# Network Coding for Robust Wireless Networks

by

MinJi Kim

B.Sc., Massachusetts Institute of Technology, 2006
B.Sc., Massachusetts Institute of Technology, 2006
M.Eng., Massachusetts Institute of Technology, 2007

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2012



Author . . . . . .
Department of Electrical Engineering and Computer Science
January 11, 2012

Certified by . . . . . .
Muriel Médard
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . . . .
Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Students

# Network Coding for Robust Wireless Networks

by

MinJi Kim

Submitted to the
Department of Electrical Engineering and Computer Science
on January 11, 2012, in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

## Abstract

Wireless networks and communications promise to allow improved access to services and information, ubiquitous connectivity, and mobility. However, current wireless networks are not well-equipped to meet the high bandwidth and strict delay requirements of future applications. Wireless networks suffer from frequent losses and low throughput.

We aim to provide designs for robust wireless networks. This dissertation presents protocols and algorithms that significantly improve wireless network performance and effectively overcome interference, erasures, and attacks. The key idea behind this dissertation is in understanding that wireless networks are fundamentally different from wired networks, and recognizing that directly applying techniques from wired networks to wireless networks limits performance. The key ingredient underlying our algorithms and protocols is *network coding*. By recognizing the algebraic nature of information, network coding breaks the convention of routing networks, and allows mixing of information in the intermediate nodes and routers. This mixing has been shown to have numerous performance benefits, *e.g.* increase in throughput and robustness against losses and failures.

We present three protocols and algorithms, each using network coding to harness a different characteristic of the wireless medium. We address the problem of interference, erasures, and attacks in wireless networks with the following network coded designs.

- *Algebraic NC* exploits strategic interference to provide a distributed, randomized code construction for multi-user wireless networks. Network coding framework simplifies the multi-user wireless network model, and allows us to describe the multi-user wireless networks in an algebraic framework. This algebraic framework provides a randomized, distributed code construction, which we show achieves capacity for multicast connections as well as a certain set of non-multicast connections.

- *TCP/NC* efficiently and reliably delivers data over unreliable lossy wireless networks. TCP, which was designed for reliable transmission over wired networks, often experiences severe performance degradation in wireless networks. TCP/NC combines network coding's erasure correction capabilities with TCP's congestion control mechanism and reliability. We show that TCP/NC achieves significantly higher throughput than TCP in lossy networks; therefore, TCP/NC is well suited for reliable communication in lossy wireless networks.

- *Algebraic Watchdog* takes advantage of the broadcast nature of wireless networks to provide a secure global self-checking network. Algebraic Watchdog allows nodes to detect malicious behaviors probabilistically, and police their neighbors locally using overheard messages. Unlike traditional detection protocols which are receiver-based, this protocol gives the senders an active role in checking the nodes downstream. We provide a trellis-based inference algorithm and protocol for detection, and analyze its performance.

The main contribution of this dissertation is in providing algorithms and designs for robust wireless networks using network coding. We present how network coding can be applied to overcome the challenges of operating in wireless networks. We present both analytical and simulation results to support that network coded designs, if designed with care, can bring forth significant gains, not only in terms of throughput but also in terms of reliability, security, and robustness.

Thesis Supervisor: Muriel Médard
Title: Professor of Electrical Engineering and Computer Science

*To umma and appa*

# Acknowledgments

It has been almost ten years since I arrived at MIT. Starting with my undergraduate, I have spent most of my last ten years on MIT campus. During those years, I have learned a great deal but more importantly had a tremendous amount of fun. I am thankful for all that MIT has offered me.

I could not have had so much fun without the help of many people. I met some of the most talented and amazing people during my career at MIT, and they have been invaluable to me. They have guided me through my ups-and-downs, both personally and professionally. My achievements are credit to their continued support and advice.

I would like to thank first and foremost my supervisor, Muriel Médard. She has been a fantastic advisor, who has guided me throughout my graduate program. I started working with her in 2006 while I was debating between a career in software engineering and continuing to a Ph.D. program. Her constant encouragement contributed significantly to my decision to continue with my graduate education. Muriel's insight and advice have kept me focused, often stopping me from going astray. Her enthusiasm motivated me to keep moving forward, and her patience encouraged me to continue even during the most difficult times. She has been a great source of inspiration.

