence of network nodes (including the sink node) might however be less than n, i.e., for every $1 < i \leq L$, $\mathcal{D}(Q_i) \leq n$.

Clearly, Q'_i is a dense matrix, and hence $\mathcal{D}(Q_i)$ is equal to the number of rows in Q'_i. Let $\mathcal{D}_i$ denote the set of (labels of) in-edges of the i^{th} node (till the end of transmission time) pertaining to the Q'_i's rows. Let M_i, a sub-matrix of Q_i, denote the collection of rows in Q_i which are not in the dense collection of rows in Q_i; i.e., the rows in M_i are $\{\boldsymbol{\mu}_e : e \in \mathcal{I}_i \setminus \mathcal{D}_i\}$. Let N_i, a sub-matrix of Q_i, denote the minimal collection of rows in $Q_i \setminus M_i$ whose removal would create a sub-matrix with linearly independent rows; i.e., the rows in N_i are $\{\boldsymbol{\mu}_e : e \in \mathcal{D}_i \setminus \boldsymbol{\mathcal{D}}_i\}$, where $\boldsymbol{\mathcal{D}}_i$ is the maximal innovative collection of packets over the in-edges of the i^{th} node (with labels) in the set $\mathcal{D}_i$.

Let $\boldsymbol{\mathcal{I}}_i$ denote the maximal innovative collection of packets in $\mathcal{I}_i$. The number $|\mathcal{I}_i \setminus \boldsymbol{\mathcal{I}}_i|$ of non-innovative packets at the i^{th} node can be bounded from above by the sum of the number of rows in M_i and N_i, i.e., $|\mathcal{I}_i \setminus \mathcal{D}_i| + |\mathcal{D}_i \setminus \boldsymbol{\mathcal{D}}_i|$. The number of rows in M_i and N_i are random variables and we shall give upper-bounds on $|\mathcal{I}_i \setminus \mathcal{D}_i|$ and $|\mathcal{D}_i \setminus \boldsymbol{\mathcal{D}}_i|$ which fail to hold w.p. b.a.b. ϵ.

For every $0 \leq i < L$, let T_i be an $n \times n$ matrix over $\mathbb{F}_2$, whose rows are the local

encoding vectors $\{\boldsymbol{\lambda}_\ell : \ell \in \mathcal{O}_i\}$. For every $j \in \mathcal{I}_i$, $(T_i)_{\ell,j} = \lambda_{\ell,j}$. We call T_i the *transfer matrix* at the i^{th} node. Then, $Q_{i+1} = T_i Q_i$.

The following lemmas provide a lower bound on $\mathcal{D}(Q_{i+1}) = \mathcal{D}(T_i Q_i)$. (The proofs of the results in this chapter can be found in Appendix A)

Lemma 3.1 *Let $\boldsymbol{v}$ be a column-vector of length n^* whose entries are independently and uniformly drawn from $\mathbb{F}_2$, and T be an arbitrary matrix of size $\gamma \times n^*$ ($\gamma \leq n^*$) over $\mathbb{F}_2$ such that* $\text{rank}(T) = \gamma$. *Then, the entries of $T\boldsymbol{v}$ are i.u.d. random variables over $\mathbb{F}_2$.*

It follows from Lemma 3.1 that a set of linear combinations of i.u.d. random variables over $\mathbb{F}_2$ are i.u.d., so long as the coefficient vectors of the linear combinations are linearly independent.

Lemma 3.2 *Let Q be a $n^* \times k^*$ ($k^* \leq n^*$) dense matrix over $\mathbb{F}_2$, and T be a matrix over $\mathbb{F}_2$ with n^* columns. If* $\text{rank}(T) \geq \gamma$, *then $\mathcal{D}(TQ) \geq \gamma$.*

Lemma 3.2 is applicable to lower bound $\mathcal{D}(T_i Q_i)$ if the decoding matrix Q_i at the i^{th} node is dense, i.e., $\mathcal{D}(Q_i) = n$. The density of Q_i might however be less than n, i.e., Q_i might not be dense.

Further, the result of Lemma 3.2 implies that if at a transmitting node of a link, the transfer matrix is not full row-rank, then at the receiving node of that link, the decoding matrix is not dense. Thus, for every $1 \leq i < L$, a packet transmitted by the i^{th} node is in the collection of globally dense packets at the $(i+1)^{\text{th}}$ node if its local encoding vector is linearly independent of those of the rest of globally dense packets at the $(i+1)^{\text{th}}$ node.

Thus, by the above argument, the size of a maximal collection of globally dense packets at a node can be lower bounded by the number of packets with linearly independent local encoding vectors at that node. The packets with linearly independent local encoding vectors are referred to as *LILE packets.* (By the above argument, the set of LILE packets at each node is a subset of the globally dense packets at that

node.[5])

Suppose that $\mathcal{D}(Q_i) = n_i$, or equivalently, $|\mathcal{D}_i| = n_i$. We rewrite T_iQ_i with respect to $\hat{Q}_i$ being Q_i restricted to its rows pertaining to the global encoding vectors of the LILE packets at the i^{th} node. Clearly, $\hat{Q}_i$ is a sub-matrix of Q'_i, and hence $\text{rank}(\hat{Q}_i) \leq \text{rank}(Q'_i) \leq \text{rank}(Q_i)$. We call $\hat{Q}_i$ the ***modified decoding matrix*** at the i^{th} node. This results in $Q_{i+1} = \hat{T}_i\hat{Q}_i$, where $\hat{Q}_i$ is a dense matrix of size $n_i \times k$ over $\mathbb{F}_2$, and $\hat{T}_i$ is a matrix of size $n \times n_i$ over $\mathbb{F}_2$ such that $\forall \ell \in \mathcal{O}_i$, and $\forall j \in \mathcal{D}_i$, $(\hat{T}_i)_{\ell,j} = \lambda_{\ell,j} + \sum_{e \in \mathcal{I}_i \setminus \mathcal{D}_i} \lambda_{\ell,e}\gamma_{e,j}$, and $\{\gamma_{e,j}\}$ are in $\mathbb{F}_2$ satisfying $\sum_{j \in \mathcal{D}_i} \gamma_{e,j}\lambda_{j,\ell} = \lambda_{e,\ell}$, $\forall \ell \in \mathcal{I}_i$. We call $\hat{T}_i$ the ***modified transfer matrix*** at the i^{th} node.

Every row of T_i is the local encoding vector of a successful packet transmitted by the i^{th} node, and every entry of a local encoding vector is either zero or chosen independently and uniformly at random from $\mathbb{F}_2$. Thus, the entries of T_i are each either zero or an i.u.d. random variable over $\mathbb{F}_2$. For every $\ell \in \mathcal{O}_i$, and every $j \in \mathcal{I}_i$, let the sets $\mathcal{T}_{i_{\text{row}}}^{(\ell)}$ and $\mathcal{T}_{i_{\text{col}}}^{(j)}$ denote the label sets of i.u.d. entries in the ℓ^{th} row and the j^{th} column of T_i, respectively.

For worst-case traffics, there are at least ℓ packets received by the i^{th} node by the time that the ℓ^{th} successful packet is to be transmitted. Thus, for every $\ell \in \mathcal{O}_i$, $|\mathcal{T}_{i_{\text{row}}}^{(\ell)}| \geq \ell$, and in particular the first ℓ entries of the ℓ^{th} row of T_i are i.u.d. random variables over $\mathbb{F}_2$. For every $j \in \mathcal{I}_i$, $|\mathcal{T}_{i_{\text{col}}}^{(j)}| \geq n - j + 1$, and in particular the last $n - j + 1$ entries of the j^{th} column of T_i are i.u.d. random variables over $\mathbb{F}_2$.

We now consider the modified transfer matrix $\hat{T}_i$. For every $\ell \in \mathcal{O}_i$, the ℓ^{th} row of $\hat{T}_i$ is representing the labels of LILE packets received by the i^{th} node which contribute to generate the ℓ^{th} successful packet to be transmitted. For every $j \in \mathcal{D}_i$, the j^{th} column of $\hat{T}_i$ is also representing the labels of successful packets in which the j^{th} LILE packet at the i^{th} node contributes. For every $\ell \in \mathcal{O}_i$, and every $j \in \mathcal{D}_i$, $(\hat{T}_i)_{\ell,j}$ is either zero or an i.u.d. random variable over $\mathbb{F}_2$.

[5]One should however note that the local encoding vectors being linearly independent is not a necessary condition for the packets to form a "globally dense" collection. In particular, the collection of all the packets successfully transmitted by the source node is globally dense (by the definition) but some packets in this collection might have local encoding vectors linearly dependent on those of the rest (and hence such packets do not belong to the collection of the LILE packets successfully transmitted by the source node).

For every $\ell \in \mathcal{O}_i$, and every $j \in \mathcal{D}_i$, we use the notations $\hat{\mathcal{T}}_{i_{\text{row}}}^{(\ell)}$ and $\hat{\mathcal{T}}_{i_{\text{col}}}^{(j)}$ to denote the label sets of i.u.d. entries in the ℓ^{th} row and the j^{th} column of $\hat{T}_i$, respectively.

Thus, $|\hat{\mathcal{T}}_{i_{\text{row}}}^{(\ell)}| \geq [\ell - n + n_i]^+$, and in particular the first $[\ell - n + n_i]^+$ entries of the ℓ^{th} row of $\hat{T}_i$ are i.u.d.. Also, $|\hat{\mathcal{T}}_{i_{\text{col}}}^{(j)}| \geq n_i - j + 1$, and in particular the last $n_i - j + 1$ entries of the j^{th} column of $\hat{T}_i$ are i.u.d..

For every $0 \leq i < L$, by using the structure of its randomness, in what follows we derive a (probabilistic) lower bound on the rank of the modified transfer matrix at the i^{th} node. It is worth noting that the rank property of such a matrix is highly dependent on the traffic, and hence the analysis of the two extremal worst-case traffics of our interest are given separately as follows.

One-In-One-Out Worst-Case Traffics:

In this setting, for every $\ell \in \mathcal{O}_i$, $|\mathcal{T}_{i_{\text{row}}}^{(\ell)}| = \ell$, and for every $j \in \mathcal{I}_i$, $|\mathcal{T}_{i_{\text{col}}}^{(j)}| = n - j + 1$. Also, for every $\ell \in \mathcal{O}_i$, $|\hat{\mathcal{T}}_{i_{\text{row}}}^{(\ell)}| \geq [\ell - n + n_i]^+$, and for every $j \in \mathcal{I}_i$, $|\hat{\mathcal{T}}_{i_{\text{col}}}^{(j)}| \geq n_i - j + 1$.[6]

Lemma 3.3 *Let T be an $r^* \times n^*$ ($n^* \leq r^*$) matrix over $\mathbb{F}_2$, such that for any $1 \leq j \leq n^*$, at least $n^* - j + 1$ entries of its j^{th} column are i.u.d. random variables. The rest of the entries are arbitrary. For every integer $0 \leq \gamma \leq n^* - 1$,*

$$\Pr\{\text{rank}(T) < n^* - \gamma\} \leq (n^* - \gamma)2^{-(\gamma+1)}.$$

Lemma 3.3 is applicable to lower bound the rank of $\hat{T}_i$, and since $Q_{i+1} = \hat{T}_i \hat{Q}_i$, and $\hat{Q}_i$ is dense, Lemma 3.2 bounds $\mathcal{D}(Q_{i+1})$ from below by the rank of $\hat{T}_i$. Combining Lemmas 3.2 and 3.3 together, we can hence give a lower bound on $\mathcal{D}(Q_{i+1})$ as follows.

Lemma 3.4 *For every $\epsilon > 0$, by applying a dense code over a line network with any one-in-one-out worst-case traffic of capacity n, for every $0 \leq i < L$, the inequality*

$$\mathcal{D}(Q_{i+1}) \geq \mathcal{D}(Q_i) - \log \mathcal{D}(Q_i) - \log(1/\epsilon)$$

fails w.p. b.a.b. ϵ.

[6]The results for the matrices T_i and $\hat{T}_i$ are consistent, in that the results for T_i are the special case of the results for $\hat{T}_i$, where $n_i = n$.

For every $0 \leq i < L$, $\mathcal{D}(Q_i) - \mathcal{D}(Q_{i+1})$ is called the *density loss* over the $(i+1)^{\text{th}}$ link. Lemma 3.4 gives a (probabilistic) upper bound on the density loss over the $(i+1)^{\text{th}}$ link with respect to $\mathcal{D}(Q_i)$, i.e., w.p. b.a.b. ϵ, $\mathcal{D}(Q_i) - \mathcal{D}(Q_{i+1}) > \log \mathcal{D}(Q_i) + \log(1/\epsilon)$.

The density of the decoding matrix at the sink node, $\mathcal{D}(Q_L)$, can be bounded from below by subtracting the density losses over the network links from the density of the decoding matrix at the first receiving node.

Lemma 3.5 *For every $\epsilon > 0$, by applying a dense code over a line network of length L with any one-in-one-out worst-case traffic of capacity n, the inequality*

$$\mathcal{D}(Q_L) \geq n - L \log(nL/\epsilon)$$

fails w.p. b.a.b. ϵ.

Lemma 3.5 helps us to upper bound the number of packets which are not LILE at the sink node, and such an upper bound serves for the number of rows in the matrix M_L, i.e., $|\mathcal{I}_L \setminus \mathcal{D}_L|$.

By Lemma 3.5, it follows that $\mathcal{D}(Q_L) < k$ w.p. b.a.b. ϵ, so long as $k \leq n - L \log(nL/\epsilon)$. Thus, for every k satisfying this inequality, w.p. no larger than ϵ, Q_L fails to include a $\mathcal{D}(Q_L) \times k$ ($k \leq \mathcal{D}(Q_L)$) dense sub-matrix. Finally, the following lemma gives an upper bound on the probability that a dense matrix fails to have rank k. This helps us to upper bound the number of non-innovative LILE packets at the sink node, and such an upper bound serves for the number of rows in the matrix N_i, i.e., $|\mathcal{D}_L \setminus \boldsymbol{\mathcal{D}}_L|$.

Lemma 3.6 *Let Q be an $n^* \times k^*$ ($k^* \leq n^*$) dense matrix over $\mathbb{F}_2$. Then,*

$$\Pr\{\text{rank}(Q) < k^*\} \leq 2^{-(n^*-k^*)}.$$

Thus, Lemma 3.5 together with Lemma 3.6 give an upper bound on the total number of packets at the sink node not belonging to a maximal innovative collection of packets, i.e., $|\mathcal{I}_L \setminus \boldsymbol{\mathcal{I}}_L|$, by respectively providing upper bounds on the number of

rows in M_L and N_L, i.e., $|\mathcal{I}_L \setminus \mathcal{D}_L|$ and $|\mathcal{D}_L \setminus \boldsymbol{\mathcal{D}}_L|$. This is used in the proof of the following theorem.

Theorem 3.1 *For every $\epsilon > 0$, a dense code fails to transmit k message vectors over a line network of length L with any one-in-one-out traffic of capacity n, w.p. b.a.b. ϵ, so long as*

$$k \leq n - L\log(nL/\epsilon) - \log(1/\epsilon) - L - 1. \tag{3.1}$$

By the result of Theorem 3.1, one can see that $k_{n,\epsilon}$ (the definition of $k_{n,\epsilon}$ is given in Section 2.1.6) is equal to $n - L\log(nL/\epsilon) - \log(1/\epsilon) - L - 1$. Thus, the ratio $(n - k_{n,\epsilon})/n$ goes to 0 (i.e., the dense coding scheme is capacity-achieving over line networks of length L with any one-in-on-out worst-case traffic), as n goes to infinity (in the asymptotic regime), so long as $L\log(nL/\epsilon) = o(n)$. Consider the case in which the number of message vectors k is equal to $k_{n,\epsilon}$ (i.e., the maximum number of message vectors such that a dense coding scheme fails over any such traffic of capacity n w.p. b.a.b. ϵ). Then, it is not hard to see that, in this case, n is equal to $(1 + o(1))k$, as k goes to infinity, and hence the latter condition can be written as $L\log(kL/\epsilon) = o(k)$. The class of dense codes is thus capacity-achieving over (line networks of length L with any traffic in) the class of one-in-one-out worst-case traffics in the asymptotic regime (as k goes to infinity), so long as $L\log(kL/\epsilon) = o(k)$. Further, in this case, a dense code fails (to transmit k message vectors over a line network of length L with any one-in-one-out worst-case traffic) w.p. b.a.b. ϵ, so long as the capacity of the underlying traffic is larger than $k+L\log(kL/\epsilon)+\log(1/\epsilon)+L+1$ (this can be seen by rewriting inequality (3.1) as $n \geq k+L\log(nL/\epsilon)+\log(1/\epsilon)+L+1$, and replacing n with $(1 + o(1))k$).

All-In-All-Out Worst-Case Traffics:

In this setting, for every $\ell \in \mathcal{O}_i$, $|\mathcal{T}^{(\ell)}_{i_{\text{row}}}| = n$, and for every $j \in \mathcal{I}_i$, $|\mathcal{T}^{(j)}_{i_{\text{col}}}| = n$. Similarly, for every $\ell \in \mathcal{O}_i$, $|\hat{\mathcal{T}}^{(\ell)}_{i_{\text{row}}}| = n_i$, and for every $j \in \mathcal{I}_i$, $|\hat{\mathcal{T}}^{(j)}_{i_{\text{col}}}| = n$. Since these conditions on $|\hat{\mathcal{T}}^{(\ell)}_{i_{\text{row}}}|$ and $|\hat{\mathcal{T}}^{(j)}_{i_{\text{col}}}|$ satisfy those required in Lemma 3.3, one can use the result of

Lemma 3.3 to upper bound $\Pr\{\text{rank}(\hat{T}) < n_i - \gamma\}$, for every integer $0 \leq \gamma \leq n_i - 1$. We, however, derive a tighter bound for this setting in Lemma 3.7.[7]

Lemma 3.7 *Let T be an $r^* \times n^*$ ($n^* \leq r^*$) dense matrix over $\mathbb{F}_2$. For every integer $1 \leq \gamma \leq n^* - 1$,*

$$\Pr\{\text{rank}(T) < n^* - \gamma\} \leq 2^{-\gamma}.$$

We, now, give a lower bound on $\mathcal{D}(Q_{i+1}) = \mathcal{D}(\hat{T}_i \hat{Q}_i)$ by using Lemmas 3.2 and 3.7.

Lemma 3.8 *For every $\epsilon > 0$, by applying a dense code over a line network with any all-in-all-out worst-case traffic of capacity n, for every $0 \leq i < L$, the inequality*

$$\mathcal{D}(Q_{i+1}) \geq \mathcal{D}(Q_i) - \log(1/\epsilon)$$

fails to hold w.p. b.a.b. ϵ.

Taking a union bound over the number of network links, and subtracting the density losses over the network links from the density of the decoding matrix at the first receiving node, a lower bound can be given on the density of the decoding matrix at the sink node; $\mathcal{D}(Q_L)$. The proof of the following is similar to that of Lemma 3.5, except that we use the result of Lemma 3.8 instead of Lemma 3.4.

Lemma 3.9 *For every $\epsilon > 0$, by applying a dense code over a line network of length L with any all-in-all-out worst-case traffic of capacity n, the inequality*

$$\mathcal{D}(Q_L) \geq n - L\log(L/\epsilon)$$

fails w.p. b.a.b. ϵ.

[7] It is worth noting that for the choice of $\gamma = 0$, Lemma 3.6 is also applicable, and in this case the result of Lemma 3.7 is not as tight as that of Lemma 3.6. However, Lemma 3.6 is not generalizable to the other choices of $\gamma > 0$, and hence Lemma 3.7 can be considered as a complement to Lemma 3.6.

Applying Lemmas 3.6 and 3.9 yields the following main result. The proof is similar to that of Theorem 3.1, and is thus omitted.

Theorem 3.2 *For every $\epsilon > 0$, a dense code fails to transmit k message vectors over a line network of length L with any all-in-all-out worst-case traffic of capacity n, w.p. no larger than ϵ, so long as*

$$k \leq n - L\log(L/\epsilon) - \log(1/\epsilon) - L - 1.$$

Similar to that by the result of Theorem 3.1 for the case of one-in-one-out worst-case traffics, by the result of Theorem 3.2, one can see that the class of dense codes is capacity-achieving over (line networks with any traffic in) the class of all-in-all-out worst-case traffics in the asymptotic regime (as k goes to infinity), so long as $L\log(L/\epsilon) = o(k)$. Also, a dense code fails (to transmit k message vectors over a line network of length L with any all-in-all-out worst-case traffic) w.p. b.a.b. ϵ, so long as the capacity of the underlying traffic is larger than $k + L\log(L/\epsilon) + \log(1/\epsilon) + L + 1$.

Coding Cost:

The worst case with regards to the encoding cost at the i^{th} node, for every $1 \leq i < L$, occurs if for every time τ at which (and every out-edge e over which) the i^{th} node transmits a packet, $|\mathcal{I}_i^{(\tau)}| = n$, and for all $j \in \mathcal{I}_i^{(\tau)}$, $\lambda_{e,j}$'s are chosen to be nonzero. In the case of the source node, the worst case with regards to the encoding cost occurs if for every time τ at which (and every out-edge e over which) the source node transmits a packet, for all $j \in \mathcal{M}$, $\mu_{e,j}$'s are chosen to be nonzero. Thus the number of packet operations for encoding at each internal node (or the source node) is $O(n^2)$ (or $O(kn)$). Thus the encoding cost of a dense code to transmit k message vectors in the underlying scenario is $O(k)$.[8]

To solve the system of linear equations at the sink node using Gaussian or Gauss-Jordan elimination, the sink node requires $O(w_{\max}n)$ row operations, where $w_{\max}$ is

[8]The number of packet operations for encoding is $O((L-1)n^2) + O(nk)$, and hence the encoding cost is $O((L-1)n^2/kL) + O(n/L) = O(k)$, since, $n = (1 + o(1))k$ in a capacity-achieving scenario, as k goes to infinity.

the bandwidth of a row vector in Q_L with the widest band.[9] Each row operation is equivalent to $O(k)$ field operations along with one packet operation. Thus the sink node requires $O(w_{\max}n)$ packet operations together with $O(w_{\max}kn)$ field operations [28].

The worst case regarding the decoding cost at the sink node occurs if $w_{\max}$ is equal to k. Thus the number of operations for decoding is $O(k^2n)$ field operations and $O(kn)$ packet operations. Thus the decoding cost of a dense code to transmit k message vectors in the underlying scenario is also $O(k)$.

Theorem 3.3 *The encoding and decoding costs of a dense code to transmit k message vectors over a line network with any worst-case traffic are $O(k)$.*

3.3.2 From Dense Codes to Chunked Codes

By putting the results of Theorems 3.1, 3.2 and 3.3 together, one can see that the class of dense codes is capacity-achieving over line networks under traffics with arbitrary deterministic transmission, loss and delay models (Theorems 3.1 and 3.2) but at the cost of large computational complexity (Theorem 3.3). Now, a question is to devise codes with lower complexity that are capacity-achieving, or, at the very least, are capacity-approaching with an arbitrarily small nonzero constant gap, over line networks with traffics as above.

One approach to reduce the coding cost is to modify the method of generating coded packets such that the global encoding vectors have smaller bandwidth. To have global encoding vectors with smaller bandwidth, one technique introduced earlier in the literature is to apply a dense code to a *chunk*, a smaller sub-message of the original message. To be more specific, for the so-called *chunked codes* (CC) [24], the set of message vectors is partitioned into chunks of equal size, and each chunk is transmitted by a dense code. The smaller is the size of chunks, the smaller is the bandwidth of the global encoding vectors of the packets. As a consequence, however, the smaller is the randomness in the global encoding vector of each packet and the

[9]The *band* of a vector is the narrowest window (in an end-around fashion) within which the nonzero entries of a vector lie. The length of a band is called *bandwidth*.

larger is its probability to be linearly dependent on the global encoding vectors of the other packets sharing the same band. Thus the probability of each packet being innovative becomes smaller in general. The problem is therefore to design chunk-based codes in which every global encoding vector has a small bandwidth but the bands are set up in a way to compensate for the reduction in the randomness of the global encoding vectors. Thus, in a more general context, the design of a chunk-based code has to deal with the following issues: (i) how to divide the message vectors into the chunks at the source node, (ii) how to schedule the chunks to be coded and transmitted by the source node and the internal nodes, and (iii) how to recover the chunks at the sink node.

In the following, we first review (and give a tighter and a more complete analysis of) the CC scheme of [24], which provably performs well over line networks with arbitrary (deterministic) traffics. Next, we propose a new chunk-based coding scheme with overlapping chunks, and analyze its performance over the network scenarios as above. Our results demonstrate that the proposed chunked coding scheme provides a better complexity-performance tradeoff in such scenarios compared to the original chunked codes with disjoint chunks, for some ranges of the chunk sizes.

3.3.3 Chunked Codes

A CC operates by dividing the set of k message vectors at the source node into q (a divisor of k) disjoint subsets, called *chunks*, each of size k/q, such that the chunk ω, for some $1 \leq \omega \leq q$, includes the last k/q contiguous message vectors with labels less than or equal to $\omega k/q$.

Encoding:

For every $0 \leq i < L$, the i^{th} node at any time τ randomly chooses a chunk and transmits it by using a dense code, i.e., the i^{th} node constructs a coded packet by randomly linearly combining all its previously received packets pertaining to the chosen chunk, and transmits the coded packet over its out-edge at time τ in the trellis if such an out-edge exists.

For any given chunk ω, let $\mathcal{I}_i^{(\tau)}(\omega)$ ($\mathcal{O}_i^{(\tau)}(\omega)$) be the set of labels of in-edges (out-edges) of the i^{th} node prior to time τ over which packets pertaining to chunk ω are received (transmitted), and let $\mathcal{M}(\omega)$ be the set of labels of message vectors in chunk ω. Let $\mathcal{I}_i(\omega)$ ($\mathcal{O}_i(\omega)$) denote the set $\mathcal{I}_i^{(\infty)}(\omega)$ ($\mathcal{O}_i^{(\infty)}(\omega)$).

Therefore, for every out-edge e of the i^{th} node at time τ in the trellis, a chunk is randomly chosen, say ω, and a packet $\boldsymbol{y}_e$ pertaining to the chosen chunk is transmitted such that $\boldsymbol{y}_e = \sum_{\ell \in \mathcal{I}_i(\omega)} \lambda_{e,\ell} \boldsymbol{y}_\ell$ if the i^{th} node is an internal node, and $\boldsymbol{y}_e = \sum_{j \in \mathcal{M}(\omega)} \mu_{e,j} \boldsymbol{x}_j$ if the i^{th} node is the source node, where for every $\ell \in \mathcal{I}_i(\omega) \setminus \mathcal{I}_i^{(\tau)}(\omega)$, $\lambda_{e,\ell}$ is 0; and for every $\ell \in \mathcal{I}_i^{(\tau)}(\omega)$, and every $j \in \mathcal{M}(\omega)$, $\lambda_{e,\ell}$ and $\mu_{e,j}$ are symbols independently and uniformly drawn from $\mathbb{F}_2$.

Decoding:

For any given chunk ω, the sink node has to solve a system of linear equations $\{\boldsymbol{y}_e = \sum_{j \in \mathcal{M}(\omega)} \mu_{e,j} \hat{\boldsymbol{x}}_j : e \in \mathcal{I}_L(\omega)\}$ for the k/q unknown packets $\{\hat{\boldsymbol{x}}_j : j \in \mathcal{M}(\omega)\}$.

The system is uniquely solvable for any given chunk (ω), and for every $j \in \mathcal{M}(\omega)$, $\hat{\boldsymbol{x}}_j$ equals $\boldsymbol{x}_j$ if there exists an innovative[10] collection of k/q packets pertaining to chunk ω at the sink node. Thus, a CC succeeds if the sink node receives a collection of k/q packets (pertaining to every chunk) which form an innovative collection.[11]

Let Q_i be the decoding matrix at the i^{th} node. For every ω, let $Q_i(\omega)$ be Q_i restricted to its rows pertaining to the global encoding vectors of the packets pertaining to chunk ω. Then, $Q_i(\omega)$ is a sub-matrix of Q_i of size $|\mathcal{I}_i(\omega)| \times |\mathcal{M}(\omega)|$ whose rows are the global encoding vectors of the packets pertaining to chunk ω at the i^{th} node, and whose columns are those with labels of the message vectors pertaining to chunk ω. There exists an innovative collection of k packets at the i^{th} node, iff, for every ω, there exists a collection of k/q linearly independent rows in $Q_i(\omega)$, i.e., $\text{rank}(Q_i(\omega)) = k/q$.

[10]We say that a (sub-) collection of packets pertaining to a given chunk, say ω, in this collection is *innovative* if the global encoding vectors of the packets therein are linearly independent.

[11]For every chunk ω, and every $j \in \mathcal{M} \setminus \mathcal{M}(\omega)$, the j^{th} element in the global encoding vector of each packet pertaining to chunk ω is 0. Thus a collection of packets is *innovative* if and only if (iff) the maximal sub-collection of packets therein pertaining to each chunk is innovative.

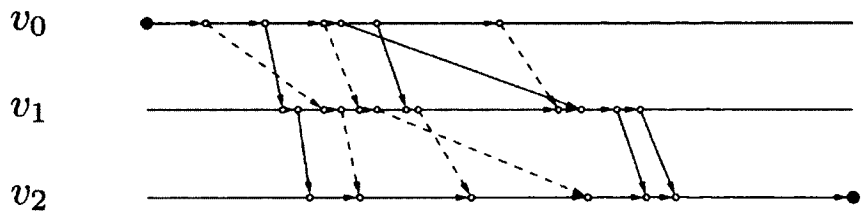

Figure 3.4: An example of a network of length 2 with a traffic of capacity 6. Consider a CC with two chunks ω_1, and ω_2. The solid (dashed) lines are the edges over which the packets pertaining to the chunk ω_1 (or the chunk ω_2) are transmitted. The capacity of the flow by the packets pertaining to the chunk ω_1 is 3, and that by the packets pertaining to the chunk ω_2 is 2.

Analysis:

We need to derive a lower bound on the probability that $\text{rank}(Q_i(\omega)) = k/q$, for every chunk ω, i.e., the probability of receiving an innovative collection of k/q packets pertaining to every chunk by the i^{th} node when a collection of n packets is received by that node.

Each chunk has size k/q and is transmitted by a dense code, and hence, for any given chunk, we can use the result of Theorem 3.1 or 3.2 in the case of each type of extremal worst-case traffics, by replacing k and n, respectively, with k/q and the capacity of the traffic by the flow of the packets pertaining to that chunk. The capacity of the traffic for any given chunk is a random variable. This capacity is equal to the number of paths between the two nodes $v_0^{(0)}$ and $v_L^{(\infty)}$ (in the trellis) whose (traffic) edges, which carry packets pertaining to the given chunk, are disjoint. Such paths are referred to as ***flow paths*** (see Figure 3.4).

To deal with the randomness in the capacity of the traffic for different chunks, for any given chunk, we derive an upper bound on the number of packets (pertaining to that chunk) received by any internal node that cannot be "matched up" with a packet (pertaining to the same chunk) which is transmitted subsequently by that node and is not yet coupled with a packet (pertaining to the same chunk) received earlier by the node. For any given chunk, those packets received by the node whose

(traffic) edges do not contribute in a maximal collection of flow paths are called *unusable*, and the rest of the packets (pertaining to that chunk) are called *usable*. The term "unusable" reflects the fact that these packets are not part of the flow paths that contribute to the capacity of the traffic pertaining to the given chunk.

One-In-One-Out Worst-Case Traffics:

For any given chunk, we take a union bound over internal nodes by subtracting the number of unusable packets pertaining to that chunk received by every node from the number of usable packets pertaining to the same chunk transmitted by the source node. This allows us to derive a lower bound on the number of usable packets pertaining to the given chunk at the sink node yielding a lower bound on the capacity of the traffic pertaining to that chunk.[12]

Lemma 3.10 *By applying a CC with q chunks, over a line network of length L with any one-in-one-out worst-case traffic of capacity n, the capacity of the traffic pertaining to every chunk is not larger than*

$$\left(1 - O\left(\left(L^3(q/n)\log(Ln/\epsilon)\right)^{\frac{1}{3}}\right)\right).(n/q) \tag{3.2}$$

w.p. b.a.b. ϵ, so long as

$$L^3 q \log\frac{Ln}{\epsilon} = o(n). \tag{3.3}$$

We, now, derive a lower bound on the capacity of traffic, n, such that w.p. b.a.b. ϵ, for at least one chunk, there is not an innovative collection of k/q packets at the sink node. For this, we use Theorem 3.1. To modify the result of Theorem 3.1 to be applicable to the transmission of a given chunk over a line network by a dense code, we replace k and n with k/q and the lower bound given in (3.2), respectively. Then the following is immediate.

[12]It is noteworthy that Lemma 3.10 is different from [24, Theorem 4.1] in the sense that the lower bound derived here is tighter, though the proofs have generally a similar structure.

Lemma 3.11 *For every $\epsilon > 0$, by applying a CC with q chunks to transmit k message vectors over a line network of length L with any one-in-one-out worst-case traffic of capacity n, the sink node fails to decode any given chunk w.p. b.a.b. ϵ/q, so long as*

$$k/q \leq \varphi_n - L\log(nL/\epsilon) - \log(q/\epsilon) - L - 1,$$

and $L^3 q \log(nL/\epsilon) = o(n)$, where φ_n is the capacity of the traffic for that chunk.

The decoding at the sink node is successful iff the sink node can decode every chunk. Taking a union bound over the number of chunks (q), we upper bound the number of message vectors (k) at the source node, such that w.p. b.a.b. ϵ, for at least one chunk, either there does not exist at least $n/q - O\left((L^3(n/q)^2\log(Ln/\epsilon))^{\frac{1}{3}}\right)$ (usable) packets at the sink node, or there do not exist k/q packets (in the set of usable packets at the sink node) which form an innovative collection. This yields the following for CC over line networks with one-in-one-out worst-case traffics.

Theorem 3.4 *For every $\epsilon > 0$, a CC with q chunks fails to transmit k message vectors over a line network of length L with any one-in-one-out worst-case traffic of capacity n, w.p. b.a.b. ϵ, so long as*

$$k \leq q\varphi - qL\log(nL/\dot{\epsilon}) - q\log(q/\dot{\epsilon}) - qL - q,$$

and $L^3 q \log(nL/\epsilon) = o(n)$, where

$$\varphi = n/q - O\left((L^3(n/q)^2\log(Ln/\epsilon))^{\frac{1}{3}}\right).$$

All-In-All-Out Worst-Case Traffics:

By the nature of the all-in-all-out traffics, it is easy to see that for every chunk, the capacity of the traffic for the packets pertaining to a chunk is equal to the minimum number of packets pertaining to that chunk transmitted over all the links in the network. Using this, the following is immediate.

Lemma 3.12 *By applying a CC with q chunks, over a line network of length L with any all-in-all-out worst-case traffic of capacity n, the capacity of the traffic pertaining to every chunk is not larger than*

$$\left(1 - O\left(((q/n)\log(Lq/\epsilon))^{\frac{1}{2}}\right)\right).(n/q) \tag{3.4}$$

w.p. b.a.b. ϵ, so long as

$$q \log \frac{Lq}{\epsilon} = o(n). \tag{3.5}$$

The proofs of the following results are similar to that of Lemma 3.11 and Theorem 3.4, except that in this case, we use Lemma 3.12 and Theorem 3.2 instead of Lemma 3.10 and Theorem 3.1, respectively.

Lemma 3.13 *For every $\epsilon > 0$, by applying a CC with q chunks to transmit k message vectors over a line network of length L with any all-in-all-out worst-case traffic of capacity n, the sink node fails to decode any given chunk w.p. b.a.b. ϵ/q, so long as*

$$k/q \leq \varphi_n - L\log(Lq/\epsilon) - \log(q/\epsilon) - L - 1,$$

and $q\log(Lq/\epsilon) = o(n)$, where φ_n is the capacity of the traffic for that chunk.

Theorem 3.5 *For every $\epsilon > 0$, a CC with q chunks fails to transmit k message vectors over a line network of length L with any all-in-all-out worst-case traffic of capacity n, w.p. b.a.b. ϵ, so long as*

$$k \leq q\varphi - qL\log(Lq/\dot{\epsilon}) - q\log(q/\dot{\epsilon}) - qL - q,$$

and $Lq\log(Lq/\epsilon) = o(n)$, where

$$\varphi = n/q - O\left(((n/q)\log(Lq/\epsilon))^{\frac{1}{2}}\right).$$

Coding Cost:

The worst-case with regards to the encoding cost occurs if, for every chunk, every internal node (or the source node) transmits all its successful packets for that chunk

after receiving all the packets for that chunk transmitted by the node in the upper layer, and if all the local encoding vectors' coefficients are chosen to be nonzero. Thus the number of packet operations for encoding at each internal node (or the source node) is $O(n^2/q)$ (or $O(kn/q)$). (For every chunk, there are $O(n/q)$ packets transmitted and received by each internal node, and there are $O(n/q)$ packets transmitted by the source node.). Thus the encoding cost of a CC with q chunks to transmit k message vectors in the underlying scenario is $O(k/q)$.[13]

To solve the system of linear equations at the sink node for every chunk (of size k/q) using Gaussian or Gauss-Jordan elimination, the sink node requires $O(kn/q)$ row operations (the band of each row in the matrix $Q_L(\omega)$, for every ω, is not wider than k/q). Thus, the sink node requires $O(k^2n/q^3)$ field operations along with $O(kn/q^2)$ packet operations (the size of the matrix $Q_L(\omega)$ is $O(n/q) \times k/q)$. The total number of operations for decoding at the sink node is $O(k^2n/q^2)$ field operations and $O(kn/q)$ packet operations, and thus the decoding cost of a CC with q chunks to transmit k message vectors in the underlying scenario is also $O(k/q)$.

Theorem 3.6 *The encoding and decoding costs of a CC with q chunks to transmit k message vectors over a line network with any worst-case traffic are $O(k/q)$.*

3.3.4 Comparison of Dense Codes and Chunked Codes: Making the Case for Overlapping Chunks

Comparing the results of Theorems 3.1 and 3.2 with Theorems 3.4 and 3.5, respectively, one can observe that, for each type of worst-case traffics, the ratio $(n - k_{n,\epsilon})/n$ in the case of a CC with q chunks converges to 0, as n goes to infinity, q times slower than that in the case of dense codes. This is the price that one has to pay for the smaller coding cost of CC. Now, the question is whether there are any chunk-based codes with better tradeoff between the speed of convergence to the capacity and the coding cost compared to CC.

[13]The number of packet operations for encoding is $O((L-1)n^2/q) + O(kn/q)$, and hence the encoding cost is $O((L-1)n^2/qkL) + O(n/qL) = O(k/q)$, since $n = (1 + o(1))k$ in a capacity-achieving scenario, as k goes to infinity.

The analysis of CC over line networks with worst-case traffics shows that the performance of a CC is affected by (i) the capacity of the traffic pertaining to each chunk; (ii) the number of innovative packets in the collection of LILE packets, and (iii) the condition of decoding completion, i.e., for every chunk, the sink node has to receive a number of innovative packets equal to the size of the chunks.

In the scenario of interest in this chapter, where the network nodes are blind to the traffic, and there is no a priori or a posteriori knowledge about the traffic at the network nodes, random coding appears to be the best strategy as far as issues (i) and (ii) are concerned. There is however room for improvement in the performance of chunk-based codes by modifying the chunking scheme to speed up the decoding process. The main idea is to devise a chunking scheme such that the sink node does not necessarily have to receive a number of innovative packets equal to the size of the chunks for every chunk to ensure that all the chunks are decodable. We demonstrate that this can be achieved by allowing chunks to overlap.

To explain the idea, we start by the simple case of a single "erasure channel" with an arbitrary traffic. It is important to note that in this case, the lack of internal network nodes significantly simplifies the analysis as one does not need to consider density losses over the network links, i.e., all the received packets at the sink node are globally dense.[14] The following results are simple to prove.

Theorem 3.7 *For every $\epsilon > 0$, a dense code fails to transmit k message vectors over an erasure channel with any arbitrary traffic of capacity n w.p. b.a.b. ϵ, so long as $k \leq n - \log(1/\epsilon)$. The encoding and decoding costs of such a code in this scenario are $O(k)$.*

Theorem 3.8 *For every $\epsilon > 0$, a CC with q chunks fails to transmit k message vectors over an erasure channel with any arbitrary traffic of capacity n w.p. b.a.b. ϵ, so long as $k \leq q\varphi - q\log(q/\dot{\epsilon}) - q$, where $\varphi = n/q - O\left(((n/q)\log(q/\epsilon))^{\frac{1}{2}}\right)$. The encoding and decoding costs of such a code in this scenario are $O(k/q)$.*

[14] It is worth noting that for a single erasure channel with an arbitrary traffic, a CC performs better than what was presented in [24], in terms of the tradeoff between the speed of convergence to the capacity and the coding cost.

The comparison of the results of Theorems 3.7 and 3.8 shows that for a single erasure channel also, CC have a slower convergence to capacity than dense codes. This is the cost for having a smaller coding cost.

Recently, Studholme and Blake [28] introduced a class of erasure codes, called *windowed erasure codes*, which is similar to CC except that for windowed erasure codes, the chunks are allowed to overlap. These codes are used to deliver k message vectors over an erasure channel with any arbitrary traffic similar to those of our interest in this chapter. To perform this task, windowed erasure codes operate on k chunks of size α, where any two contiguous chunks overlap in all but one message vector in an end-around fashion. Similar to CC, in windowed erasure codes, the chunks are scheduled at random, i.e., the source node at any time instant randomly chooses a chunk, constructs and transmits a coded packet by randomly linearly combining the message vectors in the chosen chunk. The decoding of windowed erasure codes is, however, similar to that of dense codes, not that of CC, i.e., the sink node has to solve the system of linear equations for all the chunks together.

The following theorem shows that windowed erasure codes can achieve the capacity of erasure channels with arbitrary traffics as fast as dense codes do.

Theorem 3.9 *For every $\epsilon > 0$, a windowed erasure code with any chunk size $\alpha \geq 2\sqrt{k}$ fails to transmit k message vectors over an erasure channel with any arbitrary traffic of capacity n w.p. b.a.b. ϵ, so long as $k \leq n - \log(1/\epsilon)$. The encoding and decoding costs of such a code in this scenario are also $O(\alpha)$.*

Prior to giving the proof of Theorem 3.9, let us introduce four types of structured random matrices, called collectively *random banded* (RB) matrices.

Random Banded Matrices:

Let n^*, k^*, α^* and γ_o^* be integers ($k^* \leq n^*$, $\gamma_o^* < \alpha^*$), such that $\alpha^* - \gamma_o^*$ is a divisor of k^*. Let χ^* be $k^*/(\alpha^* - \gamma_o^*)$, and I^* be the set of integers $\{i\}_{1 \leq i \leq k^*}$. We divide I^* into χ^* subsets I_i^*'s, for all $1 \leq i \leq \chi^*$, where I_i^*, called the i^{th} *aperture* of size α^*, is the set of α^* contiguous integers in I^* in an end-around fashion, starting from $(i-1)(\alpha^* - \gamma_o^*) + 1$.

We construct an $n^* \times k^*$ matrix as follows: (i) for each row, an index, say i, is randomly chosen from the set of integers $\{i\}_{1 \le i \le \chi^*}$, and (ii) the row's entries indexed by the i^{th} aperture are independently and uniformly chosen from $\mathbb{F}_2$, and the rest of the entries are set to zero. We call such a matrix a (γ_o^*, α^*) *irregular symmetric random banded* (ISRB) matrix of size $n^* \times k^*$. Now, consider a similar construction, with the difference that $\alpha^* - \gamma_o^*$ is now a divisor of $k^* - \gamma_o^*$ (not k^*), and χ^* is $(k^* - \gamma_o^*)/(\alpha^* - \gamma^*)$ (not $k^*/(\alpha^* - \gamma_o^*)$). The resulting matrix in this case is called a (γ_o^*, α^*) *irregular asymmetric random banded* (IARB) matrix of size $n^* \times k^*$. Also, consider a matrix constructed as each of the above two procedures, except that in part (i), each index in the set $\{i\}_{1 \le i \le \chi^*}$ is assigned to n^*/χ^* rows (χ^* has to be a divisor of n^* in this case). We call such a matrix a (γ_o^*, α^*) *regular symmetric/asymmetric random banded* (RSRB or RARB) matrix of size $n^* \times k^*$.

Proof of Theorem 3.9: Each packet received by the sink node pertains to a randomly chosen chunk. Each chunk contains a set of α contiguous message vectors in an end-around fashion with labels from the set of integers $\{i\}_{1 \le i \le k}$. Each packet's global encoding vector has i.u.d. random entries over $\mathbb{F}_2$ in the positions indexed by the aperture pertaining to the chosen chunk, and the rest of the entries are zero. Thus the decoding matrix Q_L at the sink node is an $(\alpha - 1, \alpha)$ ISRB matrix of size $n \times k$. The decoding at the sink node is successful iff Q_L has rank k. The following result which is a short version of [28, Conjecture 4.2] is then useful to upper bound the probability that Q_L (with an RB structure) fails to have rank k.

Conjecture 3.1 *Let Q be an $(\alpha^* - 1, \alpha^*)$ (regular/irregular) symmetric banded random matrix of size $n^* \times k^*$ ($k^* \le n^*$). For sufficiently large k^*,*

$$\Pr\{\text{rank}(Q) < k^*\} \le 2^{-(n^* - k^*)},$$

so long as $\alpha^* \ge 2\sqrt{k^*}$.

By Conjecture 3.1, the probability of failure of a windowed erasure code is b.a.b. ϵ, i.e., $\Pr\{\text{rank}(Q_L) < k\} \le \epsilon$, so long as $k \le n - \log(1/\epsilon)$. This completes the proof of the first part of the theorem. We prove the second part regarding the

coding cost in two steps. The proof that the encoding cost is $O(\alpha)$ is similar to that of Theorem 3.6. To prove that the decoding cost is $O(\alpha)$, it suffices to recall that applying Gaussian or Gauss-Jordan elimination to the decoding matrix Q_L of bandwidth of α, the sink node requires $O(\alpha n)$ row (or packet) operations. Thus the decoding cost is $O(\alpha)$, since $n = (1 + o(1))k$ in a capacity-achieving scenario, as k goes to infinity. □

The comparison of the results of Theorem 3.9 with those of Theorem 3.7 indicates that for the transmission over a single erasure channel, windowed erasure codes with sufficiently large chunks achieve the capacity as fast as dense codes but with a smaller coding cost. This motivates the application of chunked codes with overlapping chunks, referred to as *overlapped chunked codes* (OCC), to the problem of information transmission over erasure line networks with arbitrary traffics.

3.3.5 Overlapped Chunked Codes

Consider the set of k message vectors $\{\boldsymbol{x}_i : 1 \leq i \leq k\}$. An OCC operates by dividing the set of k message vectors into q overlapping chunks, each of size α, such that any two contiguous chunks overlap by $\gamma_o = \alpha - k/q$ message vectors in an end-around fashion. For every chunk ω, the set of labels of the message vectors in chunk ω, denoted by $\mathcal{M}(\omega)$, is called the *aperture* of chunk ω.

To ensure that all the message vectors appear in the same number of chunks, $(\alpha - \gamma_o)$ must be a divisor of α. For the simplicity of exposition, we consider $\gamma_o = \alpha/\tau_e$, where $\tau_e = \tau_o/(\tau_o - 1)$, for any divisor τ_o of α $(1 \leq \tau_o \leq \alpha)$.[15] For instance, there is no overlap between chunks when τ_o is equal to 1 (similar to CC of [24]); and the overlap becomes larger as τ_o increases; namely, when τ_o is equal to α, any two contiguous chunks overlap in all but one message vector (similar to the codes of [28]). We call τ_o the *overlap parameter.*

[15]The parameter τ, defined as the overlap parameter here, should not be mistaken with the same notation for the transmission time instances used earlier.

Encoding:

The encoding is performed similar to CC; i.e., for every $0 \leq i < L$, the i^{th} node at any time instant τ chooses a chunk at random and constructs a coded packet by randomly linearly combining its all previously received packets pertaining to the chosen chunk, and transmits the coded packet over its out-edge at time τ in the trellis if such an out-edge exists.

Decoding:

The decoding is performed similar to dense codes; i.e., the sink node has to solve a system of linear equations $\{\boldsymbol{y}_e = \sum_{j \in \mathcal{M}} \mu_{e,j} \hat{\boldsymbol{x}}_j : e \in \mathcal{I}_L\}$ for the k unknown packets $\{\hat{\boldsymbol{x}}_j : j \in \mathcal{M}\}$. The system is uniquely solvable, and for every $j \in \mathcal{M}$, $\hat{\boldsymbol{x}}_j$ is equal to $\boldsymbol{x}_j$, iff there exists an innovative collection of k packets at the sink node.[16] However, for OCC, unlike CC, the packets pertaining to different chunks might not be linearly independent due to the overlaps. This implies that the innovation of a packet pertaining to any given chunk at the sink node depends on all the packets pertaining to all the chunks at the sink node. Thus, an OCC succeeds if the sink node receives a collection of k packets which form an innovative collection, regardless of the number of packets pertaining to each chunk.

Let Q_L be the decoding matrix at the sink node. There exists an innovative collection of k packets at the sink node iff there exists a collection of k linearly independent rows in Q_L, i.e., $\text{rank}(Q_L) = k$.

Analysis:

We analyze OCC over the two extremal types of worst-case traffics considered in Sections 3.3.1 and 3.3.3.