Likewise, I am indebted to my long-time collaborator, João Barros. He has been a wonderful mentor. He has provided a great amount of guidance in most of my work, and without him, I would not have accomplished many of what I have during my graduate program. I am very grateful for his support and care.

I have had the great fortune to work with (late) Ralf Koetter, whose thoughtful advice and pointers have guided me through the completion of my graduate program. I would also

like to thank Dina Katabi for her valuable feedback and comments as my thesis reader. I thank my co-authors, Elona Erez, Atilla Eryilmaz, Bernhard Haeupler, Luísa Lima, Daniel Lucani, Shirley Shi, Jay Kumar Sundararajan, Edmund M. Yeh, and Fang Zhao. Working with them has been a pleasure. They have each given me invaluable lessons, and I appreciate the time and effort they have invested in me.

My life at MIT would not have been as fruitful and fun without my labmates and fellow graduate students. I thank Georgios Angelopoulo and Shirley Shi for tolerating me as their officemate and making my office a fun place to work in. I would like to thank Jay Kumar Sundararajan who generously helped me as I was starting my M.Eng program, teaching me the basics of network coding as well as what it is to do research. I also enjoyed the company of Soheil Feizi, Daniel Lucani, Ali ParandehGheibi, Arman Rezaee, Guy Weichenberg, and Fang Zhao.

I was able to enjoy my work as much as I did because of my wonderful friends and family outside of work. I have known Nathan K. Davis, Shaun J. Foley, and Marjorie Cheng since my undergraduate years. Nathan and Shaun have also been great roommates during the last few years, allowing me to come home from long days in lab, vent and relax, but most importantly, just be myself regardless of how quirky I am. Marjorie has always been available for me. Although not in Boston, she kept me company virtually so much so that I felt she never left Boston.

Finally but not least, I thank my mother Kyungsook Jeong, my father Ildoo Kim, my sister Hyojeong Kim, and Colin R. Dillard for their love and patience. No words can describe the gratitude I have for them. Without their encouragement, care, and love, I could not have made it this far. This dissertation is dedicated to my parents.

MinJi Kim

Cambridge, MA

January 2012

# Research Sponsors

This research was supported by the following grants:

- Subcontract number S0176938 issued by University of California Santa Cruz and supported by the United States Army under award number W911NF-05-1-0246,
- Subcontract number 069145 issued by BAE Systems National Security Solutions, Inc. and supported by the Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare System Center, San Diego under contract number N66001-06-C-2020,
- Grant number CNS-0627021 supported by the National Science Foundation (NSF),
- Award number N00014-05-1-0197 supported by the Office of Naval Research, (ONR),
- Award number 018955-001 supported by NBC Universal, Inc.,
- Award number 016974-002 supported by the Air Force Office of Scientific Research (AFOSR).

# Previous Publications

This dissertation is based on previous publications as listed below.

Chapter 3 is based upon the following previous publications.

- [57] M. Kim and M. Médard, Algebraic Network Coding Approach to Deterministic Wireless Relay Networks, In *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*, pages 1518–1525, September 2010.

- [54] M. Kim, E. Erez, E. M. Yeh, and M. Médard, Deterministic Network Model Revisited: An Algebraic Network Coding Approach, Submitted to *IEEE Transactions on Information Theory*, March 2011.

Chapter 4 is based upon the following previous publications.

- [61] M. Kim, M. Médard, and J. Barros, Modeling Network Coded TCP Throughput: A Simple Model and its Validation, In *Proceedings of International ICST/ACM Conference on Performance Evaluation Methodologies and Tools (Valuetools)*, May 2011.

Chapter 5 is based upon the following previous publications.

- [62] M. Kim, M. Médard, J. Barros, and R. Koetter, An Algebraic Watchdog for Wireless Network Coding, In *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pages 1159–1163, June 2009.

- [59] M. Kim, M. Médard, and J. Barros, A Multi-hop Multi-source Algebraic Watchdog, In *Proceedings of IEEE Information Theory Workshop (ITW) Dublin* (invited paper), pages 1–5, August 2010.

- [60] M. Kim, M. Médard, and J. Barros, Algebraic Watchdog: Mitigating Misbehavior in Wireless Network Coding, *IEEE Journal on Selected Areas in Communications (JSAC) Advances in Military Networking and Communications*, 29(10):1–11, December 2011.