[16] It is worth noting that in [32] and [34], the chunks were considered to be decoded in isolation. However, by performing the decoding algorithm on the set of all the chunks simultaneously (similar to [28]), the decoding of the OCC might be successful even when none of the chunks are decodable in isolation. Thus, a smaller number of packets at the sink node is sufficient to ensure successful decoding with a given probability of success.

One-In-One-Out Worst-Case Traffics:

The chunks are scheduled in OCC similar to CC. Thus, w.p. b.a.b. ϵ, the capacity of the flow by the packets pertaining to every chunk fails to be larger than the lower bound given (3.2) in Lemma 3.10. However, for successful decoding, unlike CC, in OCC, a given chunk does not have to be necessarily decodable in isolation (i.e., the rank of the decoding matrix at the sink node pertaining to a given chunk does not need to be α). This is because the decoding is performed on the set of all the packets at the sink node. The goal is to derive an upper bound on the number of packets at the sink node (n) (i.e., the capacity of the traffic in the case of worst-case traffics of capacity n), such that the rank of the decoding matrix at the sink node fails to be equal to the number of message vectors (k) at the source node, w.p. b.a.b. ϵ.

For a given n, we first lower bound the size of a dense collection of packets pertaining to each chunk at the sink node such that for all chunks, the lower bounds fail to hold w.p. b.a.b. $\dot{\epsilon}$.[17] We then derive an upper bound on n, such that, w.p. b.a.b. $\dot{\epsilon}$, there does not exist an innovative collection of k packets in the union set of maximal collections of LILE packets pertaining to at least one chunk.

Lemma 3.14 *For every $\epsilon > 0$, by applying a dense code to a given chunk over a line network of length L with any one-in-one-out worst-case traffic of capacity n, the number of LILE packets pertaining to that chunk at the sink node is not larger than $\varphi_n - L\log(\varphi_n L/\epsilon)$ w.p. b.a.b. ϵ, where φ_n is the capacity of the flow by the packets pertaining to that chunk.*

Lemma 3.15 *Let L, n, q, φ_n and ϵ be defined as above. Then the sink node fails to receive at least $\varphi_n - L\log(q\varphi_n L/\dot{\epsilon})$ LILE packets pertaining to every chunk, w.p. b.a.b. $\dot{\epsilon}$, and φ_n fails to be at least*

$$\left(1 - O\left(\left(L^3(q/n)\log(Ln/\epsilon)\right)^{\frac{1}{3}}\right)\right).(n/q),$$

[17]Note that a collection of packets pertaining to a given chunk is dense if the entries of their global encoding vectors indexed by the aperture of that chunk are i.u.d. random variables over $\mathbb{F}_2$, and a given packet pertaining to a given chunk is globally dense if it belongs to a maximal dense collection of packets pertaining to that chunk.

w.p. b.a.b. $\dot{\epsilon}$*, so long as*

$$L^3 q \log \frac{Ln}{\epsilon} = o(n). \tag{3.6}$$

By Lemma 3.15, it can be seen that, for at least one chunk, there do not exist at least $n/q - O\left((L^3(n/q)^2 \log(Ln/\epsilon))^{\frac{1}{3}}\right)$ packets at the sink node, w.p. b.a.b. $\dot{\epsilon}$, so long as $L^3 q \log(Ln/\epsilon) = o(n)$, and hence, for at least one chunk, there do not exist at least $n/q - O\left((L^3(n/q)^2 \log(Ln/\epsilon))^{\frac{1}{3}}\right) - L\log(nL/\dot{\epsilon})$ LILE packets at the sink node, w.p. b.a.b. ϵ.

Let $\hat{Q}_L$ be Q_L restricted to its rows pertaining to the LILE packets, for all the chunks. Thus, $\hat{Q}_L$ fails to include at least $n - O\left(q(L^3(n/q)^2 \log(Ln/\epsilon))^{\frac{1}{3}}\right) - qL\log(nL/\dot{\epsilon})$ rows, w.p. b.a.b. ϵ. Now, the problem is to derive an upper bound on the probability that $\hat{Q}_L$ fails to have rank k. Due to the structure of OCC, $\hat{Q}_L$ is a (γ_o, α) symmetric RB matrix, but neither necessarily irregular nor necessarily regular. However, by replacing ϵ with $\dot{\epsilon}$ in the above argument, one can see that, w.p. b.a.b. $\dot{\epsilon}$, $\hat{Q}_L$ fails to include a sub-matrix with the structure of a (γ_o, α) RSRB matrix with $n/q - O\left((L^3(n/q)^2 \log(Ln/\epsilon))^{\frac{1}{3}}\right) - L\log(nL/\dot{\epsilon}) - L$ rows pertaining to each aperture and k columns.

We now need to derive an upper bound on the probability that such a (well-structured) sub-matrix of $\hat{Q}_L$ fails to have rank k. This upper bound serves for the probability that $\hat{Q}_L$ (and also Q_L) fails to have rank k. The results of Conjecture 3.1, however, are not applicable to our setting in general, since we do not restrict the overlap size γ_o to be $\alpha - 1$. Surprisingly, similar results also hold for more general settings as it can be seen through our simulation results in Section 3.5 (no formal proof is known yet). We formalize this observation in a conjecture as follows.[18]

Conjecture 3.2 *Let* n^*, k^*, α^* *and* γ_o^* *be integers* ($k^* \leq n^*$, $\gamma_o^* < \alpha^*$)*. Let* Q *be a* (γ_o^*, α^*) *(irregular or regular) symmetric or asymmetric RB matrix of size* $n^* \times k^*$*. For sufficiently large* k^*, $\Pr\{\text{rank}(Q) < k\} \leq 2^{-(n^*-k^*)}$, *so long as* $\gamma_o^* \geq 2\sqrt{k^*}$, *or*

[18] We here briefly highlight the differences between Conjecture 3.2 and Conjecture 4.2 of [28]: (i) the latter considers a sub-class of irregular symmetric random banded matrices, yet the former considers four more general classes of regular/irregular symmetric/asymmetric random banded matrices, and (ii) unlike the latter, for a given aperture size, the overlap size in the former is not restricted to one particular value.

$\gamma_o^* \geq \tau_e^* \tau_o^* \sqrt{k^*}$, *respectively, where* $\gamma_o^* = \alpha^*/\tau_e^*$, *and* $\tau_e^* = \tau_o^*/(\tau_o^* - 1)$, *for any constant divisor* τ_o^* *of* α^*.

One should note that in the following theorems (for OCC), we use the result of Conjecture 3.2 in order to derive a tighter upper bound on the probability that the underlying sub-matrix of $\hat{Q}_L$ (with the structure of an RB matrix) fails to have full rank (and consequently, a tighter upper bound on the overhead $(n-k)$). However, it should be clear that this probability is bounded from above by the probability that not all the sub-matrices of the underlying (well-structured) matrix, corresponding to different apertures, have full rank. On the other hand, it is not hard to find an upper bound on the latter probability, by applying the result of Lemma 3.6 on the sub-matrices corresponding to all the apertures in the underlying matrix. In the asymptotic regime, however, the first largest term in the upper bound on the overhead, which indicates the speed of convergence (defined in an asymptotic sense), dominates the rest of the terms, and hence the result of Conjecture 3.2 (as can be seen in the following results) does not affect the speed of convergence. Thus, all our results (regarding the speed of convergence of OCC) hold, regardless of the validity of the results of Conjecture 3.2.

Lemma 3.15 together with Conjecture 3.2 (symmetric case) yield the following.

Theorem 3.10 *For every* $\epsilon > 0$, *an OCC with* q *chunks of size* α, *and overlap size* $\gamma_o \geq 2\sqrt{k}$, *fails to transmit* k *message vectors over a line network of length* L *with any one-in-one-out worst-case traffic of capacity* n, *w.p. b.a.b.* ϵ, *so long as*

$$k \leq q\varphi_n - \log(1/\dot{\epsilon}),$$

and $L^3 q \log(nL/\epsilon) = o(n)$, *where* $\varphi_n = n/q - O\left((L^3(n/q)^2 \log(Ln/\epsilon))^{\frac{1}{3}}\right) - L\log(nL/\dot{\epsilon}) - L$.

Let τ_o^* be the smallest integer divisor of a given chunk size α, such that by selecting τ_o to be τ_o^*, it follows that $\gamma_o \geq 2\sqrt{k}$. (Here, we suppose that the chunk size α is chosen such that τ_o^* exists. The case for which there is no such τ_o^* will be discussed

shortly.) For every $\tau_o > \tau_o^*$, $\gamma_o \geq 2\sqrt{k}$, and hence the result of Theorem 3.10 holds. The larger is the choice of τ_o, however, the larger is the number of chunks and the lower would be the speed of convergence of OCC to the capacity for a given α (i.e., for a given coding cost).[19] Thus, an OCC with sufficiently large chunks of a given size α and the overlap size $\gamma_o > 2\sqrt{k}$ has the highest speed of convergence to the capacity when its overlap size is the smallest possible value, i.e., with the overlap parameter τ_o equal to τ_o^* defined as above.

Now suppose that there is no such τ_o^* for a given α (i.e., for any divisor τ_o of α, it follows that $\gamma_o < 2\sqrt{k}$), but there exists at least a divisor τ_o of α, such that q $(= \tau_o k/\alpha)$ satisfies condition (3.6). In this case, w.p. b.a.b. ϵ, the matrix $\hat{Q}_L$ fails to include a (γ_o, α) RSRB matrix of a size similar to that discussed prior to Theorem 3.10, except that in this case, $\gamma_o < 2\sqrt{k}$ (by the assumption). Thus, Conjecture 3.2 is no longer useful to upper bound the probability that such a (sub-) matrix fails to have rank k. However, it should be clear that this probability is bounded from above by the probability that all the sub-matrices corresponding to different apertures of the underlying RSRB matrix have full rank. The following theorem summarizes the above discussion.

Theorem 3.11 *For every $\epsilon > 0$, an OCC with q chunks of size α, and overlap size $\gamma_o < 2\sqrt{k}$, fails to transmit k message vectors over a line network of length L with any one-in-one-out worst-case traffic of capacity n, w.p. b.a.b. ϵ, so long as*

$$k \leq q\varphi - qL\log(nL/\dot{\epsilon}) - q\log(q/\dot{\epsilon}) - qL - q, \tag{3.7}$$

and $L^3 q \log(nL/\epsilon) = o(n)$, where

$$\varphi = n/q - O\left((L^3(n/q)^2\log(Ln/\epsilon))^{\frac{1}{3}}\right).$$

In Theorem 3.11, the larger is the overlap size, the looser would be the upper

[19]The ratio $(n - k_{n,\epsilon})/n = \left(O(q(L^3(n/q)^2\log(Ln/\epsilon))^{\frac{1}{3}}) + qL\log(nL/\dot{\epsilon}) + qL + \log(1/\dot{\epsilon})\right)/n$ decreases as q increases. Thus, for larger values of q, this ratio becomes smaller for given L, n and ϵ, and this implies a lower speed of convergence to the capacity.

bound on k. Part of the reason that the result of Theorem 3.11 is not tight is that it is based on the assumption that each chunk has to be decodable in isolation which is a sufficient but not a necessary condition in the case of overlapping chunks. Indeed, our analysis in this case is sub-optimal, and one can expect an OCC with sufficiently large chunks, and smaller overlap size to perform even better than what Theorem 3.11 presents.

Since $k_{n,\epsilon} = n - O\left(q(L^3(n/q)^2 \log(Ln/\epsilon))^{\frac{1}{3}}\right) - qL\log(nL/\dot{\epsilon}) - q\log(q/\dot{\epsilon}) - qL - q$ is a decreasing function of q (for given n, L and ϵ), the larger is q (for a given α), the lower is the speed of convergence to the capacity (the ratio $(n - k_{n,\epsilon})/n$ becomes larger as q increases, for given n, L and ϵ). Thus, similarly as before, an OCC with sufficiently large chunks of a given size α and the overlap size $\gamma_o < 2\sqrt{k}$ provides the highest speed of convergence to the capacity when its overlap size is the smallest possible value, i.e., with the overlap parameter τ_o equal to 1 (zero overlap size). Thus, for a given coding cost, comparing OCC with sufficiently large chunks and smaller overlaps, CC (i.e., OCC with no overlap) has the fastest convergence to the capacity.

All-In-All-Out Worst-Case Traffics:

Corresponding to Lemmas 3.14 and 3.15, we have the following results for all-in-all-out traffics.

Lemma 3.16 *For every $\epsilon > 0$, by applying a dense code to a given chunk over a line network of length L with any all-in-all-out worst-case traffic of capacity n, the number of LILE packets pertaining to that chunk at the sink node is not larger than $\varphi_n - L\log(L/\epsilon)$ w.p. b.a.b. ϵ, where φ_n is the capacity of the flow by the packets pertaining to that chunk.*

Lemma 3.17 *Let L, n, q, φ_n and ϵ be defined as above. Then the sink node fails to receive at least $\varphi_n - L\log(qL/\dot{\epsilon})$ LILE packets pertaining to every chunk w.p. b.a.b. $\dot{\epsilon}$, and φ_n fails to be at least*

$$\left(1 - O\left(((q/n)\log(Lq/\epsilon))^{\frac{1}{2}}\right)\right).(n/q),$$

w.p. b.a.b. $\dot{\epsilon}$, *so long as*

$$q \log \frac{Lq}{\epsilon} = o(n).$$

Lemma 3.17 along with Conjecture 3.2 (symmetric case) yield the following results.

Theorem 3.12 *For every* $\epsilon > 0$, *an OCC with* q *chunks of size* α, *and overlap size* $\gamma_o \geq 2\sqrt{k}$, *fails to transmit* k *message vectors over a line network of length* L *with any all-in-all-out worst-case traffic of capacity* n, *w.p. b.a.b.* ϵ, *so long as*

$$k \leq q\varphi_n - \log(1/\dot{\epsilon}),$$

and $q \log(Lq/\epsilon) = o(n)$, *where*

$$\varphi_n = n/q - O\left(((n/q)\log(Lq/\epsilon))^{\frac{1}{2}}\right) - L\log(Lq/\dot{\epsilon}) - L.$$

Moreover, an OCC with the above description but with the overlap size $\gamma_o < 2\sqrt{k}$ *fails in a network scenario as above, w.p. b.a.b.* ϵ, *so long as*

$$k \leq q\varphi_n - q\log(q/\dot{\epsilon}) - q,$$

and $q \log(Lq/\epsilon) = o(n)$, *where* φ_n *is defined as above.*

Coding Cost:

The worst-case with regards to the encoding cost occurs if, for every chunk, every internal node (or the source node) transmits all its successful packets for that chunk after receiving all the packets for that chunk transmitted by the node in the upper layer, and if all the local encoding vectors' coefficients are chosen to be nonzero. The number of packet operations for encoding at each internal node (or the source node) is $O(n^2/q)$ (or $O(n\alpha)$). (For every chunk, there are $O(n/q)$ packets transmitted and received by each internal node and there are $O(n/q)$ packets transmitted by the

source node.) Thus the encoding cost of an OCC with chunks of size α to transmit k message vectors in the underlying scenario is $O(\alpha)$.[20]

To solve the system of linear equations $\{\boldsymbol{y}_e = \sum_{j \in \mathcal{M}} \mu_{e,j} \hat{\boldsymbol{x}}_j : e \in \mathcal{I}_L\}$ using Gaussian or Gauss-Jordan elimination, the sink node requires $O(kn\alpha)$ field operations along with $O(n\alpha)$ packet operations (the bandwidth and the size of the matrix Q_L are α and $n \times k$, respectively). Thus the decoding cost of an OCC with chunks of size α to transmit k message vectors in the underlying scenario is also $O(\alpha)$.

Theorem 3.13 *The encoding and decoding costs of an OCC with chunks of size α to transmit k message vectors over a line network with any worst-case traffic are $O(\alpha)$.*

3.3.6 Comparison

We now compare CC and OCC with sufficiently large chunks over one-in-one-out worst-case traffics in the asymptotic regime. Similar conclusions can also be drawn while comparing these codes over all-in-all-out worst-case traffics.

Consider a one-in-one-out worst-case traffic of capacity n over a line network of length L. Let the k message vectors be divided into $\tau_o q$ chunks of size α $(= k/q)$.[21]

We compare Theorem 3.5 with Theorems 3.10 and 3.11. (Theorem 3.5 for CC is a special case of Theorem 3.11, where τ_o is set to 1.) We study the tradeoff between the probability of failure (in recovering all the message vectors), called the *message error rate*, and the speed of convergence to the capacity. To be more specific, we compare CC and OCC with similar chunk size (similar coding cost) with respect to their speed of convergence to the capacity when the message error rate is given.

---

[20]The number of packet operations for encoding is $O((L-1)n^2/q) + O(n\alpha)$, and hence the encoding cost is $O((L-1)n^2/qkL) + O(n\alpha/kL)$, i.e., $O((L-1)n/qL) + O(\alpha/L) = O(\alpha)$, since $n = (1 + o(1))k$ in a capacity-achieving scenario, as k goes to infinity.

[21]We assume that the number of chunks in OCC with overlap parameter τ_o is $\tau_o q$, not q, as was the case in the previous sections. The reason is to be consistent in the definition of the chunk size α as k/q, for both CC and OCC, as we compare CC and OCC with different overlap parameters for a similar chunk size. Thus, q needs to be replaced with $\tau_o q$ in the results presented earlier for an OCC with overlap parameter τ_o.

Table 3.1: Comparison of Various Network Codes over Line Networks with Worst-Case Traffics

Network Codes	Redundancy Rate	MER	PER	Chunk Size (α)	Constraints
Dense Codes	$O\left(\frac{1}{k}\log\left(\frac{k}{\epsilon}\right)\right)$	ϵ	-	k	-
	$O\left(\frac{1}{k}\log\left(\frac{1}{\epsilon}\right)\right)$	ϵ	-	k	-
CC: Large α	$O\left(\left(\frac{1}{\alpha}\log\left(\frac{k}{\epsilon}\right)\right)^{\frac{1}{3}}\right)$	ϵ	-	$\omega\left(\log\left(\frac{k}{\epsilon}\right)\right)$	-
	$O\left(\left(\frac{1}{\alpha}\log\left(\frac{k}{\alpha\epsilon}\right)\right)^{\frac{1}{2}}\right)$	ϵ	-	$\omega\left(\log\left(\frac{k}{\alpha\epsilon}\right)\right)$	-
CC: Relatively Small α	γ_c	$\delta k/\alpha$	δ	$\Omega\left(\frac{1}{\gamma_c^3}\log\left(\frac{1}{\gamma_c\delta}\right)\right)$	$\delta = o\left(\frac{\alpha}{k}\right)$
	γ_c	$\delta k/\alpha$	δ	$\Omega\left(\frac{1}{\gamma_c^2}\log\left(\frac{1}{\delta}\right)\right)$	$\delta = o\left(\frac{\alpha}{k}\right)$
CC: Very Small α	γ_c	-	δ	$\Omega\left(\frac{1}{\gamma_c^3}\log\left(\frac{1}{\gamma_c\delta}\right)\right)$	$\delta = O(1)$
	γ_c	-	δ	$\Omega\left(\frac{1}{\gamma_c^2}\log\left(\frac{1}{\delta}\right)\right)$	$\delta = O(1)$
OCC: Large α	$O\left(\left(\frac{\tau_o}{\alpha}\log\left(\frac{k}{\epsilon}\right)\right)^{\frac{1}{3}}\right)$	ϵ	-	$\omega\left(\tau_o\log\left(\frac{k}{\epsilon}\right)\right)$	-
	$O\left(\left(\frac{\tau_o}{\alpha}\log\left(\frac{k}{\alpha\epsilon}\right)\right)^{\frac{1}{2}}\right)$	ϵ	-	$\omega\left(\tau_o\log\left(\frac{k}{\alpha\epsilon}\right)\right)$	-
OCC: Relatively Small α	γ_c	$\delta^*\tau_o k/\alpha$	δ^*	$\Omega\left(\frac{\tau_o}{\gamma_c^3}\log\left(\frac{\tau_o}{\gamma_c\delta}\right)\right)$	$\delta = o\left(\frac{\alpha}{k}\right)$
	γ_c	$\delta^*\tau_o k/\alpha$	δ^*	$\Omega\left(\frac{\tau_o}{\gamma_c^2}\log\left(\frac{\tau_o}{\gamma_c\delta}\right)\right)$	$\delta = o\left(\frac{\alpha}{k}\right)$
OCC: Very Small α	γ_c	-	δ^*	$\Omega\left(\frac{\tau_o}{\gamma_c^3}\log\left(\frac{\tau_o}{\gamma_c\delta}\right)\right)$	$\delta = O(1)$
	γ_c	-	δ^*	$\Omega\left(\frac{\tau_o}{\gamma_c^2}\log\left(\frac{\tau_o}{\gamma_c\delta}\right)\right)$	$\delta = O(1)$

Theorem 3.14 *For a given message error rate and for sufficiently large chunks, an OCC with larger overlap size has a lower speed of convergence compared to an OCC with smaller overlap size.*

Since CC is a special case of OCC with zero overlap size, i.e., τ_o is equal to 1, the following is a result of Theorem 3.14.

Corollary 3.1 *For a given message error rate and for sufficiently large chunks, a CC has a higher speed of convergence to the capacity compared to an OCC with any nonzero overlap size.*

Table 3.1 summarizes the performance results of dense codes, CC and OCC over two types of worst-case traffics. In particular, the performance measures are the overhead per message, i.e., $(n-k)/k$, and the message error rate (MER) or the

packet error rate (PER), for various chunk sizes (α). The coding cost of each code (in this table) is linear in the size of chunks, i.e., $O(\alpha)$. In the table, the upper (or the lower) row in front of each network code corresponds to the one-in-one-out (or all-in-all-out) worst-case traffics. For example, for each type of worst-case traffics, the comparison of the corresponding rows of Table 3.1 for CC: large α and OCC: large α shows that for a given overhead per message (i.e., a given speed of convergence to the capacity) and a given MER, the chunk size α for OCC must be larger than that for CC by a factor of τ_o. This implies that for the same MER and speed of convergence to the capacity, the coding cost of OCC is τ_o times that of CC. Also, for both CC and OCC with large α, the lower bound on α is a super-logarithmic function of k.

3.3.7 From Sufficiently Large Chunks to Smaller Chunks

Our asymptotic results so far indicate that for sufficiently large chunks, which guarantee the convergence to the capacity, CC (i.e., OCC with zero overlap size) is superior to OCC with any nonzero overlap size. One may then wonder about whether there is any provable advantage in using OCC in the asymptotic regime. We give an affirmative answer to this question in Section 3.4, and the following provides a motivation.

From Table 3.1, it can be seen that for CC and OCC with large chunks to be capacity-achieving (i.e., for $(n-k)/k$ to go to 0, as k goes to infinity), the chunk size needs to be bounded from below by a super-logarithmic function of k. However, this lower bound might be larger than the affordable coding cost in many practical scenarios. Therefore, in Section 3.4, we study CC and OCC with chunks of smaller sizes and show that in such cases, OCC can outperform CC in the asymptotic regime (for relatively/very small chunk sizes, comparing an OCC and a CC with similar coding cost and speed of convergence to the capacity, the former has a smaller MER/PER). The performance results of CC and OCC with relatively or very small chunks (given in Section 3.4) are also summarized in Table 3.1.

3.4 Capacity-Approaching Codes: Towards Linear-Time Network Codes

Focusing on the design of CC and OCC with smaller coding cost when compared to those with sufficiently large chunk sizes discussed in Section 3.3, in this part, we analyze CC and OCC with smaller chunk sizes.

3.4.1 CC with Small Chunks

One-In-One-Out Worst-Case Traffics:

Consider applying a CC with q chunks of size α to transmit k message vectors over a line network of length L with an arbitrary one-in-one-out worst-case traffic of capacity $n = (1+\gamma_c)k$, for an arbitrarily constant $0 < \gamma_c < 1$. Consider a case that α is not sufficiently large to comply with condition (3.3). Then, by the result of Lemma 3.10, it follows that not all the chunks are allocated a flow of capacity of at least $n/q - O\left((L^3(n/q)^2\log(Ln/\epsilon))^{\frac{1}{3}}\right)$. Thus Lemma 3.10 and Theorems 3.4 and 3.5 are not applicable to this case. This implies the need for a new analysis technique in such a setting. Our approach in this case is to fix a chunk, and give a lower bound on the capacity of the traffic pertaining to that chunk. We are then able to lower bound the probability of receiving a collection of α $(= k/q)$ innovative packets pertaining to that chunk by the sink node.

The expected value of the capacity of the traffic pertaining to a given chunk is $n/q = (1+\gamma_c)\alpha$. Let $\mu = (1+\gamma_c)\alpha$. The deviation of the actual value of this capacity from its expectation is upper bounded as follows.

Lemma 3.18 *For every $\delta > 0$, and an arbitrary constant $0 < \gamma_c < 1$, by applying a CC with chunk size α to transmit k message vectors over a line network of length L with any one-in-one-out worst-case traffic of capacity $(1+\gamma_c)k$, the capacity of the traffic pertaining to a given chunk fails to be larger than*

$$\left(1 - O\left(\left((L^3/\mu)\log(L\mu/\delta)\right)^{\frac{1}{3}}\right)\right)\cdot\mu \tag{3.8}$$

w.p. b.a.b. $\dot{\delta}/2$, *where* $\mu = (1+\gamma_c)\alpha$.

Lemma 3.19 *For every* $\delta > 0$, *and an arbitrary constant* $0 < \gamma_c < 1$, *by applying a dense code to a given chunk over a line network of length* L *with any one-in-one-out worst-case traffic, the number of LILE packets pertaining to that chunk is not larger than* $\varphi_n - L\log(L\varphi_n/\dot{\delta}) - L$ *w.p. b.a.b.* $\dot{\delta}/2$, *where* φ_n *is the capacity of the traffic pertaining to that chunk.*

Lemma 3.18 together with Lemma 3.19 show that the sink node fails to receive at least $\varphi_n - L\log(L\varphi_n/\dot{\delta}) - L$ LILE packets pertaining to a given chunk w.p. b.a.b. $\dot{\delta}$; and by applying Lemma 3.6, the probability of decoding failure of a given chunk can be upper bounded as follows.

Lemma 3.20 *For every* $\delta > 0$, *and an arbitrary constant* $0 < \gamma_c < 1$, *by applying a CC with chunk size* α *to transmit* k *message vectors over a line network of length* L *with any one-in-one-out worst-case traffic of capacity* $(1+\gamma_c)k$, *the probability that a given chunk fails to be decoded is b.a.b.* δ, *so long as*

$$\alpha \leq \varphi - L\log(L\varphi/\dot{\delta}) - \log(1/\delta) - L - 1, \tag{3.9}$$

where

$$\varphi = \left(1 - O\left(\left((L^3/\mu)\log(L\mu/\delta)\right)^{\frac{1}{3}}\right)\right)\mu,$$

and $\mu = (1+\gamma_c)\alpha$.

It is worth noting that, despite its appearance, inequality (3.9) imposes a lower bound on α (the right hand side of inequality (3.9) is itself a function of α). By replacing μ with $(1+\gamma_c)\alpha$, inequality (3.9) can be written as

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\log\left(\frac{L}{\gamma_c\delta}\right)\right). \tag{3.10}$$

The details of the derivation of (3.10) are deferred to Appendix C.1. The above result shows that, over any one-in-one-out worst-case traffic, for a CC with chunk

size satisfying (3.10), the probability of decoding failure for a given chunk is b.a.b. δ. Therefore, the expected number of undecodable chunks is b.a.b. δq. For sufficiently small choice of δ (i.e., δq goes to 0, as k goes to infinity), not all the chunks are decodable w.p. b.a.b. ϵ, where ϵ is equal to δq, and then the following is immediate.

Theorem 3.15 *For every $\delta > 0$, when δ goes to 0 **sufficiently fast, as k goes to infinity,**[22] by applying a CC with chunk size α to transmit k message vectors over a line network of length L with any one-in-one-out worst-case traffic of capacity $(1+\gamma_c)k$, not all the chunks are decodable w.p. b.a.b. ϵ, so long as*

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\log\frac{L}{\gamma_c\delta}\right),$$

where $0 < \gamma_c < 1$ is an arbitrary constant, and $\epsilon \doteq \delta q$.

In some cases, however, such small choices of δ might not be practical in that the corresponding chunk size α (and the coding cost) might be too large as the lower bound on α grows logarithmically with the inverse of δ. Now, let us assume larger values of δ (i.e., δ does not go to 0 sufficiently fast, as k goes to infinity) up to a constant. We show that in such cases the actual number of undecodable chunks is tightly concentrated around its expectation.[23]

Theorem 3.16 *For every $0 < \epsilon < 1$, and $0 < \delta < 1$, and an arbitrary constant $0 < \gamma_c < 1$, by applying a CC with chunk size α to transmit k message vectors over a line network of length L with any one-in-one-out worst-case traffic of capacity $(1+\gamma_c)k$, for an arbitrary constant $0 < \gamma_a < 1$, the fraction of undecodable chunks is larger than $(1+\gamma_a)\delta$, w.p. b.a.b. ϵ, so long as*

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\log\frac{L}{\gamma_c\delta}\right),$$

[22] We say that δ goes to 0 "sufficiently fast," if δq goes to 0, as k goes to infinity.

[23] We consider the worst-case assuming that the probability of decoding failure of any chunk is the largest possible, i.e., δ (as shown in Lemma 3.20), and hence the expected fraction of undecodable chunks is δ.

and $\alpha^2/\gamma_a^2\delta^2 = o(k/\log(1/\epsilon))$.

Therefore, for an arbitrary constant $0 < \gamma_a < 1$, a CC with chunk size α satisfying both conditions in Theorem 3.16 fails to recover more than $(1+\gamma_a)\delta k$ message vectors w.p. b.a.b. ϵ. This is because the number of chunks is q, and that any undecodable chunk accounts for at most k/q unrecovered message vectors.

In such cases, however, the expected fraction of undecodable chunks and unrecovered message vectors are bounded away from zero, for constant values of δ. Thus, a CC, alone, might not recover all the message vectors. Now, a question is how a CC with small chunks can recover all the message vectors? One solution is to devise a proper precoding scheme in order to guarantee the completion of decoding of all the message vectors.[24] The combination of a CC and a precode is called a *chunked code with precoding* (CCP).

The precoding works as follows: The set of k message vectors constitute the input of an erasure code, called *precode*, of rate R. The coded packets at the output of the precode are called the *intermediate packets*. The number of intermediate packets is k/R. The intermediate packets are transmitted over the network by using a CC. By applying a CC with chunks of size α satisfying both conditions in Theorem 3.16, the number of the intermediate packets that are not recoverable at the output of the CC decoder is larger than $(1+\gamma_a)\delta k/R$, w.p. b.a.b. ϵ, for an arbitrary constant $0 < \gamma_a < 1$ (as shown in Theorem 3.16). Now, let us assume that the CC decoder recovers at least $(1-(1+\gamma_a)\delta)k/R$ intermediate packets (this assumption fails w.p. b.a.b. ϵ). These packets constitute the input of the precode decoder. The precode needs to recover the k message vectors from the set of recovered intermediate packets. Thus, the precode needs to be capable of recovering a fraction $(1+\gamma_a)\delta$ of erasures.[25]

[24]Prior to the work of [24], the idea of precoding to guarantee the completion of the decoding of all the message vectors was previously used in the (related) context of (rateless) erasure codes (over a single erasure channel with an arbitrary worst-case traffic), e.g., Raptor codes [48] and Online codes [49].

[25]The precode can be capacity-achieving or non-capacity-achieving. In the case of using a capacity-achieving precode, the rate of the precode has to be at least $1-(1+\gamma_a)\delta$. In the case of using a non-capacity-achieving precode, its rate can be arbitrarily close to $1-(1+\gamma_a)\delta$, yet, the precode has to correct up to a fraction $(1+\gamma_a)\delta$ of erasures.

Table 3.2: Comparison of Various Linear-Time Erasure Codes

Erasure Codes	Coding Cost	Probability of Failure	Error-Exponent
Tornado/LDPC Codes in [50 53]	$O\left(\log \frac{1}{\delta_\gamma}\right)$	poly$(1/k)$	0
LDPC Codes in [54]	$O\left(\frac{1}{\delta_\gamma} \log \frac{1}{\delta_\gamma}\right)$	poly$(1/k)$	0
Codes in [55]	$O\left(\frac{1}{\delta_\gamma^2} \log \frac{1}{\delta_\gamma}\right)$	$e^{-\Omega(k)}$	poly(δ_γ)
	$O\left(\frac{1}{\delta_\gamma^4} \log \frac{1}{\delta_\gamma}\right)$	0	-

We are, further, interested in a CCP scheme with linear-time encoding/decoding algorithms (i.e., with a constant coding cost with respect to k). Therefore, both the CC and the precode must be linear-time codes. It should be clear that CC is linear-time so long as the size of chunks is a constant, and therefore, the problem is to look for a linear-time precode.[26]

Table 3.2 lists a number of linear-time erasure codes in the literature with the best known tradeoffs between the coding cost and the probability of failure. In this table, each code has dimension k, rate R, and is able to recover a fraction of erasures up to γ, for some $0 < \gamma < \frac{R}{1+R}$, and the coding cost of each code is expressed in terms of the parameter $\delta_\gamma \doteq (1-\gamma)(1+R) - 1$.

In Table 3.2, from top to bottom, for given k and δ_γ, the speed of convergence of the failure probabilities to zero and the coding cost of the codes increase. By choosing R and γ to be constant, the coding cost will also be constant. In our case, on the one hand, the precode needs to recover a fraction of erasures up to $\gamma = (1+\gamma_a)\delta$, for arbitrary constants $0 < \gamma_a, \delta < 1$. The codes listed in Table 3.2, on the other hand, are each able to recover a fraction of erasures up to γ with a constant coding cost, and are thus each a good candidate for a precode to be combined with a CC in order to complete the design of a linear-time CCP.

[26]The specifications of all the precodes discussed in the rest of this thesis are the same as those discussed here, and hence not repeated for brevity.

All-In-All-Out Worst-Case Traffics:

The results of CC with small chunks over all-in-all-out worst-case traffics are very similar to those over one-in one-out traffics, except that the lower bounds on the chunk size in the two cases differ. The difference arises from the difference between the conditions (3.3) and (3.5) on the chunk size in Lemmas 3.10 and 3.12, respectively. Thus, for the sake of brevity, we state the main lemmas, and the resulting theorems would be similar to Theorems 3.15 and 3.16 (and not repeated), except that the lower bound on the chunk size needs to be replaced with a new lower bound derived below.

Lemma 3.21 *For every $\delta > 0$, and an arbitrary constant $0 < \gamma_c < 1$, by applying a CC with chunk size α, to transmit k message vectors over a line network of length L with any all-in-all-out worst-case traffic of capacity $(1+\gamma_c)k$, the capacity of the traffic pertaining to a given chunk fails to be larger than*

$$\left(1 - O\left((\log(L/\delta)/\mu)^{\frac{1}{2}}\right)\right) \cdot \mu \tag{3.11}$$

w.p. b.a.b. $\dot{\delta}/2$, where $\mu = (1+\gamma_c)\alpha$.

Lemma 3.22 *For every $\delta > 0$, by applying a dense code to a given chunk over a line network of length L with any all-in-all-out worst-case traffic, the number of LILE packets pertaining to that chunk fails to be larger than $\varphi_n - L\log(L/\dot{\delta}) - L$ w.p. b.a.b. $\dot{\delta}/2$, where φ_n is the capacity of the traffic pertaining to that chunk.*

Lemma 3.21 together with Lemma 3.22 show that the sink node fails to receive at least $\varphi - L\log(L/\dot{\delta}) - L$ LILE packets pertaining to a given chunk w.p. b.a.b. $\dot{\delta}$, where $\varphi = (1 - O((\log(L/\delta)/\mu)^{\frac{1}{2}}))\mu$, and $\mu = (1+\gamma_c)\alpha$. Thus the following gives an upper bound on the probability of decoding failure of a given chunk.

Lemma 3.23 *For every $\delta > 0$, and $0 < \gamma_c < 1$, by applying a CC with chunk size α, to transmit k message vectors over a line network of length L with any all-in-all-out*

worst-case traffic of capacity $(1+\gamma_c)k$*, the probability of a given chunk not to be decodable is b.a.b.* δ*, so long as*

$$\alpha \leq \varphi - L\log(L/\delta) - \log(1/\delta) - L - 1, \tag{3.12}$$

where

$$\varphi = \left(1 - O\left((\log(L/\delta)/\mu)^{\frac{1}{2}}\right)\right) \cdot \mu,$$

and $\mu = (1+\gamma_c)\alpha$.

By replacing μ with $(1+\gamma_c)\alpha$, inequality (3.12) can be written as

$$\alpha = \Omega\left(\frac{L}{\gamma_c^2}\log\frac{L}{\delta}\right). \tag{3.13}$$

The details are similar to that of (3.10), and hence omitted. Thus, over any all-in-all-out worst-case traffic, for a CC with a chunk size as given in (3.13), the probability of decoding failure of a given chunk is b.a.b. δ.

3.4.2 OCC with Small Chunks

One-In-One-Out Worst-Case Traffics:

Consider applying an OCC with q chunks, each of size α, and overlap size γ_o (and overlap parameter τ_o, an arbitrary constant divisor of α), to transmit k message vectors over a line network of length L with an arbitrary one-in-one-out worst-case traffic of capacity $n = (1+\gamma_c)k$, for an arbitrary constant $0 < \gamma_c < 1$.

The same approach used in the analysis of CC with small chunks is not applicable to OCC with small chunks. Indeed, in the analysis of CC with small chunks, upper bounding the fraction of undecodable chunks (by using a martingale argument) yields a trivial upper bound on the fraction of unrecovered message vectors as the chunks do not share any message vectors. However, unlike CC, in OCC, a martingale argument alone does not provide a tight upper bound on the the fraction of unrecoverable message vectors. The reason is that in OCC the chunks are to be decoded together,

and this implies the need for a new approach for the analysis of OCC with small chunks.

In the following, we provide a sketch of our analysis. Consider the decoding matrix Q_L at the sink node, and let $\hat{Q}_L$ be Q_L restricted to the set of rows in Q_L pertaining to the set of LILE packets at the sink node. In the case of OCC with small chunks, the result of Lemma 3.15 is not applicable to lower bound the probability that there exists a set of rows in $\hat{Q}_L$, such that these rows form a (γ_o, α) RSRB matrix with a sufficiently large number of rows and k columns. Consequently, the result of Conjecture 3.2 (symmetric case) is no longer useful.

Let χ be an integer sufficiently smaller than the number of chunks. Consider a particular set of χ contiguous chunks (we will precisely specify the choice of χ later). Focus on the set of LILE packets pertaining to the given set of chunks. We first lower bound the probability that a subset of rows pertaining to these chunks in $\hat{Q}_L$ forms a (γ_o, α) RARB matrix with a sufficiently large number of rows and $\chi(\alpha - \gamma_o) + \gamma_o$ columns (i.e., the number of distinct message vectors in χ contiguous chunks). By using Conjecture 3.2 (asymmetric case), we next upper bound the probability that such a set of chunks fails to be decodable. Studying all such sets of χ contiguous chunks (i.e., lower bounding the probability that any such set is decodable), the fraction of recoverable message vectors can be lower bounded. (All the message vectors in a chunk belonging to a decodable set of chunks are recoverable).

It is worth noting that our analysis is sub-optimal in the sense that there might be some recoverable message vectors that we declare as unrecoverable. This is because the decoding, in our setting, is performed on the set of all the chunks together, not on the subsets of chunks in isolation.

We formalize the above process as follows. We call each set of χ contiguous chunks, chosen in an end-around fashion, a *hyperchunk*. The first hyperchunk starts from the first chunk, the second hyperchunk starts from the second chunk, and so forth.[27] The hyperchunks are overlapping in chunks, and regardless of the choice of χ, the number of hyperchunks is q. We also call each (disjoint) set of α/τ_o $(= k/q)$

[27]We will later determine the (order) optimal value of χ that results in the tightest possible bounds as part of our analysis.

contiguous message vectors a *block*. The first block starts from the first message vector, the second block starts from the message vector next to the last message vector in the first block, and so forth. By definition, each hyperchunk consists of $\chi + \tau_o - 1$ contiguous blocks. We say that a given hyperchunk is not decodable if it fails to be decoded *in isolation*. We also say that a given block is not recoverable if it does not belong to any decodable hyperchunk.

We shall upper bound the probability that a given hyperchunk is not decodable (by lower bounding the probability of receiving an innovative collection of $(\chi + \tau_o - 1)k/q$ packets belonging to this hyperchunk). Lemma 3.19 serves this purpose when the capacity of the flow by the packets pertaining to any chunk in this hyperchunk is given. We lower bound this capacity in the following.

The expected number of packets for a given chunk is $\mu = (1 + \gamma_c)\alpha/\tau_o$. The capacity of the traffic pertaining to that chunk is a random variable and its deviation from the expectation is upper bounded as follows. (Lemmas 3.24 and 3.25 follow from Lemmas 3.18 and 3.19, by replacing δ with δ/χ.)

Lemma 3.24 *For every $0 < \delta < 1$, and an arbitrary constant $0 < \gamma_c < 1$, and an arbitrary constant integer $\chi > 0$, by applying an OCC with chunk size α, and overlap parameter τ_o, to transmit k message vectors over a line network of length L with any one-in-one-out worst-case traffic of capacity $(1+\gamma_c)k$, the capacity of the flow by the packets pertaining to a given chunk fails to be larger than*

$$\left(1 - O\left(\left((L^3/\mu)\log(L\mu\chi/\delta)\right)^{\frac{1}{3}}\right)\right) \cdot \mu, \tag{3.14}$$

w.p. b.a.b. $\dot{\delta}\dot{\chi}$, where $\mu = (1+\gamma_c)\alpha/\tau_o$.

Lemma 3.25 *For every $0 < \delta < 1$, and an arbitrary constant integer $\chi > 0$, by applying a dense code to a given chunk over a line network of length L with any one-in-one-out worst-case traffic, the number of LILE packets pertaining to that chunk fails to be larger than $\varphi_n - L\log(L\varphi_n\chi/\dot{\delta}) - L$, w.p. b.a.b. $\dot{\delta}\dot{\chi}$, where φ_n is the capacity of the traffic pertaining to that chunk.*

Lemma 3.24 together with Lemma 3.25 show that, for a given chunk, w.p. b.a.b. $\dot{\delta}/\chi$, there are less than $\varphi - L\log(L\varphi\chi/\dot{\delta}) - L$ LILE packets pertaining to that chunk at the sink node, where $\varphi = (1 - O(((L^3/\mu)\log(L\mu\chi/\delta))^{\frac{1}{3}}))\mu$, and $\mu = (1+\gamma_c)\alpha/\tau_o$. Taking a union bound over a set of χ chunks (i.e., a hyperchunk), one can see that, w.p. b.a.b. $\dot{\delta}$, there does not exist a subset of LILE packets pertaining to a given hyperchunk such that their corresponding rows in $\hat{Q}_L$ form a (γ_o, α) RARB matrix with $\chi\varphi - \chi L\log(L\varphi\chi/\dot{\delta}) - \chi L$ rows and $\chi(\alpha - \gamma_o) + \gamma_o$ columns. Thus, by using Conjecture 3.2 (asymmetric case), the probability that a given hyperchunk is not decodable can be upper bounded as follows.

Lemma 3.26 *For every $0 < \delta < 1$, and an arbitrary constant $0 < \gamma_c < 1$, and an arbitrary constant integer $\chi > 0$, by applying an OCC with chunk size α, and overlap parameter τ_o, to transmit k message vectors over a line network of length L with any one-in-one-out worst-case traffic of capacity $(1+\gamma_c)k$, a given hyperchunk of size χ is not decodable w.p. b.a.b. δ, so long as*

$$r_o\alpha \leq \chi\varphi - \chi L\log(L\varphi\chi/\dot{\delta}) - \log(1/\dot{\delta}) - \chi L, \tag{3.15}$$

and

$$\gamma_o \geq \tau_e\tau_o\sqrt{\tau_o\alpha}, \tag{3.16}$$

where

$$\varphi = \left(1 - O\left(\left((L^3/\mu)\log(L\mu\chi/\delta)\right)^{\frac{1}{3}}\right)\right)\cdot\mu,$$

$\mu = (1+\gamma_c)\alpha/\tau_o$, *and* $r_o = (\chi-1)/\tau_o + 1$.

By replacing μ with $(1+\gamma_c)\alpha/\tau_o$, and by selecting $\chi \gg (\tau_o - 1)/\gamma_c$ (i.e., χ is an arbitrary constant integer sufficiently larger than $(\tau_o - 1)/\lambda$), inequality (3.15) can be written as

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\tau_o\log\left(\frac{L}{\gamma_c\delta}\tau_o\right)\right). \tag{3.17}$$

The details are deferred to Appendix C.2. For such a choice of χ, one can see that

condition (3.16) is met so long as

$$\alpha = \Omega\left(\frac{\tau_o^2}{\gamma_c}\right).$$

It is not hard to see that this condition is met so long as condition (3.17) is met.

Thus, by applying an OCC with a chunk size as given in (3.17), over any one-in-one-out worst-case traffic, the probability that a given hyperchunk is not decodable is b.a.b. δ; and thus the expected fraction of undecodable hyperchunks is at most δ. By using a martingale argument over the hyperchunks, it can be shown that the actual fraction of undecodable hyperchunks does not deviate far from the expectation w.h.p..

Theorem 3.17 *For every* $0 < \delta < 1$, *and an arbitrary constant* $0 < \gamma_c < 1$, *by applying an OCC with chunk size* α, *and overlap parameter* τ_o, *to transmit* k *message vectors over a line network of length* L *with any one-in-one-out worst-case traffic of capacity* $(1+\gamma_c)k$, *for an arbitrary constant* $0 < \gamma_a < 1$, *the fraction of undecodable hyperchunks is larger than* $(1+\gamma_a)\delta$, *w.p. b.a.b.* ϵ, *so long as*

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\tau_o \log\left(\frac{L}{\gamma_c \delta}\tau_o\right)\right)$$

and $(\alpha^2/\gamma_a^2\delta^2)\cdot(1/\gamma_c\tau_o) = o(k/\log(1/\epsilon))$.

Now, the problem is to upper bound the fraction of message vectors (or blocks) that are not recoverable. The fraction of unrecoverable blocks depends on the relative location of undecodable hyperchunks. We explain this dependency through an example as follows.

Consider a case with only two undecodable hyperchunks. Suppose that these two undecodable hyperchunks share a given block which does not belong to any other hyperchunk. This block would therefore be an unrecoverable block. Now, consider the case where these two undecodable hyperchunks either do not share any blocks, or any block that they share is in some other hyperchunks as well. In both cases, there

is no unrecoverable block, because each block belongs to at least one hyperchunk which is decodable.

It should be clear that among different positioning of undecodable hyperchunks, the one in which all such hyperchunks are adjacent results in the largest fraction of unrecoverable blocks. Since the expected fraction of undecodable hyperchunks is δ, the expected fraction of unrecoverable blocks is upper bounded by δ. Each block, itself, contains k/q message vectors, and hence, the expected number of message vectors that are unrecoverable is upper bounded by δk. This, however, is not a tight bound since the probability that a large number of undecodable hyperchunks are adjacent is very small. This implies the need for the distribution of undecodable hyperchunks to derive a tighter bound on the expected number of unrecoverable message vectors. To obtain such a distribution is however too complex. We, instead, analyze two extremal types of dependency structures of hyperchunks as defined below.

Let I be the set of integers $\{i\}_{1\leq i\leq q}$. For every $i \in I$, let $\mathcal{G}_i$ ($\mathcal{B}_i$) be the set of indices of the message vectors in the i^{th} hyperchunk (block). We use the same notation $\mathcal{G}_i$ ($\mathcal{B}_i$) to refer to the i^{th} hyperchunk (block) unless there is a danger of confusion. We, further, let G_i (B_i) be the event that $\mathcal{G}_i$ ($\mathcal{B}_i$) is not decodable (recoverable).

Let G_I be the set of events $\{G_i\}_{i\in I}$. Let I_i be an arbitrary non-empty subset of $I\backslash\{i\}$. For every I_i, we write $i \prec I_i$, $i \succ I_i$, or $i \asymp I_i$, respectively, if $\Pr\{G_i|\cap_{j\in I_i}G_j\} < \Pr\{G_i\}$, $\Pr\{G_i|\cap_{j\in I_i}G_j\} > \Pr\{G_i\}$, or $\Pr\{G_i|\cap_{j\in I_i}G_j\} = \Pr\{G_i\}$. For a given probability measure on the set G_I, for every i, and I_i, it should be clear that either $i \prec I_i$, or $i \succ I_i$, or $i \asymp I_i$. We call the set of relationships $\{(i \prec I_i) \vee (i \succ I_i) \vee (i \asymp I_i)\}_{i\in I, I_i\subset I\backslash\{i\}}$, the characteristic set of the given probability measure on G_I. The set of all probability measures on G_I, whose characteristic set is the same, is said to have the same type of dependency, and hence any characteristic set defines a *type of dependency.*

For every $i \in I$, let $N_{\mathcal{G}}(i)$ be an ordered set (in an increasing cyclic order) of indices of hyperchunks that overlap with the i^{th} hyperchunk, and I_i be an arbitrary subset of $I \setminus \{i\}$. The first dependency type is the one that the occurrence of any subset of G_j's, for every $j \in N_{\mathcal{G}}(i)$ $(j \neq i)$, increases the probability that G_i occurs,

i.e., for every I_i such that $I_i \setminus N_{\mathcal{G}}(i) = \emptyset$, it follows that either $i \succ I_i$, or $i \asymp I_i$. The second is the one that, for every I_i such that $I_i \setminus N_{\mathcal{G}}(i) = \emptyset$, it follows that either $i \prec I_i$, or $i \asymp I_i$. In the case of both dependency types, for every i and I_i such that $I_i \cap N_{\mathcal{G}}(i) = \emptyset$, one should note that either $i \prec I_i$, or $i \asymp I_i$, and this is, indeed, the case for any possible type of dependency between the hyperchunks.[28] For any other I_i, no consideration is made. We refer to the first (second) type of dependency as the *dependency with positively (negatively) dependent neighborhoods.*

The following provides an explanation about these two dependency types. Note that a hyperchunk fails to be decodable either (i) if the number of LILE packets pertaining to some of its chunks is not sufficiently large, or (ii) the set of LILE packets pertaining to all its chunks do not form an innovative collection of sufficiently large size. Let us first consider a scenario that the condition (i) is the only reason which makes a hyperchunk undecodable. Then, in the worst case, when the number of LILE packets pertaining to any chunk belonging to this hyperchunk is not sufficiently large, all the hyperchunks which overlap with this hyperchunk are undecodable (dependency with positively dependent neighborhoods). Now, consider another scenario that the condition (ii) is the only reason which makes a hyperchunk undecodable. Then, in the best case, when the number of LILE packets pertaining to each chunk belonging to a given hyperchunk is sufficiently large, all the hyperchunks which overlap with this hyperchunk are decodable/undecodable independently (dependency with negatively dependent neighborhoods). In our problem, however, we deal with both conditions (i) and (ii), and hence the actual dependency between the hyperchunks is neither as pessimistic as the former scenario, nor as optimistic as the latter scenario.

We upper bound, for each type of dependency, (i) the probability that not all the blocks are recoverable, and (ii) the probability that a block is unrecoverable. Such bounds are "outer" upper bounds for the class of dependency structures of the underlying type in that they hold for any dependency structure in the class.

[28]The undecodability of a subset of hyperchunks that does not share any chunk with a given hyperchunk $\mathcal{G}$, increases the probability that a chunk in $\mathcal{G}$ has been allocated a sufficiently large number of LILE packets. This thus increases the probability of decodability of $\mathcal{G}$.

We say that a bound is "tight" for a given dependency structure if it fails to hold w.p. b.a.b. ϵ, for a given $0 < \epsilon < 1$. We also say that a (outer) bound is "tight" over the class of dependency structures of a given type, if, in the limit of interest (i.e., as k goes to infinity, δ goes to zero sufficiently fast, or it is bounded away from zero), the bound is tight for any worst-case structure in the class.

We derive tight outer upper bounds for each type, by studying the worst case, i.e., given that any arbitrary subset of hyperchunks is not decodable, the conditional probability of undecodability of any given hyperchunk is the largest possible. For each of the probabilities (i) and (ii), these (tight) outer upper bounds act as the two limits of an interval that for any possible type of dependency, a tight outer upper bound lies within.[29]

Theorem 3.18 *For every $0 < \delta < 1$, when δ goes to 0 sufficiently fast, as k tends to infinity, and for an arbitrary constant $0 < \gamma_c < 1$, consider applying an OCC with chunk size α, and overlap parameter τ_o, to transmit k message vectors over a line network of length L with any one-in-one-out worst-case traffic of capacity $(1+\gamma_c)k$. Then, for any type of dependency between hyperchunks, there exists a tight outer upper bound on the probability that not all the blocks are recoverable between $\delta^{\chi+\tau_o-1}q$, and $\delta^2 q$, for an arbitrary constant integer χ sufficiently larger than $(\tau_o-1)/\gamma_c$, so long as*

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\tau_o \log\left(\frac{L}{\gamma_c \delta}\tau_o\right)\right).$$

Theorem 3.19 *For every $0 < \delta < 1$, and an arbitrary constant $0 < \gamma_c < 1$, consider applying an OCC with chunk size α, and overlap parameter τ_o, to transmit k message vectors over a line network of length L with any one-in-one-out worst-case traffic of capacity $(1+\gamma_c)k$. Then, for any type of dependency between hyperchunks, there exists a tight outer upper bound on the probability that a block is unrecoverable between $\delta^{\chi+\tau_o-1}$, and δ^2, for an arbitrary constant integer χ sufficiently larger than $(\tau_o-1)/\gamma_c$,*

[29] Theorems 3.18 and 3.19 correspond to Theorems 3.15 and 3.16, respectively. The proofs have similar structure, yet, the bounds on the chunk size and the probability of failure of CC are replaced with those for OCC.

so long as

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\tau_o \log\left(\frac{L}{\gamma_c\delta}\tau_o\right)\right).$$

All-In-All-Out Worst-Case Traffics:

The results of OCC with small chunks over all-in-all-out worst-case traffics are similar to those over one-in-one-out worst-case traffics, except that the lower bound on the chunk size in this case needs to be revised due to the difference between the conditions (3.3) and (3.5) on the chunk size in Lemmas 3.15 and 3.17, respectively. For the sake of brevity, we present the main lemmas below. The resulting theorems are similar to Theorems 3.18 and 3.19, with the only difference being the lower bound on the chunk size, which needs to be modified accordingly.

Lemma 3.27 *For every $0 < \delta < 1$, and an arbitrary constant $0 < \gamma_c < 1$, by applying an OCC with chunk size α, and overlap parameter τ_o, to transmit k message vectors over a line network of length L with any all-in-all-out worst-case traffic of capacity $(1+\gamma_c)k$, the sink node fails to receive at least $\varphi_n - L\log(L/\dot{\delta}) - L$ LILE packets pertaining to a given chunk, w.p. b.a.b. $\dot{\delta}/2$, and φ_n fails to be larger than*

$$\left(1 - O\left(((1/\mu)\log(L/\delta))^{\frac{1}{2}}\right)\right)\cdot\mu, \tag{3.18}$$

w.p. b.a.b. $\dot{\delta}/2$, where $\mu = (1+\gamma_c)\alpha/\tau_o$.

Lemma 3.28 *For every $0 < \delta < 1$, and an arbitrary constant $0 < \gamma_c < 1$, by applying an OCC with chunk size α, and overlap parameter τ_o, to transmit k message vectors over a line network of length L with any all-in-all-out worst-case traffic of capacity $(1+\gamma_c)k$, the probability of a given hyperchunk of size χ to be undecodable is b.a.b. δ, so long as*

$$r_o\alpha \leq \chi\varphi - \chi L\log(L\chi/\dot{\delta}) - \log(1/\dot{\delta}) - \chi L, \tag{3.19}$$

and

$$\gamma_o \geq \tau_e\tau_o\sqrt{r_o\alpha}, \tag{3.20}$$

where

$$\varphi = \left(1 - O\left(((1/\mu)\log(L/\delta))^{\frac{1}{2}}\right)\right) \cdot \mu,$$

$\mu = (1+\gamma_c)\alpha/\tau_o$, *and* $r_o = (\chi - 1)/\tau_o + 1$.

By replacing μ with $(1+\gamma_c)\alpha/\tau_o$, and by selecting $\chi \gg (\tau_o - 1)/\gamma_c$, inequality (3.19) can be written as

$$\alpha = \Omega\left(\frac{L}{\gamma_c^2}\tau_o \log\left(\frac{L}{\gamma_c\delta}\tau_o\right)\right). \tag{3.21}$$

Similarly as before, for this choice of χ, condition (3.20) is met so long as

$$\alpha = \Omega\left(\frac{\tau_o^2}{\gamma_c}\right).$$

Thus, by applying an OCC with a chunk size as given in (3.21), over any all-in-all-out worst-case traffic, the probability of (in-isolation) decoding failure of a given hyperchunk is b.a.b. δ.

3.4.3 Comparison

Now, we compare our analytical results for CC and OCC with small chunks over one-in-one-out worst-case traffics. Similar comparisons are also valid for all-in-all-out worst-case traffics.

We consider a one-in-one-out worst-case traffic of capacity $(1+\gamma_c)k$, for an arbitrary constant $0 < \gamma_c < 1$, over a line network of length L. Similar to the notations used in Section 3.3.6, suppose that the k message vectors are divided into $\tau_o q$ chunks of size α $(= k/q)$. We partition CC and OCC (with constant overlap parameter τ_o) with small chunks into two categories based on their chunk size depending on k, L, τ_o, γ_c and δ.[30] We say that a chunk size satisfying condition (3.17) is "relatively" or "very" small, if $\delta\tau_o q$ goes to 0, or does not, as k goes to infinity, respectively.

[30]CC is a special case of OCC with overlap parameter τ_o equal to 1 (i.e., when there is no overlap between the chunks).

We compare (i) Theorem 3.15 with Theorem 3.18 for CC and OCC with relatively small chunks, and (ii) Theorem 3.16 with Theorem 3.19 for CC and OCC with very small chunks. In particular, we study the tradeoff between the probability of failure in recovering all the message vectors (MER) or the probability of failure in recovering a given message vector (PER) and the speed of convergence.

Theorem 3.20 *For a sufficiently small MER and for relatively small chunks, an OCC with a larger overlap size has higher speed of convergence than an OCC with a smaller overlap size.*

Corollary 3.2 *For a sufficiently small MER and for relatively small chunks, an OCC has a higher speed of convergence compared to a CC.*

The method of proving the following results are similar to that of Theorem 3.20 and Corollary 3.2, and hence omitted. The only difference is that, instead of Theorems 3.15 and 3.18, Theorems 3.16 and 3.19 are used for comparison.

Theorem 3.21 *For a sufficiently small PER and for very small chunks, an OCC with a larger overlap size has higher speed of convergence than an OCC with a smaller overlap size.*

Corollary 3.3 *For a sufficiently small PER and for very small chunks, an OCC has a higher speed of convergence compared to a CC.*

These results for CC and OCC with relatively/very small chunks over two types of worst-case schedules are also summarized in Table 3.1. The comparison of the rows corresponding to CC and OCC with small chunks, as shown in Theorems 3.20 and 3.21, reveals that while τ_o increases, the speed of decreasing MER/PER is higher than the speed of increasing the lower bound on α. Thus, for a given α, and for sufficiently large γ_c, the MER/PER of OCC with larger overlap size (larger τ_o) are smaller than that of OCC with smaller overlap size (e.g., CC). Moreover, for a given γ_c, and for relatively small α (i.e., when δ goes to 0 sufficiently fast, as k goes to infinity), the lower bound on α is logarithmic in k. For very small α (i.e., when δ is a

constant with respect to k), however, the lower bound on α is constant with respect to k. This implies that, for very small α, the coding cost is constant with respect to k, and hence the code is linear-time. By combining this with the preceding comparison results, one can thus conclude that linear-time OCC can outperform linear-time CC over line networks with worst-case traffics.

3.5 Simulation Results

In this section, we present our simulation results in two parts. The first part is concerned with the probability that random banded matrices are full rank; and the second part is about the performance of dense codes, chunked codes and overlapped chunked codes over line networks with worst-case traffics.

3.5.1 Random Banded Matrices

We study both irregular symmetric and asymmetric RB matrices (with entries from $\mathbb{F}_2$). The results for regular symmetric and regular asymmetric cases are similar to those for irregular symmetric and irregular asymmetric cases, respectively, and hence not presented.

Setup:

In symmetric case, we consider matrices with $k^* = 128, 256, 512$ columns, and $n^* = k^* + m^*$ rows, where $m^* = 0, 1, ..., 10$. For $k^* = 128, 256$, we consider the chunk sizes $\alpha^* = k^*/8, k^*/4, k^*/2$, and for $k^* = 512$, we consider the chunk sizes $\alpha^* = k^*/16, k^*/8, k^*/4$. In asymmetric case, we consider matrices with $k^* = 512, 1024, 2048$ columns, and $n^* = k^* + m^*$ rows, where $m^* = 0, 1, 2, 4, 8, 16$; and for each k^*, we consider the chunk sizes $\alpha^* = k^*/8, k^*/4, k^*/2$. For a given k^*, the lower bound on the overlap size in asymmetric cases is larger than that in symmetric cases, and hence we need to consider larger matrices for asymmetric cases. In both symmetric and asymmetric cases, we consider the overlap sizes $\gamma_o^* = \alpha^*/\tau_e^*$, where $\tau_e^* = \tau_o^*/(\tau_o^* - 1)$, for the overlap parameters $\tau_o^* = 2, 4, 8$.

We simulate 10000 irregular symmetric or asymmetric RB matrices for each k^*, n^*, α^*, and γ_o^*, and plot the relative frequency of the number of times that the simulated matrices have full column-rank (the probability that a (γ_o^*, α^*) irregular (symmetric or asymmetric) RB matrix of size $n^* \times k^*$ has rank k^*). For each k^* and n^*, for comparison, we also plot the theoretical (asymptotic) result (derived in [56]) for the probability that a fully random matrix of size $n^* \times k^*$ has rank k^* (in a fully random matrix, each entry is chosen uniformly at random from $\mathbb{F}_2$).

Results:

Figures 3.5-3.7 show that for given k^* and n^*, so long as the overlap size $\gamma_o^* \geq 2\sqrt{k^*}$, a (γ_o^*, α^*) ISRB matrix of size $n^* \times k^*$ behaves similar to a fully random matrix of the same size in terms of the probability of being full column-rank.

Figures 3.8-3.10 show similar results for (γ_o^*, α^*) IARB matrices of size $n^* \times k^*$. The results show that the probability that such matrices have full column-rank is similar to that of a fully random matrix of the same size when the overlap size $\gamma_o^* \geq \tau_e^* \tau_o^* \sqrt{k^*}$.

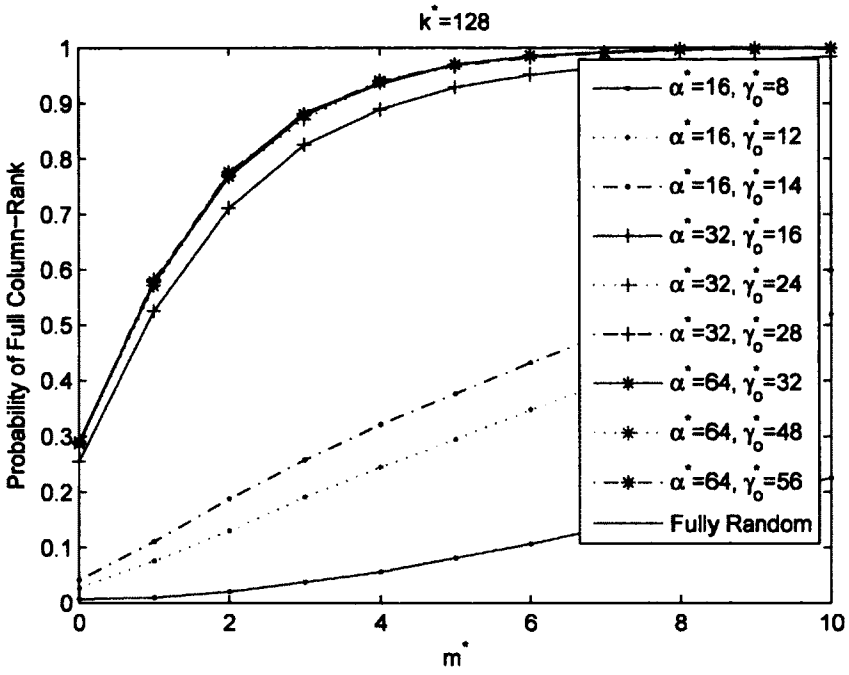

Figure 3.5: The rank property of symmetric RB matrices: $k^* = 128$.

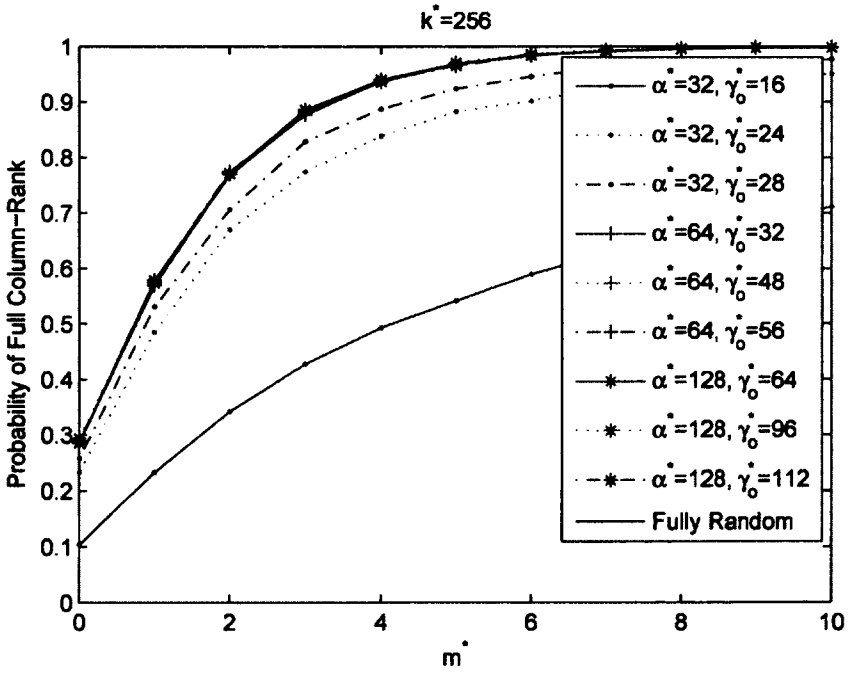

Figure 3.6: The rank property of symmetric RB matrices: $k^* = 256$.

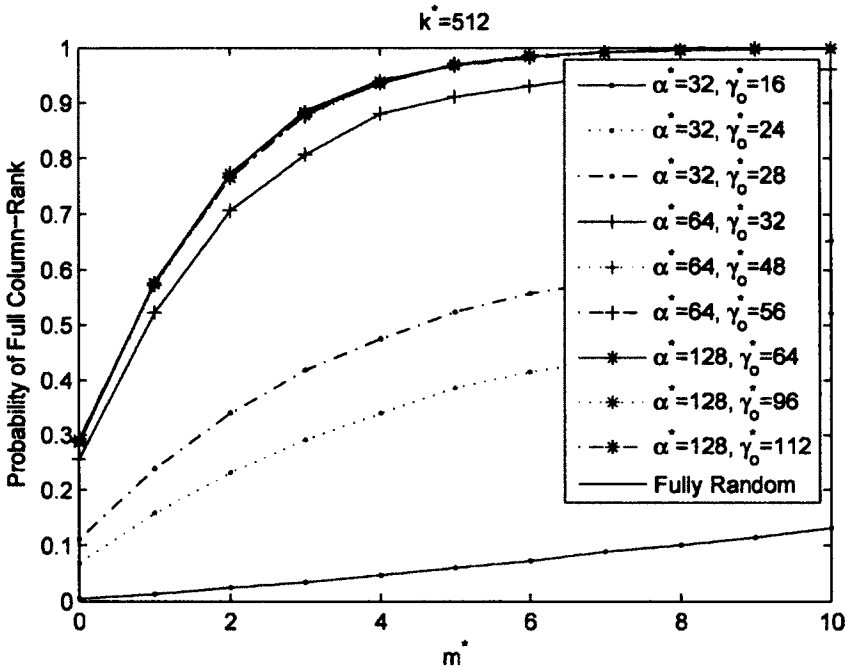

Figure 3.7: The rank property of symmetric RB matrices: $k^* = 512$.

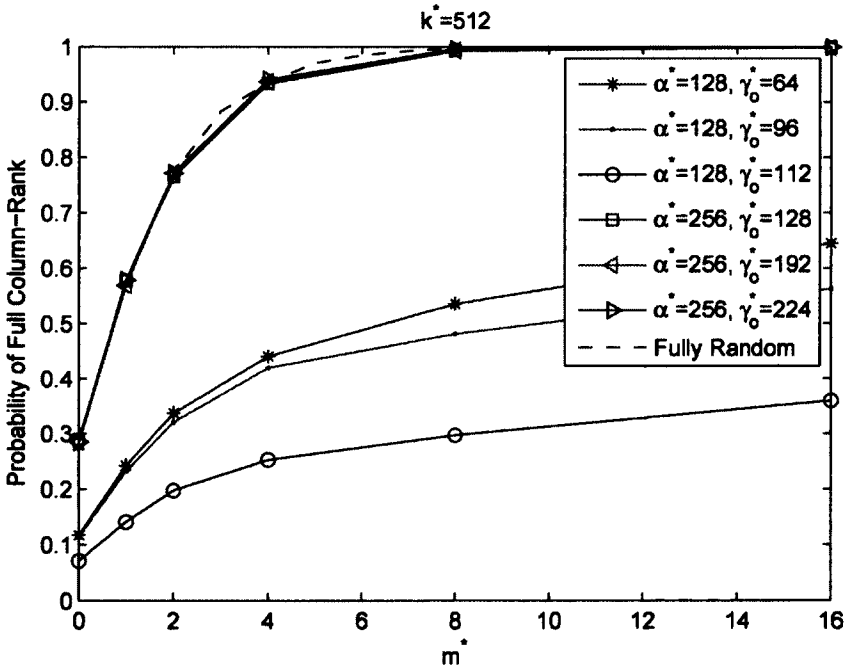

Figure 3.8: The rank property of asymmetric RB matrices: $k^* = 512$.

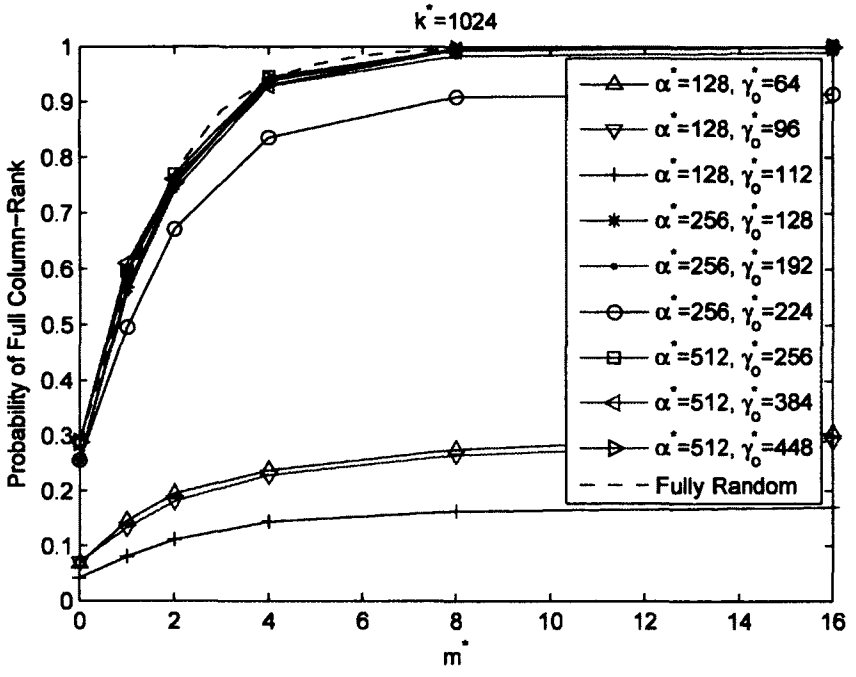

Figure 3.9: The rank property of asymmetric RB matrices: $k^* = 1024$.

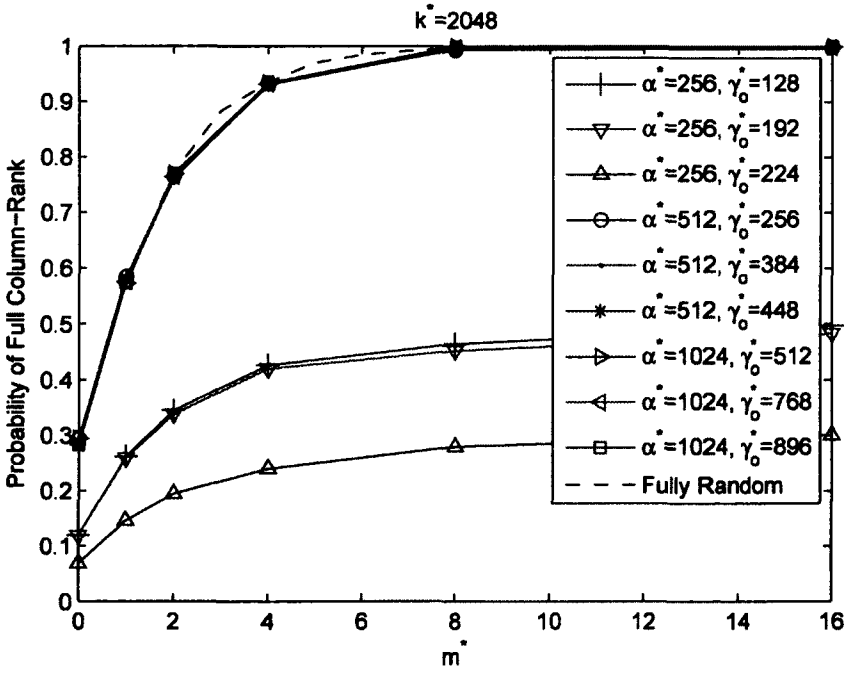

Figure 3.10: The rank property of asymmetric RB matrices: $k^* = 2048$.

3.5.2 Comparison of Dense Codes, CC and OCC

We compare the performance of finite-length dense codes, chunked codes and overlapped chunked codes over line networks with one-in-one-out and all-in-all-out worst-case traffics. The comparisons are in terms of the tradeoff between (i) the overhead per message $(n-k)/k$, and the probability of decoding failure of all the message vectors (message error rate, MER), and (ii) the overhead per message and the probability of decoding failure of a given message vector (packet error rate, PER). The variables in these comparisons are the message, chunk and overlap sizes as well as the network length and the traffic type.

Setup:

We consider line networks of lengths $L = 2, 4$. The networks are simulated with random codes over one-in-one-out or all-in-all-out worst-case traffics of capacity $n = (1+\gamma)k$, for some $0 \leq \gamma \leq 3$ (i.e., the overhead per message γ_c), and with the message sizes $k = 64, 256$. For each k, we consider the chunk sizes $\alpha = k/8, k/4, k/2, k$, and the overlap sizes $\gamma_o = \alpha/\tau_e$, where $\tau_e = \tau_o/(\tau_o - 1)$, for the overlap parameters $\tau_o = 1$ (i.e., CC), 2, and 4. We are interested in the MER and PER, each as a function of the overhead per message, for a given chunk size (i.e., for a given coding cost). We would also like to investigate how each of these functions changes with the chunk and overlap sizes as well as the length of the network and the type of the worst-case traffic. For each L, n, k, α and γ_o (and τ_o), each coding scheme is applied to the underlying networks until 1000 decoding failures occur.

One-In-One-Out Worst-Case Traffics:

Figures 3.11 and 3.12 depict the MER vs. overhead per message for the cases where $L = 2$, and k is 64, and 256, respectively (for different chunk and overlap sizes). Figures 3.13 and 3.14, respectively, depict the same scenarios as in Figures 3.11 and 3.12, only for longer networks of length $L = 4$. Similar results for PER, instead of MER, are presented in Figures 3.15-3.18. Since the trends for MER and PER are similar, in the following, we just discuss the MER results.

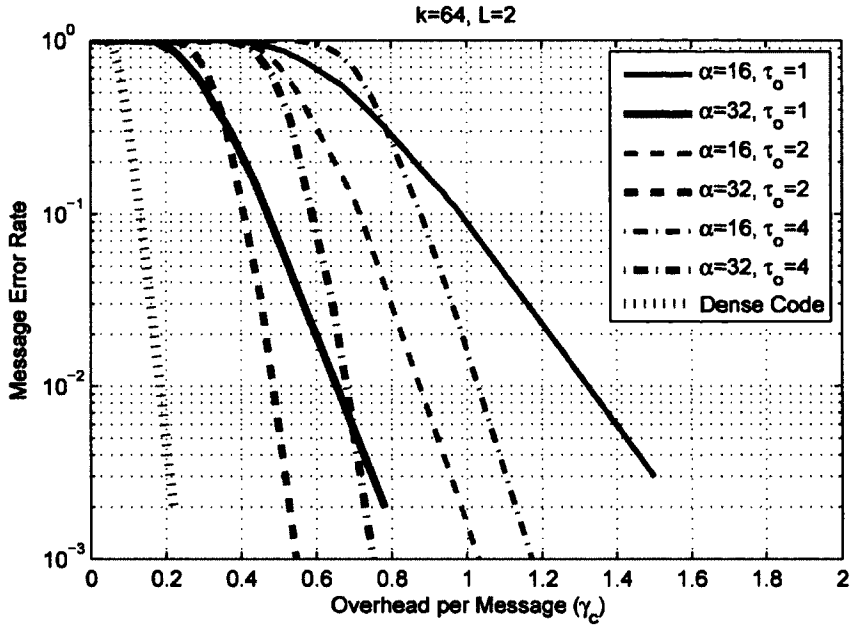

Figure 3.11: CC and OCC over one-in-one-out traffics: smaller message size and shorter network.

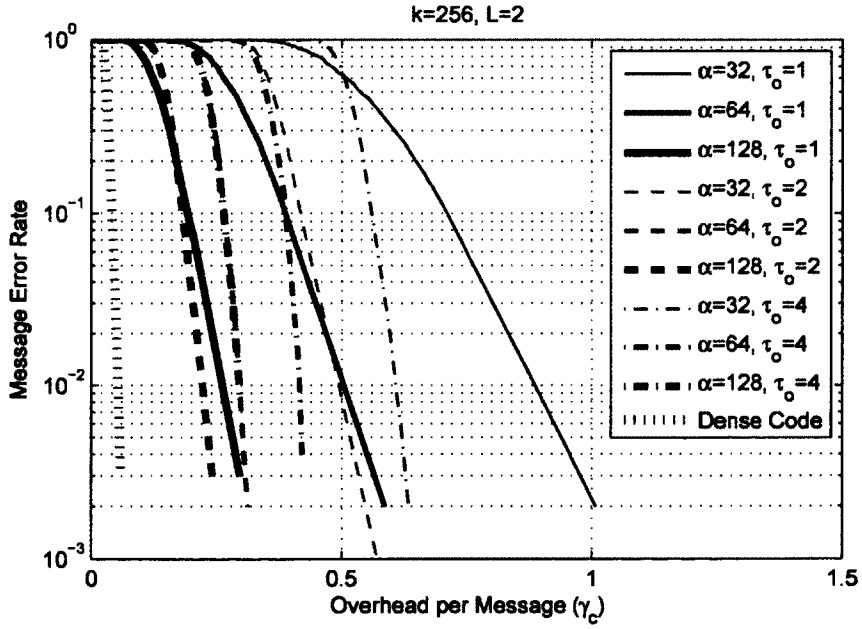

Figure 3.12: CC and OCC over one-in-one-out traffics: larger message size and shorter network.

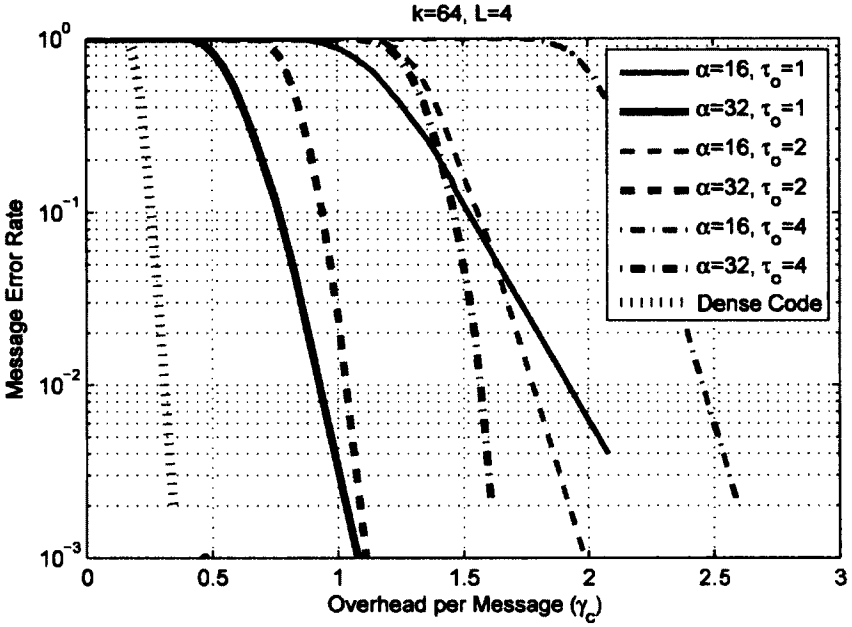

Figure 3.13: CC and OCC over one-in-one-out traffics: smaller message size and longer network.

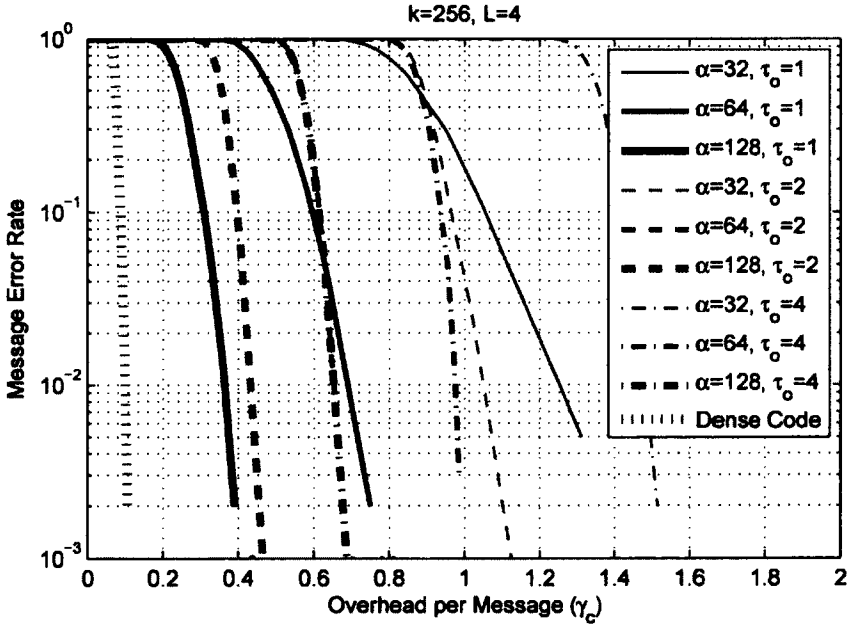

Figure 3.14: CC and OCC over one-in-one-out traffics: larger message size and longer network.

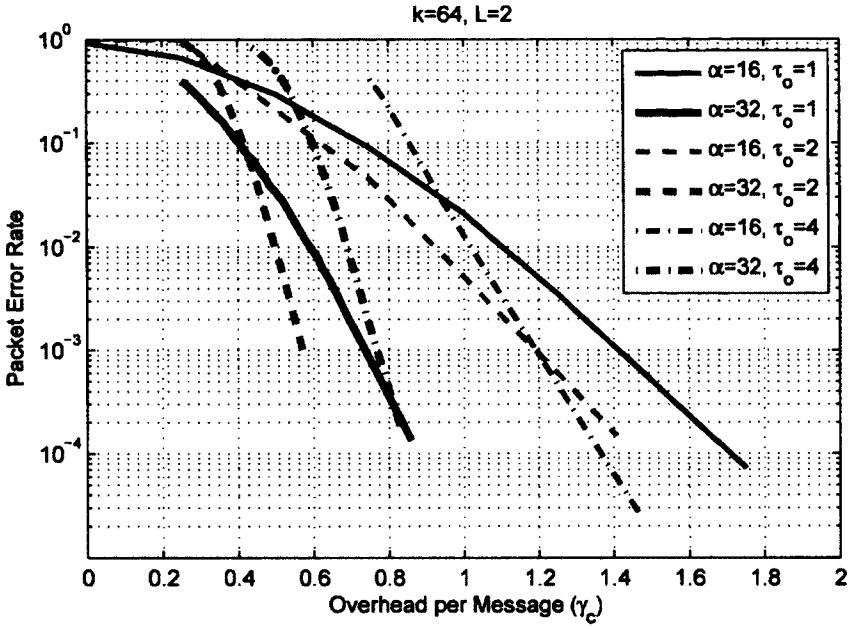

Figure 3.15: CC and OCC over one-in-one-out traffics: smaller message size and shorter network.

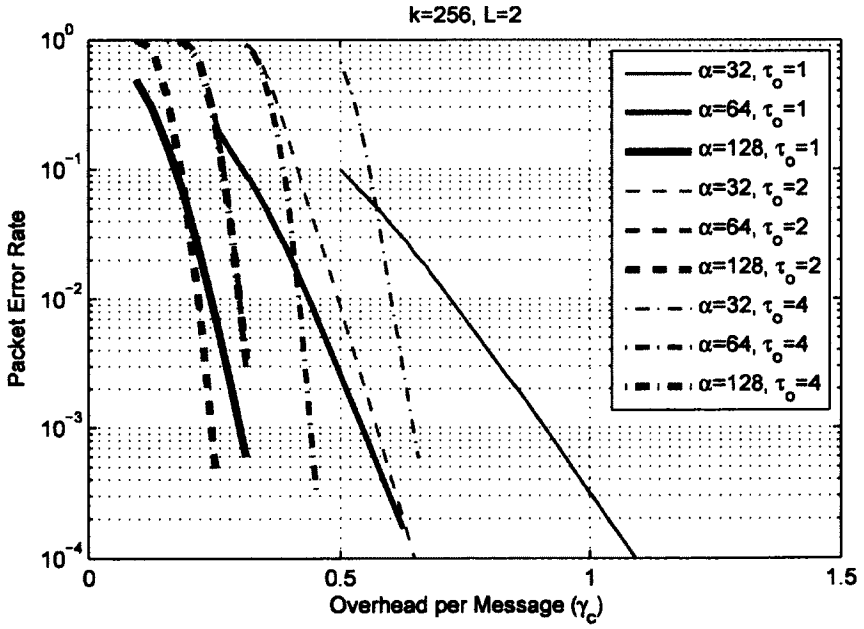

Figure 3.16: CC and OCC over one-in-one-out traffics: larger message size and shorter network.

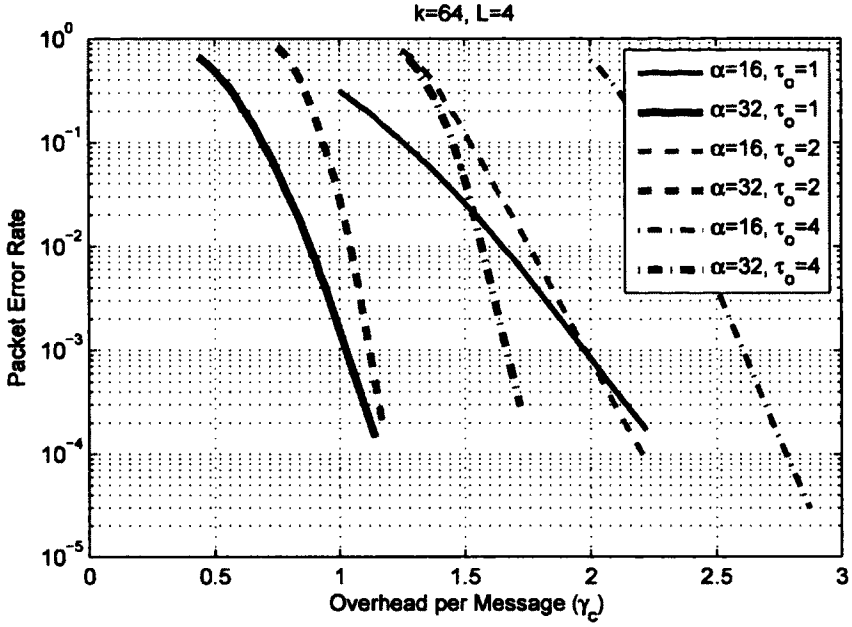

Figure 3.17: CC and OCC over one-in-one-out traffics: smaller message size and longer network.

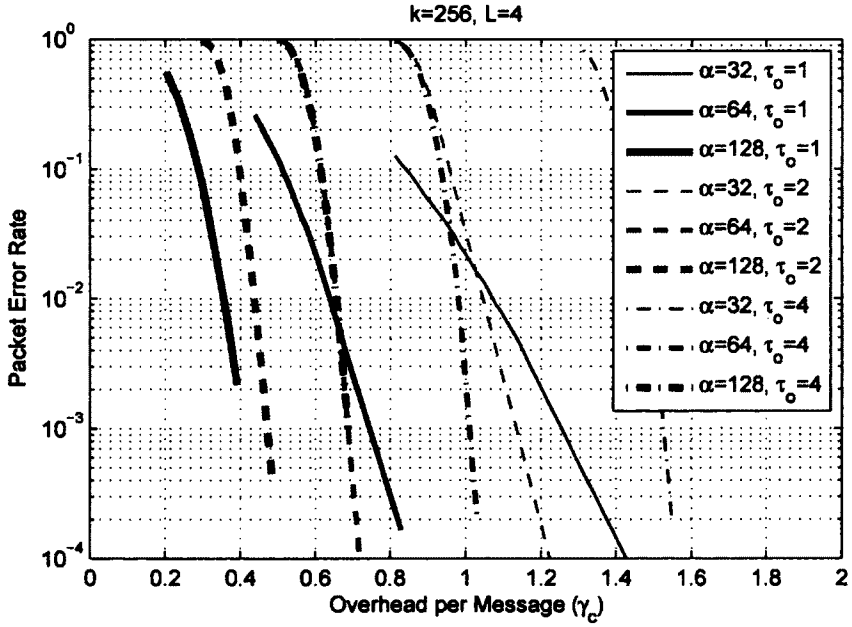

Figure 3.18: CC and OCC over one-in-one-out traffics: larger message size and longer network.

For given k and L, investigating the results in each of the Figures 3.11-3.14 shows that for the same chunk size (same coding cost), OCC with larger overlap size is more efficient (requires less overhead per message) than CC (OCC with smaller overlap), for sufficiently small MER. More detailed analysis of the simulation results is provided in the following.

The Effect of the Message Size: A comparison of Figures 3.11 and 3.12 shows that for a given chunk size, and a given network length, the MER below which OCC (OCC with larger overlap size) outperforms CC (OCC with smaller overlap size) is an increasing function of the message size. So, OCC are more advantageous over CC for a given coding cost (a given chunk size), when the message size is larger. For example, in Figure 3.11, it can be seen that for $k = 64$, and $\alpha = 32$, for MERs below about 0.35, OCC with $\tau_o = 2$ requires a smaller overhead per message compared to CC ($\tau_o = 1$). However, as can be seen in Figure 3.12, for larger message size $k = 256$, and similar chunk size $\alpha = 32$, OCC with $\tau_o = 2$ is superior to CC, for MERs below about 1 (for almost all MERs). By comparing Figures 3.13 and 3.14, similar behavior can be observed for longer networks.

The Effect of the Network Length: Comparison between Figures 3.11 and 3.13, or between Figures 3.12 and 3.14 demonstrates that OCC (OCC with larger overlap size) are more advantageous over CC (OCC with smaller overlap size) for shorter networks. For example, Figure 3.11 (shorter network) shows that for $\alpha = 16$, OCC with $\tau_o = 2$ is superior to CC for MERs below 0.3; however, as can be seen in Figure 3.13 (longer network), this occurs for MERs below 0.035.

The Effect of the Overlap Size: In Figures 3.11-3.14, one can see that for a given chunk size, the MER below which OCC (OCC with larger overlap size) outperforms CC (OCC with smaller overlap size) is a decreasing function of the overlap size. Therefore, for a given coding cost, and sufficiently small MERs, OCC with larger overlap sizes are more advantageous. For example, as can be seen in Figure 3.12, for $\alpha = 32$, OCC with $\tau_o = 4$ (larger overlap size) outperforms CC for MERs below about 0.6; and OCC with $\tau_o = 2$ (smaller overlap size) is superior to CC for MERs below about 1. Moreover, the slope of the curves for OCC with $\tau_o = 4$ and OCC with $\tau_o = 2$ shows that the former crosses (outperforms) the latter for a MER somewhere

below 0.001.

The Effect of the Chunk Size: It can be seen in Figures 3.11-3.14 that for a given overlap parameter, and given message size and network length, the smaller the chunk size (the smaller the coding cost), the larger the overhead per message.

All-In-All-Out Worst-Case Traffics:

Figures 3.19-3.22 and 3.23-3.26 represent similar scenarios to those in Figures 3.11-3.14 and 3.15-3.18, for MER and PER, respectively.

While the general trends for all-in-all-out traffics are similar to those of one-in-one-out traffics, discussed in the previous part, those trends are more pronounced for the all-in-all-out traffics. As an example, comparisons between Figures 3.11 and 3.19 reveal that OCC with $\alpha = 16$ and $\tau_o = 4$ outperforms CC of the same chunk size for MERs below 0.3 for one-in-one-out traffics. This crossover MER for the all-in-all-out traffics is improved to 0.85. The corresponding values of MER if α is increased to 32 are 0.005 and 0.9, for the two types of traffics, respectively. Therefore, for a randomly generated worst-case traffic, one expects to see improvements in the performance/complexity tradeoff of OCC versus CC that are in between those reported for one-in-one-out and all-in-all-out traffics.

Example: Consider downloading a 1MB file from a file server 4 hops away ($L = 4$). Assuming the worst-case scenario, the underlying network is under a worst-case traffic. Suppose that packets of length 4KB are used for the transmission. This implies that the number of message vectors $k = 256$. Consider a target PER of 10^{-4}, and two possible transmission scenarios: (a) using a CC with $\alpha = 64$, and (b) using an OCC with $\alpha = 64$ and $\tau_o = 2$ ($\gamma_o = 32$). From Figure 3.18 (for one-in-one-out traffics), one can see that the overhead per message γ_c for the two scenarios (a) and (b) is about 0.85 and 0.7, respectively. From Figure 3.26 (for all-in-all-out traffics), it can be seen that the overhead per message γ_c for the two scenarios is about 0.5 and 0.35, respectively. This implies that downloading the file by scenario (b) is from about 17% to about 30% faster than scenario (a), depending on the type of the worst-case traffic under consideration.

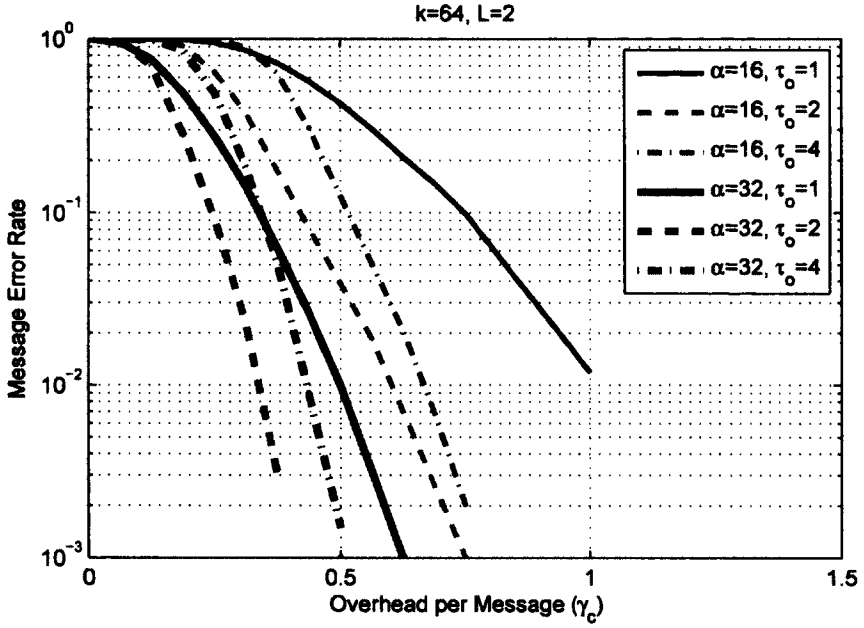

Figure 3.19: CC and OCC over all-in-all-out traffics: smaller message size and shorter network.

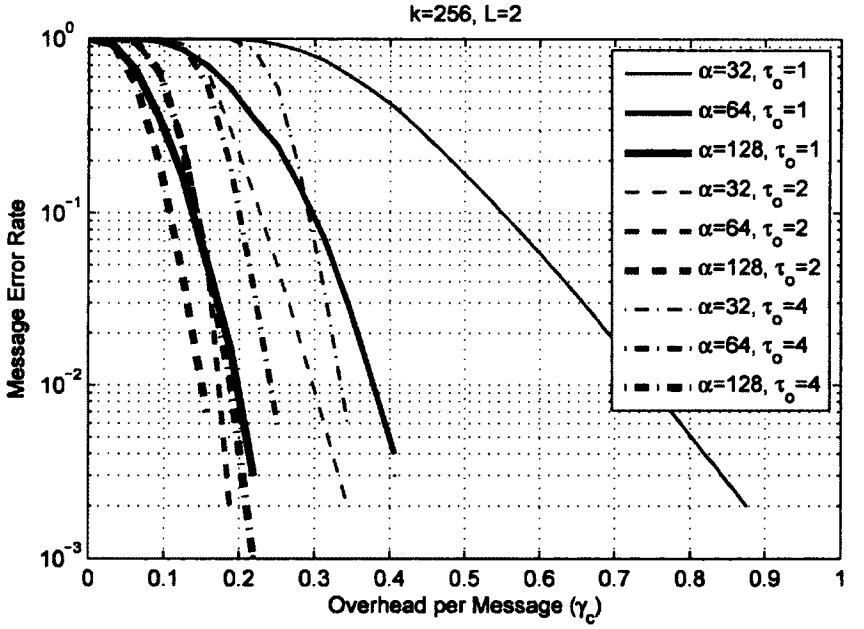

Figure 3.20: CC and OCC over all-in-all-out traffics: larger message size and shorter network.