There are several other previous publications, which are not directly incorporated into this dissertation. The following publications may be of interest to the readers as supplementary readings.

- [58] M. Kim, M. Médard, and J. Barros, Counteracting Byzantine Adversaries with Network Coding: An Overhead Analysis, In *Proceedings of IEEE Conference for Military Communications (MILCOM)*, pages 1–7, November 2008.

- [56] M. Kim, D. Lucani, X. Shi, F. Zhao, and M. Médard, Network Coding for Multi-Resolution Multicast, In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 1–9, March 2010.

- [55] M. Kim, L. Lima, F. Zhao, J. Barros, M. Médard, R. Koetter, T. Kalker, and K. Han, On Counteracting Byzantine Attacks in Network Coded Peer-to-Peer Networks, *IEEE Journal on Selected Areas in Communications (JSAC) Mission Critical Networking*, 28(5):692–702, June 2010.

- [39] B. Haeupler, M. Kim, and M. Médard, Optimality of Network Coding in Packet Networks, *IEEE Information Theory Workshop (ITW) Paraty*, pages 1–5, October 2011.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Wireless networks and communications have emerged as a dominant mode of communications, as they facilitate mobility and ubiquitous connectivity. A surge of technical advances in wireless communications has brought faster and more affordable network access. However, the technological advances are often matched, if not outmatched, by users' appetite for even faster, cheaper, and more robust wireless networks. For instance, with the rise of new forms of data, in particular high quality multimedia, we are faced with a challenge of transforming our communication networks to handle unparalleled growth in traffic and strict delay constraints. In order to meet the future demands, we need to manage the existing wireless networks more efficiently in terms of, but not exclusively, energy, latency, and bandwidth; and to build new infrastructure and design novel protocols that take into account the high-bandwidth, dynamic, and diverse traffic that needs to be served across wired and wireless medium.

There are several sources of challenges in designing a robust wireless networks, which are not as prominent in wired networks.

- **Wireless is a shared medium.** Unlike in wired networks, senders and receivers are limited to a certain number of channels, which they ultimately have to share either in time, frequency, or space. When senders and receivers fail to share the wireless medium appropriately, we observe a *collision* or an *interference*, which hinders efficient communication.

- **Wireless is stochastic in nature.** The time varying nature of wireless results in not only ever-changing capacity and delay but also *errors* and *erasures*. Various physical phenomena, such as fading and interference, often cause packets to be lost or delivered erroneously. Consequently, reliability is much more difficult to attain in wireless networks than in wired networks.

- **Wireless is a broadcast medium.** Wireless networks are often insecure, prone to *adversarial eavesdropping* and *contamination attacks*. There are tools and software, easily accessible, that can allow an unauthorized user to access nearby wireless networks, overhear other users' messages, and even jam and intercept signals from users.

These properties of wireless make designing and operating wireless networks vastly different from those of wired networks. However, our currently deployed wireless networks are built using architecture rooted in wired networks, which does not adequately address these problems.

Often measures to address the problems of wireless networks are appended to the existing architecture. For instance, the current design to combat erasures and errors is retransmissions. In wired networks, in which erasures and errors are rare, retransmissions are an efficient way to correct erasures and errors; however, this is not the case in wireless networks. Furthermore, interference and collisions in current systems are deemed unacceptable, and wireless networks are designed to avoid interference and collisions. An unfortunate consequence of such design decision is that, when there is interference, the system is unable to cope with such scenario efficiently.

Motivated by these observations, we propose novel algorithms and protocols to build more efficient, high performance wireless networks. In this dissertation, we present algorithms that can better manage interference, overcome losses, and provide secure communications by recognizing and harnessing the characteristics of wireless.

The key idea behind this dissertation is in understanding that wireless networks are fundamentally different from wired networks, and recognizing that directly applying techniques from wired networks to wireless networks limits throughput and performance. The

key ingredient underlying our algorithms and protocols is *network coding*.

Network coding promises a fundamentally new way to operate networks. At present, networks are based on *routing* solutions. In a *router network*, each data packet that enters a node can only be relayed onto some outgoing link(s). Sources and destinations may generate and modify the information; however, the intermediate nodes can only store, forward, or replicate the information they receive. This "store-and-forward" approach is closely related to the multi-commodity flow problem, and has been studied extensively owing to its wide applications to communication networks. This approach views data packets as objects that cannot be altered, *e.g.* commodities such as apples or books. However, this view that data are non-malleable commodities can result in inefficient operations of the networks.

In an essence, network coding questions the fundamental assumption in our "store-and-forward" network designs. The theory of *network coding*, first introduced in their seminal paper by Ahlswede *et al.* [3], breaks the convention of router networks. Intermediate nodes, *e.g.* routers, are allowed to algebraically combine flows, packets, and symbols. By recognizing the malleable algebraic nature of data, network coding can bring performance gains. This simple yet fundamental shift in network paradigm can yield significant throughput improvements compared to the state of the art routing protocols.

We consider the problems of robust communication in wireless networks using network coding. This dissertation shows that network coding is a powerful tool that can efficiently and effectively overcome interference, erasures, and adversarial attacks. We provide support for network coding as a new paradigm to operate networks and show that network coding can deliver on the promise of a more efficient wireless network with higher throughput and reliability.

We introduce three new designs: *Algebraic NC*, *TCP/NC*, and *Algebraic Watchdog*. Algebraic NC, founded on the algebraic network coding framework introduced in [67], is a novel code construction which exploits strategic interference and symbol-level network coding to enhance throughput in wireless networks and, more importantly, achieve capacity in wireless multi-user networks. TCP/NC uses a packet-level network coding to overcome erasures in wireless networks, allowing for efficient and reliable transport of data. TCP/NC focuses on providing reliability and congestion control in intrinsically unreliable networks. Alge-

braic Watchdog harnesses interference and broadcast nature of wireless to provide secure, self-checking network. Network nodes monitor their neighborhood locally in a distributed manner to ensure that adversaries do not contaminate the information flow. At first glance, network coding, which performs algebraic transformation to the information, may seem to hinder the nodes from checking their neighbors. However, with a studied understanding, this algebraic transformation can be harnessed to provide secure communication.

We begin, in Sections 1.1, 1.2, and 1.3, by briefly describing the three network coding designs presented in this dissertation. Then, in Section 1.4, we outline the body of the dissertation.

## 1.1 Algebraic NC: Network Coding for Interference Networks

In Chapter 3, we present Algebraic NC, a simple code construction using symbol-level network coding and strategic interference to achieve capacity in multi-user wireless networks. Finding the capacity as well as the code construction for the multi-user wireless networks, unlike its wired counterparts, are generally open problems. Even the relatively simple relay network with one source, one sink, and one relay, has not been fully characterized. There are two sources of disturbances in multi-user wireless networks – channel noise and interference among users in the network. In order to better approximate the Gaussian multi-user wireless networks, Avestimehr *et al.* [6][7] proposed a binary linear deterministic network model (known as the ADT model), which takes into account the multi-user interference but not the channel noise.

The main role of network coding in Chapter 3 is to simplify the ADT model. In Chapter 3, we draw a connection between the ADT network and network coding, in particular algebraic network coding introduced by Koetter and Médard [67]. We show that the ADT network problems, including that of computing the min-cut and constructing a code, can be captured by the algebraic network coding framework. We prove that the ADT network problem can be described by a single matrix, and show that the min-cut of an ADT network is the rank of this matrix; thus, eliminating the need to optimize over exponential number of cuts between two nodes to compute the min-cut of an ADT network. We extend the

capacity characterization for ADT networks to a more general set of connections, including a single unicast or multicast connection and non-multicast connections such as multiple multicast, disjoint multicast, and two-level multicast. We also provide sufficiency conditions for achievability in ADT networks for any general connection set. Furthermore, we extend the ADT networks to those with random erasures and cycles; thus, allowing bi-directional links.

In addition, we show that the random linear network coding [41], a randomized distributed algorithm for network code construction, achieves capacity for the connections listed above. By using algebraic network coding framework, we show the solvability of a given ADT network problem is reduced to determining an algebraic property of a single matrix. The randomized, distributed code construction naturally falls from this reduction; it is a by-product of understanding and reducing the problem of multi-user network to that of algebraic network coding framework. Chapter 3 illustrates the elegance of network coding, allowing a simple yet powerful framework for theoretical analysis.