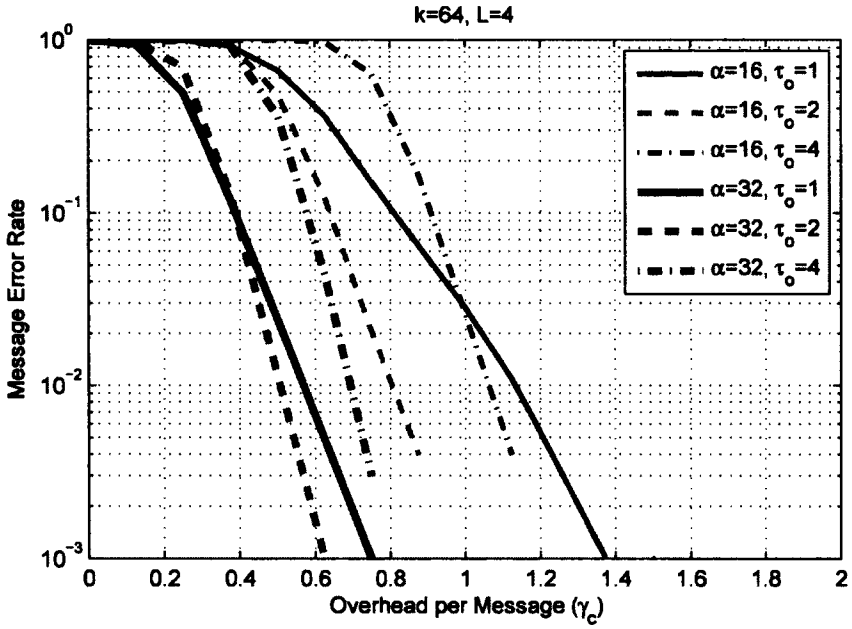

Figure 3.21: CC and OCC over all-in-all-out traffics: smaller message size and longer network.

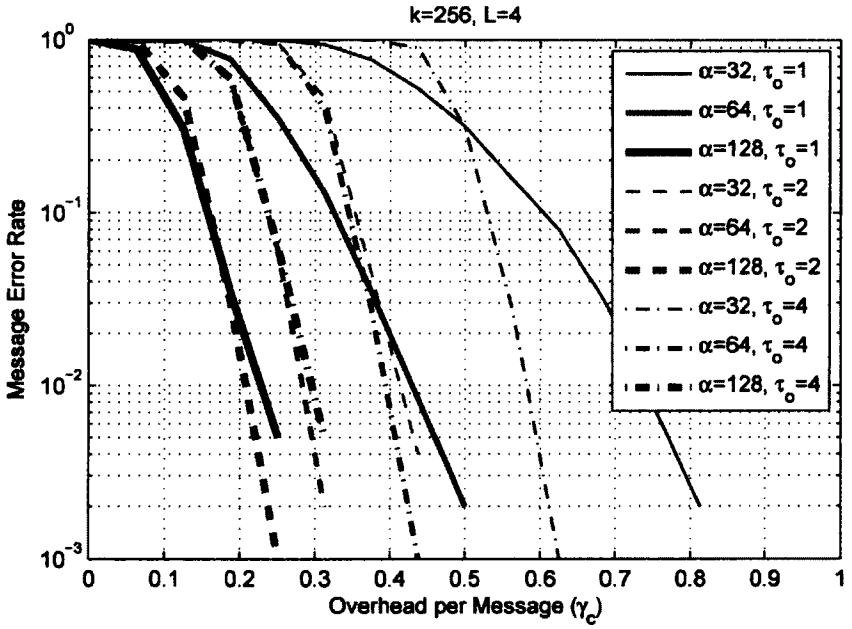

Figure 3.22: CC and OCC over all-in-all-out traffics: larger message size and longer network.

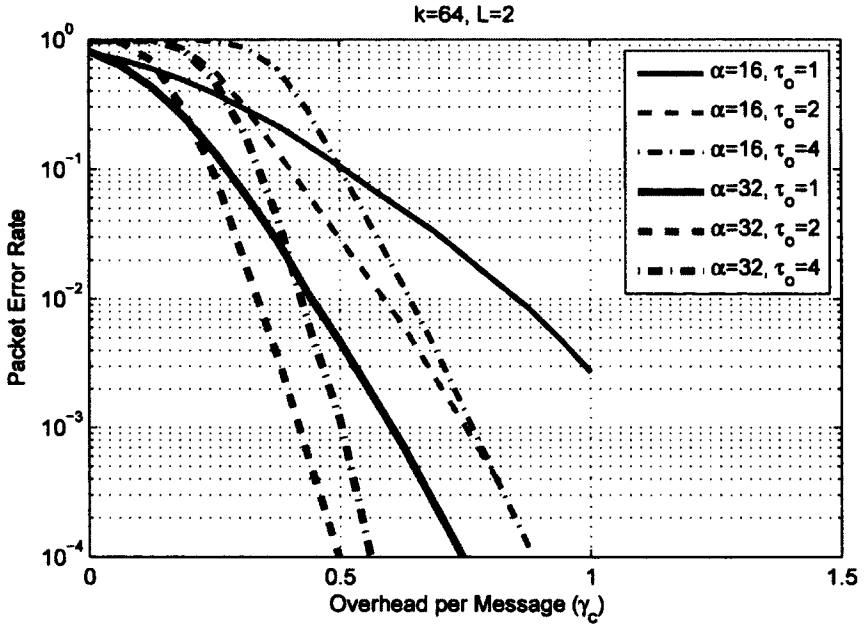

Figure 3.23: CC and OCC over all-in-all-out traffics: smaller message size and shorter network.

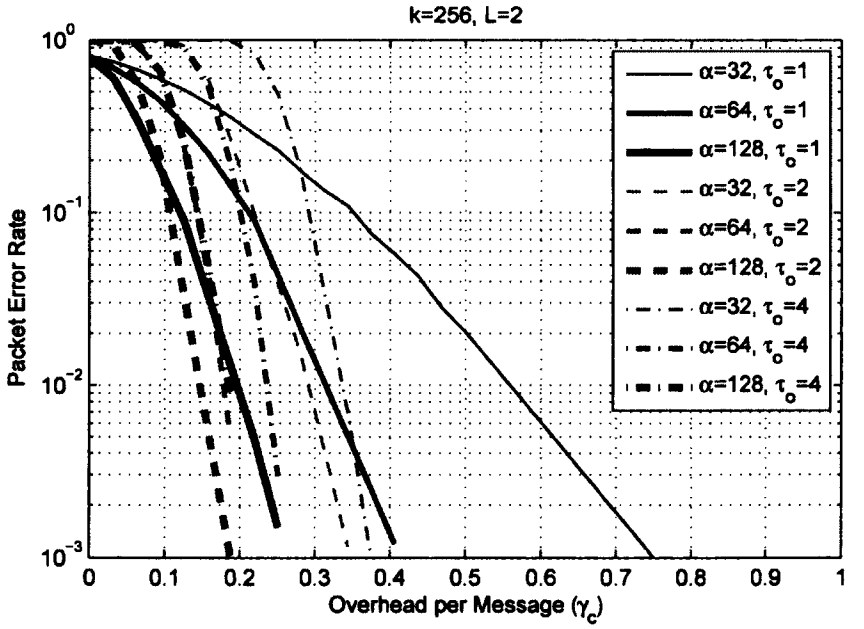

Figure 3.24: CC and OCC over all-in-all-out traffics: larger message size and shorter network.

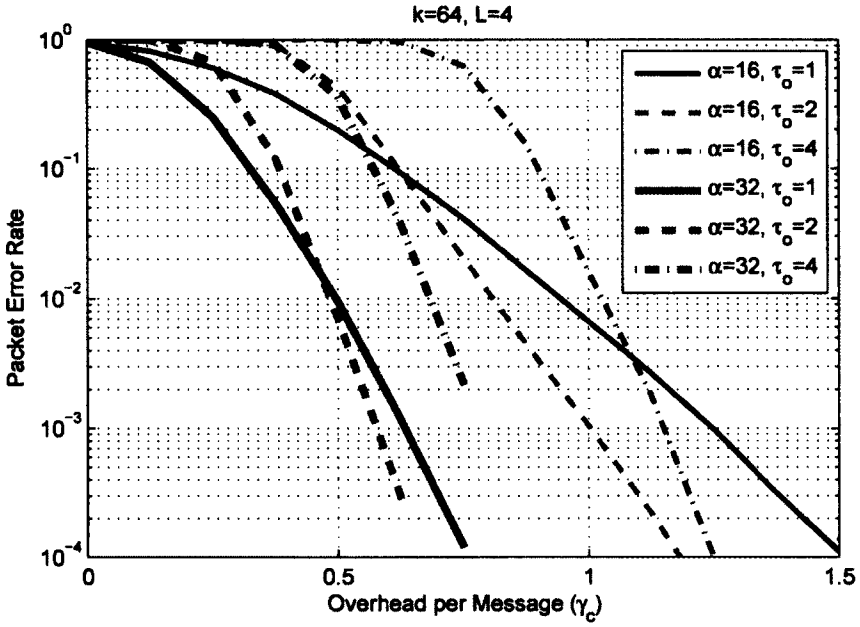

Figure 3.25: CC and OCC over all-in-all-out traffics: smaller message size and longer network.

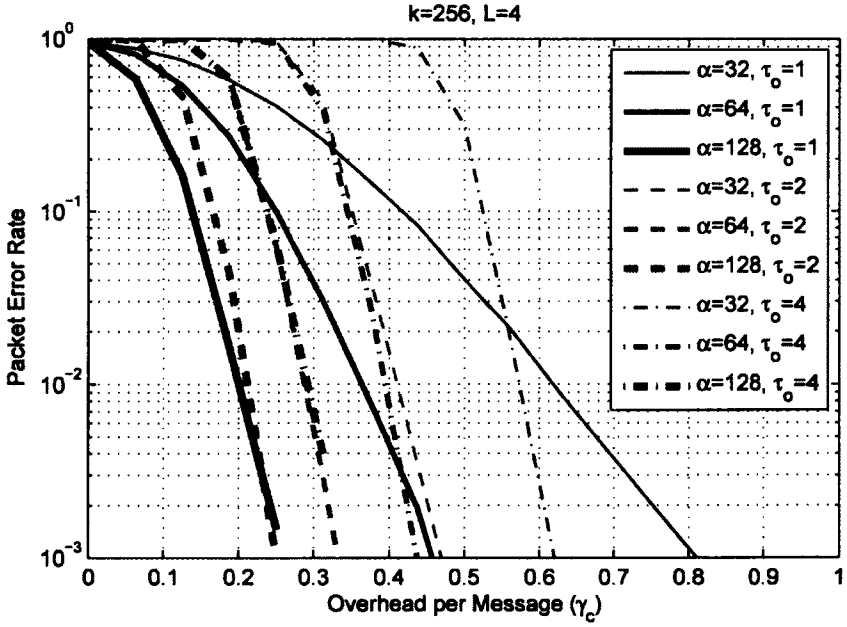

Figure 3.26: CC and OCC over all-in-all-out traffics: larger message size and longer network.

Chapter 4

Network Coding over Networks with Probabilistic Traffics

4.1 Network Model and Assumptions

4.1.1 Transmission, Loss and Delay Models

We consider a unicast problem similar to that in Chapter 3 over a line network with L links connecting $L+1$ nodes $\{v_i\}_{0\leq i\leq L}$ in tandem. The source node v_0 has a message of k vectors, called *message vectors*, from a vector space $\mathcal{F}$ over $\mathbb{F}_2$, and the sink node v_L requires all the message vectors.

Each (non-sink) node at each transmission time transmits a (coded) packet, which is a vector in $\mathcal{F}$. Unlike the traffic model used in Chapter 3 with arbitrary deterministic transmission model, in this chapter, the packet transmissions are assumed to occur in discrete-time, and the transmission times over different links are assumed to follow independent stochastic processes. The transmission times over the i^{th} link are specified by (i) a deterministic process where there is a packet transmission at each time instant, or (ii) a Poisson process with parameter $\lambda_i : 0 < \lambda_i \leq 1$, where λ_i is the average number of transmissions per time unit over the i^{th} link. The transmission schedules resulting from (i) and (ii) are referred to as *deterministic regular* and *Poisson*, respectively.

Each transmitted packet either succeeds (*successful packet*) or fails (*erased*

packet) to be received. Unlike the arbitrary deterministic models for delay and loss considered in Chapter 3, the delay and loss models of our interest in this chapter are as follows: The successful packets are assumed to arrive with zero delay (and hence the name *delay free transmissions*), and the erased packets will never arrive. The packets are assumed to be successful independently over different links. The successful packets over the i^{th} link are specified by a Bernoulli process with (success) parameter $p_i : 0 < p_i \leq 1$, where p_i is the average number of successes per transmission over the i^{th} link. The loss model defined as above is referred to as *Bernoulli.*

For each traffic model as above, the parameters $\{p_i\}$ or $\{\lambda_i p_i\}$ are called the *traffic parameters.* In the case of traffics with parameters $\{p_i\}$ or $\{\lambda_i p_i\}$, the *min-cut capacity* is equal to the ratio of the message size k to the minimum traffic parameter $\min_{1\leq i\leq L} p_i$ or $\min_{1\leq i\leq L} \lambda_i p_i$, respectively [18]. For simplifying the terminology, hereafter, we refer to the "min-cut capacity" as the "capacity."

4.1.2 Assumptions

Similar to the case in Chapter 3, in this chapter, we assume that (i) the network nodes are blind to the traffic, (ii) there is no (a priori or a posteriori) information about the state of the channels at the network nodes before the time that the sink node recovers all the message packets (e.g., there is no feedback information at the network nodes during the course of transmissions), and (iii) the size of the memory at the network nodes is unbounded.

4.2 Problem Setup

The goal in this chapter is to derive upper bounds on the coding delay and the average coding delay of dense codes and chunked codes (with disjoint chunks) over line networks with delay-free deterministic regular or Poisson transmissions and Bernoulli losses.

For some fixed $0 < \epsilon < 1$, the coding delay of a class of codes over a network with

a class of traffics is upper bounded by N with failure probability (w.f.p.) bounded above by (b.a.b.) ϵ, so long as the coding delay of a randomly chosen code over the network with a randomly chosen traffic is larger than N with probability (w.p.) b.a.b. ϵ. The average coding delay of a class of codes over a network with respect to a class of traffics is upper bounded by N w.f.p. b.a.b. ϵ, so long as the average coding delay of a randomly chosen code over the network with respect to the class of traffics is larger than N w.p. b.a.b. ϵ.

4.3 Delay-Free Deterministic Regular Transmissions and Bernoulli Losses

In this section, for each coding scheme, we first consider arbitrary traffic parameters $\{p_i\}$; and next, we consider a special case with unequal traffic parameters.

4.3.1 Dense Codes

As was previously discussed in Chapter 3, in a dense coding scheme, the source node, at each transmission opportunity, transmits a packet by randomly linearly combining the message vectors, and each non-source non-sink (interior) node transmits a packet by randomly linearly combining its previously received packets.The sink node can recover all the message vectors as long as it receives an innovative collection of packets of the size equal to the number of message vectors at the source node.

The first step in our analysis in this chapter, with a similar approach to that used in Chapter 3, is to lower bound the size of a maximal collection of globally dense packets (the packets whose global encoding vectors' entries are independent and uniformly distributed (i.u.d.) Bernoulli random variables) at any non-source node until a certain time, referred to as the *decoding time.* Based on the result of Lemma 3.2, the size of a maximal collection of globally dense packets at a node can be lower bounded by the number of LILE packets (the packets whose local encoding vectors are linearly independent) at that node. The set of LILE packets are thus of main importance in our analysis. In particular, by studying the linear

dependence/independence of the local encoding vectors of the successful packets over a link, the number of LILE packets, and further, the size of a maximal collection of globally dense packets, at the receiving node of that link can be lower bounded. We, next, lower bound the decoding time such that the probability that the underlying collection fails to include an innovative sub-collection of a sufficiently large size (equal to the message size) is upper bounded (this probability upper bounds the probability of the failure of a dense code to recover all the message packets till the underlying decoding time).

Following the same notation as in Chapter 3, let us define $\mathcal{O}_i$ ($\mathcal{I}_i$) as the set of labels of the successful (i.e., not lost) packets transmitted (received) by the i^{th} node till the decoding time, and define $\mathcal{D}_i$ as the set of labels of the LILE packets at the i^{th} node till the decoding time. Let Q_{i+1} and Q_i, with entries over $\mathbb{F}_2$, be the decoding matrices at the $(i+1)^{\text{th}}$ and i^{th} nodes, respectively, and T_i, a matrix over $\mathbb{F}_2$, be the transfer matrix at the i^{th} node (i.e., $Q_{i+1} = T_i Q_i$). The rows of T_i are the local encoding vectors of the successful packets transmitted by the i^{th} node. Let $\hat{Q}_i$ be the modified decoding matrix at the i^{th} node (i.e., $\hat{Q}_i$ is Q_i restricted to its rows pertaining to the global encoding vectors of the LILE packets). Let $\hat{T}_i$, a matrix over $\mathbb{F}_2$, be the modified transfer matrix at the i^{th} node (i.e., $Q_{i+1} = \hat{T}_i \hat{Q}_i$). The ℓ^{th} row of $\hat{T}_i$ indicates the labels of LILE packets at the i^{th} node which contribute to the ℓ^{th} successful packet sent by the i^{th} node, and the j^{th} column of $\hat{T}_i$ indicates the labels of successful packets sent by the i^{th} node to which the j^{th} LILE packet contributes.

Since $Q_{i+1} = \hat{T}_i \hat{Q}_i$, and $\hat{Q}_i$ is dense,[1] by applying the result of Lemma 3.2, it follows that $\mathcal{D}(Q_{i+1})$ can be bounded from below so long as $\text{rank}(\hat{T}_i)$ is bounded from below. The rank of the modified transfer matrix $\hat{T}_i$ is a function of the structure of $\hat{T}_i$, and the structure of such a matrix depends on the number of LILE packet arrivals at the i^{th} node and the number of successful packet departures from the i^{th} node before or after any given point in time. Such parameters depend on the traffic over the i^{th} and $(i+1)^{\text{th}}$ links, and are therefore random variables. It is however not

[1]The rows in $\hat{Q}_i$ are the global encoding vectors of the LILE packets at the i^{th} node, and based on an earlier argument, the set of LILE packets at a node belong to the set of globally dense packets at that node. Thus, the entries of all the rows in $\hat{Q}_i$ are i.u.d. over $\mathbb{F}_2$.

straightforward to find the distribution of such random variables. We thus use a probabilistic technique as follows to study such variables.

Let $(0, N_T]$ be the period of time over which the transmissions occur (N_T is the decoding time). We split the time interval $(0, N_T]$ into w disjoint subintervals (partitions) of length N_T/w. The first partition represents the time interval $(0, N_T/w]$; the second partition represents the time interval $(N_T/w, 2N_T/w]$, and so forth. For every $1 \leq i < L$ and $1 \leq j < w$, all the arrivals over the i^{th} link in the first j partitions, i.e., in the time interval $(0, jN_T/w]$, occur before any departure over the $(i+1)^{\text{th}}$ link in the $(j+1)^{\text{th}}$ partition, i.e., in the time interval $(jN_T/w, (j+1)N_T/w]$. Thus the number of arrivals at the i^{th} node before any given point in time within the $(j+1)^{\text{th}}$ partition is bounded from below by the sum of the number of arrivals at this node in the first j partitions.

This method of counting is however suboptimal since there might be some extra arrivals in the $(j+1)^{\text{th}}$ partition, which arrive before the given point in time within this partition. To control the impact of sub-optimality, the length of partitions needs to be chosen with some care. To be specific, the length of partitions, on the one hand, needs to be sufficiently small such that there is not a large number of arrivals in one partition compared to the total number of arrivals in all the partitions. This should be the case so that ignoring a subset of arrivals in one partition does not result in a significant difference in the number of arrivals before each point in time within the same partition. On the other hand, the partitions need to be long enough such that the deviation of the number of arrivals from the expectation in one partition is negligible in comparison with the expectation itself. This ensures the validity of our analysis and the tightness of our results.

Let I_{ij} represent the j^{th} partition pertaining to the i^{th} link for every i and j. We focus on the set of all the packets over the i^{th} link in the *active partitions* pertaining to this link, where, for every i, j, I_{ij} is an active partition if and only if $i \leq j \leq w-L+i$. Such a partition is active in the sense that (i) there exists some other partition over the upper link so that all its packets arrive before the departure of any packet in the underlying active partition, and (ii) there exists some other partition over the lower link so that all its packets depart after the arrival of any packet in the underlying

active partition. In particular, the first $w - L + 1$ partitions pertaining to the first link are all active; the $w - L + 1$ partitions pertaining to the second link starting from the second partition are all active and so forth. Let w_T represent the total number of active partitions pertaining to all the links, i.e.,

$$w_T \doteq L(w - L + 1). \tag{4.1}$$

We start off with lower bounding the number of successful packets in all the w_T active partitions. Let I_{ij} be an active partition, and φ_{ij} be the number of (successful) packets in I_{ij}. Since the length of the partition I_{ij} is N_T/w, and by the assumption the packet successes over the i^{th} link follow a Bernoulli process with the parameter p_i, φ_{ij} is a binomial random variable with the expected value $\varphi_i \doteq p_i N_T/w$. Let

$$p \doteq \min_{1 \le i \le L} p_i, \tag{4.2}$$

and

$$\varphi \doteq pN_T/w. \tag{4.3}$$

By applying Lemma A.3, one can show that φ_{ij} is not larger than or equal to

$$r \doteq (1 - \gamma^*)\varphi \tag{4.4}$$

w.p. b.a.b. $\dot{\epsilon}/w_T$, so long as $0 < \gamma^* < 1$ is chosen such that r is an integer, and γ^* goes to 0 as N_T goes to infinity, where

$$\gamma^* \sim \left(\frac{1}{\dot{\varphi}} \ln \frac{w_T}{\dot{\epsilon}}\right)^{\frac{1}{2}}. \tag{4.5}$$

For all i, j, suppose that φ_{ij} is larger than or equal to r. This assumption fails if the number of packets in some active partition is less than r. Hence, the failure occurs w.p. b.a.b. $\dot{\epsilon}$.

Next, for every $1 < i \le L$ and $1 \le j \le w - L + 1$, we lower bound the number of LILE packets in the first j active partitions over the i^{th} link. Before explaining the

lower bounding technique in detail, let us introduce two lemmas which will be useful to lower bound the rank of the modified transfer matrix at each node (depending on whether the number of LILE packet arrivals at the i^{th} node over the i^{th} link in a given partition is larger or smaller than the number of packet departures from that node over the $(i+1)^{\text{th}}$ link in the partition with the same index as the underlying partition pertaining to the i^{th} link).

Let w^*, r^* and $\{r_l^*\}_{1\leq l\leq w^*}$ be arbitrary non-negative integers, and let $r_{\max}^* = \max_{1\leq l\leq w^*} r_l^*$ and $r_{\min}^* = \min_{1\leq l\leq w^*} r_l^*$. For any pair (i',j') such that $1 \leq j' \leq i' \leq w^*$, let $T_{i',j'}$ be an $r^* \times r_{j'}^*$ dense matrix over $\mathbb{F}_2$; for any other pair (i',j'), let $T_{i',j'}$ be an arbitrary $r^* \times r_{j'}^*$ matrix over $\mathbb{F}_2$.[2] Let $T = [T_{i',j'}]_{1\leq i',j'\leq w^*}$. The matrix T is called *random block lower-triangular* (RBLT) (see Figure 4.1). (The proofs of the results in this chapter are given in Appendix B)

Lemma 4.1 *Let T be an RBLT matrix with parameters w^*, r^* and $\{r_l^* : 0 \leq r_l^* \leq r^*\}_{1\leq l\leq w^*}$. Let $n^* = \sum_{1\leq l\leq w^*} r_l^*$. For every integer $0 \leq \gamma \leq n^* - 1$,*

$$\Pr\{\text{rank}(T) < n^* - \gamma\} \leq u^* \left(1 - 2^{-r_{\max}^*}\right) 2^{-\gamma + n^* - w^* r^* + (r^* - r_{\min}^*)(u^* - 1)},$$

where $u^* = \lceil (n^* - \gamma)/r_{\min}^* \rceil$.

Lemma 4.2 *Let T be an RBLT matrix with parameters w^*, r^* and $\{r_l^* : 0 \leq r^* \leq r_l^*\}_{1\leq l\leq w^*}$. Let $n^* = w^* r^*$. For every integer $0 \leq \gamma \leq n^* - 1$,*

$$\Pr\{\text{rank}(T) < n^* - \gamma\} \leq u^* \left(1 - 2^{-r^*}\right) 2^{-\gamma + n^* - w^* r_{\min}^* + (r_{\min}^* - r^*)(u^* - 1)},$$

where $u^* = \lceil (n^* - \gamma)/r^* \rceil$.

Now, for every $1 < i \leq L$ and $1 \leq j \leq w - L + 1$, we explain how to lower bound the number of LILE packets in the first j active partitions over the i^{th} link. The lower bounding technique works in a recursive manner as follows:

[2]For any pair (i',j') such that $1 \leq i' < j' \leq w^*$, the entries of $T_{i',j'}$ might be dependent on the entries of $T_{i'',j''}$, for any other pair (i'',j'') such that $1 \leq j'' \leq i'' \leq w^*$.

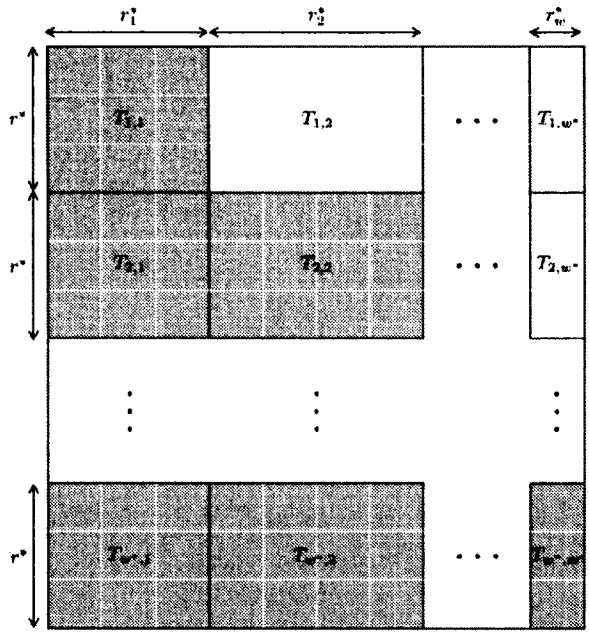

Figure 4.1: The structure of a random block lower-triangular (RBLT) matrix T with parameters w^*, r^* and $\{r_l^*\}_{1 \leq l \leq w^*}$. The shaded blocks represent the dense sub-matrices of T with i.u.d. Bernoulli entries, and the blank blocks represent those sub-matrices of T with arbitrarily dependent or independent entries with respect to the entries of the dense sub-matrices of T.

For every $1 \leq l \leq j$, suppose that the number of LILE packets in the first l active partitions over the $(i-1)^{\text{th}}$ link is lower bounded. Let $\hat{T}_i^j$ be the modified transfer matrix at the i^{th} node, restricted to the successful packet transmissions within the first j active partitions over the i^{th} link (by the assumption, the number of such packets in each partition is bounded from below by r). Then, one can see that the matrix $\hat{T}_i^j$ includes a sub-matrix $\hat{T}'$ with a structure similar to that in Lemma 4.1 or the one in Lemma 4.2.[3] This can be seen precisely by the following replacements in Lemma 4.1 or Lemma 4.2: (i) w^* with j (i.e., the number of underlying active partitions), (ii) r^* with r (i.e., the lower bound on the number of successful packet

[3]In the case of identical links, the modified transfer matrix at each node includes a sub-matrix similar to that in Lemma 4.1. However, in the case of non-identical links, depending on the traffic parameters, the modified transfer matrix at a node might include a sub-matrix similar to that in Lemma 4.1 or the one in Lemma 4.2.

transmissions in each of the first j active partitions pertaining to the i^{th} link), and (iii) r_l^*, for every $1 \leq l \leq w^*$, with the difference between the two lower bounds on the number of LILE packets in the first l and the first $l-1$ active partitions pertaining to the $(i-1)^{\text{th}}$ link (note that r_1 is equal to the lower bound on the number of LILE packets in the first active partition).

The lower bounding process then proceeds as follows. Each successful packet in any of the first j active partitions, say the m^{th} active partition, for some $1 \leq m \leq j$, pertaining to the i^{th} link can be written as a linear combination of the LILE packets in the l^{th} active partition pertaining to the $(i-1)^{\text{th}}$ link, for all $1 \leq l \leq m$, and perhaps some extra LILE packets in the $(m+1)^{\text{th}}$ active partition pertaining to the $(i-1)^{\text{th}}$ link. Thus, for every $1 \leq l \leq m$, each row of the sub-matrix $\hat{T}'_{i-1,l}$ (in the matrix $\hat{T}'$) indicates the labels of (some subset of)[4] the LILE packets in the l^{th} active partition pertaining to the $(i-1)^{\text{th}}$ link which contribute to the linear combination of one packet (from the set of the r chosen successful packets) in the m^{th} active partition pertaining to the i^{th} link; and each column of $\hat{T}'_{i-1,l}$ indicates the labels of the successful packets to which one LILE packet (from the set of the r_l chosen LILE packets) in the l^{th} active partition pertaining to the $(i-1)^{\text{th}}$ link contributes. For any other $m < l \leq w$, the set of LILE packets in the l^{th} active partition pertaining to the $(i-1)^{\text{th}}$ link which contribute to the linear combination of one packet in the m^{th} active partition pertaining to the i^{th} link is not tractable in our analysis. Hence, the rows (or the columns) of each sub-matrix $\hat{T}'_{i-1,l}$, for such values of l (i.e., $m < l \leq w$), might have independent or dependent entries (over $\mathbb{F}_2$) with respect to the entries of the rows (or the columns) of the sub-matrices in the set of $\{\hat{T}'_{i-1,l}\}_{1 \leq l \leq m}$. Next, by applying the proper lemma, the rank of the modified transfer matrix at the i^{th} node, and finally, by applying Lemma 3.2, the number of LILE packets in the first j active partitions over the i^{th} link can be bounded from below. This completes the lower bounding process.

Note that, because of its recursive nature, the above technique lower bounds the

[4]It is worth noting that there might be a number of LILE packets which contribute to the linear combination of some packet transmission, but are not included in our lower bounding analysis. The exclusion of such (LILE) packets weakens the tightness of the results, but does not affect the correctness of the analysis.

number of LILE packets in the first j active partitions over the i^{th} link as a function of the number of LILE packets in the active partitions pertaining to the first link. Further, the packets over the first link are all globally dense (by the definition), and hence by using the recursion, the required results can be derived as follows.

Let $\mathcal{D}(Q_i^j)$ be the number of "globally dense" packets in the first j active partitions over the i^{th} link. By the definition, $\mathcal{D}(Q_i^j)$ is bounded from below by the number of "LILE" packets in the first j active partitions over the i^{th} link. Let $\mathcal{D}_p(Q_i^j)$ be a (probabilistic) lower bound on the number of LILE packets in the first j active partitions over the i^{th} link, and of course a lower bound on $\mathcal{D}(Q_i^j)$,[5] such that: if $\mathcal{D}(Q_s^\tau) \geq \mathcal{D}_p(Q_s^\tau)$, for every $1 \leq s \leq i$ and $1 \leq \tau \leq j$, except $(s,\tau) = (i,j)$, then the inequality $\mathcal{D}(Q_i^j) \geq \mathcal{D}_p(Q_i^j)$ fails w.p. b.a.b. $\dot{\epsilon}/w_T$. Let $\hat{r}_{ij}$ be defined in a recursive fashion as the largest integer satisfying

$$\hat{r}_{ij} \leq \mathcal{D}_p(Q_i^j) - \sum_{1 \leq \tau < j} \hat{r}_{i\tau}. \tag{4.6}$$

Note that, at each step of our lower bounding process, the number of LILE packets in a collection of active partitions, but not the number of LILE packets in one individual active partition, is lower bounded. Further, the difference between the two lower bounds corresponding to the two collections of the first j and the first $j-1$ active partitions does not lower bound the number of LILE packets in the j^{th} active partition. However, due to the recursion, we need to choose a certain number of LILE packets at each step of the process (and ignore the rest, if any), and study the density of the packets in the next partition, at the next step of the process, with respect to the LILE packets chosen till the previous step. We, thus, construct a collection of LILE packets at the i^{th} node as follows: starting with an empty collection (at the step zero), for every $1 \leq j \leq w - L + 1$, at the j^{th} step, we expose the packets in the active partitions over the i^{th} link in order, one by one. We add a packet to the collection whenever the packet is LILE (with respect to the current collection), until revealing $\hat{r}_{ij}$ new LILE packets. The size of such a collection

[5] $\mathcal{D}_p(Q_i^j)$ is a "probabilistic" lower bound on $\mathcal{D}(Q_i^j)$ and hence the subscript "p."

lower bounds the number of LILE packets at the i^{th} node; and in order to study the structure of the modified transfer matrix at this node, we fix the packets in the subsets of the underlying collection (each subset pertaining to one of the collection steps) and ignore the rest of packets.

The set of packets over the first link are globally dense, and hence, for every $1 \leq j \leq w - L + 1$, $\mathcal{D}(Q_1^j) \geq rj$ (by the assumption, each partition includes more than or equal to r packets). Further, for every $1 < i \leq L$ and $1 \leq j \leq w - L + 1$, $\mathcal{D}(Q_i^j)$ is bounded from below as follows.

Lemma 4.3 *Consider applying a dense code over a line network of L links with delay-free deterministic regular transmissions and Bernoulli losses with parameters $\{p_i\}$. Let w_T and r be defined as in* (4.1) *and* (4.4)*, respectively. For every $1 < i \leq L$, the inequality*

$$\mathcal{D}(Q_i^1) \geq r - \log(w_T/\epsilon) - 1$$

fails w.p. b.a.b. $i\dot{\epsilon}/w_T$.

Lemma 4.4 *Consider a scenario similar to the one in Lemma 4.3. Let p be defined as in* (4.2)*. For every $1 < i \leq L$ and $1 < j \leq w - L + 1$, the inequality*

$$\mathcal{D}(Q_i^j) \geq rj - \mathcal{L}_{ij}$$

fails w.p. b.a.b. $ij\dot{\epsilon}/w_T$, so long as

$$w \log \frac{w_T}{\epsilon} = o(pN_T) \tag{4.7}$$

where $\mathcal{L}_{ij} = j(1 + o(1))(\log(w_T/\epsilon) + 1) + \log((j(1 + o(1)) + 1)/\epsilon) + \log w_T + 1$, and the $o(1)$ term is $(\log(w_T/\epsilon) + 1)/r$.

The result of Lemma 4.4 lower bounds the number of LILE packets at the sink node, $\mathcal{D}(Q_L)$, as follows.

Lemma 4.5 *Consider a scenario similar to the one in Lemma 4.3. Let p and φ be defined as in* (4.2) *and* (4.3)*, respectively. The inequality*

$$\begin{aligned}\mathcal{D}(Q_L) \geq\ & w_T\varphi/L - w_T\varphi/L\sqrt{(1/\dot{\varphi})\log(w_T/\dot{\epsilon})} \\ & - (w_T/L)\log(w_T/\dot{\epsilon}) - (w_T/L\varphi)\log^2(w_T/\epsilon) \\ & - (w_T/L\varphi)\log(w_T/\epsilon) - \log(w_T/\epsilon) \\ & - \log(w_T/L) - 1 \end{aligned} \tag{4.8}$$

fails w.p. b.a.b. ϵ, so long as

$$w\log\frac{w_T}{\epsilon} = o(pN_T)$$

where $w \sim (pN_TL^2/\log(pN_TL/\epsilon))^{\frac{1}{3}}$.

Let n_T be equal to the right-hand side of inequality (4.8). Thus, Q_L fails to include an $n_T \times k$ dense sub-matrix w.p. b.a.b. ϵ. By applying Lemma 3.6, $\Pr\{\text{rank}(Q_L) < k\}$ is b.a.b. ϵ, so long as $k \leq n_T - \log(1/\epsilon)$. By replacing ϵ with $\dot{\epsilon}$, it follows that the sink node fails to recover all the message vectors w.p. b.a.b. ϵ, so long as $k \leq n_T - \log(1/\epsilon) - 1$. In the asymptotic setting, as N_T goes to infinity, n_T can be written as

$$pN_T - (1+o(1))(pN_TL/w + \sqrt{pN_Tw\log(wL/\epsilon)} + w\log(wL/\epsilon)).$$

We rewrite the last inequality as

$$\begin{aligned} k \leq\ & pN_T - (1+o(1))(pN_TL/w \\ & + \sqrt{pN_Tw\log(wL/\epsilon)} + w\log(wL/\epsilon)) - \log(1/\epsilon) - 1. \end{aligned}$$

Let $k_{\max}$ be the largest integer k satisfying this inequality. Thus, $k_{\max} \sim pN_T$, and by replacing N_T with k/p ($N_T \sim k/p$), the following result is immediate.

Theorem 4.1 *The coding delay of a dense code over a line network of L links with delay-free deterministic regular transmissions and Bernoulli losses with parameters*

$\{p_i\}$ is larger than

$$\frac{1}{p}\left(k+(1+o(1))\left(\frac{kL}{w}+\sqrt{k\left(w\log\frac{wL}{\epsilon}\right)}+w\log\frac{wL}{\epsilon}\right)\right)$$

w.p. b.a.b. ϵ, so long as

$$w\log\frac{wL}{\epsilon}=o(k)$$

where $w\sim(kL^2/\log(kL/\epsilon))^{\frac{1}{3}}$, $p\doteq\min_{1\le i\le L}p_i$, and the $o(1)$ term goes to 0 as k goes to infinity.[6]

We now study the average coding delay of dense codes over the traffics with delay-free deterministic regular transmissions and Bernoulli losses. In this case, the deviation of the number of packets per partition should not be taken into account. Thus, by replacing r with φ in the analysis of the coding delay, the following result can be shown.

Theorem 4.2 *The average coding delay of a dense code over a network similar to Theorem 4.1 is larger than*

$$\frac{1}{p}\left(k+(1+o(1))\left(\frac{kL}{w}+w\log\frac{wL}{\epsilon}\right)\right)$$

w.p. b.a.b. ϵ, so long as

$$w\log\frac{wL}{\epsilon}=o(k)$$

where $w\sim(kL/\log(kL/\epsilon))^{\frac{1}{2}}$.

The choice of w in Theorem 4.2 is much larger than the one in Theorem 4.1. This is because, in this case, there is no gap between the lower bound on the number of packet transmissions in each partition and its expectation; and hence, the partitions do not need to be sufficiently long.

[6] In the rest of the theorems in this chapter, the $o(1)$ term is defined similarly.

It is worth noting that the preceding results might not provide a very clear picture of how the coding delay or the average coding delay are related to the traffic parameters of the links other than the one(s) with the minimum traffic parameter. However, by applying our analysis technique, while taking into consideration the actual values (and the ordering) of the traffic parameters of the links, new upper bounds (with more details) on the coding delay and the average coding delay can be derived. To be more specific, in such an analysis, for every $1 \leq i < L$, depending on whether the i^{th} or the $(i+1)^{\text{th}}$ link has a larger traffic parameter, either Lemma 4.1 or Lemma 4.2 can be used to lower bound the rank of the modified transfer matrix at the i^{th} node, respectively. The rest of the analysis, however, remains the same. For example, the coding delay and the average coding delay of dense codes for the special case with unequal traffic parameters, where no two parameters are equal, can be upper bounded as follows. In particular, the upper bounds, in this case, demonstrate the dependence of the coding delay or the average coding delay on the minimum of the (absolute value of the) difference between the traffic parameters of any two consecutive links in the network.

Let us assume $p_1 > p_2 > \cdots > p_L$, without loss of generality. Let $p \doteq \min_{1 \leq i \leq L} p_i$, $\gamma_e \doteq \min_{1 < i \leq L} \gamma_{e_i}$, and $\gamma_{e_i} \doteq |p_i - p_{i-1}|$. Let $r_i \doteq (1 - \gamma_i^*)\varphi_i$, where $\varphi_i = p_i N_T / w$ and $\gamma_i^* \sim \sqrt{(1/\varphi_i)\log(w_T/\dot{\epsilon})}$. For every $1 \leq i \leq L$ and $1 \leq j \leq w - L + 1$, let φ_{ij} be defined as before (i.e., φ_{ij} is the number of successful packets in the j^{th} active partition pertaining to the i^{th} link). For all i, j, suppose that φ_{ij} is larger than or equal to r_i, i.e., there exist a sufficiently large number of successful packet transmissions in each partition over each link. (This assumption fails if, for some $1 \leq i \leq L$, the number of packets in some active partition over the i^{th} link is less than r_i. Hence, the failure occurs w.p. b.a.b. $\dot{\epsilon}$.)

Since all the packet transmissions over the first link are globally dense, for every $1 \leq j \leq w - L + 1$, $\mathcal{D}(Q_1^j) \geq r_1 j$. Further, by applying Lemma 4.2, it can be shown that, for every $1 < i \leq L$ and $1 \leq j \leq w - L + 1$, the inequality $\mathcal{D}(Q_i^j) \geq r_i j$ fails

w.p. b.a.b. $ij\epsilon/w_T$, so long as

$$w \log \frac{w_T}{\epsilon} = o\left(\min\left\{\frac{\gamma_e}{p}, 1\right\} \cdot pN_T\right). \tag{4.9}$$

Let p, φ, γ^* and r denote p_L, φ_L, γ_L^* and r_L, respectively. Thus, the inequality $\mathcal{D}(Q_L) \geq (1-\gamma^*)\varphi w_T/L$ fails w.p. b.a.b. ϵ. By replacing φ with pN_T/w, the right-hand side of the last inequality can be written as:

$$pN_T - O(pN_T L/w) - O(\sqrt{pN_T w \log(wL/\epsilon)}). \tag{4.10}$$

The rest of the analysis is similar to that of Theorem 4.1, except that (4.10) excludes the last term in (B.2), and the choice of w needs to satisfy condition (4.9), instead of condition (4.7).

Theorem 4.3 *Consider a sequence of unequal parameters $\{p_i\}_{1\leq i\leq L}$. The coding delay of a dense code over a line network of L links with delay-free deterministic regular transmissions and Bernoulli losses with parameters $\{p_i\}$ is larger than*

$$\frac{1}{p}\left(k + (1+o(1))\left(\frac{kL}{w} + \sqrt{k\left(w \log \frac{wL}{\epsilon}\right)}\right)\right)$$

w.p. b.a.b. ϵ, so long as

$$w \log \frac{wL}{\epsilon} = o\left(\min\left\{\frac{\gamma_e}{p}, 1\right\} \cdot k\right)$$

where $w \sim (kL^2/\log(kL/\epsilon))^{\frac{1}{3}}$, $p \doteq \min_{1\leq i\leq L} p_i$, $\gamma_e \doteq \min_{1<i\leq L} \gamma_{e_i}$, and $\gamma_{e_i} \doteq |p_i - p_{i-1}|$.

In the case of the average coding delay, the analysis follows the same line as that of Theorem 4.2, except that the choice of w needs to maximize

$$pN_T - O(pN_T L/w) \tag{4.11}$$

subject to condition (4.9), instead of (B.3) subject to condition (4.7).

Theorem 4.4 *The average coding delay of a dense code over a network similar to Theorem 4.3 is larger than*

$$\frac{1}{p}\left(k+(1+o(1))\left(\frac{kL}{w}\right)\right)$$

w.p. b.a.b. ϵ*, so long as*

$$w\log\frac{wL}{\epsilon}=o\left(\min\left\{\frac{\gamma_e}{p},1\right\}\cdot k\right),$$

i.e., $w\sim k/(f(k)\log(kL/\epsilon))$*, and* $f(k)$ *goes to infinity, as* k *goes to infinity, such that* $f(k)=o\left(k/(L\log(kL/\epsilon))\right)$.

4.3.2 Chunked Codes

Consider applying a chunked code (CC) to the set of k message vectors with q disjoint chunks, each of size $\alpha=k/q$.

Capacity-Achieving Scenarios:

In a CC, at each transmission time, a chunk is chosen w.p. $1/q$, and a packet transmission over the i^{th} link is successful w.p. p_i. Thus the probability that a given packet transmission over the i^{th} link is successful and pertains to a given chunk is p_i/q. Thus by replacing p_i with p_i/q in the analysis of dense codes in Section 4.3.1, the coding delay and the average coding delay of CC in a capacity-achieving scenario will be upper bounded.

The results of dense codes are indeed a special case of those of CC with one chunk of size k. It is, however, worth noting that, due to the change in the parameters, the number of partitions w needs to satisfy a new condition: $wq\log\frac{w_Tq}{\epsilon}=o(pN_T)$ or $wq\log\frac{w_Tq}{\epsilon}=o\left(\min\left\{\frac{\gamma_e}{p},1\right\}\cdot pN_T\right)$, instead of condition (4.7) or (4.9), in the proofs of Theorems 4.5 and 4.6, or those of Theorems 4.7 and 4.8, respectively. Further,

by replacing w with its optimal choice in the new version of (B.2), (B.3), (4.10) and (4.11), each $O(.)$ term needs to be $o(pN_T/q)$ in order to ensure that CC are capacity-achieving in the underlying case. Such a condition lower bounds the size of chunks (α) by a function super-logarithmic in the message size (k).

Theorem 4.5 *The coding delay of a CC with q chunks over a line network of L links with delay-free deterministic regular transmissions and Bernoulli losses with parameters $\{p_i\}$ is larger than*

$$\frac{1}{p}\left(k+(1+o(1))\left(\frac{kL}{w}+\sqrt{k\left(wq\log\frac{wqL}{\epsilon}\right)}+wq\log\frac{wqL}{\epsilon}\right)\right)$$

w.p. b.a.b. ϵ, so long as

$$q=o(k/(L\log(kL/\epsilon))),$$

and

$$wq\log\frac{wLq}{\epsilon}=o(k)$$

where $w\sim(kL^2/(q\log(kL/\epsilon)))^{\frac{1}{3}}$, and $p\doteq\min_{1\le i\le L}p_i$.

Theorem 4.6 *The average coding delay of a CC with q chunks over a network similar to Theorem 4.5 is larger than*

$$\frac{1}{p}\left(k+(1+o(1))\left(\frac{kL}{w}+wq\log\frac{wqL}{\epsilon}\right)\right)$$

w.p. b.a.b. ϵ, so long as

$$q=o(k/(L\log(kL/\epsilon))),$$

and

$$wq\log\frac{wLq}{\epsilon}=o(k)$$

where $w\sim(kL/(q\log(kL/\epsilon)))^{\frac{1}{2}}$.

In the case of unequal traffic parameters, the coding delay and the average coding delay are upper bounded as follows.

Theorem 4.7 *The coding delay of a CC with q chunks over a line network of L links with delay-free deterministic regular transmissions and Bernoulli losses with unequal parameters $\{p_i\}$ is larger than*

$$\frac{1}{p}\left(k+(1+o(1))\left(\frac{kL}{w}+\sqrt{k\left(wq\log\frac{wqL}{\epsilon}\right)}\right)\right)$$

w.p. b.a.b. ϵ, so long as

$$q=o\left(\min\left\{\frac{\gamma_e}{p},1\right\}\cdot k/(L\log(kL/\epsilon))\right),$$

where $w\sim(kL^2/(q\log(kL/\epsilon)))^{\frac{1}{3}}$, $p\doteq\min_{1\leq i\leq L}p_i$, $\gamma_e\doteq\min_{1<i\leq L}\gamma_{e_i}$, and $\gamma_{e_i}\doteq|p_i-p_{i-1}|$.

Theorem 4.8 *The average coding delay of a CC with q chunks over a network similar to Theorem 4.7 is larger than*

$$\frac{1}{p}\left(k+(1+o(1))\left(\frac{kL}{w}\right)\right)$$

w.p. b.a.b. ϵ, so long as

$$q=o\left(\min\left\{\frac{\gamma_e}{p},1\right\}\cdot k/(f(k)L\log(kL/\epsilon))\right),$$

where $w\sim k/(qf(k)\log(kL/\epsilon))$, and $f(k)$ goes to infinity, as k goes to infinity, such that $f(k)=o\left(k/(L\log(kL/\epsilon))\right)$.

Capacity-Approaching-with-a-Gap Scenarios:

By the results of Section 4.3.2, one can conclude that CC are not capacity-achieving if the size of the chunks does not comply with condition $\alpha=\omega(L\log(kL/\epsilon))$. Also, the analysis of Section 4.3.1 does not apply to CC with chunks of small sizes violating the above condition. From a computational complexity perspective, CC with chunks

of smaller sizes are, however, of more practical interest (e.g., linear-time CC with constant-size chunks). In the following, we study CC with chunks of a size constant in the message size.