## 1.2 TCP/NC: Network Coding for Erasure Networks

We present TCP/NC in Chapter 4 where the application of interest is reliable and efficient transport of data across lossy and faulty networks. The Transmission Control Protocol (TCP) is one of the core protocols of the Internet Protocol Suite. TCP provides reliable, ordered delivery of a stream of bytes across networks. Many Internet applications, such as the World Wide Web, email, file transfer, peer-to-peer file sharing, and multimedia streaming, rely on TCP's promise to deliver correctly the data stream without losses or duplications. Furthermore, TCP plays an essential role in end-to-end flow control and congestion control, which are important in enabling a diverse set of devices to share the network resources. TCP uses several mechanisms to detect congestion in the network, and controls its rate to avoid congestion collapse and to allow fair sharing of the network resources among many flows.

However, TCP was designed for reliable transmission over wired networks, in which the lower layers such as the MAC and the PHY layers can effectively correct for random losses

and errors. As a result, in wired networks, losses are generally indication of buffer overflow or congestion. Therefore, TCP considers losses as congestion and reacts accordingly. However, this is not the case in wireless networks, where losses are often due to fading, interference, and other physical phenomena. With the proliferation of wireless networks and devices, the assumption that MAC and PHY layers will effectively fix the problems of efficiency and losses is no longer true. This results in TCP's poor performance in wireless networks compared to the wired counterparts.

To combat these harmful effects of wireless networks to TCP, we combine TCP with network coding. In Chapter 4, we introduce network coding to complement TCP and make it more robust to erasures. Using network coding's erasure correction capability, we show that TCP/NC is a novel reliable transport protocol which is resilient to losses while maintaining TCP's functionality in end-to-end flow control and congestion control. Our results confirm that TCP/NC has significant throughput gain over TCP, where the gain increases with the packet loss rate in the network. The key challenge in inserting network coding into TCP is understanding how much coding is needed when and the effect of network coding on the other TCP functionalities.

## 1.3 Algebraic Watchdog: Network Coding for Secure Communications

In Chapter 5, we consider the problem of secure communication in wireless networks. Wireless networks and communications have great potential to improve access to services and information. Unfortunately, wireless networks are often insecure, prone to adversarial eavesdropping and contamination. These dangers are not just theoretical. There are tools and software, easily accessible, that can allow an unauthorized user to access nearby wireless networks, overhear other users' messages, and even jam and intercept signals from users.

Countering these types of threats is important in any type of networks. One key application of secure wireless communications is in military communications and networking, which are highly dynamic in nature and must not fail when adversaries succeed in compromising some of the nodes in the network. However, security often comes at a cost of

network performance. For instance, many secure protocols require a significant amount of control information to be sent and received among nodes, or need a certain infrastructure (such as a public key infrastructure and certificate authorities) to manage and control the communications network.

In Chapter 5, we leverage the performance gain provided by network coding to integrate security without incurring significant transmission overhead or requiring additional infrastructure. By taking advantage of the broadcast nature of wireless, we present a scheme that allows network nodes to monitor other nodes in the network in a distributed manner.

Algebraic Watchdog, the subject of Chapter 5, considers the problem of detecting malicious behavior in wireless networks. The goal of Algebraic Watchdog is to allow network nodes to monitor their neighborhood locally, and together the nodes can provide a secure global self-checking network. Network coding plays two significant roles in Algebraic Watchdog. First, network coding bolsters higher throughput in the wireless network; second, network coding protects the network and prevents adversaries from injecting invalid information without being detected. The latter benefit is obtained by understanding the algebraic nature of network coding. Network coding performs algebraic transformation to the information, which at first glance may seem to hinder the nodes from checking their neighbors. However, with a studied understanding, this algebraic transformation can be harnessed to provide secure communication.

## 1.4 Outline

The dissertation is organized as follows. Chapter 2 provides a brief overview of network coding and its evolution. The dissertation focuses on designing a coded system that overcomes the challenges posed by wireless networks, in particular interference, erasures, and attacks. These solutions are contained in Chapters 3, 4, and 5. Each application is entirely contained within each chapter; therefore, depending on what the reader is looking for, each chapter can be read independently.