Let $\{p_i\}_{1\leq i\leq L}$ be an arbitrary sequence of traffic parameters, and let $p \doteq \min_{1\leq i\leq L} p_i$. Let the size of the chunks α $(= k/q)$ be a constant in the message size k, i.e., $\alpha = O(1)$. Let the time interval $(0, N_T]$ and its w disjoint partitions be defined as in Section 4.3.1. Let φ_{ij} be the number of packets (pertaining to a given chunk) in the partition I_{ij}, and φ_i be the expected value of φ_{ij}. Let $\varphi \doteq \min_{1\leq i\leq L} \varphi_i$. Then, $\varphi_i = p_i N_T/wq$, and $\varphi = pN_T/wq$. Let $N_T = (1+\gamma_c)k/p$, where $0 < \gamma_c < 1$ is an arbitrarily small constant. By replacing N_T with $(1+\gamma_c)k/p$, it follows that $\varphi = (1+\gamma_c)\alpha/w$. Further, it is not hard to see that $\varphi = O(1)$, since w has to be a constant (otherwise, if w goes to infinity, as N_T goes to infinity, then φ goes to 0, and for such a case, our analysis is not valid).

By applying Lemma A.3, it can be shown that $\Pr\{\varphi_{ij} < (1-\gamma^*)\varphi\} \leq e^{-\gamma^{*2}\dot{\varphi}}$, for every $0 < \gamma^* < 1$. Taking $e^{-\gamma^{*2}\dot{\varphi}} \leq \dot{\delta}/w_T$, for an arbitrary $0 < \delta < 1$, it follows that φ_{ij} is not larger than or equal to $r \doteq (1-\gamma^*)\varphi$ w.p. b.a.b. $\dot{\delta}/w_T$, where γ^* is chosen to be the smallest real number larger than or equal to $\sqrt{(1/\dot{\varphi})\ln(w_T/\dot{\delta})}$ such that r $(= (1-\gamma^*)\varphi)$ is an integer. It is not hard to see that $\gamma^* = O(1)$. Taking a union bound over all the active partitions of all links, it follows that φ_{ij} is not larger than or equal to r w.p. b.a.b. $\dot{\delta}$.

Let $\mathcal{D}(Q_i^j)$ be the number of globally dense packets pertaining to a given chunk in the first j active partitions over the i^{th} link.

By applying Lemma 4.2, it can be shown that: (i) for every $1 \leq j \leq w-L+1$, $\mathcal{D}(Q_1^j) \geq rj$, (ii) for every $1 < i \leq L$, the inequality $\mathcal{D}(Q_i^1) \geq r - \log(w_T/\dot{\delta})$ fails w.p. b.a.b. $i\dot{\delta}/w_T$, and (iii) for every $1 < i \leq L$ and $1 < j \leq w-L+1$, the inequality $\mathcal{D}(Q_i^j) \geq r - j\log(w_T/\dot{\delta}) - \log((j+1)w_T/\dot{\delta})$ fails w.p. b.a.b. $ij\dot{\delta}/w_T$, so long as

$$\alpha = \Omega\left(w^2 \log\frac{w_T}{\delta}\right). \tag{4.12}$$

By using the above results, it follows that the number of LILE packets pertaining

to a given chunk at the sink node fails to be lower bounded by

$$\frac{w_T\varphi}{L} - O\left(\frac{w_T}{L}\sqrt{\varphi \log \frac{w_T}{\delta}}\right) - O\left(\frac{w_T}{L}\log\frac{w_T}{\delta}\right) \tag{4.13}$$

w.p. b.a.b. δ. The lower bound is non-negative so long as $\alpha = \Omega\left(w \log(w_T/\delta)\right)$, and this condition holds so long as condition (4.12) holds. We select w to be $\sqrt[3]{\alpha L^2/\log(\alpha L/\delta)}$ to maximize (4.13). By replacing w with this value, (4.12) can be rewritten as

$$\alpha = \Omega\left(L^4 \log\frac{L}{\delta}\right). \tag{4.14}$$

By replacing δ with $\dot{\delta}$, and by applying Lemma 3.6, it follows that the sink node fails to decode a given chunk w.p. b.a.b. δ, so long as (4.13) is larger than $\alpha + \log(1/\dot{\delta})$. By replacing our choice of w in (4.13), it can be seen that, excluding the first term, the second term dominates the rest. By replacing φ with $(1+\gamma_c)\alpha/w$, the decoding condition becomes

$$\alpha = \Omega\left(\frac{L}{\gamma_c^3}\log\frac{L}{\gamma_c\delta}\right). \tag{4.15}$$

Thus, a given chunk fails to be decodable w.p. b.a.b. δ so long as both conditions (4.14) and (4.15) are met. In other words, the expected fraction of undecodable chunks is bounded from above by δ. By using a martingale argument similar to the one in [31] (by constructing a martingale sequence over the number of undecodable chunks), the concentration of the fraction of undecodable chunks around the expectation can be shown as follows. The proof is omitted to avoid repetition.

Lemma 4.6 *By applying a CC with chunks of size α, satisfying both conditions* (4.14) *and* (4.15)*, the fraction of undecodable chunks at the sink node until time $N_T = (1+\gamma_c)k/p$ is larger than $(1+\gamma_a)\delta$, w.p. b.a.b. ϵ, so long as*

$$\alpha^2/\gamma_a^2\delta^2 = o(k/\log(1/\epsilon)), \tag{4.16}$$

where $0 < \gamma_a, \delta, \gamma_c < 1$ are arbitrary constants.

By the result of Lemma 4.6, the fraction of chunks which are not decodable until time N_T becomes larger than $(1+\gamma_a)\delta$, w.p. b.a.b. ϵ. Since γ_a, δ are nonzero constants, a CC, alone, might not decode all the chunks. However, the completion of decoding of all the chunks is guaranteed by devising a proper precoding scheme similar to what was used in Section 3.4. The precoding works as follows: The set of k message vectors at the source node constitute the input of a precode, i.e., a capacity-achieving erasure code.[7] The rate of the precode is $1-(1+\gamma_a)\delta$, i.e., the precode decoder can correct up to a fraction $(1+\gamma_a)\delta$ of erasures, and the number of the intermediate packets, i.e., the coded packets at the output of the precode, is $(1+(1+\gamma_a)\delta+O(\delta^2))\,k$. By applying a CC with chunks of size α, satisfying conditions (4.14), (4.15) and (4.16), the fraction of the intermediate packets that are not recoverable at the output of the CC decoder until time $(1+\gamma_c)\,(1+(1+\gamma_a)\delta+O(\delta^2))\,\frac{k}{p}$ is larger than $(1+\gamma_a)\delta$, w.p. b.a.b. ϵ. Then, the precode decoder can recover all the k message vectors from the set of recovered intermediate packets. Therefore, the coding delay of a CC with precoding (CCP) is upper bounded as follows.

Theorem 4.9 *The coding delay of a CCP with chunks of size α and a capacity-achieving erasure code of rate $1-(1+\gamma_a)\delta$, over a line network of L links with delay-free deterministic regular transmissions and Bernoulli losses with parameters $\{p_i\}$ is larger than $(1+\gamma_c)\,(1+(1+\gamma_a)\delta+O(\delta^2))\,\frac{k}{p}$, w.p. b.a.b. ϵ, so long as*

$$\alpha=\Omega\left(\left\{\left(\frac{L}{\gamma_c^3}\log\frac{L}{\gamma_c\delta}\right),\left(L^4\log\frac{L}{\delta}\right)\right\}\right),$$

and $\alpha^2/\gamma_a^2\delta^2=o(k/\log(1/\epsilon))$, where $0<\gamma_a,\delta,\gamma_c<1$ are arbitrary constants, and $p\doteq\min_{1\le i\le L}p_i$.

In the case of the average coding delay of a CC with precoding, the following can be shown similar to Theorem 4.9 by replacing r with φ.

[7]For simplifying the terminology in this section, the precode is assumed to be capacity-achieving. In Section 3.4, however, we discussed about the use of non-capacity-achieving codes as a precode.

Theorem 4.10 *The average coding delay of a CCP with chunks of size α and a capacity-achieving erasure code of rate $1-(1+\gamma_a)\delta$, over a network similar to Theorem 4.9 is larger than $(1+\gamma_c)\left(1+(1+\gamma_a)\delta+O(\delta^2)\right)\frac{k}{p}$, w.p. b.a.b. ϵ, so long as*

$$\alpha = \Omega\left(\frac{L}{\gamma_c}\log\frac{L}{\gamma_c\delta}\right),$$

and $\alpha^2/\gamma_a^2\delta^2 = o(k/\log(1/\epsilon))$, where $0<\gamma_a,\delta,\gamma_c<1$ are arbitrary constants.

In the special case of unequal traffic parameters, the coding delay and the average coding delay of CC with precoding can be upper bounded as follows. The proofs follow the same line as in the general case except that a new set of conditions needs to be satisfied based on the assumption that no two traffic parameters are equal.

Theorem 4.11 *The coding delay of a CCP with chunks of size α and a capacity-achieving erasure code of rate $1-(1+\gamma_a)\delta$, over a line network of L links with delay-free deterministic regular transmissions and Bernoulli losses with unequal parameters $\{p_i\}$ is larger than $(1+\gamma_c)\left(1+(1+\gamma_a)\delta+O(\delta^2)\right)\frac{k}{p}$, w.p. b.a.b. ϵ, so long as*

$$\alpha = \Omega\left(\left\{\left(\frac{L}{\gamma_c^3}\log\frac{L}{\gamma_c\delta}\right),\left(\frac{L}{\gamma_e^3}\log\frac{L}{\gamma_e\delta}\right)\right\}\right),$$

and $\alpha^2/\gamma_a^2\delta^2 = o(k/\log(1/\epsilon))$, where $0<\gamma_a,\delta,\gamma_c<1$ are arbitrary constants, $p \doteq \min_{1\le i\le L} p_i$, $\gamma_e \doteq \min_{1<i\le L}\gamma_{e_i}$, and $\gamma_{e_i} \doteq |p_i - p_{i-1}|$.

Theorem 4.12 *The average coding delay of a CCP with chunks of size α and a capacity-achieving erasure code of rate $1-(1+\gamma_a)\delta$, over a network similar to Theorem 4.11 is larger than $(1+\gamma_c)\left(1+(1+\gamma_a)\delta+O(\delta^2)\right)\frac{k}{p}$, w.p. b.a.b. ϵ, so long as*

$$\alpha = \Omega\left(\frac{L}{\gamma_e^2\gamma_c}\log\frac{L}{\gamma_c\delta}\right),$$

and $\alpha^2/\gamma_a^2\delta^2 = o(k/\log(1/\epsilon))$, where $0<\gamma_a,\delta,\gamma_c<1$ are arbitrary constants.

4.4 Delay-Free Poisson Transmissions and Bernoulli Losses

In the case of Bernoulli losses and delay-free Poisson transmissions with parameters $\{p_i\}_{1\leq i\leq L}$ and $\{\lambda_i\}_{1\leq i\leq L}$, the points in time at which the arrivals/departures occur over the i^{th} link follow a Poisson process with parameter $\lambda_i p_i$. Thus the number of packets pertaining to a given chunk (note that a dense code is a CC with only one chunk), in each partition pertaining to the i^{th} link, has a Poisson distribution with the expected value $\lambda_i p_i N_T/wq$. Since the result of Lemma A.3 also holds for Poisson random variables (see [57, Theorem A.1.15]), the main results in Section 4.3 apply to this case by replacing p with $\min_{1\leq i\leq L} \lambda_i p_i$.

4.5 Comparison with Existing Bounds

4.5.1 Dense Codes

The upper bounds on the coding delay and the average coding delay, derived in this chapter, are valid for any arbitrary choice of ϵ. However, in the following, to compare our results with those of [10] and [25], we focus on the case where ϵ goes to 0 polynomially fast, as k goes to infinity (i.e., $\epsilon = 1/k^c$, for some constant $c > 0$). For such a choice of ϵ, the upper bounds on the coding delay and the average coding delay hold w.p. 1, as k goes to infinity.

In [10], the average coding delay of dense codes over the networks of length 2 with delay-free deterministic regular transmissions and Bernoulli losses with equal parameters (p) is shown to be upper bounded by $\frac{1}{p}(k + O(\sqrt{k \log k}))$. The result of Theorem 4.2 indicates that the average coding delay of dense codes over the networks of length L with similar traffics as above (i.e., the special case of identical links with equal parameters)[8] is upper bounded by $\frac{1}{p}(k + (1 + o(1))(\sqrt{kL \log(kL)}))$. This is consistent with the result of [10], although the bound presented here provides more

[8]One should note that Theorems 4.1 and 4.2 are not restricted to the special case of identical links, and hold true for any arbitrary sequence of parameters.

details on the smaller terms in the $O(.)$ term.

The result of Theorem 4.1 suggests that the coding delay of dense codes over network scenarios as above is upper bounded by $\frac{1}{p}(k + (1 + o(1))(k^2 L \log(kL))^{\frac{1}{3}})$. One should note that there has been no result on the coding delay of dense codes over identical links in the existing literature. In fact, this was posed as an open problem in [25]. It is also noteworthy that unlike the analysis of [25], our analysis does not rely on the existence of a single worst link, and hence is applicable to the special case of identical links.

In [25], the average coding delay of dense codes over the networks of length L with delay-free deterministic regular transmissions and Bernoulli losses with parameters $\{p_i\}$ was upper bounded by $\frac{k}{p} + \sum_{i \neq i_{\min}} \frac{1-p}{p_i - p}$, where $p = \min_i p_i$ is the unique minimum and $i_{\min} = \arg\min_i p_i$. This result was derived under the (impractical) assumption that the size of the finite field over which the coding scheme operates is infinitely large.

Related to this result, Theorem 4.2 or Theorem 4.4 indicate that the average coding delay of dense codes over line networks with traffics as above, but with arbitrary or unequal parameters,[9] is upper bounded by $\frac{1}{p}(k + (1 + o(1))(\sqrt{kL \log(kL)}))$, or $\frac{1}{p}(k + (1 + o(1))(Lf(k) \log(kL)))$, respectively, where $f(k)$ goes to infinity sufficiently slow (see Theorem 4.4), as k goes to infinity. It is important to note that both Theorems 4.2 and 4.4 do not have the limiting assumption of the result of [25] regarding the size of the finite field. The bounds of Theorems 4.2 and 4.4 are larger than that of [25], which is expected, since the former, unlike the latter, are derived based on the practical assumption of operating over a finite field of size as small as two.

The results of Theorems 4.1 and 4.3 indicate that for both traffics with arbitrary or unequal parameters, the coding delay is upper bounded by $\frac{1}{p}(k + (1 + o(1))(k^2 L \log(kL))^{\frac{1}{3}})$. This is while, in [25], the coding delay is upper bounded by $\frac{1}{p}(k + O(k^{\frac{3}{4}}))$. This bound is looser than the bound in Theorem 4.1, or the one

[9] The special case of traffic parameters with a unique minimum can fall into each category of arbitrary or unequal traffic parameters. For example, aside from the uniqueness of the parameter with the minimum value, some other parameters might be equal, and hence such a case does not belong to the category of unequal parameters but it does belong to the category of arbitrary parameters.

in Theorem 4.3, although it is derived under the same limiting assumption as the one used in [25] for the average coding delay (i.e., the size of the finite field being infinitely large). Such an assumption makes the bound appear smaller than what it would be at the absence of the assumption. This demonstrates the strength of the bounding technique used in this work.

By combining Theorems 4.1 and 4.2, or Theorems 4.3 and 4.4, it can be seen that the coding delay might be much larger than the average coding delay. This highlights the fact that the analysis of the average coding delay does not provide a complete picture of the speed of convergence of dense codes to the capacity of line networks.

4.5.2 Chunked Codes

Table 4.1 shows the upper bounds[10] (w.p. of failure b.a.b. ϵ) on the overhead and the average overhead (i.e., the difference between the coding delay or the average coding delay and the capacity) of CC over various traffics for different ranges of the size of the chunks based on the results in Section 4.3 and those in Chapter 3.[11] The traffics under consideration are: arbitrary deterministic traffics, and traffics with delay-free deterministic regular transmissions and Bernoulli losses. We refer to the latter traffics as the *probabilistic traffics* for simplification.[12] The probabilistic traffics are categorized into two sub-categories: traffics with arbitrary parameters and traffics with unequal parameters. We say that a code is "capacity-achieving" (c.-a.) if the ratio of the overhead to the capacity goes to 0, as k goes to infinity. Similarly, a code is "capacity-achieving on average" (c.-a.a.) if the ratio of the average overhead to the capacity goes to 0, as k goes to infinity. In Table 4.1, the upper (or the lower) row in front of each case of traffic parameters corresponds to a c.-a. (or a c.-a.a.)

[10]With a slight abuse of language, we refer to the "upper bound" on the overhead or the average overhead as the "overhead" or the "average overhead."

[11]The results of Section 4.3.2 and those of Section 4.3.2 were stated in terms of q and α, respectively. In this section, for the ease of comparison, the former results are also restated in terms of α by replacing q with k/α.

[12]In the case of arbitrary deterministic traffics, the capacity is equal to k, and in the case of probabilistic traffics with parameters $\{p_i\}_{1\leq i\leq L}$, the capacity is equal to k/p, where $p = \min_{1\leq i\leq L} p_i$.

Table 4.1: Comparison of Overhead and Average Overhead for CC over Line Networks with Various Traffics

Traffic	Traffic Parameters	Overhead (N_o) and Average Overhead ($\overline{N_o}$)	Size of Chunks (α)	w	Comments
Arbitrary Deterministic	-	$N_o = \overline{N_o} = O\left(kL\left(\frac{1}{\alpha}\log\frac{kL}{\epsilon}\right)^{\frac{1}{3}}\right)$	$\omega\left(L^3\log\frac{kL}{\epsilon}\right)$	-	$f(k) = \left\{o\left(\frac{k}{L\log\frac{kL}{\epsilon}}\right), \omega(1)\right\}$ $m = \frac{kw}{\alpha}\log\left(\frac{kLw}{\alpha\epsilon}\right)$ $\gamma_p = \min\left\{\frac{\gamma_\epsilon}{p}, 1\right\}$ $\gamma_{e_i} = \lvert p_i - p_{i-1}\rvert$ $\gamma_e = \min_{1<i\leq L}\gamma_{e_i}$ $p = \min_{1\leq i\leq L} p_i$
Delay-Free Deterministic Regular Transmissions and Bernoulli Losses	Arbitrary	$N_o = \frac{1}{p}\left((1+o(1))\left(\frac{kL}{w} + k^{\frac{1}{2}}m^{\frac{1}{2}} + m\right)\right)$	$\omega\left(L\log\frac{kL}{\epsilon}\right)$	$\left(\frac{\alpha L^2}{\log\frac{kL}{\epsilon}}\right)^{\frac{1}{3}}$	
		$\overline{N_o} = \frac{1}{p}\left((1+o(1))\left(\frac{kL}{w} + m\right)\right)$		$\left(\frac{\alpha L}{\log\frac{kL}{\epsilon}}\right)^{\frac{1}{2}}$	
	Unequal	$N_o = \frac{1}{p}\left((1+o(1))\left(\frac{kL}{w} + k^{\frac{1}{2}}m^{\frac{1}{2}}\right)\right)$	$\omega\left(\frac{L}{\gamma_p}\log\frac{kL}{\epsilon}\right)$	$\left(\frac{\alpha L^2}{\log\frac{kL}{\epsilon}}\right)^{\frac{1}{3}}$	
		$\overline{N_o} = \frac{1}{p}\left((1+o(1))\left(\frac{kL}{w}\right)\right)$	$\omega\left(f(k)\cdot\frac{L}{\gamma_p}\log\frac{kL}{\epsilon}\right)$	$\frac{1}{f(k)}\left(\frac{\alpha}{\log\frac{kL}{\epsilon}}\right)$	

scenario.

In the table, one can see that, for each category of traffics, the size of the chunks (α) has to be sufficiently large so that CC are c.-a. or c.-a.a.. For arbitrary deterministic traffics, the lower bound on α is super-logarithmic in k, i.e., $\omega(\log k)$, and super-log-cubic in L, i.e., $\omega(L^3\log L)$. For the probabilistic traffics with arbitrary or unequal parameters, the lower bound on α has a similar growth rate with k, but a smaller (super-log-linear) growth rate with L, i.e., $\omega(L\log L)$. The coding cost of CC (i.e., the number of the coding (packet) operations per message packet), is, on the other hand, linear in α. Thus, CC can perform as fast over both the arbitrary deterministic traffics and the probabilistic traffics, but with a lower coding cost (smaller chunks) in the latter case compared to the former.

Moreover, as it can be seen in Table 4.1, for both arbitrary deterministic and probabilistic traffics (in each case of arbitrary or unequal traffic parameters), the overhead grows sub-log-linearly with k, i.e., $O(k\log^{\frac{1}{3}}k)$, and decays sub-linearly with α, i.e., $O(1/\alpha^{\frac{1}{3}})$. However, for arbitrary deterministic traffics, the overhead grows with $O(L\log^{\frac{1}{3}}L)$, and for the probabilistic traffics, it only grows with $O(L^{\frac{1}{3}}\log^{\frac{1}{3}}L)$. This implies a faster speed of convergence to the capacity in the latter case compared to the former. Similar comparison results can also be observed in terms of the

Table 4.2: Comparison of Overhead and Average Overhead for CCP over Line Networks with Various Traffics

Traffic	Traffic Parameters	Overhead (N_o) and Average Overhead ($\overline{N_o}$)	Size of Chunks (α)		Comments
Arbitrary Deterministic	-	$N_o = \overline{N_o} = \gamma_o k$	$\Omega\left(\frac{L^3}{\gamma_c^3}\log\frac{L}{\gamma_c\delta}\right)$	$o\left(\sqrt{\frac{\gamma_a^2\delta^2 k}{\log\frac{1}{\epsilon}}}\right)$	$0 < \gamma_a, \delta, \gamma_c < 1$ $\{\gamma_a, \delta, \gamma_c\} = O(1)$ $\gamma_o = \gamma_c + (1+\gamma_c)\gamma_o'$ $\gamma_o' = (1+\gamma_a)\delta + O(\delta^2)$ $\gamma_{e_i} = \lvert p_i - p_{i-1}\rvert$ $\gamma_e = \min_{1<i\leq L}\gamma_{e_i}$ $p = \min_{1\leq i\leq L} p_i$
Delay-Free Deterministic Regular Transmissions and Bernoulli Losses	Arbitrary	$N_o = \gamma_o\frac{k}{p}$	$\Omega\left(\left\{\left(\frac{L}{\gamma_c^3}\log\frac{L}{\gamma_c\delta}\right), \left(L^4\log\frac{L}{\delta}\right)\right\}\right)$		
		$\overline{N_o} = \gamma_o\frac{k}{p}$	$\Omega\left(\frac{L}{\gamma_c}\log\frac{L}{\gamma_c\delta}\right)$		
	Unequal	$N_o = \gamma_o\frac{k}{p}$	$\Omega\left(\left\{\left(\frac{L}{\gamma_c^3}\log\frac{L}{\gamma_c\delta}\right), \left(\frac{L}{\gamma_e^3}\log\frac{L}{\gamma_e\delta}\right)\right\}\right)$		
		$\overline{N_o} = \gamma_o\frac{k}{p}$	$\Omega\left(\frac{L}{\gamma_e^2\gamma_c}\log\frac{L}{\gamma_c\delta}\right)$		

average overhead, except that in the case of unequal traffic parameters, the average overhead decays linearly with α, i.e., $O(1/\alpha)$, but grows poly-log-linearly with k, i.e., $O(k\log^2 k)$, for the choice of $f(k) = O(\log k)$, and log-linearly with L, i.e., $O(L\log L)$.

Table 4.2 shows the results for CC with precoding (CCP) in the scenarios similar to those considered in Table 4.1, where the precode is a capacity-achieving erasure code of dimension k and rate $1-(1+\gamma_a)\delta$. In particular, one can see that CCP are "capacity-approaching" or "capacity-approaching on average" with an arbitrary small "nonzero constant" gap γ_o (i.e., the ratio of the overhead or the average overhead to the capacity goes to γ_o, as k goes to infinity) if α is sufficiently large. For simplifying the terminology, we drop the term "with a nonzero constant gap." The upper (or the lower) row in front of each case of traffic parameters corresponds to a capacity-approaching (or a capacity-approaching on average) scenario. For arbitrary deterministic traffics, the lower bound on α is constant in k, and log-cubic in L, i.e., $O(L^3\log L)$. For the probabilistic traffics with arbitrary or unequal parameters, the lower bound on α is also constant in k, but has a smaller (log-linear) growth rate with L, i.e., $O(L\log L)$. Thus, in the case of CCP, one can make a conclusion similar to the one made in the case of stand-alone CC, with respect to the arbitrary deterministic and the probabilistic traffics.

Chapter 5

New Scheduling Policies for Chunked Network Coding over Networks with Feedback

5.1 Network Model and Assumptions

5.1.1 Network Topology

In this chapter, we generally consider a unicast problem over a network with L (directed) links with any arbitrary topology, not necessarily a line network. In other words, we consider an arbitrary network topology over which one *source* node (which is not the receiving node of any link), and one *sink* node (which is not the transmitting node of any link), are connected through the rest of the network nodes, called *internal* nodes. It is however worth mentioning that the simulations in this chapter are performed only for the scenario of unicast over line networks. Similar to the problem of our interest in Chapters 3 and 4, we assume that the source node has k message vectors, each is a string of bits (i.e., a vector in a vector space $\mathcal{F}$ over $\mathbb{F}_2$), and the sink node demands the message vectors.

Consider an arbitrary ordering of the L links in the network, and associate a label (i.e., a unique integer in $\{i\}_{1\leq i\leq L}$) to each link. For every $1 \leq i \leq L$, let $\mathcal{I}_{R_i}$ (or $\mathcal{I}_{T_i}$) be the set of labels of the links whose receiving nodes (or transmitting nodes) are

the same as the receiving node (or the transmitting node) of the i^{th} link.

5.1.2 Transmission, Loss and Delay Models

In the following, we describe the transmission, loss and the delay models used in this chapter. One, however, should note that the application of the scheduling policies discussed here is not restricted to a specific model of transmission, loss or delay.

We assume that the packet transmissions occur in discrete-time, and there is one packet transmission by each (non-sink) node at each time instant (i.e., deterministic regular transmissions). Each link is modeled by a memoryless erasure channel with a constant probability of success (i.e., Bernoulli losses), i.e., for every $1 \leq i \leq L$, the i^{th} link has a probability of success p_i, for some $0 < p_i \leq 1$ (each packet transmitted over the i^{th} link is either erased with probability $1-p_i$, or is successfully received with probability p_i). We also assume that the links are affected by erasures independently. The special case with no erasure, i.e., $p_i = 1$, for all i, is referred to as the *lossless* case.

One of the main differences between the model used in this chapter and those used in the previous chapters is the delay model. Unlike the case of arbitrary deterministic delays or the delay-free case, here, we assume a probabilistic model for the delay as follows: Each successful (not erased) packet transmitted at any time τ over the i^{th} link is assumed to experience a delay $Z_i^{(\tau)} \in \mathbb{Z}^+ \setminus \{0\}$, i.e., the packet arrives at time $n + Z_i^{(\tau)}$, where $Z_i^{(\tau)}$ is a random variable with the probability mass function

$$P_{Z_i^{(\tau)}}[z^*] = \int_{z^*-1}^{z^*} f_{R_i^{(\tau)}}(r^*)dr^*, \tag{5.1}$$

for every $z^* \in \mathbb{Z}^+ \setminus \{0\}$, where $f_{R_i^{(\tau)}}(r^*)$, $r^* \in \mathbb{R}^+$, is a probability density function. From (5.1), it should be clear that $Z_i^{(\tau)}$ is a discrete version of the continuous random variable $R_i^{(\tau)}$. For all τ, $Z_i^{(\tau)}$'s are assumed to be independent and identically distributed.[1] The special case with all the delays equal to 1, i.e., when $Z_i^{(\tau)} = 1$, for

[1]Here, without loss of generality, we have assumed that the time unit is equal to the inverse of the packet transmission rate at each network node.

all i and τ, is referred to as the *unit-delay* model.

5.1.3 Assumptions

Similar to that in the previous chapters, in this chapter, we also assume that the size of the memory of the network nodes is unbounded. However, as opposed to a former assumption in the previous chapters that the network nodes are blind to the traffic, now, we consider the case of networks where some information about the traffic is available at the network nodes as follows.

Each node is assumed to: (i) know the transmission, loss and delay models of each link over which it transmits a packet, called the *link model*; (ii) store all the packets it transmits along with their departure times, and (iii) store a new packet right after its reception if the packet is innovative to the set of all its previously received innovative packets, or discards the packet, otherwise.[2] In the case of transmitted packets, it suffices that each node stores the global encoding vector of each packet included in the packet header (which is often much smaller than the packet payload), instead of storing both the packet header and the packet payload. In the case of the received packets, both the packet header and the packet payload need to be stored. This is not however a burden when the internal nodes also demand all the message packets (e.g., in the application of peer-to-peer file sharing).

We also consider two cases of networks with or without feedback information at the network nodes. The type of the feedback of our interest in this chapter and its properties will be discussed shortly.

5.2 Problem Setup

Consider a chunked code (CC) with q disjoint chunks, each of size $\alpha = k/q$. Each non-sink node, at each time instant τ, chooses a chunk, say ω, based on a scheduling policy, and by applying a specific coding algorithm to its previously received packets

[2]It is noteworthy that none of the assumptions (i)–(iii) was used for the analysis in Chapters 3 and 4.

pertaining to chunk ω generates/transmits a new packet pertaining to chunk ω. The sink node is able to decode the chunk ω so long as it receives k/q innovative packets pertaining to chunk ω.

Let $\mathcal{T}_i^{(\tau)}$ and $\mathcal{R}_i^{(\tau)}$ be the set of packets transmitted and received over the i^{th} link till time τ, respectively, and $\mathcal{T}_i^{(\tau)}(\omega)$ and $\mathcal{R}_i^{(\tau)}(\omega)$ be the set of the packets for chunk ω in $\mathcal{T}_i^{(\tau)}$ and $\mathcal{R}_i^{(\tau)}$, respectively. By the assumption made in Section 5.1.3, all the packets in $\mathcal{R}_i^{(\tau)}(\omega)$ are innovative, and none of the packets in $\mathcal{T}_i^{(\tau)}(\omega) \setminus \mathcal{R}_i^{(\tau)}(\omega)$ is received yet.

Based on the presence or absence of feedback in the network, one can devise different scheduling policies. If no feedback is available, for every $1 \leq i \leq L$, at time τ, the transmitting node of the i^{th} link knows $\bigcup_{j \in \mathcal{I}_{T_i}} \mathcal{T}_j^{(\tau)}$, but has no information about $\mathcal{R}_j^{(\tau)}$, for any $j \in \mathcal{I}_{R_i}$. However, in the presence of feedback, whenever a packet arrives, the receiving node sends a delay-free and error/erasure-free acknowledgment to the transmitting node, along with a message containing information about the departure time of the arrived packet. The receiving node will also send messages to all the transmitters of the links in $\mathcal{I}_{R_i}$ to convey information about the received packet. In addition to being delay-free, the feedback channels are assumed to have no error/erasure.[3] Thus, in the presence of feedback, the transmitting node has full knowledge of $\bigcup_{j \in \mathcal{I}_{T_i}} \mathcal{T}_j^{(\tau)}$ and $\bigcup_{j \in \mathcal{I}_{R_i}} \mathcal{R}_j^{(\tau)}$.

The problem, at the transmitting node of the i^{th} link, for every i, at every time τ, is how to select a chunk and to code it over the i^{th} link, given the link model, in order to minimize the average delivery time (where the expectation is taken over all the realizations of the code and the network), i.e., the expected time required for all the chunks to be decodable, when only $\bigcup_{j \in \mathcal{I}_{T_i}} \mathcal{T}_j^{(\tau)}$, or both $\bigcup_{j \in \mathcal{I}_{T_i}} \mathcal{T}_j^{(\tau)}$ and $\bigcup_{j \in \mathcal{I}_{R_i}} \mathcal{R}_j^{(\tau)}$ are known.

[3]The assumptions of delay-free and error/erasure-free feedback are reasonable because the data rate over the channel used for feedback is often very low compared to that of the channels used for forward packet transmission.

5.3 Existing Solutions

5.3.1 Random Scheduling Policy

As discussed earlier, originally, CC were designed for networks with no feedback (similar to the scenarios considered in Chapters 3 and 4). In this scenario, one possible strategy for a transmitting node is to use a *fully random scheduling policy* (as was used in the previous chapters), specified as follows: The node chooses a chunk, say ω, uniformly at random; if the node is source, it generates/transmits a random linear combination of all the packets belonging to the chunk ω, and if it is internal, it generates/transmits a random linear combination of all its previously received packets pertaining to chunk ω. Note that, when there is no information about $\bigcup_{j\in\mathcal{I}_{R_i}} \mathcal{R}_j^{(\tau)}$ at the transmitting node of the i^{th} link at time τ, it is not clear how to use the information about $\bigcup_{j\in\mathcal{I}_{T_i}} \mathcal{T}_j^{(\tau)}$.

To speed up the transmission of information over packet networks with feedback, afterwards, CC were adopted in [26] and [27] with feedback-based scheduling policies. The idea behind such scheduling policies is that in the random scheduling policy, a transmitting node might misuse a number of transmission opportunities by transmitting some information which is not useful at the receiving node as it might be contained in previously received packets. This, therefore, increases the average delivery time. The feedback, however, can inform the transmitting node about the set of innovative packets previously received at the receiving node, and hence the transmitting node can, in turn, avoid transmitting packets which are not innovative (with respect to the set of packets available at the receiving node) at the time of transmission.

5.3.2 RP and LRF Scheduling Policies

In [26], Wang and Li proposed a *priority-based* randomized scheduling policy, referred to as *random push* (RP), based on the number of innovative packets at the receiving node. In RP, for every $1 \leq i \leq L$, the node transmitting over the i^{th} link, at each time

τ, randomly chooses a chunk, say ω, from the set of chunks satisfying the condition

$$\left| \bigcup_{j \in \mathcal{I}_{R_{i^*}}} \mathcal{R}_j^{(\tau)}(\omega) \right| > \left| \bigcup_{j \in \mathcal{I}_{R_i}} \mathcal{R}_j^{(\tau)}(\omega) \right|, \tag{5.2}$$

where i^* is the label of some link whose receiving node is the transmitting node of the i^{th} link. The transmitting node, then, generates/transmits an innovative packet (pertaining to chunk ω) with respect to the set $\bigcup_{j \in \mathcal{I}_{R_i}} \mathcal{R}_j^{(\tau)}(\omega)$, by random linear combination of its previously received innovative packets for the chunk ω.[4] Otherwise, if there is no chunk ω, such that condition (5.2) holds, the transmitting node does not transmit a packet, since, in this case, all the information available at the transmitting node is already available at the receiving node.

More recently, in [27], Xu *et al.* introduced a deterministic scheduling policy, referred to as ***local-rarest-first*** (LRF), by prioritizing the chunks based on the same metric as in [26]. In LRF, for every $1 \leq i \leq L$, the node transmitting over the i^{th} link, at each time τ, selects a chunk, say ω, such that (i) chunk ω satisfies condition (5.2) and (ii) the size of the set $\bigcup_{j \in \mathcal{I}_{R_i}} \mathcal{R}_j^{(\tau)}(\omega)$ is the minimum; and generates/transmits a packet (pertaining to chunk ω) innovative to the set $\bigcup_{j \in \mathcal{I}_{R_i}} \mathcal{R}_j^{(\tau)}(\omega)$. If there exist multiple chunks satisfying both conditions (i) and (ii), one of these chunks will be selected uniformly at random.

5.4 Proposed Scheduling Policies

5.4.1 Motivation

The existing scheduling policies based on feedback, as discussed in Section 5.3, prioritize the chunks according to the number of innovative packets at the receiving nodes at the time of transmission. In networks with delay, however, there is no guarantee that a packet which is innovative with respect to the set of packets at a receiving

[4]The transmitting node keeps generating random linear combinations till it generates an innovative packet with respect to the set of the packets at the receiving node.

node at the time of transmission, would still stay innovative at the time of reception. There might be packets transmitted earlier that arrive at the receiving node at some point in time later than the time of the current transmission, but before the reception of the current transmission. Thus the set of received packets at the time of the reception of the currently transmitted packet might differ from the set at the time of the current transmission, and at that point, the currently transmitted packet might no longer be innovative.

This event particularly depends on the set of packets that are transmitted earlier but have not been received yet. The earlier a packet is transmitted, the more likely it generally is for that packet to arrive sooner, but the less likely is for that packet to deliver some useful information if it arrives.

In this chapter, given the link model, and the information about the set of packets transmitted by the transmitting node of a given link and the set of packets received by the receiving node of that link until a given time,[5] we calculate the probabilities of the above mentioned events.[6] We then use these probabilities in the proposed scheduling policies. In particular, we use the expected number of innovative packets "transmitted" prior to the next or the current transmission time, as the metric. It should be clear that this is in contrast to the number of innovative packets "received" prior to the current transmission time, used as the metric in both [26] and [27]. The proposed scheduling policies are referred to as *minimum-distance-first* (MDF) and *minimum-current-metric-first* (MCMF), respectively.

[5]If the information about the packets that were transmitted over the other links connected to the receiving node and still not received was also available at the transmitting node, a more accurate decision could be made about which chunk to choose and what packet to transmit. However, attaining such information might not be possible due to the network topology. We thus assume that such information is not available in the rest of this chapter. However, in the case of line networks simulated in this chapter, since every receiving node only receives information from one node, such situations do not apply.

[6]In earlier works [26] and [27], no assumption has been made about the link model, and hence such probabilities could not be calculated.

5.4.2 MDF Scheduling Policy

For every pair of chunks ω, υ, let $x_i^{(\tau)}(\upsilon|\omega)$ represent the expected number of innovative υ-packets transmitted over the i^{th} link prior to the next transmission time $(\tau+1)$, given that, at the current transmission time (τ), an innovative ω-packet (with respect to the packets in the set $\bigcup_{j\in\mathcal{I}_{R_i}} \mathcal{R}_j^{(\tau)}$) is transmitted over the i^{th} link.[7] The calculation of $x_i^{(\tau)}(\upsilon|\omega)$ is deferred to Section 5.4.4. For every ω, let $d_i^{(\tau)}(\omega)$ denote the Euclidean distance between the vector $[x_i^{(\tau)}(1|\omega), \ldots, x_i^{(\tau)}(q|\omega)]$ and the (q-dimensional) vector $[k/q, \ldots, k/q]$.

In MDF, the node transmitting over the i^{th} link, at each time τ, selects the chunk ω such that (i) chunk ω satisfies condition (5.2) and (ii) $d_i^{(\tau)}(\omega)$ is minimized. That is, the transmitting node chooses a chunk whose transmission at the present time minimizes the distance between the vector of the "expected" number of innovative packets transmitted over the i^{th} link prior to the next transmission time and the vector $[k/q, \ldots, k/q]$. It should be clear that the goal of the network coding solution is to reach the latter vector (i.e., all the chunks can be successfully decoded so long as there are k/q innovative packets pertaining to each chunk). Therefore, the MDF scheduling policy is devised to achieve this goal in a greedy fashion by taking the largest possible step towards (by obtaining the smallest Euclidian distance from) the target. Despite the fact that the MDF scheduling policy is heuristic, in Section 5.5.3, we present some experimental results that indicate the (near) optimality of this scheme (in the sense of minimizing the average delivery time) over line networks.[8] Similar to LRF and RP, in MDF, for any chosen chunk at each time, the transmitting node randomly generates/transmits a packet (pertaining to that chunk) innovative to the packets at the receiving node.

---

[7] Note that, in the definition of the metric $x_i^{(\tau)}(\upsilon|\omega)$, the expectation is taken based on the feedback information available at time τ.

[8] It should be noted that, currently, no analytical result on the proposed or the existing scheduling policies, for a given link model, is available. The difficulty of such analysis stems from the high-level of dependency between the large number of random variables involved in the process.

5.4.3 MCMF Scheduling Policy

For every chunk v, let $y_i^{(\tau)}(v)$ represent the expected number of innovative packets pertaining to chunk v transmitted over the i^{th} link prior to the current transmission time τ. It should be clear that, by the definition, for any given v, $y_i^{(\tau)}(v) = x_i^{(\tau)}(v|\omega)$, for every $\omega \neq v$.[9] In MCMF, the node transmitting over the i^{th} link, at each time τ, selects the chunk ω such that (i) chunk ω satisfies condition (5.2) and (ii) $y_i^{(\tau)}(\omega)$ is minimized. For any chosen chunk at each time, the packet generation/transmission process is similar to that in MDF.

Lemma 5.1 *For networks with unit-delay links (defined in Section 5.1.2), MDF policy reduces to MCMF policy.*

Proof: In this case, the delay values are all one, and hence, there is no randomness in the number of innovative packet transmissions pertaining to any chunk prior to the current transmission time. Thus, by transmitting a given chunk at the current transmission time, the expected number of innovative packet transmissions (prior to the next transmission time) pertaining to that chunk increases, yet, this number does not change for the rest of the chunks. The amount of this increase by transmitting any given chunk is the same as that by transmitting any other chunk, and hence, in such a case, MDF reduces to MCMF, which operates by choosing a chunk which has the smallest (expected) number of innovative packets transmitted prior to the current transmission. □

For a network with a general delay model, however, MDF outperforms MCMF in terms of the average delivery time. In particular, the performance advantage is more profound for random delays with larger mean and variance, for larger networks and for smaller chunks. The performance improvement for MDF policy however, comes at the expense of higher computational complexity (this issue will be discussed in details in Section 5.5).

[9]Both the metrics $y_i^{(\tau)}(v)$ and $x_i^{(\tau)}(v|\omega)$, i.e., the expected number of innovative packet transmissions prior to the current and the next transmission time, respectively, pertaining to any chunk v ($v \neq \omega$), are equal since they both rely on the same (feedback) information till the current transmission time τ.

5.4.4 Metric Calculation

For every pair of chunks ω, υ, every link i, and every time τ, we need to calculate the metric $x_i^{(\tau)}(\upsilon|\omega)$ for the MDF policy. Note that, by the definition, in order to calculate $x_i^{(\tau)}(\upsilon|\omega)$, we focus on the sets $\bigcup_{j\in\mathcal{I}_{R_i}} \mathcal{R}_j^{(\tau)}(\upsilon)$ and $\mathcal{T}_i^{(\tau)}(\upsilon)$, and assume that, at the current time τ, an innovative packet pertaining to chunk ω with respect to the set $\bigcup_{j\in\mathcal{I}_{R_i}} \mathcal{R}_j^{(\tau)}$ is transmitted over the i^{th} link.

For every chunk υ, every link i, and every time τ, let $r_i^{(\tau)}(\upsilon) = |\bigcup_{j\in\mathcal{I}_{R_i}} \mathcal{R}_j^{(\tau)}(\upsilon)|$ and $t_i^{(\tau)}(\upsilon) = |\mathcal{T}_i^{(\tau)}(\upsilon) \setminus \mathcal{U}_i^{(\tau)}(\upsilon)|$, where $\mathcal{U}_i^{(\tau)}(\upsilon)$ denotes the set $\bigcup_{j\in\mathcal{I}_{R_i}} \mathcal{R}_j^{(\tau)}(\upsilon)$. For the ease of notation, hereafter, we often drop the argumant υ, the subscript i, and superscript τ, unless there is a possibility for confusion. For example, we use the notations r and t, instead of $r_i^{(\tau)}(\upsilon)$ and $t_i^{(\tau)}(\upsilon)$, respectively.

Let $\mathcal{N}_t$ and $\mathcal{N}_r$ be the set of the time indices that the packets (pertaining to chunk υ) in $\mathcal{T}\setminus\mathcal{U}$ and $\mathcal{U}$ are transmitted and received, respectively, in an increasing order, i.e., $\mathcal{N}_t = \{\tau_1, \ldots, \tau_t\}$, and $\mathcal{N}_r = \{\tau_1^*, \ldots, \tau_r^*\}$, such that $\tau_1 \leq \cdots \leq \tau_t$, and $\tau_1^* \leq \cdots \leq \tau_r^*$. To lower the computational complexity of the scheduling policy, for some constant integer $1 \leq m \leq t$, we focus on the set of m packets in $\mathcal{T}\setminus\mathcal{U}$, transmitted at the time indices $\mathcal{N}_{t,m} = \{\tau_{t-m+1}, \ldots, \tau_t\}$, i.e., the last m packets transmitted but not received up to time τ. Taking into account only m out of t delayed packets, however, results in an approximation of $x_i^{(\tau)}(\upsilon|\omega)$.[10]

Let $t^* = t - m + 1$. For the case of $\omega = \upsilon$, we define $\boldsymbol{z} = \{z_{\tau_{t^*}}, \ldots, z_{\tau_t}, z_\tau\}$, called the *delay sequence*, as the sequence of the delays that the packets transmitted at time indices $\{\mathcal{N}_{t,m}, \tau\}$ experience, assuming that all these $m+1$ packets arrive (the last packet is the one that, we assume, is transmitted at the time τ), i.e., for every $t^* \leq j \leq t$, the packet transmitted at time τ_j arrives at time $\tau_j + z_{\tau_j}$, and the packet transmitted at time τ arrives at time $\tau + z_\tau$. For the reason that none of these packets has been received till time τ, for every j, the delay z_{τ_j} is bounded from below by $\tau - \tau_j$, for every possible delay sequence $\boldsymbol{z}$. For the other cases of $\omega \neq \upsilon$, due to the fact that the packet which is assumed to be transmitted at time τ is a

[10]The smaller is the value of m, the lower is the complexity of the scheduling policy (and the smaller is the memory requirement at the network nodes). This is at the expense of larger approximation error.

packet pertaining to chunk ω, the delay sequence $\boldsymbol{z}$ excludes the term z_τ. In such cases, we denote the truncated delay sequence $\boldsymbol{z}$ by $\boldsymbol{z}_\tau$.

The delays are, however, random variables that can sometimes take very large values, and it is thus not practical to consider the set of all possible delay sequences $\boldsymbol{z}$. To lower the computational complexity of the scheduling policy, we introduce a constant integer $z_{\max}$, so that if a packet transmitted prior to time τ (or transmitted at time τ) is assumed to arrive later than $z_{\max}$ time units after the time τ (or $\tau+1$), it will be treated as an erased packet in our calculations. We, thus, focus on a subset of all possible delay sequences, referred to as the *desirable delay sequences*, such that at time τ, for every $\omega \neq \upsilon$, the delay of the j^{th} packet ($t^* \leq j \leq t$) is bounded above by $\tau - \tau_j + z_{\max}$, and otherwise, i.e., for $\omega = \upsilon$, the delay of the last packet (assumed to be transmitted at time τ) is bounded above by $z_{\max}$. For the desirable delay sequences, one can see that: for every $t^* \leq j \leq t$, $\tau - \tau_j < z_{\tau_j} \leq \tau - \tau_j + z_{\max}$, and $0 < z_\tau \leq z_{\max}$.[11]

For the sake of brevity, hereafter, we focus on the case of $\omega = \upsilon$. Clearly, by removing the terms related to the packet transmission at time τ and its delay value z_n, the other cases of $\omega \neq \upsilon$ will be covered.[12]

Let $\tau_{t+1} \doteq \tau$. For every desirable delay sequence $\boldsymbol{z}$, suppose that its elements are reordered as follows: let the sequence $\{\tau'_{t^*} + z_{\tau'_{t^*}}, \ldots, \tau'_{t+1} + z_{\tau'_{t+1}}\}$ represent the sequence $\{\tau_{t^*} + z_{\tau_{t^*}}, \ldots, \tau_{t+1} + z_{\tau_{t+1}}\}$ sorted in an increasing order, i.e., $\tau'_{t^*} + z_{\tau'_{t^*}} \leq \cdots \leq \tau'_{t+1} + z_{\tau'_{t+1}}$, and for every $t^* \leq j \leq t+1$, there exists a unique $t^* \leq j' \leq t+1$, such that $t_j = t'_{j'}$. For every delay sequence $\boldsymbol{z}$, hereafter, we use its corresponding reordered sequence based on the reception time indices, and adopt the same notation $\boldsymbol{z}$ to represent it.

For every desirable delay sequence $\boldsymbol{z}$, the probability that a packet, which is transmitted over the i^{th} link at time τ'_j, but not received till time τ, arrives after a

[11]The smaller is the choice of $z_{\max}$, the smaller is the number of desirable delay sequences to be taken into account and hence the lower is the computational complexity of the scheduling policy. This however comes at the expense of larger approximation error.

[12]For any given chunk υ, the metrics $x_i^{(\tau)}(\upsilon|\omega)$, for all $\omega \neq \upsilon$, are the same (independent of ω), and hence need to be calculated only once.

delay $\tau - \tau'_j < z_{\tau'_j} \leq \tau - \tau'_j + z_{\max}$, for every $t^* \leq j \leq t$, is

$$p[z_{\tau'_j}] = \frac{P_{Z_i^{(\tau)}}[z_{\tau'_j}]}{1 - \sum_{1 \leq z \leq \tau - \tau'_j} P_{Z_n^{(\tau)}}[i]} \cdot p_i, \tag{5.3}$$

and

$$p[z_{\tau'_{t+1}}] = P_{Z_i^{(\tau)}}[z_{\tau'_{t+1}}] \cdot p_i, \tag{5.4}$$

where $P_{Z_i^{(\tau)}}$ is given by (5.1), and p_i is the probability of success over the i^{th} link.