The main body of the dissertation is organized in the order of "difficulty of management". We start with Algebraic NC in Chapter 3, which studies the effect of interference

among users in wireless networks. We investigate how network coding can manage and exploit strategic interference to better understand multi-user wireless networks. We then consider the problem of erasures in Chapter 4. Interference is generated by participants of the networks, and therefore, can be managed and designed using appropriate scheduling algorithms and coding techniques; however, erasures are an innate property of wireless which we have to cope with. The subject of Chapter 4 is TCP/NC, which aims to provide reliable and efficient transport of data in lossy and faulty networks. Finally, in Chapter 5, we introduce Algebraic Watchdog, a sophisticated inference algorithm that takes advantage of interference to provide secure communication in wireless networks. Algebraic Watchdog detects misbehaviors or contamination within the network, and ultimately limits the adversarial attacks to that of channel noise.

Finally, in Chapter 6, we present a summary and give a final perspective on this dissertation. Chapter 6 also provides a few pointers for potential extensions to the body of work presented in this dissertation.

# Chapter 2

# Overview of Network Coding

NETWORK coding [3] has emerged as a new paradigm to operate networks, promising a more efficient network with higher throughput and reliability. Network coding allows and encourages mixing of data at intermediate nodes, which has been shown to increase throughput and robustness against failures and erasures [67]. The growth in network coding has been explosive. This chapter presents only a few selected works in this area, and does not attempt to provide a comprehensive overview of network coding. A more detailed background of network coding related to the specific applications is presented in each chapter separately. In this chapter, we describe in a few broad brush strokes how the field of network coding has evolved.

Network coding has evolved significantly from its inception. Initially, network coding was proposed for *wired multicast networks*, in which one or more sources may wish to deliver information to many receivers. References [3] showed that, if additional computing tasks are performed at the intermediate nodes, the multicast capacity can be achieved. Thus, network coding has been shown to extend the Max-flow Min-cut theorem for unicast connections to multicast connections, which was not possible with traditional routing ideas.

The butterfly network [3], shown in Figure 2-1, is an iconic example used to illustrate the advantage of network coding over routing in a multicast network. Assume that all links in Figure 2-1 have capacity one. The two source nodes $S_1$ and $S_2$ wish to send information $A$ and $B$, respectively, to both destinations $D_1$ and $D_2$. In a routing network, the middle link between the two intermediate nodes $V_1$ and $V_2$ becomes a bottleneck; the middle link needs

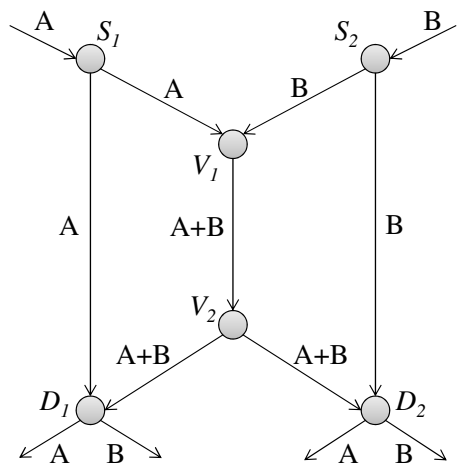



Figure 2-1: The butterfly network. The example illustrates how network coding can be used to achieve multicast capacity.

to be time shared between the two sources. On the other hand, if the intermediate node $V_1$ is allowed to perform an XOR of $A$ and $B$ and send the result to $V_2$, both destinations can recover both $A$ and $B$, as shown in Figure 2-1.

In the example in Figure 2-1, a simple XOR operation was sufficient to achieve capacity. It was later shown by Li *et al.* [71] that linear operations are sufficient to achieve the multicast capacity. Subsequently, Koetter and Médard [67] showed an algebraic framework for linear network coding, and presented an equivalence between a network coding problem and a certain algebraic condition. As a result, there has been a significant interest in linear network coding due to its potential as practical, simple code with near-optimum performance.

Efficient linear network codes were quickly introduced. References [44][46][43][41] introduced low complexity linear network codes for multicast. In particular, Ho *et al.* [41] proposed a randomized, distributed code construction algorithm, called *random linear network coding*, which achieves multicast capacity with probability exponentially approaching one with the field size. The network code introduced in [41] is particularly attractive for its distributed nature.

Soon after the introduction of efficient linear network codes, network coding evolved

beyond wired multicast networks. Lun *et al.* [77] extended network coding from reliable networks to unreliable networks with packet losses. A typical example of such unreliable networks is wireless networks in which losses and failures are common. Therefore, reference [77] showed that the benefit of network coding is not limited to multicast networks; a single point-to-point connection may benefit from network coding in the presence of losses and failures.