The packets which will (or will not) arrive at the receiving node till the next $z_{\max}$ time units are referred to as *on-time* (or *late*). From one perspective, some "late" packets might be erased (not be successful) and will never arrive at the receiving node. By the definition, however, all the "on-time" packets are successful. From another perspective, some of the $m+1$ packets might not be on-time, and the on-time packets might not arrive in the same order that they were transmitted (any possible subset of the $m+1$ packets might be on-time with any possible ordering). The innovation of a packet at the time of reception, however, is dependent on the set of packets that arrived earlier along with the order in which they arrive. We thus need to differentiate between the two partitions of on-time and late packets.

For every possible subset of on-time packets, let us consider a binary sequence of $m+1$ elements $\boldsymbol{s} = \{s_{\tau'_{t^*}}, \ldots, s_{\tau'_{t+1}}\}$, called the *status sequence*, such that, for every $t^* \leq j \leq t+1$, $s_{\tau'_j}$ is 1, if the packet transmitted at the time τ'_j is assumed to be on-time, and it is 0, otherwise. In particular, for every $\boldsymbol{z} = \{z_{\tau'_{t^*}}, \ldots, z_{\tau'_{t+1}}\}$, the packet transmitted at the time τ'_j is assumed to be on-time and to experience a delay $z_{\tau'_j}$, if $s_{\tau'_j}$ is 1, and the packet will be late (i.e., either is successful but does not arrive on-time, or is not successful and is erased), if $s_{\tau'_j}$ is 0. Thus the (joint) probability that all the packets whose corresponding status indicators are 1 arrive on-time with their corresponding delays, and that the rest of the packets are late (regardless of their corresponding delay values), is

$$p_{\boldsymbol{s},\boldsymbol{z}} = \prod_{t^* \leq j \leq t+1} \left(s_{\tau'_j} p[z_{\tau'_j}] + (1 - s_{\tau'_j}) \left(2 - \sum_{\tau - \tau'_j < z_j \leq \tau - \tau'_j + z_{\max}} p[z_j] - p_i \right) \right),$$

for every status sequence $\boldsymbol{s} \in \{0,1\}^{m+1}$, and every desirable delay sequence $\boldsymbol{z}$, where $p[z_{\tau_j'}]$ is given in (5.3) and (5.4), for every $t^* \leq j \leq t$, and $j = t+1$, respectively. (For the cases of $\omega \neq v$, the status sequence $\boldsymbol{s}$ excludes the term s_{m+1}, and we denote the truncated status sequence $\boldsymbol{s}$ with $\boldsymbol{s}_T \in \{0,1\}^m$.)

For every status sequence $\boldsymbol{s}$, let m^* denote the number of 1's in $\boldsymbol{s}$. Now, consider the subset $\{z_{j_1}, \ldots, z_{j_{m^*}}\}$ of the elements of the delay sequence $\boldsymbol{z}$ whose corresponding elements in the status sequence $\boldsymbol{s}$ are 1. Correspondingly, let $\{j_1, \ldots, j_{m^*}\}$ denote the associated sequence of the transmission time indices. For every $1 \leq \ell \leq m^*$, let us define $\mathcal{N}_{\boldsymbol{s},\boldsymbol{z}|\ell}$ as the subset of all the reception time indices $\{j_1 + z_{j_1}, \ldots, j_\ell + z_{j_\ell}\}$ whose corresponding packets are innovative to the set of packets with the reception time indices $\mathcal{N}_{\boldsymbol{s},\boldsymbol{z}|\ell-1} \cup \mathcal{N}_\tau$. Clearly, $\mathcal{N}_{\boldsymbol{s},\boldsymbol{z}|0}$ is the empty set.

To indicate whether the ℓ^{th} packet is innovative at the time of reception, we introduce an indicator variable $I_{\boldsymbol{s},\boldsymbol{z}|\ell}$ defined as follows: $I_{\boldsymbol{s},\boldsymbol{z}|\ell}$ is 1, if the packet with the reception time $j_\ell + z_{j_\ell}$ is innovative to the set of packets with the reception time indices $\mathcal{N}_{\boldsymbol{s},\boldsymbol{z}|\ell-1} \cup \mathcal{N}_\tau$, and $I_{\boldsymbol{s},\boldsymbol{z}|\ell}$ is 0, otherwise. Thus, for every v, at time τ, the expected number of innovative packets pertaining to chunk v transmitted over the i^{th} link prior to the next transmission time, given that an innovative packet pertaining to chunk v is transmitted over the i^{th} link at time τ, can be calculated as

$$x_i^{(\tau)}(v|v) = \sum_{\boldsymbol{s}} \sum_{\boldsymbol{z}} p_{\boldsymbol{s},\boldsymbol{z}} \cdot \left(r + \sum_{1 \leq \ell \leq m^*} I_{\boldsymbol{s},\boldsymbol{z}|\ell} \right). \tag{5.5}$$

Similarly, for every $\omega \neq v$, $x_i^{(\tau)}(v|\omega)$ can be calculated by (5.5), where $\boldsymbol{s}$ and $\boldsymbol{z}$ are replaced with $\boldsymbol{s}_T$ and $\boldsymbol{z}_T$, respectively, i.e.,

$$x_i^{(\tau)}(v|\omega) = \sum_{\boldsymbol{s}_T} \sum_{\boldsymbol{z}_T} p_{\boldsymbol{s}_T,\boldsymbol{z}_T} \cdot \left(r + \sum_{1 \leq \ell \leq m_T^*} I_{\boldsymbol{s}_T,\boldsymbol{z}_T|\ell} \right), \tag{5.6}$$

where m_T^* denotes the number of 1's in $\boldsymbol{s}_T$. Since $y_i^{(\tau)}(v) = x_i^{(\tau)}(v|\omega)$ for every ω $(\neq v)$, the metric $y_i^{(\tau)}(v)$ for MCMF can be calculated by (5.6).

5.4.5 On the Amount of Feedback and the Computational Complexity of the Proposed Scheduling Policies

It is worth noting that both MDF and MCMF require more feedback and more computations compared to RP and LRF. Part of the feedback in MDF and MCMF is required to transmit the link parameters, estimated at the receiver, to the transmitter. This however is not needed for RP and LRF policies. Moreover, in RP and LRF policies, the transmitting node only requires the set of innovative packets at the receiving node. It however does not require the departure/arrival time of such packets. Such information, on the other hand, is required to calculate the metrics in MDF and MCMF policies.

Unlike RP and LRF policies, MDF and MCMF policies need to estimate the link parameters (for loss and delay). This increases the computational complexity and transmission overhead of the proposed policies.[13] The main part of the computational complexity of MDF and MCMF policies is however dedicated to the calculation of $x_i^{(\tau)}(\upsilon|\omega)$, for every pair of chunks ω, υ. This complexity corresponds to the calculation of the double-summations in (5.5) and/or (5.6) over all the desirable delay sequences $\boldsymbol{z}$ and/or $\boldsymbol{z}_T$, and the status sequences $\boldsymbol{s}$ and/or $\boldsymbol{s}_T$. Part of the computations of the argument of the double-summations can be carried out offline, and the results can be stored for online use. (This part is the calculation of the values of $p_{\boldsymbol{s},\boldsymbol{z}}$ and $p_{\boldsymbol{s}_T,\boldsymbol{z}_T}$.) The values of $I_{\boldsymbol{s},\boldsymbol{z}|\ell}$ and $I_{\boldsymbol{s}_T,\boldsymbol{z}_T|\ell}$ however need to be computed online, as they depend on the set of received packets. To determine each value of $I_{\boldsymbol{s},\boldsymbol{z}|\ell}$ or $I_{\boldsymbol{s}_T,\boldsymbol{z}_T|\ell}$, one needs to find the rank of a matrix formed by the global encoding vectors of the packets under consideration. It should however be noted that such operations are performed in the finite field associated with the linear coding scheme, and are in general negligible in comparison with packet operations required for encoding, particularly for larger packet sizes (see, e.g., [24]). In addition, for coding schemes operating over finite fields of large size, the summations $\sum_{1\leq\ell\leq m^*} I_{\boldsymbol{s},\boldsymbol{z}|\ell}$ and $\sum_{1\leq\ell\leq m_T^*} I_{\boldsymbol{s}_T,\boldsymbol{z}_T|\ell}$ in (5.5) and (5.6) can be simply approximated based on the number

[13] Efficient techniques for link estimation can be found in [45, 46], and are beyond the scope of this thesis.

and the ordering of the on-time packets depending on the sequences $\boldsymbol{s}$ and $\boldsymbol{z}$, or $\boldsymbol{s}_T$ and $\boldsymbol{z}_T$, respectively.

Finally, as mentioned earlier, the computational complexity of MCMF is smaller than that of MDF. This arises from the following facts: for every link i, every time instant τ and every chunk v, (i) in MCMF, the metric $y_i^{(\tau)}(v)$ (which is equal to $x_i^{(\tau)}(v|\omega)$, for every $\omega \neq v$) needs to be calculated. However, in MDF, the two metrics $x_i^{(\tau)}(v|v)$ and $x_i^{(\tau)}(v|\omega)$, for some $\omega \neq v$ need to be calculated. This implies that the computational complexity of MDF is at least twice the computational complexity of MCMF; (ii) in MDF, in order to calculate the metric $x_i^{(\tau)}(v|v)$, the sequences $\boldsymbol{s}$ and $\boldsymbol{z}$ are each of length $m+1$. However, in MCMF, in order to calculate the metric $y_n^{(\tau)}(v) = x_n^{(\tau)}(v|\omega)$, for some $\omega \neq v$, the sequences $\boldsymbol{b}_T$ and $\boldsymbol{z}_T$ are each of length m, and hence the calculation of the double-summation in (5.6) requires less computations compared to that in (5.5); and (iii) In MDF, having the metric vectors $[x_i^{(\tau)}(1|\omega), \ldots, x_i^{(\tau)}(q|\omega)]$, for all chunks ω, the Euclidian distances $d_i^{(\tau)}(\omega)$ need to be calculated, and then the chunk with minimum distance will be chosen. However, in MCMF, having the metrics $y_i^{(\tau)}(v)$, for all chunks v, the chunk with minimum metric will be chosen, and there is no need for further computation.

5.5 Simulation Results

We compare random, RP, LRF and MDF scheduling policies over line networks with one source node, one sink node and $L-1$ internal nodes connected in tandem. The comparisons are in terms of the average delivery time (i.e., the expected time it takes for all the chunks to be decodable). The variables involved in the comparisons are the size of the chunks, the length of the network and the parameters of the delay and the loss models. We present the simulation results in two parts: lossless links with (random) delays, and lossy links with unit delays. By combining these results, one can easily generalize the results to the case of the links with both loss and delay. In each part, we consider two cases: identical links and non-identical links.

Table 5.1: Parameters of Delay Models Used in the Simulations

Network Length	L				
Delay Model	I	II	III	IV	V
(μ_i, σ_i)	$(0.5, 0.5)$	$(1, 0.5)$	$(1, 1)$	$(0.5, 0.5)$, if $1 \le i \le L/2$; $(1, 1)$, otherwise.	$(1, 1)$, if $1 \le i \le L/2$; $(0.5, 0.5)$, otherwise.
$(\mathbf{E}(R_i^{(\tau)}), \mathbf{Var}(R_i^{(\tau)}))$	$(1.86, 0.99)$	$(3.08, 2.69)$	$(4.48, 34.51)$	$(1.86, 0.99)$, if $1 \le i \le L/2$; $(4.48, 34.51)$, otherwise.	$(4.48, 34.51)$, if $1 \le i \le L/2$; $(1.86, 0.99)$, otherwise.

5.5.1 Lossless Links with Delay

We consider line networks of lengths $L = 2, 8$. The links are assumed to be lossless, and for every $1 \le i \le L$, the delay model of the i^{th} link is specified as follows: The (continuous delay) probability distribution $f_{R_i^{(\tau)}}(r^*)$, used in (5.1), is assumed to be log-normal[14] with the location and scale parameters (μ_i, σ_i), i.e.,

$$f_{R_i^{(\tau)}}(r^*) = \frac{1}{r^* \sigma_i \sqrt{2\pi}} e^{-\frac{(\ln r^* - \mu_i)^2}{2\sigma_i^2}},$$

where $\{(\mu_i, \sigma_i)\}$ are specified in Table 5.1. The mean and the variance of a log-normal random variable with the location and scale parameters (μ, σ) are $e^{\mu+\sigma^2/2}$ and $(e^{\sigma^2} - 1)e^{2\mu+\sigma^2}$, respectively. In the case of identical links, we consider three delay models, labeled as delay models I, II and III; and in the case of non-identical links, we consider two delay models, labeled as delay models IV and V.

We assume that the message size $k = 64$. We consider two sizes of chunks $\alpha = 8, 32$. For each set of chunk size, loss/delay model and network length, 100 network realizations are simulated, and the chunked coding scheme (over the binary field) along with each scheduling policy is applied to each network realization for 100 trials. For the MDF and MCMF scheduling policies, the parameters m and $z_{\max}$ are set to 4 and 4, respectively. The selected values of m and $z_{\max}$ strike a good balance between the complexity of simulations and the accuracy of the results for

[14]It has been recently shown that, in a variety of real-world packet networks, the delay can be modeled by a heavy-tailed distribution (i.e., the right, or left, or both tail(s) of the probability distribution function are not exponentially bounded), see, e.g., [58]. Examples of such distributions are log-normal, Pareto, and Lévy.

Table 5.2: Average Delivery Time for Various Scheduling Policies over Identical Lossless Links with Delay

	Delay Model		I				II				III			
	Network Length		2		8		2		8		2		8	
	Chunk Size		8	32	8	32	8	32	8	32	8	32	8	32
Lossless	Scheduling Policy	Random	156.45	89.46	331.78	150.56	162.80	91.66	345.86	158.49	167.66	94.14	351.62	168.38
		RP	102.12	81.82	170.23	135.08	106.01	85.19	191.57	149.30	111.64	88.59	199.53	155.91
		LRF	102.00	79.81	182.16	130.21	107.71	83.63	205.81	143.21	111.82	86.15	215.78	151.56
		MDF	69.52	70.77	91.81	96.26	73.02	76.07	103.95	111.75	82.20	86.05	130.26	130.64
		MCMF	76.41	76.19	111.12	104.59	91.15	81.33	142.42	124.53	107.73	86.10	153.48	131.85
	I_1 (%)		31.84	11.33	46.07	26.07	31.12	9.04	45.74	13.04	26.37	0.10	34.72	13.80
	I_2 (%)		25.08	4.53	34.72	19.67	14.01	2.75	25.65	13.04	3.50	0.05	23.07	13.00

Table 5.3: Average Delivery Time for Various Scheduling Policies over Non-Identical Lossless Links with Delay

	Delay Model		IV				V			
	Network Length		2		8		2		8	
	Chunk Size		8	32	8	32	8	32	8	32
Lossless	Scheduling Policy	Random	161.80	91.89	340.62	159.40	162.38	91.71	341.56	159.45
		RP	107.39	86.00	187.55	145.37	103.49	84.84	182.85	144.87
		LRF	107.12	83.69	205.51	141.70	105.21	83.38	194.70	140.80
		MDF	76.42	79.83	117.43	118.78	73.85	77.04	112.37	117.80
		MCMF	86.21	80.77	148.91	129.88	94.59	78.98	137.45	119.52
	I_1 (%)		28.65	4.61	37.38	16.17	28.64	7.60	38.54	16.33
	I_2 (%)		19.52	3.48	20.60	8.34	8.59	5.27	24.82	15.11

the purpose of the comparisons in this chapter. To determine the average delivery time, for each case, the expectation is taken over the 100 network realizations and the 100 trials of the coding scheme.

Tables 5.2 and 5.3 list the average delivery time for each scheduling policy in the case of identical and non-identical lossless links with delay, respectively. Each table quickly reveals that all the scheduling policies significantly outperform the random scheduling policy. The last two rows of each table present the relative performance of the proposed scheduling policies compared to the existing feedback-based scheduling policies. Parameters I_1 and I_2 are defined as $I_1 = \frac{\min\{RP,LRF\}-MDF}{\min\{RP,LRF\}}$ and $I_2 = \frac{\min\{RP,LRF\}-MCMF}{\min\{RP,LRF\}}$, respectively, where, e.g., LRF denotes the average delivery

time of the LRF scheduling policy.

As it can be seen in Table 5.2, both MDF and MCMF policies outperform RP and LRF policies. The largest improvement in this table is 46.07% and corresponds to MDF policy over a network of length 8 with delay model I where the chunk size is 8. The improvement of MDF/MCMF policies over RP/LRF policies is larger for delays with smaller mean and variance. For example, considering MDF policy, for the case of the chunk size 8 and the network length 8, it can be observed that $I_1 = 46.07\%$ for the delay model I. It is then reduced to $I_1 = 45.74\%$, for the delay model II (with larger mean and variance), and is further reduced to 34.72% for the delay model III. Furthermore, the advantage of MDF/MCMF over RP/LRF becomes more for smaller chunks and larger networks. For example, in the case of MDF over the delay model I and the network length 8, for the (larger) chunk size 32, one can see that $I_1 = 26.07\%$, which is smaller than that for the (smaller) chunk size 8 (i.e., $I_1 = 45.74\%$); or in the case of MDF over the delay model I and the chunk size 8, for the (smaller) network length 2, it can be seen that $I_1 = 31.84\%$, which is smaller than that for the (larger) network length 8 (i.e., $I_1 = 46.07\%$). Similar trends can also be observed for MCMF policy. Furthermore, comparing the advantages of MDF and MCMF over RP/LRF (by comparing the values of I_1 and I_2), it can be easily seen that MDF always outperforms MCMF (i.e., for each case, $I_1 \geq I_2$).

Similarly, in Table 5.3, for the case of non-identical links, one can observe similar trends as in the case of identical links, for a given delay model, i.e., the advantage of MDF/MCMF over RP/LRF is more pronounced for smaller chunks and larger networks.

Based on the results in Tables 5.2 and 5.3, the relative performance of LRF and RP (or MDF and MCMF) compared to each other and compared to the random scheduling policy, are listed in Tables 5.4 and 5.5. For each scheduling policy, e.g., LRF, I_R is defined as $I_R = \frac{R-LRF}{R}$, where R denotes the average delivery time of the random scheduling policy. For the pair of scheduling policies RP and LRF (or MDF and MCMF), the parameter I_E (or I_P) is defined as $I_E = \frac{LRF-RP}{LRF}$ (or

Table 5.4: Relative Performance of Scheduling Policies over Identical Lossless Links with Delay

Lossless	Delay Model			I				II				III			
	Network Length			2		8		2		8		2		8	
	Chunk Size			8	32	8	32	8	32	8	32	8	32	8	32
	Policy	RP	I_R (%)	34.72	8.54	48.69	10.28	34.88	7.05	44.61	5.79	33.41	5.89	43.25	7.40
		LRF		34.80	10.78	45.09	13.51	33.83	8.76	40.49	9.64	33.30	8.48	38.63	9.98
		MDF		55.56	20.89	72.32	36.06	55.14	17.00	69.94	29.49	50.97	8.59	62.95	22.41
		MCMF		51.16	14.83	66.50	30.53	44.01	11.26	58.82	21.42	35.74	8.54	56.35	21.69
	I_E (%)			−0.11	−2.51	+6.54	−3.74	+1.57	−1.86	+6.91	−4.25	+0.16	−2.83	+7.53	−2.87
	I_P (%)			9.01	7.11	17.37	7.96	19.89	6.46	27.01	10.26	23.69	0.05	15.12	0.91

Table 5.5: Relative Performance of Scheduling Policies over Non-Identical Lossless Links with Delay

Lossless	Delay Model			IV				V			
	Network Length			2		8		2		8	
	Chunk Size			8	32	8	32	8	32	8	32
	Policy	RP	I_R (%)	33.62	6.40	44.93	8.80	36.26	7.49	46.46	9.14
		LRF		33.79	8.92	39.66	11.10	35.20	9.08	42.99	11.69
		MDF		52.76	13.12	65.52	25.48	54.52	15.99	67.10	26.12
		MCMF		46.71	12.10	56.28	18.51	41.74	13.88	59.75	25.04
	I_E (%)			−0.25	−2.76	+8.73	−2.59	+1.63	−1.75	+6.08	−2.89
	I_P (%)			11.35	1.16	21.14	8.54	21.92	2.45	18.24	1.43

$I_P = \frac{MCMF-MDF}{MCMF}$).

We first focus on the existing scheduling policies RP and LRF and their relative performance (the rows related to I_R for RP and LRF, and I_E, in both Tables 5.4 and 5.5). In the case of identical links (Table 5.4), for RP or LRF, as the mean and the variance of the delay become larger (i.e., moving from delay model I, with the smallest mean and variance, towards the delay model III, with the largest mean and variance), the parameter I_R decreases, i.e., RP or LRF is more advantageous over the random scheduling policy for networks with delays with smaller mean and variance. For example, focusing on the results for RP, in the case with the chunk size 8 and the network length 8, for the delay model I, $I_R = 48.69\%$, and for the delay models II and III, I_R is reduced to 44.61% and 43.25%, respectively. It is also worth noting that, for a given delay model, as the size of the chunks is decreased or the length of

the network is increased, the parameter I_R increases. More interestingly, the results of the second last row of the table (I_E) demonstrates that the relative performance of RP and LRF compared to each other also depends on the delay model. In particular, as the mean and the variance of the delay are increased, or the size of the chunks is decreased, or the length of the network is increased, the relative performance of RP and LRF changes to the benefit of RP. In particular, for a given delay model and network length, RP outperforms LRF for a sufficiently small chunk size.[15] For example, considering the case for the chunk size 8 and the network length 8, and focusing on the comparison between LRF and RP over identical links (in Table 5.4), one can see that for the delay model I, RP is superior ($I_E = +6.54\%$). For the delay model II, the advantage of RP becomes more ($I_E = +6.91\%$), and for the delay model III with the largest mean and variance, RP is even more advantageous ($I_E = +7.53\%$). Similar trends can also be observed for the larger chunk size 32. However, a closer look reveals that, for larger chunk sizes, the transition between the relative superiority of LRF over RP occurs at delays with larger mean and variance. For example, for the network length 8 and the delay model II, RP is superior to LRF for the chunk size 8 ($I_E = +6.92\%$), but for the larger chunk size 32, LRF is still superior ($I_E = -4.25\%$). For delays with smaller mean and variance, LRF is superior to RP, since, in this case, there is a higher chance for a smaller difference between the set of packets at the receiving node at the time of transmission and that at the time of reception. Thus by giving the opportunity of transmission to a chunk with the smallest number of packets at the receiving node, there is a higher chance in balancing the number of packets for all the chunks. For delays with larger mean and variance, however, there is a higher chance for a bigger difference between the underlying sets, and hence, distributing the transmission opportunities over a larger set of chunks yields more balance.

In the case of non-identical links (Table 5.5), for a given delay model, similar to the case of identical links, the performance improvement of RP and LRF over

[15]It is worth noting that, in [27], LRF and RP policies were compared over a number of network scenarios, and for the tested cases, it was concluded that LRF is superior to RP in terms of the average delivery time. However, our simulation results on line networks demonstrate that the relative performance of these policies highly depends on the link model.

random scheduling improves as the chuck size is reduced or the network length is increased. Also, as it can be seen for sufficiently small chunks and sufficiently large networks, RP outperforms LRF (i.e., I_E is positive). For larger chunks or smaller networks, I_E becomes smaller and for sufficiently large chunks and sufficiently small networks, I_E crosses zero and becomes negative (i.e., LRF outperforms RP).

Similarly, by comparing MDF and MCMF, for fixed parameters m and $z_{\max}$, and their relative performance compared to the random scheduling (the rows representing I_R for MDF and MCMF, and I_P, in both Tables 5.4 and 5.5), one can conclude that (i) for each scheduling policy, I_R is decreased for delays with larger mean and variance, and for a given delay model, I_R is increased for smaller chunks and larger networks; (ii) for (a given delay model with) delays with sufficiently small mean and variance, I_P is increased (i.e., the performance gap between MDF and MCMF is increased) as the size of the chunks decreases or the length of the network increases. (Similarly, for sufficiently small chunks and sufficiently large networks, as the mean and the variance of the delay decrease, I_P is decreased.) For example, for the delay model I, considering the chunk size 8, for (smaller) network of length 2, $I_P = 9.01\%$, and for (larger) network of length 8, I_P is increased to 17.37%. Similarly, considering the network of length 2, for the smaller chunk size of 8, $I_P = 9.01\%$, and for the larger chunk size of 32, $I_P = 7.11\%$. Similar comparison results hold true for the delay model II. One should however note that for the delay model III, with the largest mean and variance, similar trends do not seem to hold true. For example, considering the chunk size 8, for the networks of length 2, $I_P = 23.69\%$, and it is reduced down to 15.12% for the (larger) network of length 8.

To justify the different trend for the delay model III, we note that the results of Tables 5.4 and 5.5 are based on fixed parameters m and $z_{\max}$, and as a consequence, for delays with larger mean and variance, the approximation error in the calculation of the metrics is increased. In other words, for sufficiently large m and $z_{\max}$ (and given chunk size and network length), the (monotonically improving) trend of the relative performance of MDF compared to MCMF indeed does not change as the mean and the variance of delays are increased. To verify this claim, we have performed another

Table 5.6: Average Delivery Time for MDF/MCMF with Sufficiently Large Parameters m and $z_{\max}$

Lossless	Network Length			2							
	Chunk Size			4							
	Delay Model			II				III			
	Scheduling Policy		m / $z_{\max}$	2	3	4	5	2	3	4	5
		MDF	2	15.44	15.34	15.31	15.30	19.74	19.64	19.50	19.35
			3	15.33	15.10	14.98	14.97	19.54	18.80	18.75	18.68
			4	15.07	14.98	14.97	14.97	19.37	18.73	18.69	18.65
			5	14.99	14.97	14.97	14.97	19.20	18.68	18.65	18.65
			m / $z_{\max}$	2	3	4	5	3	3	4	5
		MCMF	2	15.89	15.73	15.54	15.44	20.15	19.88	19.64	19.51
			3	15.66	15.37	15.34	15.33	19.84	19.24	19.14	19.13
			4	15.47	15.36	15.33	15.33	19.63	19.24	19.14	19.12
			5	15.38	15.35	15.33	15.33	19.48	19.24	19.13	19.12
		Random		21.88				24.47			

experiment described below.

Consider the transmission of a message of size 8 over a line network of length 2 with CC where the chunk size is 4 (two chunks). In this experiment, we only consider the delay models II and III. For both MDF and MCMF policies, the parameters m and $z_{\max}$ vary between 2 and 5, i.e., $2 \leq m \leq 5$, and $2 \leq z_{\max} \leq 5$. For each delay model, 100 network realizations are simulated, and for each pair of choices of m and $z_{\max}$, CC with MDF or MCMF policy is applied to each network realization for 100 trials. The average delivery time, for each case, is the average of the delivery time over all the simulated realizations of the code and the network realization. These results are presented in Table 5.6. The corresponding relative performances, I_R (for each policy) and I_P, are also listed in Table 5.7. The results in Table 5.6 demonstrate that, for MDF or MCMF, the average delivery time reaches a limit as m and $z_{\max}$ grow large (i.e., the approximation error in the metrics can be made sufficiently small by choosing m and $z_{\max}$ sufficiently large). In the case of delay model II, for MDF and MCMF, this limit is equal to 14.97 and 15.33, respectively; and in the case of delay model III, it is equal to 18.65 and 19.12, respectively. It can be seen that the limit of the average delivery time itself does change with the delay model. For each

Table 5.7: Relative Performance of MDF/MCMF with Sufficiently Large Parameters m and $z_{\max}$

Lossless	Network Length			2	
	Chunk Size			4	
	Delay Model			II	III
	Policy	MDF	I_R (%)	31.58	23.78
		MCMF		29.93	21.86
	I_P (%)			2.34	2.45

Table 5.8: Parameters of Loss Models Used in the Simulations

Network Length	L		
Loss Model	I	II	III
p_i	$\frac{2}{3}$	$1-\frac{1}{3}\cdot\frac{i}{L}$	$1-\frac{1}{3}\cdot\frac{L-i+1}{L}$

scheduling policy, MDF or MCMF, with sufficiently large m and $z_{\max}$, as can be seen in Table 5.7, I_R decreases (and it is not constant) as the mean and the variance of the delay increases. Furthermore, MDF becomes more advantageous compared to MCMF for delays with larger mean and variance (i.e., I_P becomes larger as the mean and the variance of the delay are increased).

5.5.2 Lossy Links with Unit Delays

The scenarios considered in this part are very similar to those in Section 5.5.1, except that the links are lossy and their loss model is specified in Table 5.8, and that the delay model of each link is the unit-delay model. (In particular, the loss model I considers a case with identical links with erasure probability $\frac{1}{3}$, and the two loss models II and III represent two cases with non-identical links.)

Tables 5.9 and 5.10 list the average delivery time for each scheduling policy in the case of identical and non-identical lossy links with unit delay, respectively. Based on the results in these tables, the relative performance of LRF and RP (or MDF and

Table 5.9: Average Delivery Time for Various Scheduling Policies over Identical Lossy Links with Unit Delays

Unit-Delay	Loss Model		I			
	Network Length		2		8	
	Chunk Size		8	32	8	32
	Scheduling Policy	Random	321.21	181.03	656.89	312.18
		RP	191.00	163.31	322.59	269.00
		LRF	192.14	162.25	334.02	265.20
		MDF	148.47	148.47	224.46	224.46
		MCMF	148.47	148.47	224.46	224.46
	I_1 (%)		22.26	8.49	30.41	15.36
	I_2 (%)		22.26	8.49	30.41	15.36

Table 5.10: Average Delivery Time for Various Scheduling Policies over Non-Identical Lossy Links with Unit Delays

Unit-Delay	Loss Model		II				III			
	Network Length		2		8		2		8	
	Chunk Size		8	32	8	32	8	32	8	32
	Scheduling Policy	Random	269.87	159.19	478.79	225.47	280.45	157.52	494.84	224.17
		RP	169.75	144.27	239.15	195.80	161.50	141.35	227.77	190.32
		LRF	171.92	142.74	248.04	193.79	163.55	140.72	232.04	191.44
		MDF	131.91	131.91	158.85	158.85	131.40	131.40	157.26	157.26
		MCMF	131.91	131.91	158.85	158.85	131.40	131.40	157.26	157.26
	I_1 (%)		22.29	7.58	33.57	18.02	18.63	6.62	30.95	17.37
	I_2 (%)		22.29	7.58	33.57	18.02	18.63	6.62	30.95	17.37

MCMF) compared to each other and compared to the random scheduling policy, are also listed in Tables 5.11 and 5.12.

Based on the results in the tables, we observe the followings: (i) MDF and MCMF are always superior to RP and LRF ($I_1 = I_2 > 0$); (ii) The advantage of MDF/MCMF over RP/LRF is larger for smaller chunks and larger networks (i.e., I_1 $(= I_2)$ increases, as the size of the chunks is decreased, or as the length of the network is increased); (iii) The relative performance of RP vs. LRF depends on the chunk size and network length. For sufficiently small chunk sizes and sufficiently large networks, the relative performance of RP with respect to LRF improves (i.e., I_E is increased) as the chunk size is reduced or as the network length is increased. This trend however reverses for sufficiently large chunks or sufficiently small networks; (iv) In all lossy

Table 5.11: Relative Performance of Scheduling Policies over Identical Lossy Links with Unit Delays

Unit-Delay	Loss Model			I			
	Network Length			2		8	
	Chunk Size			8	32	8	32
	Policy	RP	I_R (%)	40.53	9.78	50.89	13.83
		LRF		40.18	10.37	49.15	15.04
		MDF		53.77	17.98	65.82	28.09
		MCMF		53.77	17.98	65.82	28.09
	I_E (%)			+0.59	−0.65	+3.42	−1.43
	I_P (%)			0.00	0.00	0.00	0.00

Table 5.12: Relative Performance of Scheduling Policies over Non-Identical Lossy Links with Unit Delays

Unit-Delay	Loss Model			II				III			
	Network Length			2		8		2		8	
	Chunk Size			8	32	8	32	8	32	8	32
	Policy	RP	I_R (%)	37.09	9.37	50.05	13.15	42.41	10.26	53.97	15.10
		LRF		36.29	10.33	48.19	14.05	41.68	10.66	53.10	14.60
		MDF		51.12	17.13	66.82	29.54	53.14	16.58	68.22	29.84
		MCMF		51.12	17.13	66.82	29.54	53.14	16.58	68.22	29.84
	I_E (%)			+1.26	−1.07	+3.58	−1.03	+1.25	−0.44	+1.84	+0.58
	I_P (%)			0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

scenarios with unit delay, MDF and MCMF perform the same ($I_P = 0$). This is expected based on Lemma 5.1.

5.5.3 On the (Near) Optimality of the MDF Scheduling Policy over Line Networks

To verify the fact that the MDF scheduling policy is (near) optimal in the sense of minimizing the average delivery time over line networks, we have performed the following experiment.

Consider a simple line network of length 1 (one transmitting node and one receiving node). The link is assumed to be lossless, and two cases with two different delay models I and II are considered. The message size is 8, and we consider chunks of size 4 (i.e., the case of CC with two chunks). The parameters m and $z_{\max}$ vary between 2

and 4, i.e., $2 \leq m \leq 4$, and $2 \leq z_{\max} \leq 4$. For each delay model, 100 network realizations are simulated, and for each choice of parameters m and $z_{\max}$, CC with the MDF scheduling policy is applied to each network realization. For each case (i.e., a given network realization), we consider all the possible choices of chunks (to be selected) at each transmission time, and set the maximum transmission time equal to $\tau_{\max}$, for some $\tau_{\max} \in \{16, 32\}$ (i.e., we consider all the possible sequences of the chunk indices till the time $\tau_{\max}$). We further focus on those sequences for which the (decoding) success occurs (i.e., for each chunk, among all the packets transmitted till the time $\tau_{\max}$ by the transmitting node, 4 innovative packets pertaining to that chunk are received at the receiving node). For each such successful sequence, we record the choice (index) of the chunk selected (to be transmitted) at the time τ_0, for some $\tau_0 \in \{4, 8\}$ (we only pick two values of τ_0 as considering all the possible values between 1 and $\tau_{\max}$ is too complex). For each chunk $\omega \in \{1, 2\}$, we calculate the average of delivery times over all those (successful) sequences in which the chunk ω is selected at the time τ_0, and denote it by $\overline{N}(\omega)$. We also calculate (approximate) the two Euclidian distances $d_1^{(\tau_0)}(1)$ and $d_1^{(\tau_0)}(2)$ between the two metrics vectors $[x_1^{(\tau_0)}(1|1), x_1^{(\tau_0)}(2|1)]$ and $[x_1^{(\tau_0)}(1|2), x_1^{(\tau_0)}(2|2)]$ (defined in Section 5.4.2), and the target vector $[4, 4]$, respectively. Let $\omega_{\text{opt}} \doteq \arg\min_\omega \overline{N}(\omega)$ and $\omega_{\text{MDF}} \doteq \arg\min_\omega d_1^{(\tau_0)}(\omega)$. Let us define an indicator variable I such that $I = 1$, if $\omega_{\text{opt}} = \omega_{\text{MDF}}$; and $I = 0$, otherwise. From the above definitions, one can see that ω_{MDF} is the chunk which will be selected based on the MDF scheduling policy at the transmission time τ_0, and ω_{opt} is the chunk whose selection at the transmission time minimizes the average delivery time. Therefore, if $I = 1$, then both events coincide, and in other words, the MDF policy minimizes the average delivery time.

Table 5.13 lists the average of the indicator variables I (in percentage) for all the 100 network realizations, for each delay model and each pair of parameters m and $z_{\max}$. The closer is the expected value of I to 100%, the closer the MDF policy would be to minimize the average delivery time. As can be seen in the table, for each delay model, for sufficiently large values of m and $z_{\max}$ (and for sufficiently large choice of $\tau_{\max}$), I is equal to 100%. This is particularly the case for sufficiently large $\tau_{\max}$, and

Table 5.13: On the Optimality of the MDF Scheduling Policy over Line Networks

Lossless	Network Length			1											
	Chunk Size			4											
	τ_0			4						8					
	$\tau_{\max}$			16			32			16			32		
	Delay Model		$z_{\max}$ \ m	2	3	4	2	3	4	2	3	4	2	3	4
		I	2	98.5	100	100	100	100	100	97.5	100	100	100	100	100
			3	100	100	100	100	100	100	100	100	100	100	100	100
			4	100	100	100	100	100	100	100	100	100	100	100	100
			$z_{\max}$ \ m	2	3	4	2	3	4	2	3	4	2	3	4
		II	2	90	90	90	100	100	100	80.9	87.0	93.2	84.6	94.0	99.6
			3	90	90	90	100	100	100	80.9	93.4	93.7	89.2	99.5	100
			4	90	90	90	100	100	100	78.7	93.2	93.7	90.9	99.5	100

for sufficiently small τ_0 in comparison with $\tau_{\max}$. This would indicate that the MDF policy with sufficiently large m and $z_{\max}$ (depending on the delay model parameters, the chunk size and the network length) achieves the minimum average delivery time in each case.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In Chapter 3, we analyzed and designed communicationally and computationally efficient network codes over line networks with arbitrary deterministic traffics in the absence of feedback. We first presented a detailed analysis of random linear network codes (dense codes). This analysis and its building blocks then served as part of our machinery to analyze chunk-based coding schemes. In particular, tighter bounds on the performance of "chunked codes" (CC) than those by Maymounkov *et al.* were derived. It was shown that for sufficiently large chunks (super-logarithmic chunk size in the message size), chunked codes asymptotically achieve the capacity, yet with a slower speed of convergence (smaller communicational efficiency) compared to dense codes. This is the price for the superior computational efficiency of CC.

Searching for codes that are more computationally efficient than dense codes and more communicationally efficient than chunked codes, in Chapter 3, we proposed and analyzed chunked codes with overlapping chunks, called "overlapped chunked codes" (OCC). We showed that (i) for sufficiently large chunks (super-logarithmic in the message size), an OCC with a constant overlap parameter is asymptotically capacity-achieving, and (ii) for a given chunk size (i.e., for a similar coding cost), an OCC with sufficiently large chunks performs inferior to a CC (an OCC with zero overlap size) with regards to the tradeoff between the speed of convergence to the

capacity and the message error rate. We also proved that the larger is the overlap size in OCC with sufficiently large chunks, the worse is the tradeoff between the speed of convergence to the capacity and the message error rate.

Targeting practical scenarios, where computational resources are scarce, we further analyzed CC and OCC with smaller chunk sizes (logarithmic or sub-logarithmic in the message size down to some constant) in Chapter 3. We showed that there exists a lower bound on the chunk size logarithmic in the message size, such that a code with relatively small chunks satisfying the given bound, yet not satisfying the bound for sufficiently large chunks, asymptotically approaches the capacity with an arbitrarily small nonzero constant gap. We also proved that for codes with relatively small chunk sizes, the larger is the overlap size, the better would be the tradeoff between the speed of convergence to the capacity and the message error rate. We, next, showed that there exists a lower bound on the chunk size constant in the message size, such that a code with very small chunks, i.e., with a chunk size satisfying the given bound, but not satisfying the bound for relatively small chunks, asymptotically approaches the capacity with an arbitrarily small nonzero constant gap. In this case, the message error rate is bounded away from zero, yet still, the packet error rate can be made arbitrarily small. We also proved that for codes with very small chunk sizes, the larger is the overlap size, the better would be the tradeoff between the speed of convergence to the capacity with a nonzero constant gap and the packet error rate. In fact, one of the main contributions of Chapter 3 was to design linear-time network codes (OCC with very small chunks) that are both computationally and communicationally efficient, and that outperform linear-time CC.

In line with the asymptotic analytical results in Chapter 3, we also presented finite-length simulation results which verified that OCC with larger overlap sizes are more efficient, both communicationally and computationally, when sufficiently small message or packet error rates are desired.

In Chapter 4, we analyzed the coding delay and the average coding delay of dense codes, CC and CC with precoding (CCP) over line networks with some probabilistic traffics studied earlier in the literature, in the absence of feedback in the network.

In particular, we considered the traffics with Bernoulli losses and (delay-free) deterministic regular or Poisson transmissions. For such traffics, we derived tight upper bounds on the delay and the average delay of the underlying codes in an asymptotic regime. In the case of dense codes, we showed that our upper bounds on the average coding delay are in some cases more general, and in some other cases tighter than the existing bounds in the literature. Our results on the coding delay also demonstrated that the coding delay may have large deviation from the average coding delay, and thus the average coding delay, alone, might not provide a clear picture of the speed of convergence to the capacity over such probabilistic traffics. In particular, for the analysis of dense codes, we extended the analysis framework used in Chapter 3, and developed a more general analysis technique which is applicable to various network codes (and to more general network scenarios than the unicast over line networks). By using this technique, we were also able to analyze the coding delay and the average coding delay of CC and CCP, for the first time, over the traffics similar to those considered in the case of dense codes. Our results in such cases indicated that CC and CCP provide a better tradeoff between the computational complexity and the speed of convergence to the capacity over the underlying probabilistic traffics compared to arbitrary deterministic traffics which were studied earlier in Chapter 3.

Finally, in Chapter 5, we proposed two feedback-based policies, called minimum-distance-first (MDF) and minimum current metric first (MCMF), for scheduling the chunks in chunked codes over networks with delay and loss, where MCMF is a low-complexity version of MDF with a rather small loss in the performance. In contrast with the existing scheduling policies, random push (RP) and local-rarest-first (LRF), that prioritize the chunks based on the number of innovative received packets over the links (by using the feedback information) up to the time of the new transmission, MDF and MCMF incorporate the loss and delay models in the scheduling process, and operate based on the expected number of innovative successful packet transmissions at each node of the network prior to the next and current transmission time, respectively. To study the performance of the proposed scheduling policies in comparison with the existing ones, we used the log-normal and the unit delay models as well as the lossless and the Bernoulli loss model, over line networks. Our simulations

showed that MDF and MCMF significantly outperform (by up to about 46% and 34.72%, respectively, for the tested cases) the existing policies of RP and LRF in terms of the expected time required for all the chunks to be decodable. The performance improvements are specially larger for smaller chunks and larger networks. Such scenarios are of particular practical interest as smaller chunks translate to lower coding cost, and larger networks are just a fact of life with the continuous increase in the number of communication devices. The improvements come at the cost of more computations, and a slight increase in the amount of feedback. Our results also indicate that the relative performance of RP vs. LRF changes depending on the delay and loss models, the length of network and the size of chunks.

The performance comparison of the proposed and the existing scheduling policies over more general network topologies is also an interesting problem that requires more investigation. While such an investigation was not performed in this thesis, we expect that the proposed policies still outperform RP and LRF policies over more general network topologies. This would be particularly the case, if the network topology allows for the inclusion of the transmission times of the delayed packets of all the adjacent transmitter neighbors of a receiving node in the decision making process at each such transmitting neighbor.

6.2 Future Work

There are many interesting open problems and research directions yet to be addressed in this area of research. In the following, a number of such directions which are considered to be of significant theoretical and practical value are highlighted.

- Increasing the size of the finite field over which we build our codes of interest, and the comparison of the performance of codes over larger fields with those over the binary field (of size two) is an interesting direction for future research. Such results could be very useful in practical scenarios, as, today, most of packet networks are operating over finite fields of sizes as large as 256.

- One another topic for future research is to generalize the analysis technique

used in this thesis for upper bounding the coding delay and the average coding delay of dense codes and CC to the case of OCC. The result of such an analysis would be very helpful in order to compare the performance-complexity tradeoffs provided by CC and OCC over some well-known probabilistic traffic models similar to those discussed in this thesis.

- A theoretical analysis of the performance of the scheduling policies for chunk-based codes discussed in this thesis is still an open problem, even for the special case of line networks. Moreover, the performance comparison of such policies over more general network topologies needs further investigation. Another related open problem is to design new scheduling policies for chunks with overlap and compare such policies with the existing policies in the literature (all of which, to the best of our knowledge, are originally proposed for non-overlapping chunks).

- The analysis of codes discussed in this thesis are based on the assumption of unbounded memory size at the network nodes. However, in some practical situations, and in particular in large-scale applications, it might not be valid to assume that the size of the memory of any network node is unbounded, or even, is very large. Another interesting direction for future work is thus to revise the analysis technique used in this thesis under the assumption that the size of the memory is finite or bounded from above with a function of the message size (and/or the length of the network, or even the size of the chunks in the case of chunked network codes). Such an analysis would be very useful in practice, particularly when the internal network nodes need a large percentage of their memory for some other usages.

- The performance analysis and comparison of dense codes, CC and OCC over more general network scenarios, e.g., (multiple) unicast and/or multicast over arbitrary network topologies with worst-case (arbitrary deterministic) traffics, in the asymptotic and/or finite-length regimes are also important topics. The

importance of such analysis in real-world applications, e.g., large-scale file sharing, emerges from the following facts:

  - There might be more than one sink node demanding the same file simultaneously.
  - The topology of the network between the source node and the sink nodes may vary over time and hence it may seem to be arbitrary from the perspective of the active network nodes.
  - Multicasting through the network generally results in reducing the average delivery time for all the sink nodes than unicasting over link-disjoint line sub-networks to the individual sink nodes.

- In this thesis, our analytical results are asymptotic. Thus, for the purpose of real-world applications, one might not be able to give precise estimates of the parameters of interest (e.g., the message or packet error rate), by using the asymptotic results. This illustrates the importance of finite-length analysis of CC and OCC as another important topic for future research.

List of References

[1] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network Information Flow," *IEEE Trans. on Info. Theory*, vol. 46, pp. 1204–1216, July 2000.

[2] S.-Y. Li, R. Yeung, and N. Cai, "Linear Network Coding," *IEEE Trans. on Info. Theory*, vol. 49, pp. 371–381, Feb. 2003.

[3] R. Koetter and M. Medard, "An Algebraic Approach to Network Coding," *IEEE/ACM Trans. on Networking*, vol. 11, pp. 782–795, Oct. 2003.

[4] C. Gkantsidis and P. Rodriguez, "Network Coding for Large Scale Content Distribution," in *Proc. IEEE INFOCOM'05*, vol. 4, pp. 2235–2245, March 2005.

[5] C. Gkantsidis, J. Miller, and P. Rodriguez, "Comprehensive View of a Live Network Coding P2P System," in *Proc. Internet Measurement Conference, IMC'06*, 2006.

[6] C. Gkantsidis, J. Miller, and P. Rodriguez, "Anatomy of a P2P Content Distribution System with Network Coding," in *Proc. Int. Workshop on P2P Systems, IPTPS'06*, 2006.

[7] J. H. D. M. Chiu, R. W. Yeung and B. Fan, "Can Network Coding Help in P2P Networks?," in *Proc. NetCod'06*, 2006.

[8] D. Tuninetti and C. Fragouli, "Processing Along the Way: Forwarding vs. Coding," in *Proc. of the Int. Sympo. on Info. Theory and Its Applications*, 2004.

[9] D. Tuninetti and C. Fragouli, "On the Throughput Improvement due to Limited Complexity Processing at Relay Nodes," in *Proc. IEEE Int. Symp. on Info. Theory, ISIT'05*, pp. 1081–1085, Sept. 2005.

[10] P. Pakzad, C. Fragouli, and A. Shokrollahi, "Coding Schemes for Line Networks," in *Proc. IEEE Int. Symp. Info. Theory, ISIT'05*, pp. 1853–1857, Sept. 2005.

[11] M. Vehkapera and M. Medard, "A Throughput-Delay Trade-off in Packetized Systems with Erasures," in *Proc. IEEE Int. Symp. on Info. Theory, ISIT'05*, pp. 1858–1862, Sept. 2005.

[12] C. Fragouli, J. Widmer, and J.-Y. Le Boudec, "Efficient Broadcasting Using Network Coding," *IEEE/ACM Transactions on Networking*, vol. 16, pp. 450–463, April 2008.

[13] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting," in *Proc. IEEE Int. Symp. on Info. Theory, ISIT'03*, p. 442, June-July 2003.

[14] T. Ho, *Networking from a Network Coding Perspective.* PhD thesis, Massachusetts Institute of Technology, 2004.

[15] D. S. Lun, M. Médard, and M. Effros, "On Coding for Reliable Communication over Packet Networks," in *in Proc. 42nd Annual Allerton Conference on Communication, Control, and Computing*, 2004.

[16] D. Lun, M. Médard, R. Koetter, and M. Effros, "Further Results on Coding for Reliable Communication over Packet Networks," in *Proc. IEEE Int. Symp. Info. Theory, ISIT'05*, pp. 1848–1852, Sept. 2005.

[17] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A Random Linear Network Coding Approach to Multicast," *IEEE Trans. Info. Theory*, vol. 52, pp. 4413–4430, Oct. 2006.

[18] D. Lun, M. Médard, R. Koetter, and M. Effros, "On Coding for Reliable Communication over Packet Networks," *Physical Communication*, vol. 1, no. 008542, pp. 3–20, 2008.

[19] P. Chou, Y. Wu, and K. Jain, "Practical Network Coding," in *Proc. 41st Annual Allerton Conference on Communication Control and Computing*, pp. 40–49, 2003.

[20] M. Wang and B. Li, "How Practical is Network Coding?," in *Proc. 14th IEEE Int. Workshop on Quality of Service, IWQoS'06*, pp. 274–278, June 2006.

[21] C. Fragouli, D. Lun, M. Medard, and P. Pakzad, "On Feedback for Network Coding," in *Proc. 41st Annual Conference on Information Sciences and Systems, CISS'07*, pp. 248–252, March 2007.

[22] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the Air: Practical Wireless Network Coding," *IEEE/ACM Trans. on Networking*, vol. 16, pp. 497–510, June 2008.

[23] Y. Wu, "A Trellis Connectivity Analysis of Random Linear Network Coding with Buffering," in *Proc. IEEE Int. Symp. Info. Theory, ISIT'06*, pp. 768–772, July 2006.

[24] P. Maymounkov, N. Harvey, and D. Lun, "Methods for Efficient Network Coding," in *Proc. 44th Annual Allerton Conference on Communication Control and Computing*, pp. 482–491, 2006.

[25] T. Dikaliotis, A. Dimakis, T. Ho, and M. Effros, "On the Delay of Network Coding over Line Networks," in *Proc. IEEE Int. Symp. Info. Theory, ISIT'09*, pp. 1408–1412, July 2009.

[26] 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*, vol. 25, pp. 1655–1666, Dec. 2007.

[27] J. Xu, J. Zhao, X. Wang, and X. Xue, "Swifter: Chunked Network Coding for Peer-to-Peer Content Distribution," in *Proc. IEEE Int. Conference on Communications, ICC'08*, pp. 5603–5608, May 2008.

[28] C. Studholme and I. Blake, "Random Matrices and Codes for the Erasure Channel," *Algorithmica*, vol. 56, pp. 605–620, 2010.

[29] A. Heidarzadeh and A. Banihashemi, "Overlapped Chunked Network Coding," in *Proc. IEEE Info. Theory Workshop, ITW'10*, pp. 1–5, Jan. 2010.

[30] A. Heidarzadeh and A. Banihashemi, "Analysis of Overlapped Chunked Codes with Small Chunks over Line Networks," in *Proc. IEEE Int. Symp. on Info. Theory, ISIT'11*, pp. 801–805, Aug. 2011.

[31] A. Heidarzadeh and A. Banihashemi, "Network Codes with Overlapping Chunks over Line Networks: A Case for Linear-Time Codes," *Submitted to IEEE Trans. Info. Theory*, May 2011. Available [Online]: http://arxiv.org/abs/1105.5736.

[32] D. Silva, W. Zeng, and F. Kschischang, "Sparse Network Coding with Overlapping Classes," in *Proc. Workshop on Network Coding, Theory, and Applications, NetCod'09*, pp. 74–79, June 2009.

[33] Y. Li, E. Soljanin, and P. Spasojevic, "Collecting Coded Coupons over Overlapping Generations," in *IEEE Int. Sympo. on Network Coding, NetCod'10*, pp. 1–6, June 2010.

[34] Y. Li, E. Soljanin, and P. Spasojevic, "Effects of the Generation Size and Overlap on Throughput and Complexity in Randomized Linear Network Coding," *IEEE Trans. Info. Theory*, vol. 57, pp. 1111–1123, Feb. 2011.

[35] G. Ma, Y. Xu, M. Lin, and Y. Xuan, "A Content Distribution System Based on Sparse Linear Network Coding," in *Proc. Workshop on Network Coding, Theory and Applications, NetCod'07*, March 2007.

[36] J.-P. Thibault, S. Yousefi, and W.-Y. Chan, "Throughput Performance of Generation-Based Network Coding," in *10th Canadian Workshop on Information Theory, CWIT'07*, pp. 89–92, June 2007.

[37] D. Niu and B. Li, "On the Resilience-Complexity Tradeoff of Network Coding in Dynamic P2P Networks," in *Proc. 15th IEEE Int. Workshop on Quality of Service, IWQoS'07*, pp. 38–46, June 2007.

[38] J.-P. Thibault, W.-Y. Chan, and S. Yousefi, "A Family of Concatenated Network Codes for Improved Performance with Generations," *Journal of Communications and Networks*, vol. 10, pp. 384–395, Dec. 2008.

[39] M. Halloush and H. Radha, "Network Coding with Multi-Generation Mixing: Analysis and Applications for Video Communication," in *Proc. IEEE Int. Conference on Communications, ICC'08*, pp. 198–202, May 2008.

[40] S.-H. Lee, U. Lee, K.-W. Lee, and M. Gerla, "Content Distribution in VANETs Using Network Coding: The Effect of Disk I/O and Processing O/H," in *5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON'08*, pp. 117–125, June 2008.

[41] S. L. W. R. G. P. U. Lee, J. Park and M. Gerla, "Efficient Peer-to-Peer File Sharing Using Network Coding in MANET," *Journal of Communications and Networks (JCN), Special Issue on Network Coding*, vol. 10, pp. 422–429, June 2008.

[42] A. Heidarzadeh and A. Banihashemi, "How Fast Can Dense Codes Achieve the Min-Cut Capacity of Line Networks?," in *Proc. IEEE Int. Symp. on Info. Theory, ISIT'12*, pp. 2471–2475, July 2012.

[43] A. Heidarzadeh and A. Banihashemi, "Coding Delay Analysis of Chunked Codes over Line Networks," in *Proc. IEEE Int. Symp. on Network Coding, NetCod'12*, pp. 1–5, June 2012.

[44] A. Heidarzadeh and A. Banihashemi, "Coding Delay Analysis of Dense and Chunked Network Codes over Line Networks," *Submitted to IEEE Trans. Info. Theory*, July 2012. Available [Online]: http://arxiv.org/abs/1203.1643.

[45] V. E. Paxson, *Measurements and Analysis of End-to-End Internet Dynamics.* PhD thesis, University of California, Berkeley, 1997.

[46] S. B. Moon, *Measurement and Analysis of End-to-End Delay and Loss in the Internet.* PhD thesis, University of Massachusetts Amherst, 2000.

[47] S. C. Althoen and R. McLaughlin, "Gauss-Jordan Reduction: a Brief History," *Am. Math. Monthly*, vol. 94, pp. 130–142, February 1987.

[48] A. Shokrollahi, "Raptor Codes," *IEEE Trans. Info. Theory*, vol. 52, pp. 2551–2567, June 2006.

[49] P. Maymounkov, "Online Codes," *Technical report, New York University*, 2002.

[50] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical Loss-Resilient codes," in *Proc. of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, STOC'97*, pp. 150–159, 1997.

[51] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Efficient Erasure Correcting Codes," *IEEE Trans. Info. Theory*, vol. 47, pp. 569–584, Feb. 2001.

[52] P. Oswald and A. Shokrollahi, "Capacity-Achieving Sequences for the Erasure Channel," *IEEE Tran. Info. Theory*, vol. 48, pp. 3017–3028, Dec. 2002.

[53] H. Saeedi and A. Banihashemi, "New Sequences of Capacity-Achieving LDPC Code Ensembles over the Binary Erasure Channel," *IEEE Tran. Info. Theory*, vol. 56, pp. 6332–6346, Dec. 2010.

[54] A. Ashikhmin and V. Skachek, "Decoding of Expander Codes at Rates Close to Capacity," *IEEE Trans. Info. Theory*, vol. 52, pp. 5475–5485, Dec. 2006.

[55] N. Alon and M. Luby, "A Linear Time Erasure-Resilient Code with Nearly Optimal Recovery," *IEEE Trans. Info. Theory*, vol. 42, pp. 1732–1736, Nov. 1996.

[56] C. Cooper, "On the Distribution of Rank of a Random Matrix over a Finite Field," *Random Struct. Algorithms*, vol. 17, pp. 197–212, Oct. 2000.

[57] N. Alon and J. Spencer, *The Probabilistic Method.* 3rd ed. Wiley Interscience, 2008.

[58] J. Tan, *New Origins of Heavy Tails with Applications to Information Networks.* PhD thesis, Columbia University, 2009.

[59] G.-Z. Xiao and J. Massey, "A Spectral Characterization of Correlation-Immune Combining Functions," *IEEE Trans. Info. Theory*, vol. 34, pp. 569–571, May 1988.

[60] S. Janson, "Large Deviations for Sums of Partly Dependent Random Variables," *Random Struct. Algorithms*, vol. 24, pp. 234–248, May 2004.

Appendix A

Proofs of the Results in Chapter 3

A.1 Proofs of the Lemmas

Proof of Lemma 3.1: Let $\boldsymbol{u} = T\boldsymbol{v}$, and $\boldsymbol{u} = [u_1; ...; u_\gamma]$. The proof of uniformity of u_i's is straightforward. To prove the independence of u_i's, it suffices to show that for every $1 < i \leq \gamma$, u_i is independent of the set of u_j's, for all $1 \leq j < i$. First, the independence of u_2 and u_1: The vector of coefficients in the linear combination of u_1 is linearly independent of that of u_2, due to the assumption that T has full row-rank. Thus there exists at least one element in $\boldsymbol{v}$ which only appears in either u_1, or u_2. The elements in $\boldsymbol{v}$ are all independent, and hence the independence of u_2 and u_1. Second, the independence of u_3 and u_2, u_1: The following lemma shows that u_3 is independent of u_2, u_1, if and only if u_3 is independent of any nonzero linear combination of u_2 and u_1, given that u_2 and u_1 are independent.

Lemma A.1 *[59] A discrete random variable z is independent of a set of independent Bernoulli random variables if and only if z is independent of any nonzero linear combination of the Bernoulli random variables with coefficients from $\mathbb{F}_2$.*

Since u_1 and u_2 are independent, then by Lemma A.1, it suffices to show that u_3 is independent of any nonzero linear combination of u_2 and u_1. The vector of coefficients in the linear combination of u_3 is linearly independent of those of u_2 and u_1, and hence is linearly independent of any nonzero linear combination of u_2 and u_1. Then, for any given nonzero linear combination $u_{1,2}$ of u_2 and u_1, there exists at

least one element in $\boldsymbol{v}$ which only appears in either u_3 or $u_{1,2}$. Thus, u_3 and $u_{1,2}$ are independent. Similarly, it can be shown that for every $1 < i \leq \gamma$, u_i is independent of the set of u_j's, for all $1 \leq j < i$, as was to be shown. □

Proof of Lemma 3.2: By the assumption of $\text{rank}(T) \geq \gamma$, there exist at least γ linearly independent rows in T. Let $\boldsymbol{T}$ be T restricted to the γ linearly independent rows in T, i.e., $\boldsymbol{T}$ is a $\gamma \times n^*$ sub-matrix of T. Then, $\boldsymbol{TQ}$ has γ rows which are a subset of rows in TQ. Clearly $\mathcal{D}(TQ) \geq \mathcal{D}(\boldsymbol{TQ})$. For every $1 \leq i \leq \gamma$, and every $1 \leq j \leq k^*$, $(\boldsymbol{TQ})_{i,j} = \sum_{1 \leq \ell \leq n^*} \boldsymbol{T}_{i,\ell} Q_{\ell,j}$ and $\{Q_{\ell,j} : 1 \leq \ell \leq n^*, 1 \leq j \leq k^*\}$ are all i.u.d. Then, for every $1 \leq i, i' \leq \gamma$, and every $1 \leq j, j' \leq k^*$ such that $i \neq i'$ or $j \neq j'$, it follows that $(\boldsymbol{TQ})_{i,j}$ and $(\boldsymbol{TQ})_{i',j'}$ are i.u.d.[1] Thus, $\boldsymbol{TQ}$ is a dense matrix, i.e., $\mathcal{D}(\boldsymbol{TQ}) = \gamma$, and hence $\mathcal{D}(TQ) \geq \gamma$. □

Proof of Lemma 3.3: For every integer $0 \leq \gamma \leq n^* - 1$, let T' be T restricted to its first $n^* - \gamma$ columns, i.e., T' is an $r^* \times (n^* - \gamma)$ sub-matrix of T. Since T' is a sub-matrix of T, $\Pr\{\text{rank}(T) < n^* - \gamma\} \leq \Pr\{\text{rank}(T') < n^* - \gamma\}$. Suppose that $\text{rank}(T') < n^* - \gamma$. Thus, there exists a nonzero column vector $\boldsymbol{v}$ of length $(n^* - \gamma)$ whose entries are from $\mathbb{F}_2$ and that the column-vector $T'\boldsymbol{v}$ of length r^* is an all-zero vector. Suppose that the first nonzero element of $\boldsymbol{v}$ is the j^{th} element, for a given $1 \leq j \leq n^* - \gamma$. There are $2^{n^* - \gamma - j}$ such vectors. Since there exist at least $n^* - j + 1$ i.u.d. random entries over $\mathbb{F}_2$ in j^{th} column of T', there exist at least $n^* - j + 1$ i.u.d. random entries over $\mathbb{F}_2$ in the vector $T'\boldsymbol{v}$. The probability of all these entries being zero is $2^{-(n^* - j + 1)}$, and thus the probability of $T'\boldsymbol{v}$ being an all-zero vector given that $\boldsymbol{v}$ is a vector with the first nonzero entry being the j^{th} is not larger than $2^{-(n^* - j + 1)}$. Taking a union bound over all such vectors $\boldsymbol{v}$, for a given $1 \leq j \leq n^* - \gamma$, the probability of $T'\boldsymbol{v}$ being an all-zero vector is not larger than $2^{n^* - \gamma - j} \times 2^{-(n^* - j + 1)} = 2^{-(\gamma + 1)}$. Taking a union bound over all $1 \leq j \leq n^* - \gamma$, the probability of $T'\boldsymbol{v}$ being an all-zero vector is not larger than $(n^* - \gamma)2^{-(\gamma+1)}$. □

[1]For $j \neq j'$, $(\boldsymbol{TQ})_{i,j}$ and $(\boldsymbol{TQ})_{i',j'}$ are linear combinations of disjoint sets of i.u.d. random variables over $\mathbb{F}_2$ and hence they are i.u.d.; and for $j = j'$ and $i \neq i'$, $(\boldsymbol{TQ})_{i,j}$ and $(\boldsymbol{TQ})_{i',j}$ are linear combinations of the same set of random variables over $\mathbb{F}_2$ with linearly independent vectors of coefficients, and hence they are i.u.d. (by Lemma 3.1).

Proof of Lemma 3.4: Since $\hat{T}_i$ is a $n \times \mathcal{D}(Q_i)$ matrix over $\mathbb{F}_2$ such that for every $j \in \mathcal{I}_i$, $|\hat{T}^{(j)}_{i_{\text{col}}}| \geq \mathcal{D}(Q_i) - j + 1$, Lemma 3.3 implies that the inequality $\text{rank}(\hat{T}_i) \geq \mathcal{D}(Q_i) - \gamma$, for every integer $0 \leq \gamma \leq \mathcal{D}(Q_i) - 1$, fails to hold w.p. b.a.b. $(\mathcal{D}(Q_i) - \gamma)2^{-\gamma-1}$. Lemma 3.2 together with Lemma 3.3 show that the inequality $\mathcal{D}(Q_{i+1}) = \mathcal{D}(\hat{T}_i\hat{Q}_i) \geq \mathcal{D}(Q_i) - \gamma$ fails to hold w.p. b.a.b. $(\mathcal{D}(Q_i) - \gamma)2^{-\gamma-1}$. Setting $(\mathcal{D}(Q_i) - \gamma)2^{-\gamma-1} \leq \epsilon$, it follows that

$$\gamma \geq \log(\mathcal{D}(Q_i) - \gamma) + \log(1/\epsilon) - 1. \tag{A.1}$$

Let γ be the smallest integer equal to or larger than $\log \mathcal{D}(Q_i) + \log(1/\epsilon)$.[2] Thus, the inequality $\mathcal{D}(Q_{i+1}) \geq \mathcal{D}(Q_i) - \log \mathcal{D}(Q_i) - \log(1/\epsilon)$ fails to hold w.p. b.a.b. ϵ. □

Proof of Lemma 3.5: Since $\mathcal{D}(Q_{i+1}) \leq n$ for every $0 \leq i < L$, by replacing ϵ with $\epsilon/(L-1)$ in Lemma 3.4, it follows that the inequality

$$\begin{aligned} \mathcal{D}(Q_i) - \mathcal{D}(Q_{i+1}) &\leq \log \mathcal{D}(Q_i) + \log((L-1)/\epsilon) \\ &\leq \log(n(L-1)/\epsilon) \\ &\leq \log(nL/\epsilon), \end{aligned}$$

which fails to hold w.p. b.a.b. $\epsilon/(L-1)$. Thus, the inequality

$$\sum_{1 \leq i < L} (\mathcal{D}(Q_i) - \mathcal{D}(Q_{i+1})) \leq (L-1)\log(nL/\epsilon)$$

fails to hold w.p. b.a.b. ϵ. Further,

$$\mathcal{D}(Q_L) = \mathcal{D}(Q_1) - \sum_{1 \leq i < L} (\mathcal{D}(Q_i) - \mathcal{D}(Q_{i+1})).$$

[2]Such a choice of γ satisfies (A.1) because for every $\gamma \geq 0$, $\log \mathcal{D}(Q_i) + \log(1/\epsilon) \geq \log(\mathcal{D}(Q_i) - \gamma) + \log(1/\epsilon) > \log(\mathcal{D}(Q_i) - \gamma) + \log(1/\epsilon) - 1$.

Since $\mathcal{D}(Q_1) = n$, the inequality

$$\begin{aligned}\mathcal{D}(Q_L) &= n - \sum_{1 \le i < L} (\mathcal{D}(Q_i) - \mathcal{D}(Q_{i+1})) \\ &\ge n - (L-1)\log(nL/\epsilon) \\ &\ge n - L\log(nL/\epsilon)\end{aligned}$$

fails to hold w.p. b.a.b. ϵ. □

Proof of Lemma 3.6: Clearly, $\mathrm{rank}(Q) < k^*$ if there exists a nonzero column vector $\boldsymbol{v}$ of length k^*, such that $Q\boldsymbol{v}$ is an all-zero vector. For a given nonzero vector $\boldsymbol{v}$, the probability of $Q\boldsymbol{v}$ being an all-zero vector is equal to 2^{-n^*}. Taking a union bound over all such vectors $\boldsymbol{v}$, $\Pr\{\mathrm{rank}(Q) < k\}$ is upper bounded by $(2^{k^*} - 1)2^{-n^*} \le 2^{-(n^*-k^*)}$. □

Proof of Lemma 3.7: The following lemma is useful in order to give an upper bound on $\Pr\{\mathrm{rank}(T) < n^* - \gamma\}$, for every integer $0 \le \gamma \le n^* - 1$.

Lemma A.2 *[56, Theorem 1] Let T be an $r^* \times n^*$ matrix over $\mathbb{F}_2$ whose entries are all i.u.d. random variables. Then, for every integer $1 \le \gamma \le n^* - 1$,*

$$\Pr\{\mathrm{rank}(T) = n^* - \gamma] = c_\gamma \cdot 2^{-\gamma(r^*-n^*+\gamma)},$$

for $c_\gamma = \prod_{i=r^*-n^*+\gamma+1}^{\infty}(1-2^{-i}) / \prod_{i=1}^{\gamma}(1-2^{-i})$.

For every integer $1 \le \gamma \le n^* - 1$, it can be shown by induction that $c_\gamma \le 2^\gamma$. Thus,

$$\Pr\{\mathrm{rank}(T) = n^* - \gamma\} \le 2^\gamma \cdot 2^{-\gamma(r^*-n^*+\gamma)} \le 2^\gamma \cdot 2^{-\gamma^2}.$$

The probability of $\text{rank}(T) < n^* - \gamma$ is the sum of the probabilities of $\text{rank}(T)$ being $n^* - (\gamma+1)$, $n^* - (\gamma+2)$, ..., or 1. Thus,

$$\begin{aligned}
\Pr\{\text{rank}(T) < n^* - \gamma\} &= \sum_{\gamma+1 \le i < n^*} \Pr\{\text{rank}(T) = n^* - i\} \\
&\le \sum_{\gamma+1 \le i < n^*} 2^{-i(i-1)} \\
&\le \sum_{\gamma+1 \le i < n^*} 2^{-i} \\
&= 2^{-\gamma} - 2^{-(n^*-1)} \\
&\le 2^{-\gamma}.
\end{aligned}$$

□

Proof of Lemma 3.8: Since $\hat{T}_i$ is an $n \times \mathcal{D}(Q_i)$ matrix over $\mathbb{F}_2$, such that for every $j \in \mathcal{I}_i$, $|\hat{T}_{i_{\text{col}}}^{(j)}| = n$, Lemma 3.7 implies that the inequality $\text{rank}(\hat{T}_i) \ge \mathcal{D}(Q_i) - \gamma$ fails to hold w.p. b.a.b. $2^{-\gamma}$, for every integer $0 \le \gamma \le \mathcal{D}(Q_i) - 1$. Lemma 3.2 together with Lemma 3.7 then shows that the inequality $\mathcal{D}(Q_{i+1}) = \mathcal{D}(\hat{T}_i \hat{Q}_i) \ge \mathcal{D}(Q_i) - \gamma$ fails to hold w.p. b.a.b. $2^{-\gamma}$. Setting $2^{-\gamma} \le \epsilon$, it follows that $\gamma \ge \log(1/\epsilon)$. Let γ be the smallest integer equal to or larger than $\log(1/\epsilon)$. Thus, the inequality $\mathcal{D}(Q_{i+1}) \ge \mathcal{D}(Q_i) - \log(1/\epsilon)$ fails to hold w.p. b.a.b. ϵ. □

Proof of Lemma 3.9: The proof follows the same steps as the proof of Lemma 3.5, except that the bound of Lemma 3.4 is replaced by that of Lemma 3.8. □

Proof of Lemma 3.10: We show that with high probability for every chunk, the most of packets pertaining to a chunk in any given subset of packets consecutively received by a node (when the size of the subset is chosen with some care) are matched up with the most of the packets pertaining to that chunk in a particular subset of packets consecutively transmitted afterwards by that node. We formalize this idea below.

Consider the i^{th} and $(i+1)^{\text{th}}$ nodes for some $0 \leq i < L$, and the transmissions over the $(i+1)^{\text{th}}$ link. We partition the n packets transmitted (received) by the i^{th} node (the $(i+1)^{\text{th}}$ node) into b *departure* (*arrival*) buckets from the perspective of the i^{th} node (the $(i+1)^{\text{th}}$ node) such that the first bucket includes the first n/b packets transmitted (or received) by the node, the second bucket includes the second n/b packets and so forth. The packets transmitted by the i^{th} node might not be received in order by the $(i+1)^{\text{th}}$ node, and hence, for every $1 \leq j \leq b$, the packets in the j^{th} departure bucket of the i^{th} node might be different from those in the j^{th} arrival bucket of the $(i+1)^{\text{th}}$ node.

The expected number of packets pertaining to any given chunk in any departure or arrival bucket is $\mu = n/bq$, for any divisor bq of n; and the expected number of packets pertaining to any given chunk in all the departure or arrival buckets is n/q. The chunks are however randomly scheduled by the nodes and hence the actual number of packets pertaining to any given chunk in any bucket is a random variable.

Let Y denote the number of packets pertaining to a given chunk in a given bucket. Thus, $\mathbf{E}[Y] = \mu$. The set of labels of packets in this bucket is denoted by $\mathcal{I}$. Thus, $|\mathcal{I}| = n/b$. Every packet in this bucket happens to pertain to the given chunk independent of any other packet. The random variable Y can thus be written as the sum of independent indicator random variables $\{X_i : i \in \mathcal{I}\}$ where X_i is 1, if the i^{th} packet pertains to the given chunk and, X_i is 0, otherwise.

Lemma A.3 *[57, Corollary A.1.14] Let Y^* be the sum of mutually independent indicator (arbitrarily distributed) random variables. For every $0 < \gamma < 1$, $\Pr\{Y^* \geq (1+\gamma)\mathbf{E}[Y^*]\} \leq e^{-\gamma^2\mathbf{E}[Y^*]/2}$, and $\Pr\{Y^* \leq (1-\gamma)\mathbf{E}[Y^*]\} \leq e^{-\gamma^2\mathbf{E}[Y^*]/2}$.*

Let μ', and μ'' be $\mu - c\sqrt{\mu}$, and $\mu + c\sqrt{\mu}$, respectively, for some positive c. Taking $\gamma = c/\sqrt{\mu}$, and $c = \sqrt{2\ln(2/\epsilon)} = O(\sqrt{\log(1/\epsilon)})$ in Lemma A.3, it follows that $\Pr\{Y > \mu''\} < \dot{\epsilon}$, and $\Pr\{Y < \mu'\} < \dot{\epsilon}$, i.e., Y is less than μ', or greater than μ'', w.p. b.a.b. ϵ. Thus the number of packets for a given chunk in a given bucket fails to lie within μ' and μ'' w.p. b.a.b. ϵ.

We are interested in deriving an upper bound (or a lower bound) on the number of unusable (or usable) packets for every chunk.

Fix a particular non-source non-sink node, say the i^{th} node, for some $0 < i < l$. There exists at least one packet transmitted by the i^{th} node at some time $\tau' \geq \tau$, after receiving a packet at time τ. Thus, for every $1 \leq j < b$, all the packets in the j^{th} arrival bucket of the i^{th} node land before any of the packets in the $(j+1)^{\text{th}}$ departure bucket of the i^{th} node leaves. Thus, a given packet pertaining to a given chunk in the j^{th} arrival bucket of the i^{th} node can be matched up with any packet pertaining to that chunk in the $(j+1)^{\text{th}}$ departure bucket of the i^{th} node.

Consider the packets pertaining to a given chunk over the i^{th} and $(i+1)^{\text{th}}$ links (note that the $(i-1)^{\text{th}}$ node or the $(i+1)^{\text{th}}$ node might be the source node or the sink node, respectively). We randomly choose μ' packets (pertaining to the given chunk) in each departure (arrival) bucket of the $(i-1)^{\text{th}}$ and i^{th} nodes (the i^{th} and $(i+1)^{\text{th}}$ nodes) and call them *half-good* from the perspective of the $(i-1)^{\text{th}}$ node, or the i^{th} node (the i^{th} node, or the $(i+1)^{\text{th}}$ node), respectively. Since each packet over these two links belongs to one departure bucket (of the $(i-1)^{\text{th}}$ node, or the i^{th} node) and one arrival bucket (of the i^{th} node, or the $(i+1)^{\text{th}}$ node), we call a packet *good* if it is half-good from the perspective of both the $(i-1)^{\text{th}}$ and i^{th} nodes (or the i^{th} and $(i+1)^{\text{th}}$ nodes).

For every $1 < j \leq b$, there exists at least one particular good packet pertaining to any given chunk in the $(j-1)^{\text{th}}$ arrival bucket of the i^{th} node to be matched up with any given good packet pertaining to that chunk in the j^{th} departure bucket of the i^{th} node. Thus, the set of good packets pertaining to any given chunk in all but the last arrival bucket of the i^{th} node are a subset of usable packets for that chunk received by the i^{th} node, and hence the number of good packets for any given chunk in these buckets gives a lower bound on the number of usable packets for that chunk at the i^{th} node.

Taking $c = o(\sqrt{\mu})$, the number of those packets pertaining to a given chunk in any given bucket of any node which are not good fails to be less than $\mu'' \cdot O(c/\sqrt{\mu}) = O(c\sqrt{\mu})$ w.p. b.a.b. ϵ.[3] Taking a union bound over all but the last arrival bucket

[3]Hereafter, we operate under the assumption that the actual number of packets pertaining to any given chunk in any bucket lies within μ' and μ''. Let μ^* be the actual number of packets for a given chunk in a given bucket. We randomly choose μ' packets from the set of μ^* packets (i.e.,

of the i^{th} node, w.p. b.a.b. $(b-1)\epsilon$, the number of packets pertaining to a given chunk which are not good fails to be less than $(b-1)\cdot O(c\sqrt{\mu}) \leq b \cdot O(c\sqrt{\mu}) = O(bc\sqrt{n/bq}) = O(c\sqrt{nb/q})$.

The number of packets pertaining to a given chunk which are not usable due to landing at the b^{th} arrival bucket of the i^{th} node is, w.p. b.a.b. ϵ, larger than $\mu'' = \mu + c\sqrt{\mu} = (1+o(1))\mu = O(\mu) = O(n/bq)$. Such packets are unusable because there is no departure bucket at the i^{th} node, such that the packets therein can be matched up with the packets in the last arrival bucket of the i^{th} node.

Thus by adding the number of packets in both groups of unusable packets together, i.e., $O(c\sqrt{nb/q})$, and $O(n/bq)$, the probability that there are more than $O(c\sqrt{nb/q} + n/bq)$ unusable packets pertaining to a given chunk at the i^{th} node is not larger than $b\epsilon$. Summing over all the internal network nodes (i.e., for all $0 < i < L$), w.p. b.a.b. $(L-1)b\epsilon < Lb\epsilon$, the total number of unusable packets pertaining to a given chunk at the sink node is larger than

$$O((L-1)(c\sqrt{nb/q} + n/bq)) \leq O(L(c\sqrt{nb/q} + n/bq)). \tag{A.2}$$

We now specify c such that, for every chunk, the probability that the number of unusable packets pertaining to that chunk at the sink node is larger than (A.2) is upper bounded by ϵ. By replacing ϵ with ϵ/Lbq in Lemma A.3, one can see that c needs to be at least $O(\sqrt{\log(Lbq/\epsilon)})$. Let $c = O(\sqrt{\log(Ln/\epsilon)})$ (bq is a divisor of n and hence smaller than n). Minimizing the number of unusable packets for every chunk at the sink node with respect to b (by setting the derivative of (A.2) with

$\mu^* - \mu'$ packets will not be chosen). Therefore the probability that a given packet pertaining to the given chunk in the given bucket is not half-good from the perspective of its transmitting node is $(\mu^* - \mu')/\mu^*$. Since $\mu^* \leq \mu''$, the latter probability is upper bounded by $(\mu'' - \mu')/\mu''$. Similarly, from the perspective of its receiving node, the given packet (pertaining to the given chunk) fails to be half-good w.p. b.a.b. $(\mu'' - \mu')/\mu''$. Thus the probability that a given packet for a given chunk is not good (it is not half-good from the perspective of its transmitting or receiving node) is not larger than $2(\mu'' - \mu')/\mu'' \leq 2(\mu'' - \mu')/\mu' = 4c/(\sqrt{\mu} - c)$ (the inequality follows from $\mu' < \mu''$, and the equality follows from replacing μ' and μ'', respectively, with $\mu - c\sqrt{\mu}$, and $\mu + c\sqrt{\mu}$). Taking $c = o(\sqrt{\mu})$, we have $4c/(\sqrt{\mu} - c) \leq 4c/[(1-o(1))\sqrt{\mu}] \leq (1+o(1))(4c/\sqrt{\mu}) = O(c/\sqrt{\mu})$.

respect to b equal to zero and solving for b), we select

$$b \sim \left(\frac{n}{q\log(Ln/\epsilon)}\right)^{\frac{1}{3}}.$$

Such choices of c and b ensure that $c = o(\sqrt{\mu}) = o(\sqrt{n/bq})$, because $o(\sqrt{n/bq}) = o((n/q)^{\frac{1}{3}}\log^{\frac{1}{6}}(Ln/\epsilon))$, and $c = O(\sqrt{\log(Ln/\epsilon)}) = o((n/q)^{\frac{1}{3}}\log^{\frac{1}{6}}(Ln/\epsilon))$, so long as $q\log(Ln/\epsilon) = o(n)$. By substituting these values of c and b into (A.2), the total number of unusable packets for every chunk at the sink node is larger than

$$O\left(L\,(n/q)^{\frac{2}{3}}\log^{\frac{1}{3}}(Ln/\epsilon)\right),$$

w.p. b.a.b. ϵ. The expected number of packets for every chunk received by the sink node is n/q, and the actual number of such packets is lower bounded by $b\mu' = n/q - O((n/q)^{2/3}\ln^{1/3}(ln/\epsilon))$. For every chunk, the capacity of the traffic by the flow of the packets pertaining to a chunk (i.e., the number of usable packets pertaining to a chunk at the sink node) is therefore lower bounded by the difference between the total number of packets for that chunk at the sink node and the number of unusable packets for that chunk at the sink node. Thus, the capacity of the flow by the packets pertaining to every chunk fails to be larger than

$$\left(1 - O\left(\left(L^3(q/n)\log(Ln/\epsilon)\right)^{\frac{1}{3}}\right)\right).(n/q),$$

w.p. b.a.b. ϵ, so long as

$$L^3 q\log\frac{Ln}{\epsilon} = o\,(n)\,.^4$$

One should note that the last condition ensures that the number of unusable packets for every chunk at the sink node is asymptotically smaller than the total number of packets for that chunk at the sink node (otherwise, if this condition is violated, the lower bound given above on the capacity of the traffic by the packets for every chunk is not valid. The analysis of such cases will be given later in Section 3.4).□

[4]Assuming $L^3 q\log(Ln/\epsilon) = o(n)$, one can conclude that $O((L^3(q/n)\log(Ln/\epsilon))^{\frac{1}{3}})$ is asymptotically $o(1)$ with respect to n.

Proof of Lemma 3.12: By applying Lemma A.3, it can be shown that, for a given chunk, the number of packets over a given link fails to be larger than

$$\left(1 - O\left(((q/n)\log(1/\epsilon))^{\frac{1}{2}}\right)\right).(n/q), \tag{A.3}$$

w.p. b.a.b. ϵ.[5] To lower bound the number of packets for all the chunks, and over all the links, we take a union bound over the number of links and the number of chunks by setting $\epsilon = \epsilon/Lq$ in (A.3) and (A.4), yielding (3.4) and (3.5), respectively.□

Proof of Lemma 3.14: This is a direct result of Lemma 3.5, by replacing n with φ_n. □

Proof of Lemma 3.15: By taking a union bound over q, i.e., by replacing ϵ with $\dot{\epsilon}/q$ in Lemma 3.14, the first part of the lemma follows. By replacing ϵ with $\dot{\epsilon}$ in Lemma 3.10, the second part of the lemma follows. □

Proof of Lemma 3.16: This is a direct result of Lemma 3.9, by replacing n with φ. □

Proof of Lemma 3.17: By taking a union bound over q, i.e., by replacing ϵ with $\dot{\epsilon}/q$ in Lemma 3.16, the first part of the lemma follows. By replacing ϵ with $\dot{\epsilon}$ in Lemma 3.12, the second part of the lemma follows. □

[5]For a given chunk, let X_i, $1 \leq i \leq n$, be an indicator random variable, such that X_i is 1, if the i^{th} packet pertains to that chunk, and X_i is 0, otherwise. Since each packet pertains to a randomly (uniformly) chosen chunk, X_i's are independent Bernoulli random variables with $\Pr\{X_i = 1\} = 1/q$. Let Y be the number of packets pertaining to the given chunk over a given link, i.e., $Y = \sum_{1 \leq i \leq n} X_i$. Then, $\mathbf{E}[Y] = n/q$. By Lemma A.3, for every $0 < \gamma < 1$, $\Pr\{Y \leq (1-\gamma)n/q\} \leq e^{-\gamma^2 n/2q}$. Setting $e^{-\gamma^2 n/2q} = \epsilon$, it follows that $\gamma = \sqrt{2(q/n)\ln(1/\epsilon)} = O(\sqrt{(q/n)\log(1/\epsilon)})$, and hence we can write $\Pr\{Y \leq (1 - O(\sqrt{(q/n)\log(1/\epsilon)})) \cdot (n/q)\} \leq \epsilon$. Since the result of Lemma A.3 is valid for choices of γ such that $0 < \gamma < 1$, then, we need the condition

$$q\log(1/\epsilon) = o(n). \tag{A.4}$$

Proof of Lemma 3.18: The proof is similar to that of Lemma 3.10, except that we do not need to take a union bound over all the chunks. This comes from the fact that, in this setting, we need to find the capacity of the flow allocated to one chunk alone.□

Proof of Lemma 3.19: In this case, the traffic by the flow of the packets pertaining to a given chunk does not necessarily have the structure of a one-in-one-out worst-case traffic. However, since we are interested in analyzing the worst-case scenario, the traffic by the flow of the packets pertaining to a chunk needs to be also assumed to be a one-in-one-out worst-case traffic. Thus, the result follows from Lemma 3.5, by replacing n and ϵ with φ_n and $\dot{\delta}/2$, respectively. □

Proof of Lemma 3.20: By replacing k^* and n^* with α and $\varphi - L\log(L\varphi/\dot{\delta}) - L$, respectively, in Lemma 3.6, the result follows by a similar argument as in the proof of Theorem 3.1, except that in this case, we focus on the decoding failure probability of a given chunk alone, not all the chunks. □

Proof of Lemma 3.21: The proof is similar to that of Lemma 3.12, except that, in this case, we do not take a union bound over all the chunks, since we need to find a lower bound on the capacity of the flow allocated to one chunk alone, not all the chunks. □

Proof of Lemma 3.22: In this case, the traffic by the flow of the packets pertaining to a given chunk has itself always the structure of an all-in-all-out worst-case traffic. Thus, the result follows from Lemma 3.9, by replacing n and ϵ with φ and $\dot{\delta}/2$, respectively. □

Proof of Lemma 3.23: By replacing k^* and n^* with α and $\varphi - L\log(L/\dot{\delta}) - L$, respectively, in Lemma 3.6, the result follows from an argument similar to that of Theorem 3.2, while focusing on the decoding failure probability of a given chunk alone, not all the chunks. □

Proof of Lemma 3.26: The result follows from Conjecture 3.2 (asymmetric case), by replacing n^* and k^* with the number $(1 - O(((L^3/\mu)\log(L\mu\chi/\delta))^{\frac{1}{3}}))\chi\mu - \chi L\log(L\mu\chi/\dot{\delta}) - \chi L$ of rows pertaining to the given hyperchunk in $\hat{Q}_L$ that constitute the rows of the RARB (sub-) matrix under our consideration, and the number $\chi(\alpha - \gamma_o) + \gamma_o$ $(= r_o\alpha)$ of message vectors in the given hyperchunk, respectively. □

Proof of Lemma 3.27: The proof is similar to that of Lemma 3.17, yet, no union bound is taken over the chunks since we need the capacity of the flow allocated to one chunk alone. □

Proof of Lemma 3.28: The proof follows from Conjecture 3.2 (asymmetric case), similar to that of Lemma 3.20, by replacing n^* and k^* with the lower bound given on the number of rows pertaining to the chunks in a given hyperchunk which constitute an RARB matrix and the number of message vectors in that hyperchunk, respectively. □

A.2 Proofs of the Theorems

Proof of Theorem 3.1: The decoding fails if the number of innovative packets at the sink node is less than k. Replacing ϵ with $\dot{\epsilon}$ (for the definition of $\dot{\epsilon}$, please see Section 2.2) in Lemma 3.5, the number of LILE packets at the sink node is not larger than $n - L\log(nL/\dot{\epsilon})$ w.p. b.a.b. $\dot{\epsilon}$. Given that the sink node receives at least $n - L\log(nL/\dot{\epsilon})$ LILE packets, based on Lemma 3.6, the number of innovative packets in the set of LILE packets at the sink node is less than k w.p. b.a.b. $\dot{\epsilon}$, so long as $k \leq n - L\log(nL/\epsilon) - \log(1/\epsilon) - L - 1$. Hence, the probability of decoding failure is b.a.b. ϵ. □

Proof of Theorem 3.4: By replacing ϵ with $\dot{\epsilon}$ in Lemmas 3.10 and 3.11, it follows that: (i) the capacity of traffic by the flow of packets pertaining to every chunk

fails to be at least $\varphi \doteq n/q - O\left((L^3(n/q)^2 \log(Ln/\epsilon))^{\frac{1}{3}}\right)$, w.p. b.a.b. $\dot{\epsilon}$, so long as $L^3 q \log(nL/\epsilon) = o(n)$; and (ii) given that every chunk has been allocated a flow with capacity of at least φ, a dense code fails to transmit all the chunks successfully over the network w.p. b.a.b. $\dot{\epsilon}$, so long as $k \leq \varphi q - qL\log(nL/\dot{\epsilon}) - q\log(q/\dot{\epsilon}) - qL - q$. Thus, the probability of failure, when either (i), or (ii) occurs, is b.a.b. ϵ. □

Proof of Theorem 3.10: By replacing ϵ with $\dot{\epsilon}$ in Lemma 3.15 and Conjecture 3.2, one can see that (i) the capacity of the traffic by the flow of the packets pertaining to every chunk fails to be at least $(1 - O((L^3(q/n)\log(Ln/\epsilon))^{\frac{1}{3}})).(n/q)$, w.p. b.a.b. $\dot{\epsilon}/2$, so long as $L^3 q \log(nL/\epsilon) = o(n)$; (ii) if every chunk has been allocated a flow of capacity of at least $(1 - O((L^3(q/n)\log(Ln/\epsilon))^{\frac{1}{3}})).(n/q)$, there do not exist $n/q - O((L^3(n/q)^2\log(Ln/\epsilon))^{\frac{1}{3}}) - L\log(nL/\dot{\epsilon})$ LILE packets pertaining to every chunk at the sink node, w.p. b.a.b. $\dot{\epsilon}/2$; and (iii) if, for every chunk, the sink node receives at least $n/q - O((L^3(n/q)^2\log(Ln/\epsilon))^{\frac{1}{3}}) - L\log(nL/\dot{\epsilon})$ LILE packets, the decoding matrix Q_L at the sink node fails to have rank k, w.p. b.a.b. $\dot{\epsilon}$, so long as $k \leq n - O(q(L^3(n/q)^2\log(Ln/\epsilon))^{\frac{1}{3}}) - qL\log(nL/\dot{\epsilon}) - qL - \log(1/\dot{\epsilon})$; Thus the probability of failure, when either (i), or (ii), or (iii) occurs, is b.a.b. ϵ. □

Proof of Theorem 3.14: It follows from Theorems 3.10 and 3.11 that

$$(n - k_{n,\epsilon})/n = O\left(L\tau_o(q/n)\log(nL/\epsilon)\right),$$

for an OCC with chunk size k/q, and overlap parameter τ_o, so long as $q = o\left(n/(L^3\log(nL/\epsilon))\right)$. Thus, for given n, L, q and ϵ, the larger is τ_o, the larger is the ratio $(n - k_{n,\epsilon})/n$. □

Proof of Theorem 3.16: We define the sequence of n $(= (1+\gamma_c)k)$ independent random variables $h_1, ..., h_n$, denoted by vector h, such that h_i represents the index of the chunk to which the i^{th} packet received at the sink node pertains. We define $\Lambda(h)$ as the number of not fully decodable chunks given a specific vector h. The sequence of $X_0, X_1, ..., X_n$ defined as $X_i = \mathrm{E}[\Lambda(h)|\{h_j : 1 \leq j \leq i\}]$ yields a standard martingale

(for the definition, please see Appendix C.1). The functional Λ, as defined above, has also Lipschitz property in that the number of chunks that are not decodable differs by at most one if two sets of the indices of the chunks to which the n packets (at the sink node) pertain differ in only one index. The expected number of undecodable chunks, $\mathbf{E}[\Lambda(h)]$, is δq, and by applying Azuma's inequality (please refer to Appendix C.1), for an arbitrary constant $0 < \gamma_a < 1$, it follows that

$$\Pr\{\Lambda(h) \geq (1+\gamma_a)\delta q\} \leq e^{-\left(\frac{\gamma_a^2\delta^2}{2\alpha^2}\right)\cdot k}.$$

□

Proof of Theorem 3.17: We define the sequence of n $(= (1+\gamma_c)k)$ independent random variables $h_1, ..., h_n$, such that h_i represents the same random variable as defined in the proof of Theorem 3.16. Let $\Lambda(h)$ be the number of undecodable hyperchunks given a specific h (the vector of n outcomes h_i's). As in the proof of Theorem 3.16, a martingale sequence can be constructed based on Λ. However, unlike before, here, Λ has χ-Lipschitz property in that if two sets of output symbols differ in only one symbol, then the number of undecodable hyperchunks differ by at most χ (each output symbol pertains to one chunk, and each chunk belongs to χ hyperchunks). The expected number of undecodable hyperchunks, $\mathbf{E}[\Lambda(h)]$, is δq, and by applying Azuma's inequality, for an arbitrary constant $0 < \gamma_a < 1$, it follows that

$$\Pr\{\Lambda(h) \geq (1+\gamma_a)\delta q\} \leq e^{-\left(\frac{\gamma_a^2\delta^2}{2\alpha^2}\right)\cdot\left(\frac{\tau_o^2}{\chi}\right)k}.$$

□

Proof of Theorem 3.20: Let the parameter θ be defined as follows: in the case of OCC, $2 \leq \theta \leq \chi + \tau_o - 1$, for an arbitrary constant integer $\chi \gg (\tau_o - 1)/\gamma_c$; and in the case of CC, θ is equal to 1. For positive $\delta, \delta', \gamma_c$ and γ_c', and for some positive constants c and c', consider two OCC's with chunk sizes $\alpha = c(L/\gamma_c)^3\tau_o \log((L/\gamma_c\delta)\tau_o)$ and $\alpha' = c'(L/\gamma_c')^3\tau_o' \log((L/\gamma_c'\delta')\tau_o')$, and constant overlap parameters τ_o and τ_o' $(\tau_o' < \tau_o)$,

over two line networks of length L with two traffics of capacities $(1+\gamma_c)k$, and $(1+\gamma_c')k$, respectively. By the result of Theorem 3.18, it can be seen that for a given MER (i.e., $\delta^{\theta}\tau_o q = \delta'^{\theta'}\tau_o' q$), and for a given chunk size (i.e., $\alpha = \alpha'$), the speed of convergence of OCC with overlap parameter τ_o is higher than that of OCC with overlap parameter τ_o' (i.e., $\gamma_c < \gamma_c'$), so long as

$$\delta < \left[L^{\gamma'-\gamma} \frac{(\tau_o'^2/c')^{\frac{\gamma'}{3}}}{(\tau_o^2/c)^{\frac{\gamma}{3}}} \left(\frac{\tau_o'}{\tau_o}\right)^{\gamma'} \right]^{\frac{1}{\kappa(\gamma-\gamma')+\gamma'(\theta/\theta'-1)}},$$

and $c'/c > \tau_o/\tau_o'$, for some constant $\kappa > 0$, where $\gamma = c\tau_o/\gamma_c^3$, and $\gamma' = c'\tau_o'/\gamma_c'^3$. □

Proofs of Theorems 3.18 and 3.19: By combining the following results, Theorems 3.18 and 3.19 are immediate for any type of dependency between hyperchunks.

Theorem A.1 *For every $0 < \delta < 1$, and an arbitrary constant $0 < \gamma_c < 1$, by applying an OCC with chunk size α, and overlap parameter τ_o, to transmit k message vectors over a line network of length L with any one-in-one-out worst-case traffic of capacity $(1+\gamma_c)k$, for any type of dependency between hyperchunks with positively dependent neighborhoods, for an arbitrary constant $0 < \gamma_a < 1$, the fraction of blocks that are not recoverable is larger than $(1+\gamma_a)\xi$, w.p. b.a.b. ϵ, so long as*

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\tau_o \log\left(\frac{L}{\gamma_c \delta}\tau_o\right)\right),$$

and $(\alpha^2/\gamma_a^2\delta^4)\cdot(1/\gamma_c\tau_o) = o(k/\log(1/\epsilon))$.

Proof of Theorem A.1: The worst case in this type occurs when, for every $i \in I$, and every subset I_i of $I\backslash\{i\}$, such that $I_i \cap N_{\mathcal{G}}(i) = \emptyset$, it follows that $\Pr\{G_i | \cap_{j\in I_i} G_j\} = \Pr\{G_i\}$, and for every other I_i, $\Pr\{G_i | \cap_{j\in I_i} G_j\} = 1$.

For every $i \in I$, let $N_{\mathcal{B}}(i)$ be an ordered set (in an increasing cyclic order) of indices of the hyperchunks that overlap with the i^{th} block. Each block is unrecoverable if the hyperchunks including it are all undecodable. Thus, for every $i \in I$, $\Pr\{B_i\} = \Pr[\cap_{k_\star \in N_{\mathcal{B}}(i)} G_{k_\star}]$. Let l denote the number of blocks in a hyperchunk. For

a given $i \in I$, consider a permutation $\{k_1, ..., k_l\}$ of the elements of the set $N_{\mathcal{B}}(i)$, so that $k_1 \notin N_{\mathcal{G}}(k_2)$. Such a permutation always exists as $|N_{\mathcal{B}}(i)| > |N_{\mathcal{G}}(k_j)|$, for every $1 \leq j \leq l$. Thus, $\Pr[G_{k_2}|G_{k_1}] = \Pr[G_{k_2}]$. It can also be shown that for such a permutation, the hyperchunks $\{\mathcal{G}_{k_\star} : k_\star \in \{k_3, ..., k_l\}\}$ overlap with at least one of the hyperchunks $\mathcal{G}_{k_1}$ and $\mathcal{G}_{k_2}$. In the setting of our interest, the probability that a hyperchunk fails to be decodable is shown to be b.a.b. δ, and hence, for every $k_\star \in N_{\mathcal{B}}(i)$, $\Pr[G_{k_\star}] \leq \delta$. By the above argument along with applying chain rule, it follows as $\Pr\{B_i\} = \Pr\{\cap_{k_\star \in N_{\mathcal{B}}(i)} G_{k_\star}\} = \Pr\{G_{k_1}\} \Pr\{G_{k_2}|G_{k_1}\} = \Pr\{G_{k_1}\} \Pr\{G_{k_2}\} \leq \delta^2 \doteq \xi$.

Each block is unrecoverable w.p. b.a.b. ξ, and for any choice of δ, so long as ξ goes to zero sufficiently fast with k (i.e., ξq goes to 0, as k goes to infinity), by applying a union bound, one can see that not all the blocks are recoverable w.p. b.a.b. ξq. The tightness of this upper bound can be shown by rewriting $\Pr\{\cap_{k_\star \in N_{\mathcal{B}}(i)} G_{k_\star}\}$ as $1 - \Pr\{\cup_{k_\star \in N_{\mathcal{B}}(i)} \overline{G}_{k_\star}\}$, and using the inclusion-exclusion principle.

Since the expected fraction of unrecoverable blocks is upper bounded by ξ, for larger choices of δ up to a constant, by constructing a martingale as before (yet this time, over the blocks, not the hyperchunks), we can prove the concentration of the actual fraction of unrecoverable blocks around its expectation as follows. The proof is similar to that of Theorem 3.17, except that Λ is l-Lipschitz in this case. The expected number of unrecoverable blocks is ξq (in worst-case scenario), and by applying Azuma's inequality, for an arbitrary constant $0 < \gamma_a < 1$, it follows that

$$\Pr\{\Lambda(h) > (1 + \gamma_a)\xi q\} \leq e^{-ck},$$

by selecting $c = O((\gamma_a^2 \xi^2 / \alpha^2)(\tau_o^2 / l))$.

Each block has k/q message vectors. Therefore, the result of Theorem A.1 shows that for any type of dependency between hyperchunks with positively dependent neighborhoods, w.h.p., for an arbitrary constant $0 < \gamma_a < 1$, the number of unrecoverable message vectors is not larger than $(1 + \gamma_a)\xi k$. □

Theorem A.2 *For every $0 < \delta < 1$, and an arbitrary constant $0 < \gamma_c < 1$, by applying an OCC with chunk size α, and overlap parameter τ_o, over a line network*

of length L with any one-in-one-out worst-case traffic of capacity $(1+\gamma_c)k$, for any type of dependency between hyperchunks with negatively dependent neighborhoods, for an arbitrary constant $0 < \gamma_a < 1$, the fraction of blocks that are not recoverable is larger than $(1+\gamma_a)\eta$, w.p. b.a.b. ϵ, so long as

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\tau_o \log\left(\frac{L}{\gamma_c \delta}\tau_o\right)\right),$$

and $\alpha/\gamma_c\tau_o = o(k/\log\log(1/\epsilon))$.

Proof of Theorem A.2: The worst case in this type occurs when, for every $i \in I$, and every subset I_i of $I \setminus \{i\}$, $\Pr\{G_i | \cap_{j\in I_i} G_j\} = \Pr\{G_i\}$. That is, for every $i \in I$, G_i's are independent. Thus, $\Pr\{B_i\} = \Pr\{\cap_{k_\star \in N_{\mathcal{B}}(i)} G_{k_\star}\} = \prod_{k_\star \in N_{\mathcal{B}}(i)} \Pr\{G_{k_\star}\} \leq \delta^{|N_{\mathcal{B}}(i)|} = \delta^l \doteq \eta$.

Now, for any choice of δ, so long as η goes to zero sufficiently fast with k (i.e., ηq goes to 0, as k goes to infinity), by applying a union bound, one can see that not all the blocks are recoverable w.p. b.a.b. ηq. We prove the tightness of this upper bound below.

For every $i \in I$, let $\mathcal{A}$ be the union set of events H_i, such that H_i is either G_i, or $\overline{G_i}$. Further, let A_i be the intersection set of events $\{G_{k_\star}\}_{k_\star \in N_{\mathcal{B}}(i)}$, and B_i be the event $A_i \subseteq \mathcal{A}$ (i.e., B_i occurs if the intersection of the events $\{G_{k_\star}\}_{k_\star \in N_{\mathcal{B}}(i)}$ occurs). Then, for $i, j \in I$ $(i \neq j)$, and $N_{\mathcal{B}}(i) \cap N_{\mathcal{B}}(j) \neq \emptyset$, one can see that $i \sim j$ (for more details and the definition of relation "$\sim$," please refer to Appendix D.2); also, for $i \neq j$, and $i \not\sim j$, B_i and B_j are independent due to the dependence on two disjoint subsets of $\{G_{k_\star}\}_{k_\star \in I}$. For $i, j \in I$ $(i < j)$, and $i \sim j$, $N_{\mathcal{B}}(i) = \{k_1, ..., k_l\}$, $N_{\mathcal{B}}(j) = \{k_{j-i+1}, ..., k_l, ..., k_{l+j-i}\}$ (in cyclic order), and $N_{\mathcal{B}}(i) \cup N_{\mathcal{B}}(j) = \{k_1, ..., k_l, ..., k_{l+j-i}\}$.

Thus, $|N_{\mathcal{B}}(i) \cup N_{\mathcal{B}}(j)| = l + j - i$, and

$$\begin{aligned}
\Pr\{B_i \cap B_j\} &= \Pr\{\{\cap_{k_* \in N_{\mathcal{B}}(i)} G_{k_*}\} \cap \{\cap_{k_* \in N_{\mathcal{B}}(j)} G_{k_*}\}\} \\
&= \Pr\{\cap_{k_* \in N_{\mathcal{B}}(i) \cup N_{\mathcal{B}}(j)} G_{k_*}\} \\
&= \prod_{k_* \in N_{\mathcal{B}}(i) \cup N_{\mathcal{B}}(j)} \Pr\{G_{k_*}\} \\
&= \Pr\{G_{k_*}\}^{|N_{\mathcal{B}}(i) \cup N_{\mathcal{B}}(j)|} \\
&\leq \delta^{|N_{\mathcal{B}}(i) \cup N_{\mathcal{B}}(j)|} \\
&= \delta^{l+j-i}.
\end{aligned}$$

Thus,

$$\begin{aligned}
\Phi &= \sum_{i \sim j} \Pr\{B_i \cap B_j\} \\
&= \sum_{i \in I} \sum_{j : i < j < i+l} \Pr[B_i \cap B_j] \\
&\leq \sum_{i \in I} \sum_{j : i < j < i+l} \delta^{l+j-i} \\
&= \sum_{i \in I} \sum_{0 < j-i < l} \delta^{l+j-i} \\
&= \eta \sum_{i \in I} \sum_{0 < j-i < l} \delta^{j-i} \\
&= \eta \sum_{i \in I} (\delta - \eta)/(1 - \delta) \\
&= \eta q (\delta - \eta)/(1 - \delta).
\end{aligned}$$

For sufficiently small choice of δ (i.e., ηq goes to 0, as k goes to infinity), one can see that Φ goes to 0. Thus, in the limit of interest, the Janson's inequality results in $\Pr\{\cap_{i \in I} \overline{B_i}\} = P$. By an earlier argument, it follows that $\Pr\{B_i\} \leq \eta$, and thus $P = \prod_{i \in I} \Pr\{\overline{B_i}\} \geq (1 - \eta)^q \geq 1 - \eta q$.

For larger values of ϵ, a given block is unrecoverable w.p. b.a.b. η, and hence, an upper bound on the fraction of unrecoverable blocks is η. The actual fraction

of unrecoverable blocks can also be shown to be tightly concentrated around the expectation (in worst-case scenario) as follows. Let I be the set of integers $\{i\}_{1\leq i\leq q}$. For every $i \in I$, let X_i be an indicator random variable associated with the i^{th} block ($\mathcal{B}_i$), such that X_i is 1 if B_i occurs, and it is 0, otherwise. Then, $X = \sum_{i\in I} X_i$ is the number of unrecoverable blocks. Each block belongs to l contiguous hyperchunks, and hence the event of a given block being unrecoverable is a function of the events of decodability of the hyperchunks containing it. For every $j \in I$, j^{th} block belongs to l contiguous hyperchunks, and the next block which does not belong to any of these hyperchunks is the $(j+l)^{\text{th}}$ block. Thus, for every $j \in I$, the events $B_j, B_{j+l}, ..., B_{j+q-l}$ are independent. Taking $I_j = \{j, j+l, ..., j+q-l\}$, for every $1 \leq j \leq l$, the family $\{I_j\}$ of subsets of I is the smallest proper cover of I (for the definition, please see Appendix D.3), because for every $1 \leq j \leq l$, I_j is the largest possible subset which contains all the possible indices whose associated indicator random variables $\{X_i\}_{i\in I_j}$ are independent. The size M of such a proper cover of I is l. To give an upper bound on $M_{\min}$, let us expand the family $\{I_j\}$ as follows: the new family $U(\{I_j\})$ contains any non-empty subset of I_j, for each $1 \leq j \leq l$ (i.e., $U(\{I_j\})$ is the union of the powersets of subsets I_j's, excluding the null set).[6] By definition, $U(\{I_j\})$ is a cover. The number of non-empty subsets of I_j is $2^{q/l} - 1$. By symmetry, each of the q/l distinct indices is equally distributed in these subsets, and hence each index appears $(2^{q/l} - 1)/(q/l)$ times. By selecting ω_j to be $(q/l)/(2^{q/l} - 1)$, one can see that, for every $i \in I$, the inequality $\sum_{j:i\in I_j} \omega_j \geq 1$ holds with equality. Since there are q indices in I, $\sum_{j\in I} \omega_j = q \cdot \big((q/l)/(2^{q/l} - 1)\big)$. Thus, $M_{\min} \leq q \cdot \big((q/l)/(2^{q/l} - 1)\big)$. As discussed earlier (in worst-case scenario), $\Pr\{B_i\} = \eta$, and the expected number of unrecoverable blocks, $\mathbf{E}[X]$, is ηq. For an arbitrary constant $0 < \gamma_a < 1$, thus, by using Lemma D.2, it follows that

$$\Pr\{X \geq (1+\gamma_a)\eta q\} \leq e^{-e^{ck}},$$

by selecting $c = O(\tau/\alpha l) - \left(O(\log(k)) - O(\log((\alpha l/\tau_o)\gamma_a^2\eta^2))\right)/k$.

Therefore, for any type of dependency between hyperchunks with negatively

[6]The powerset of a set S is the set of all subsets of S, including the null set and the set S itself.

dependent neighborhoods, w.h.p., for any arbitrary constant $0 < \gamma_a < 1$, the number of unrecoverable message vectors is upper bounded by $(1 + \gamma_a)\eta k$. $\square$

Appendix B

Proofs of the Results in Chapter 4

B.1 Proofs of the Lemmas

Proof of Lemma 4.1: For any integer $0 \leq \gamma \leq n^* - 1$, let T' be T restricted to its first $n^* - \gamma$ columns. Since T' is an $w^* r^* \times (n^* - \gamma)$ sub-matrix of T, $\Pr\{\text{rank}(T) < n^* - \gamma\} \leq \Pr\{\text{rank}(T') < n^* - \gamma\}$. Suppose that $\text{rank}(T') < n^* - \gamma$. Then there exists a nonzero column vector $\boldsymbol{v}$ of length $n^* - \gamma$ over $\mathbb{F}_2$ such that the column vector $T'\boldsymbol{v}$ of length $w^* r^*$ is an all-zero vector. For a given integer $1 \leq j \leq n^* - \gamma$, suppose that the first nonzero entry of $\boldsymbol{v}$ is the j^{th}. There exists $2^{n^* - \gamma - j}$ such vectors. Let us define $r_0^* \doteq 0$ for convenience. Let τ^* be an integer satisfying $\sum_{0 \leq i \leq \tau^*} r_i^* < j \leq \sum_{0 \leq i \leq \tau^*+1} r_i^*$, and $\tau_{\max}^*$ be an integer satisfying $\sum_{0 \leq i \leq \tau_{\max}^*} r_i^* < n^* - \gamma \leq \sum_{0 \leq i \leq \tau_{\max}^*+1} r_i^*$. By the definition, it follows that $0 \leq \tau_{\max}^* \leq \min\{w^*, u^* - 1\}$. It should not be hard to see that τ^* and $\tau_{\max}^*$ are unique. For every $0 \leq \tau \leq \tau_{\max}^*$, define $s_\tau^* = \sum_{0 \leq i \leq \tau} r_i^*$. The j^{th} column of T' has at least $(w^* - \tau^*) r^*$ i.u.d. Bernoulli entries, and hence the vector $T'\boldsymbol{v}$ has at least $(w^* - \tau^*) r^*$ i.u.d. Bernoulli entries. Thus, $T'\boldsymbol{v}$ is all-zero w.p. b.a.b.

$2^{-\gamma+n^*-w^*r^*}\sum_{1\le j\le n^*-\gamma}2^{\tau^*r^*-j}$, noting that τ^* depends on j. We rewrite the sum as:

$$\sum_{0<j\le s_1^*}2^{-j}+\sum_{s_1^*<j\le s_2^*}2^{r^*-j}+$$
$$\cdots+\sum_{s^*_{\tau^*_{\max}}<j\le n^*-\gamma}2^{\tau^*_{\max}r^*-j}=$$
$$\sum_{0<j\le r_1^*}2^{-j}+2^{r^*-s_1^*}\sum_{0<j\le r_2^*}2^{-j}+$$
$$\cdots+2^{\tau^*_{\max}r^*-s^*_{\tau^*_{\max}}}\sum_{0<j\le n^*-\gamma-s^*_{\tau^*_{\max}}}2^{-j}\le$$
$$\sum_{0<j\le r^*_{\max}}2^{-j}+2^{r^*-s_1^*}\sum_{0<j\le r^*_{\max}}2^{-j}+$$
$$\cdots+2^{\tau^*_{\max}r^*-s^*_{\tau^*_{\max}}}\sum_{0<j\le r^*_{\max}}2^{-j}=$$
$$\sum_{0<j\le r^*_{\max}}2^{-j}\sum_{0\le\tau\le\tau^*_{\max}}2^{\tau r^*-s^*_\tau}\le$$
$$\sum_{0<j\le r^*_{\max}}2^{-j}\sum_{0\le\tau\le\tau^*_{\max}}2^{(r^*-r^*_{\min})\tau}=$$
$$(1-2^{-r^*_{\max}})\sum_{0\le\tau\le\tau^*_{\max}}2^{(r^*-r^*_{\min})\tau}.$$

The series $\sum_{0\le\tau\le\tau^*_{\max}}2^{(r^*-r^*_{\min})\tau}$ converges from below to $(\tau^*_{\max}+1)2^{(r^*-r^*_{\min})\tau^*_{\max}}$ if $r^*-r^*_{\min}$ goes to infinity. Thus the following is always true: $(1-2^{-r^*_{\max}})\sum_{0\le\tau\le\tau_{\max}}2^{(r^*-r^*_{\min})\tau}\le(\tau^*_{\max}+1)(1-2^{-r^*_{\max}})2^{(r^*-r^*_{\min})\tau^*_{\max}}\le u^*(1-2^{-r^*_{\max}})2^{(r^*-r^*_{\min})(u^*-1)}$. This proves the lemma. □

Proof of Lemma 4.2: We start the proof by noting that T has a smaller number of rows than columns, and the minimum number of rows and columns gives an upper bound on the rank of the matrix. Let T' be T restricted to its last $n^*-\gamma$ rows. For every $0\le\tau\le w^*$, define $s^*_\tau=\sum_{0\le j\le w^*-\tau}r^*_j$. Thus, T' is of size $(n^*-\gamma)\times s^*_0$. Suppose that there exists a nonzero row vector $\boldsymbol{v}$ of length $n^*-\gamma$ whose entries are over $\mathbb{F}_2$, and its first nonzero entry is the j^{th}, and the row vector $\boldsymbol{v}T'$ is all-zero. There

are $2^{n^*-\gamma-j}$ such vectors. Let τ^* be the largest integer smaller than j/r^*. The j^{th} row of T' has at least $s^*_{\tau^*}$ i.u.d. Bernoulli entries, and hence the vector $\boldsymbol{v}T'$ has at least $s^*_{\tau^*}$ i.u.d. Bernoulli entries. Thus, $\boldsymbol{v}T'$ is all-zero w.p. b.a.b. $2^{-\gamma+n^*}\sum_{1\le j\le n^*-\gamma}2^{-j-s^*_{\tau^*}}$. By definition, $s^*_\tau \ge (w^*-\tau)r^*_{\min}$, and the preceding sum can thus be upper bounded as follows: $\sum_{1\le j\le n^*-\gamma}2^{-j-s^*_{\tau^*}} \le \sum_{1\le j\le n^*-\gamma}2^{-j-(w^*-\tau^*)r^*_{\min}}$. The latter sum can be rewritten itself as:

$$\sum_{0<j\le r^*}2^{-j-w^*r^*_{\min}}+\sum_{r^*<j\le 2r^*}2^{-j-(w^*-1)r^*_{\min}}+$$
$$\cdots+\sum_{(u^*-1)r^*<j\le n^*-\gamma}2^{-j-(w^*-u^*+1)r^*_{\min}}=$$
$$2^{-w^*r^*_{\min}}\sum_{0<j\le r^*}2^{-j}+2^{-(w^*-1)r^*_{\min}-r^*}\sum_{0<j\le r^*}2^{-j}+$$
$$\cdots+2^{-(w^*-1)r^*_{\min}-(u^*-1)r^*}\sum_{0<j\le n^*-\gamma-(u^*-1)r^*}2^{-j}\le$$
$$2^{-w^*r^*_{\min}}\sum_{0<j\le r^*}2^{-j}\sum_{0\le\tau\le u^*-1}2^{(r^*_{\min}-r^*)\tau}=$$
$$(1-2^{-r^*})2^{-w^*r^*_{\min}}\sum_{0\le\tau\le u^*-1}2^{(r^*_{\min}-r^*)\tau}.$$

The last sum is bounded from above by $2^{(r^*_{\min}-r^*)(u^*-1)}u^*$, and this completes the proof. □

Proof of Lemma 4.3: Fix $1 < i \le L$. Let $\hat{T}_{i-1}$ be the modified transfer matrix at the starting node of the i^{th} link. Let $\hat{T}^1_{i-1}$ be $\hat{T}_{i-1}$ restricted to the packets in the first active partition over the i^{th} link. For every $1 < s < i$, suppose $\mathcal{D}(Q^1_s) \ge \mathcal{D}_p(Q^1_s)$, where $\mathcal{D}_p(Q^1_s) = r - \log(w_T/\epsilon) - 1$, and $\mathcal{D}(Q^1_1) = \mathcal{D}_p(Q^1_1) = r$. Then, by replacing w^*, r^* and $\{r^*_l\}_{1\le l\le w^*}$ with $1, r$ and $\hat{r}_1$, respectively, in Lemma 4.1, where $\hat{r}_1 \doteq \hat{r}_{i-1,1}$,[1] one can see that $\hat{T}^1_{i-1}$ includes an $r \times \hat{r}_1$ dense sub-matrix. Thus by applying Lemma 4.1, for every $0 \le \gamma \le \hat{r}_1 - 1$, $\Pr\{\text{rank}(\hat{T}^1_{i-1}) < \hat{r}_1 - \gamma\} \le u^*(1-2^{-\hat{r}_1})2^{-\gamma+\hat{r}_1-r+(r-\hat{r}_1)(u^*-1)}$, where

[1]We often drop the subscript i in the notation r_{ij} when there is no danger of confusion.

$u^* = \lceil(\hat{r}_1 - \gamma)/\hat{r}_1\rceil$. Thus, $\Pr\{\text{rank}(\hat{T}^1_{i-1}) < (1-2^{-\hat{r}_1})2^{-\gamma+\hat{r}_1-r}$, since $u^* = 1$. Taking $\gamma = \log(w_T/\epsilon) + \hat{r}_1 - r + 1$, it follows that $\Pr\{\text{rank}(\hat{T}^1_{i-1}) < \hat{r}_1 - \gamma\} \leq \dot{\epsilon}/w_T$. By Lemma 3.2, $\mathcal{D}(Q^1_i) < r - \log(w_T/\epsilon) - 1$ w.p. b.a.b. $\dot{\epsilon}/w_T$. Thus, $\mathcal{D}_p(Q^1_i) = r - \log(w_T/\epsilon) - 1$. Taking a union bound over the first i links, $\mathcal{D}(Q^1_i) < r - \log(w_T/\epsilon) - 1$ w.p. b.a.b. $i\dot{\epsilon}/w_T$. □

Proof of Lemma 4.4: Fix $1 < i \leq L$. For every $1 < s \leq i$ and $1 < \tau \leq j$, except $(s,\tau) = (i,j)$, suppose $\mathcal{D}(Q^\tau_s) \geq \mathcal{D}_p(Q^\tau_s)$, where $\mathcal{D}_p(Q^\tau_s) = r\tau - \tau(1+o(1))(\log(w_T/\epsilon)+1) - \log((\tau(1+o(1))+1)/\epsilon) - \log w_T - 1$, and the $o(1)$ term is $(\log(w_T/\epsilon)+1)/r$, and $\mathcal{D}(Q^1_s) \geq \mathcal{D}_p(Q^1_s)$, where $\mathcal{D}_p(Q^1_s) = r - \log(w_T/\epsilon) - 1$. Let $\hat{r}_\tau = \hat{r}_{i-1,\tau}$, for every $1 \leq \tau \leq j$, $\hat{r}_{\min} = \min_\tau \hat{r}_\tau$, and $\hat{r}_{\max} = \max_\tau \hat{r}_\tau$. Let $n^* = \mathcal{D}_p(Q^j_{i-1}) = \sum_{1\leq\tau\leq j}\hat{r}_\tau$. Let us define $\hat{T}_{i-1}$ as in the proof of Lemma 4.3. Let $\hat{T}^j_{i-1}$ be $\hat{T}_{i-1}$ restricted to the packets in the first j active partitions over the i^{th} link. Then, by replacing w^*, r^* and $\{r^*_l\}_{1\leq l\leq w^*}$ with j, r and $\{\hat{r}_\tau\}_{1\leq\tau\leq j}$, respectively, in Lemma 4.1, one can see that $\hat{T}^j_{i-1}$ includes an $rj \times n^*$ sub-matrix with a structure similar to the matrix T as in Lemma 4.1. Thus by applying Lemma 4.1, for every $0 \leq \gamma \leq n^* - 1$, $\Pr\{\text{rank}(\hat{T}^j_{i-1}) < n^* - \gamma\} \leq u^*(1-2^{-\hat{r}_{\max}})2^{-\gamma+n^*-rj+(r-\hat{r}_{\min})(u^*-1)}$, where $u^* = \lceil(n^* - \gamma)/\hat{r}_{\min}\rceil$. It is not difficult to see that, by our method of collecting the LILE packets, it follows that $\hat{r}_{\min} = \hat{r}_1$. Further by applying Lemma 4.3, $\hat{r}_1 = \mathcal{D}_p(Q^1_{i-1}) = r - \log(w_T/\epsilon) - 1$. Thus, $u^* \leq \lceil rj/\hat{r}_1\rceil = \lceil(1+o(1))j\rceil \leq (1+o(1))j + 1$, since $\hat{r}_1 = r(1-o(1))$, given $\log(w_T/\epsilon) = o(r)$, where the $o(1)$ term is $(\log(w_T/\epsilon)+1)/r$. Since $r \sim \varphi = pN_T/w_T$, the latter condition can be written as $w\log(w_T/\epsilon) = o(pN_T)$. Taking $\gamma = n^* - rj + (1+o(1))j(\log(w_T/\epsilon)+1) + \log(((1+o(1))j+1)/\epsilon) + \log w_T + 1$, it follows that $\Pr\{\text{rank}(\hat{T}^j_{i-1}) < n^* - \gamma\} \leq \dot{\epsilon}/w_T$. Now, by applying Lemma 3.2, $\mathcal{D}(Q^j_i) < n^* - \gamma$ w.p. b.a.b. $\dot{\epsilon}/w_T$. Thus, $\mathcal{D}_p(Q^j_i) = n^* - \gamma$. Taking a union bound over the first j active partitions of the first i links, $\mathcal{D}(Q^j_i) < rj - (1+o(1))j(\log(w_T/\epsilon)+1) - \log(((1+o(1))j+1)/\epsilon) - \log w_T - 1$ w.p. b.a.b. $ij\dot{\epsilon}/w_T$, where the $o(1)$ term is $(\log(w_T/\epsilon)+1)/r$. This completes the proof.□

Proof of Lemma 4.5: For the ease of exposition, let $v = w_T/L$. Lemma 4.4 gives a lower bound on $\mathcal{D}(Q_L^v)$. Thus, we can write: $\mathcal{D}(Q_L) \geq \mathcal{D}(Q_L^v) \geq rv - v(1+o(1))\left(\log(w_T/\epsilon)+1\right) - \log((v(1+o(1)))/\epsilon) - \log w_T - 1$, where the $o(1)$ term is $\left(\log(w_T/\epsilon)\right)/r$. This bound fails w.p. b.a.b. $\dot{\epsilon}$, given the assumption that the number of packets in each active partition is larger than or equal to r. Since this assumption fails w.p. b.a.b. $\dot{\epsilon}$, the lower bound on $\mathcal{D}(Q_L)$ fails w.p. b.a.b. ϵ. Further, $r = (1-o(1))\varphi$, where the $o(1)$ term is $\sqrt{(1/\dot{\varphi})\ln(w_T/\dot{\epsilon})}$. Thus, $\mathcal{D}(Q_L) \geq \varphi v - o(\varphi v) - v\log(w_T/\epsilon) - v - o(v\log(w_T/\epsilon)) - o(v) - \log(v/\epsilon) - \log w_T - 1$ fails w.p. b.a.b. ϵ, where $o(\varphi v) \sim O(\varphi v\sqrt{(1/\varphi)\log(w_T/\epsilon)})$, and $o(v) \sim (v/\varphi)\log(w_T/\epsilon)$. By considering the dominant terms, the right-hand side of the last inequality can be written as

$$\varphi v - O(v\sqrt{\varphi\log(w_T/\epsilon)}) - O(v\log(w_T/\epsilon)). \tag{B.1}$$

We now replace φ and v by pN_T/w and w ($v \sim w$), respectively, and rewrite (B.1) as

$$\begin{aligned} &pN_T - O(pN_TL/w) - \\ &\qquad O(\sqrt{pN_Tw\log(wL/\epsilon)}) - O(w\log(wL/\epsilon)), \end{aligned} \tag{B.2}$$

by using the fact that w_T is $O(wL)$. We select w to be

$$\sqrt[3]{\frac{pN_TL^2}{\log(pN_TL/\epsilon)}}$$

in order to maximize (B.2) subject to condition (4.7). This choice of w ensures that each $O(.)$ term in (B.2) is $o(pN_T)$, and hence the coding scheme is capacity-achieving. □

B.2 Proofs of the Theorems

Proof of Theorem 4.2: The proof follows the same line as that of Theorem 4.1, except that r needs to be replaced with φ in the proof of Lemma 4.5. Thus, the $O(v\sqrt{\varphi \log(w_T/\epsilon)})$ term in (B.1) and the $O(\sqrt{pN_T w \log(wL/\epsilon)})$ term in (B.2) disappear. Then, it should not be hard to see that the choice of w needs to maximize

$$pN_T - O(pN_T L/w) - O(w \log(wL/\epsilon)), \tag{B.3}$$

instead of (B.2), subject to condition (4.7). This can be done by selecting w to be

$$\sqrt{\frac{pN_T L}{\log(pN_T L/\epsilon)}}.$$

□

Proof of Theorem 4.5: The proof follows the same line as that of Theorem 4.1 by implementing the following modifications. Let us replace p and ϵ with p/q and ϵ/q, respectively. Then, $\varphi = pN_T/wq$, and $r = (1-\gamma^*)\varphi$, where $\gamma^* \sim \sqrt{(1/\dot{\varphi}) \ln(w_T q/\dot{\epsilon})}$. For every $1 \le i \le L$, and $1 \le j \le w - L + 1$, let $\mathcal{D}(Q_i^j)$, $\mathcal{D}_p(Q_i^j)$, and r_{ij} be defined as in Section 4.3.1, but only restricted to the packets pertaining to a given chunk (not all the chunks). For every i, j, $\mathcal{D}(Q_i^j)$ can be lower bounded as follows (the proofs are very similar to those of Lemmas 4.3 and 4.4): for every $1 \le j \le w - L + 1$, $\mathcal{D}(Q_1^j) \ge rj$, and for every $1 < i \le L$ and $1 \le j \le w - L + 1$, $\mathcal{D}(Q_i^j)$ fails to be larger than $rj - j(1 + o(1)) \log(w_T q/\epsilon)$, w.p. b.a.b. $ij\dot{\epsilon}/w_T q$, so long as

$$wq \log \frac{w_T q}{\epsilon} = o(pN_T). \tag{B.4}$$

Thus the number of LILE packets pertaining to a given chunk at the sink node fails

to be larger than

$$\frac{pN_T}{q} - O\left(\frac{pN_TL}{wq}\right) - O\left(\sqrt{\frac{pN_Tw}{q}\log\frac{wqL}{\epsilon}}\right) - O\left(w\log\frac{wqL}{\epsilon}\right) \tag{B.5}$$

w.p. b.a.b. ϵ/q. In order to maximize (B.5) subject to condition (B.4), we select w to be

$$\sqrt[3]{\frac{pN_TL^2}{q\log(pN_TL/\epsilon)}}.$$

Now let us assume that N_T is $(1+o(1))k/p$. By replacing ϵ with $\dot{\epsilon}$, in the preceding results, and by replacing k and ϵ with k/q and $\dot{\epsilon}/q$, respectively, in Lemma 3.6, it follows that the sink node fails to decode a given chunk w.p. b.a.b. ϵ/q, so long as N_T is larger than

$$\frac{1}{p}\left(k+(1+o(1))\left(\frac{kL}{w}+\sqrt{k\left(wq\log\frac{wqL}{\epsilon}\right)}+wq\log\frac{wqL}{\epsilon}\right)\right). \tag{B.6}$$

Taking a union bound over all the chunks, it follows that the sink node fails to decode all the chunks w.p. b.a.b. ϵ, so long as N_T is larger than (B.6). To ensure that the lower bound on N_T is $(1+o(1))k/p$, all the terms in (B.6), excluding the first one, need to be $o(k/p)$. This condition is met so long as q is

$$o\left(\frac{k}{L\log(kL/\epsilon)}\right).$$

□

Proof of Theorem 4.6: The proof is similar to that of Theorem 4.5, except that r needs to be replaced with φ. This implies that the third term in (B.5) disappears.

Thus, by selecting w to be

$$\sqrt{\frac{pN_TL}{q\log(pN_TL/\epsilon)}}$$

in order to maximize a new version of (B.5) (i.e., where the third term in (B.5) is excluded), subject to condition (B.4), it follows that the sink node fails to decode all the chunks w.p. b.a.b. ϵ, so long as N_T is larger than

$$\frac{1}{p}\left(k+(1+o(1))\left(\frac{kL}{w}+wq\log\frac{wqL}{\epsilon}\right)\right). \tag{B.7}$$

The rest of the proof follows that of Theorem 4.5. □

Proof of Theorem 4.7: By replacing p and ϵ with p/q and ϵ/q, respectively, in the proof of Theorem 4.3, it follows that the number of LILE packets pertaining to a given chunk at the sink node fails to be larger than

$$\frac{pN_T}{q}-O\left(\frac{pN_TL}{wq}\right)- O\left(\sqrt{\frac{pN_Tw}{q}\log\frac{wqL}{\epsilon}}\right) \tag{B.8}$$

w.p. b.a.b. ϵ/q, so long as

$$wq\log\frac{w_Tq}{\epsilon}=o\left(\min\left\{\frac{\gamma_e}{p},1\right\}\cdot pN_T\right). \tag{B.9}$$

The rest of the proof is similar to that of Theorem 4.5, except that (B.8) excludes the last term in (B.5), and the choice of w needs to satisfy condition (B.9), instead of condition (B.4). By selecting w to be

$$\sqrt[3]{\frac{pN_TL^2}{q\log(pN_TL/\epsilon)}}$$

in order to maximize (B.8) subject to condition (B.9), it follows that the sink node

fails to decode all the chunks w.p. b.a.b. ϵ, so long as N_T is larger than

$$\frac{1}{p}\left(k+(1+o(1))\left(\frac{kL}{w}+\sqrt{k\left(wq\log\frac{wqL}{\epsilon}\right)}\right)\right). \tag{B.10}$$

In (B.10), each term, except the largest one, needs to be $o(k/p)$, and this condition is met so long as q is

$$o\left(\min\left\{\frac{\gamma_e}{p},1\right\}\cdot\frac{k}{L\log(kL/\epsilon)}\right).$$

□

Proof of Theorem 4.8: The proof follows the same line as that of Theorem 4.5, except that the choice of w needs to maximize

$$\frac{pN_T}{q}-O\left(\frac{pN_TL}{wq}\right) \tag{B.11}$$

subject to condition (B.9). To do so, we select w to be

$$\frac{pN_T}{qf(pN_T)\log(pN_TL/\epsilon)},$$

where $f(n)$ goes to infinity, as n goes to infinity, such that $f(n) = o\left(n/(L\log(nL/\epsilon))\right)$. The sink node fails to decode all the chunks w.p. b.a.b. ϵ, so long as N_T is larger than

$$\frac{1}{p}\left(k+(1+o(1))\left(\frac{kL}{w}\right)\right). \tag{B.12}$$

The second term in (B.12) needs to be $o(k/p)$, and this condition is met so long as q is

$$o\left(\min\left\{\frac{\gamma_e}{p},1\right\}\cdot\frac{k}{f(k)L\log(kL/\epsilon)}\right).$$

□

Proof of Theorem 4.11: Let us assume $p_1 > p_2 > \cdots > p_L$, without loss of generality. Let $p \doteq \min_{1 \le i \le L} p_i$, $\gamma_e \doteq \min_{1<i\le L} \gamma_{e_i}$, and $\gamma_{e_i} \doteq |p_i - p_{i-1}|$. Let $r_i \doteq (1-\gamma_i^*)\varphi_i$, where $\varphi_i = p_i N_T/wq$ and $\gamma_i^* \sim \sqrt{(1/\dot{\varphi}_i)\log(w_T/\dot{\delta})}$, and $0 < \delta < 1$ is an arbitrary constant. For every $1 \le i \le L$ and $1 \le j \le w - L + 1$, let φ_{ij} be the number of packets (pertaining to a given chunk) in the partition I_{ij} (the j^{th} partition pertaining to the i^{th} link), where the time interval $(0, N_T]$ is split into w partitions of length N_T/w, and let φ_i be the expected value of φ_{ij} over all the partitions pertaining to the i^{th} link. For all i, j, suppose that φ_{ij} is larger than or equal to r_i. Let $N_T = (1+\gamma_c)k/p$, where $0 < \gamma_c < 1$ is an arbitrarily small constant. By replacing N_T with $(1+\gamma_c)k/p$, it follows that $\varphi_i = (1+\gamma_c)p_i\alpha/pw$, and $\varphi = O(1)$, similar to those in the proof of Theorem 4.9.

For every $1 \le i \le L$ and $1 \le j \le w-L+1$, let $\mathcal{D}(Q_i^j)$, $\mathcal{D}_p(Q_i^j)$, and r_{ij} be defined as in the proof of Theorem 4.5. For every $1 \le j \le w - L + 1$, $\mathcal{D}(Q_1^j) \ge r_1 j$ (since all the packets pertaining to any chunk over the first link are globally dense). For every $1 < i \le L$ and $1 \le j \le w - L + 1$, by applying Lemma 4.2, it can be shown that the inequality $\mathcal{D}(Q_i^j) \ge r_i j$ fails w.p. b.a.b. $ij\dot{\delta}/w_T$, so long as

$$\alpha = \Omega\left(\frac{w}{\gamma_e^2}\log\frac{w_T}{\delta}\right). \tag{B.13}$$

Let φ, γ^* and r denote φ_L, γ_L^* and r_L, respectively. Thus, the number of LILE packets pertaining to a given chunk at the sink node fails to be larger than

$$(1+\gamma_c)\alpha - O\left(\frac{\alpha L}{w}\right) - O\left(\sqrt{\alpha w \log\frac{w_T}{\delta}}\right). \tag{B.14}$$

We select w to be

$$\sqrt[3]{\frac{\alpha L^2}{\log(w_T/\delta)}}$$

to maximize (B.14) subject to condition (B.13). For this choice of w, condition (B.13) is met so long as

$$\alpha = \Omega\left(\frac{L}{\gamma_e^3}\log\frac{L}{\gamma_e\delta}\right). \tag{B.15}$$

By replacing δ with $\dot{\delta}$ in the preceding results, and substituting the selected value of w in (B.14), the result of Lemma 3.6 shows that the sink node fails to decode a given chunk w.p. b.a.b. δ, so long as (B.14) is larger than $\alpha + \log(1/\dot{\delta})$. Based on the properties of the notation $\Omega(.)$, the latter condition is met so long as

$$\alpha = \Omega\left(\frac{L}{\gamma_c^3}\log\frac{L}{\gamma_c\delta}\right). \tag{B.16}$$

The rest of the proof is similar to the proof of Theorem 4.9, except that in this case conditions (B.15) and (B.16) need to be met, instead of conditions (4.14) and (4.15).□

Proof of Theorem 4.12: The proof follows the same line as that of Theorem 4.11, except that the choice of w needs to maximize

$$(1+\gamma_c)\alpha - O\left(\frac{\alpha L}{w}\right) \tag{B.17}$$

subject to condition (B.13). To do so, the choice of w needs to be $\Omega(L/\gamma_c)$, and hence, condition (B.13) becomes

$$\alpha = \Omega\left(\frac{L}{\gamma_e^2\gamma_c}\log\frac{L}{\gamma_c\delta}\right).$$

□

Appendix C

Derivation of Some Equations in Chapter 3

C.1 Details of Equation (3.10)

Inequality (3.9) can be rewritten as

$$\begin{aligned}\mu \geq \alpha + O\left(L\mu^{\frac{2}{3}}\log^{\frac{1}{3}}\frac{L\mu}{\delta}\right)\\ +O\left(L\log\frac{L\varphi}{\delta}\right)\\ +O\left(\log\frac{1}{\delta}\right).\end{aligned}$$

By replacing μ with $(1+\gamma_c)\alpha$, the latter inequality reduces to

$$\begin{aligned}\alpha = \Omega\left(\frac{L}{\gamma_c}\alpha^{\frac{2}{3}}\log^{\frac{1}{3}}\frac{\alpha L}{\delta}\right)\\ +\Omega\left(\frac{L}{\gamma_c}\log\frac{\alpha L}{\delta}\right)\\ +\Omega\left(\frac{1}{\gamma_c}\log\frac{1}{\delta}\right).\end{aligned}$$

The first term in this relation dominates the other two, and hence this relation can be rewritten as

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\log\frac{L\alpha}{\delta}\right). \tag{C.1}$$

Further, condition (C.1) is met so long as

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\log\frac{L}{\gamma_c\delta}\right).$$

C.2 Details of Equation (3.17)

Inequality (3.15) can be rewritten as

$$\begin{aligned}\mu \geq (r_o/\chi)\alpha &+ O\left(L\mu^{\frac{2}{3}}\log^{\frac{1}{3}}\frac{L\chi\mu}{\delta}\right)\\ &+O\left(L\log\frac{L\chi\mu}{\delta}\right)\\ &+O\left(\frac{1}{\chi}\log\frac{1}{\delta}\right).\end{aligned}$$

By replacing μ with $(1+\gamma_c)\alpha/\tau_o$, the latter inequality can be written as

$$\begin{aligned}\alpha = \Omega&\left(L\left(\frac{\chi}{\rho}\right)\cdot\left(\frac{\alpha}{\tau_o}\right)^{\frac{2}{3}}\log^{\frac{1}{3}}\frac{L\chi\alpha}{\tau_o\delta}\right)\\ &+\Omega\left(L\left(\frac{\chi}{\rho}\right)\log\frac{L\chi\alpha}{\tau_o\delta}\right)\\ &+\Omega\left(\frac{1}{\rho}\log\frac{1}{\delta}\right),\end{aligned}$$

where $\rho = (\chi\gamma_c + 1)/\tau_o - 1$. The first term in this relation dominates the other two, and thus it can be rewritten as

$$\alpha = \Omega\left(\frac{L^3\chi^3}{\rho^3\tau_o^2}\log\frac{L\chi\alpha}{\tau_o\delta}\right). \tag{C.2}$$

For $\chi \gg (\tau_o - 1)/\gamma_c$, it follows that $\rho \sim \chi\gamma_c/\tau_o$, and (C.2) becomes equivalent to

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\tau_o\log\frac{L\chi\alpha}{\tau_o\delta}\right).^{1} \tag{C.3}$$

Condition (C.3) is satisfied so long as

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\tau_o\log\alpha\right),$$

and

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\tau_o\log\frac{L\chi}{\tau_o\delta}\right).$$

The first condition is met so long as

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\tau_o\log\left(\frac{L}{\gamma_c}\tau_o\right)\right). \tag{C.4}$$

By selecting χ to be a constant integer sufficiently larger than $(\tau_o - 1)/\gamma_c$, the second condition can be rewritten as

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\tau_o\log\left(\frac{L}{\gamma_c\delta}\tau_e\right)\right). \tag{C.5}$$

[1]For $\chi \gtrsim (\tau_o - 1)/\gamma_c$, it follows that $\rho \sim 0$, and (C.2) becomes equivalent to

$$\alpha = \Omega\left(\frac{L^3\tau_e^3}{\gamma_c^3\tau_o\rho^3}\log\frac{L\tau_e\alpha}{\gamma_c\delta}\right).$$

Yet, for such a choice of χ, it follows that $\rho \sim 0$, and thus the lower bound on α is much larger (and not desirable) compared to that in (C.3).

Thus, both conditions (C.4) and (C.5) are met so long as

$$\alpha = \Omega\left(\frac{L^3}{\gamma_c^3}\tau_o \log\left(\frac{L}{\gamma_c \delta}\tau_o\right)\right).$$

Appendix D

Some Mathematical Notions and Tools

D.1 Martingales

In this section, we provide a brief overview of martingales (see, [57, Chapter 7]), which are useful in proving some concentration results (e.g., in the proof of Theorems 3.16 and 3.17).

Let $\Lambda : \mathcal{A}^{\mathcal{B}} \to \mathbb{R}$ be a functional in a probability space $\Omega = \mathcal{A}^{\mathcal{B}}$, where Ω denotes the set of functions $f : \mathcal{B} \to \mathcal{A}$. We define a measure by setting the values of $\Pr\{f(\boldsymbol{b}) = \boldsymbol{a}\}$, where the values $f(\boldsymbol{b})$ are assumed to be mutually independent. We also fix a gradation $\{\mathcal{B}_i\}_{0 \le i \le m}$, i.e.,

$$\emptyset = \mathcal{B}_0 \subset \mathcal{B}_1 \subset \cdots \mathcal{B}_m = \mathcal{B}.$$

We define a martingale $X_0, X_1, ..., X_m$ by setting

$$X_i(h) = \mathbf{E}[\Lambda(f) | f(\boldsymbol{b}) = h(\boldsymbol{b}) \text{ for all } \boldsymbol{b} \in \mathcal{B}_i],$$

where h is a function in Ω. X_0 is a constant, the expected value of Λ of the random f. X_m is Λ itself. The proof that the sequence $X_0, X_1, ..., X_m$ constructed as above is a martingale follows: In this case, $\mathcal{B}_i$ (for $0 \le i < m$) is defined as $\mathbf{h}_i = \{h_j\}_{j=1}^{i}$. We shall show $\mathbf{E}[X_{i+1} | \mathbf{X}_i] = X_i$, where $\mathbf{X}_i = \{X_j\}_{j=0}^{i}$. Using the independence of

h_j's, we can write

$$\begin{aligned}
\mathbf{E}[X_{i+1}|\mathbf{X}_i] &\overset{(a)}{=} \mathbf{E}\left[\mathbf{E}[\Lambda(h)|\mathcal{B}_{i+1}]|\mathbf{X}_i\right] \\
&\overset{(b)}{=} \mathbf{E}\left[\mathbf{E}[\Lambda(h)|\mathcal{B}_{i+1}]|\mathcal{B}_i\right] \\
&\overset{(c)}{=} \sum_{\mathcal{B}_{i+1}\setminus\mathcal{B}_i} \Pr\{\mathcal{B}_{i+1}|\mathcal{B}_i\}\mathbf{E}[\Lambda(h)|\mathcal{B}_{i+1}] \\
&\overset{(d)}{=} \sum_{\mathcal{B}_{i+1}\setminus\mathcal{B}_i} \Pr\{\mathcal{B}_{i+1}|\mathcal{B}_i\} \sum_{\mathcal{B}_n\setminus\mathcal{B}_{i+1}} \Pr\{\mathcal{B}_n|\mathcal{B}_{i+1}\}\Lambda(h) \\
&\overset{(e)}{=} \sum_{\mathcal{B}_n\setminus\mathcal{B}_i} \Pr\{\mathcal{B}_n|\mathcal{B}_i\}\Lambda(h) \\
&= \mathbf{E}[\Lambda(h)|\mathcal{B}_i] \\
&= X_i,
\end{aligned}$$

where (a) since $X_{i+1} = \mathbf{E}[\Lambda(h)|\mathcal{B}_{i+1}]$, (b) $\mathbf{X}_i$ is constructed based on $\mathcal{B}_i$, (c) since $\Pr\left\{\mathbf{E}[\Lambda(h)|\mathcal{B}_{i+1}]|\mathcal{B}_i\right\} = \Pr\{\mathcal{B}_{i+1}|\mathcal{B}_i\}$, (d) by expanding $\mathbf{E}[\Lambda(h)|\mathcal{B}_{i+1}]$, and (e) by using Bayes' rule.

We say the functional Λ satisfies Θ-*Lipschitz* (or *Lipschitz*) condition relative to the gradation so long as for some $0 \leq i < m$, if h and h' differ only on $\mathcal{B}_{i+1} \setminus \mathcal{B}_i$, then $|\Lambda(h') - \Lambda(h)| \leq \Theta$ (or $|\Lambda(h') - \Lambda(h)| \leq 1$). For a functional Λ with Θ-Lipschitz property, and for an arbitrary constant $0 < \gamma_a < 1$, Azuma's inequality gives

$$\Pr\left\{\Lambda(h) \geq (1+\gamma_a)\mathbf{E}[\Lambda(h)]\right\} \leq e^{-\frac{\gamma_a^2 \mathbf{E}^2[\Lambda(h)]}{2m\Theta}}.$$

D.2 Mostly Independent Events

In this section, we explain a well-known result in probability theory, called Janson's inequality, which we use in the proof of Theorem A.2.

Consider the problem of lower bounding the probability that a set of events B_i's do not occur. For mutually independent events, $\Pr\{\cap_{i\in I}\overline{B_i}\} = \prod_{i\in I}\Pr\left\{\overline{B_i}\right\}$. However, in the case that B_i's are not independent, but "mostly" independent (each is only dependent on a small subset of the others), Janson's inequality, formulated

below, provides a tight bound on $\Pr\{\cap_{i\in I}\overline{B_i}\}$ (see [57, Chapter 8]).

Let Ω be a finite universal set (a set which contains all elements of interest, including itself) and let $\mathcal{A}$ be a random subset of Ω such that, for every element $r \in \Omega$, $\Pr\{r \in \mathcal{A}\}$ is well-defined, and the events $\{r \in \mathcal{A}\}_{r\in\Omega}$ are mutually independent. Let $\{A_i\}_{i\in I}$ be subsets of Ω, and I be a finite index set. We define the event $A_i \subseteq \mathcal{A}$ as the event B_i. For $i, j \in I$, we write $i \sim j$ if $i \neq j$ and $A_i \cap A_j \neq \emptyset$; otherwise, $i \not\sim j$. For $i \neq j$, and $i \not\sim j$, B_i and B_j are independent events. Further, for $i \notin J \subset I$, and $i \not\sim j$, for every $j \in J$, B_i is mutually independent of $\{B_j\}_{j\in J}$, and is thus independent of any Boolean function of those B_j's.

Let $\Phi = \sum_{i\sim j} \Pr\{B_i \wedge B_j\}$, where the sum is over pairs (i, j), with (i, j) and (j, i) are counted as one pair, and $P = \prod_{i\in I} \Pr\{\overline{B_i}\}$. The following is known as the Janson's inequality.

Lemma D.1 *[57, Chapter 8] Let $\{B_i\}_{i\in I}, \Phi, P$ be defined as above, and for every $i \in I$, assume that $\Pr\{B_i\} \leq \gamma$. Then,*

$$P \leq \Pr\{\cap_{i\in I}\overline{B_i}\} \leq Pe^{\frac{\Phi}{1-\gamma}}.$$

D.3 Proper Covers and Proper Fractional Covers

In this section, we state a theorem based on the notion of (proper) covers and fractional covers (of a sequence of random variables) which is useful in proving concentration results for the sum of dependent indicator random variables (see [60]). In particular, we use this theorem in the proof of Theorem A.2.

For a given set I, and a sequence of random variables $\{X_i\}_{i\in I}$, a subset J of I is called *independent* if $\{X_i\}_{i\in J}$ are independent. A sequence $\{I_j\}$ of subsets of I is a *cover* of I, if $\cup_j I_j = I$. A sequence $\{(I_j, \omega_j)\}$ of pairs (I_j, ω_j), where $I_j \subseteq I$ and $0 \leq \omega_j \leq 1$, is a *fractional cover* of I, if $\sum_{j:i\in I_j} \omega_j \geq 1$, for every $i \in I$. A cover or a fractional cover is *proper* if each set I_j in it is independent. Let M be the number of subsets I_j's of the smallest proper cover of I, and $M_{\min}$ be the smallest $\sum_j \omega_j$ over all proper fractional covers $\{(I_j, \omega_j)\}$.

Lemma D.2 *[60, Corollary 2.2] Suppose that* $X = \sum_{i \in I} X_i$, *and for every* $i \in I$, X_i *is a Bernoulli random variable. For an arbitrary constant* $0 < \gamma_a < 1$,

$$\Pr\{X \geq (1 + \gamma_a)\mathbf{E}[X]\} \leq e^{-\frac{2\gamma_a^2 \mathbf{E}^2[X]}{M_{\min}|I|}}.$$