Despite its desirable properties, network coding is not an "one size fits all" solution to problems in networking. Like many ideas, network coding is a tool among many others to be chosen for appropriate circumstances; like many protocols and systems, network coded systems have to be designed and implemented with the application of interest in mind. It is crucial to understand the constraints of the network, the quality of service (QoS) required by the application, and the security and privacy concerns of the users to design a good network code.

An example of such engineering considerations is in weighing the computational cost of coding and decoding against the throughput gain provided by network coding [11][70]. This trade-off between throughput and complexity is at the core of designing a good network code. If there are abundant network resources for all users, then there is no need to code. A simple routing solution may suffice. However, this is often not the case, particularly poignant in bandwidth limited wireless networks. If there is a bottleneck or a need to manage and share network resources among many users, then the coding overhead may be negligible compared to the benefits gained from using network coding.

Another example of such engineering decisions is in determining when and where to use network coding. Network coding promises improved efficiency in the form of higher throughput, especially in challenged network situations. For instance, network coding may provide better utilization of network resources in bandwidth limited networks [52][50][49], and resilience in lossy and faulty networks [77][91][66]. On the other hand, a highly provisioned optical networks, such as the Internet backbone, may not benefit from network coding as much. On the contrary, the computational cost of coding and decoding may be undesirable in such networks.

As a result, there has been significant effort made to tailor and adjust network coding for

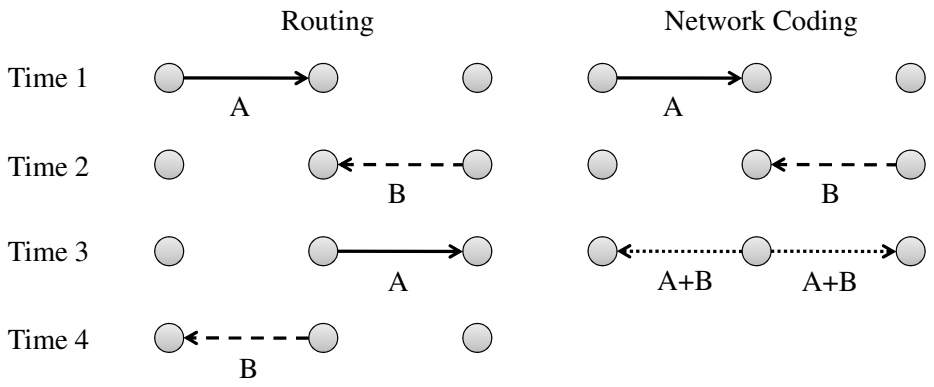



Figure 2-2: An example showing how a simple XOR code may increase throughput. The example illustrates how two nodes may exchange information in three transmissions instead of four transmissions (needed by routing solution).

various applications. As the theory of network coding developed, the use of network coding expanded to more practical and application-oriented work. Many areas of applications have been considered, of which we mention only a few here.

In keeping with the original spirit of network coding, there has been work in implementing network coding in multiple multicast scenarios, such as in multicast routers. Scheduling algorithms and buffer management protocols for multicast routers with network coding were presented in [88][90][63][64].

There are several protocols that take advantage of network coding in wireless networks [52][50][49][15][73][10]. Reference [52] may be the most intuitive and simple implementation of network coding. Figure 2-2 illustrates the basic concept behind [52], which uses simple XOR codes to achieve higher throughput in wireless networks.

For secure communication, network coding has been used in [94][45][40][96]. These works focus on algorithm and protocol designs for robustness and resilience against network attacks such as snooping, eavesdropping, and pollution. Furthermore, application of network coding in distributed storage [2][19] and peer-to-peer networks [35][34] have been of significant interest. One of the key problems in peer-to-peer networks is, as poignantly presented in [27], that of the coupon collector's problem. Network coding's ability to correct erasures and remove scarce packets is heavily exploited in these systems.

The potential benefits and applications of network coding are wide. The key observation is that there is no coding scheme which is optimal for all applications. Depending on the design parameters, the network code may differ greatly. The list of applications presented in this chapter is not an exhaustive one – there are many other applications that may benefit from the use of network coding. This dissertation aims to reinforce the theoretical and practical benefits of network coding, and to provide some guidance in designing network coded systems for robust communication over wireless networks.

# Chapter 3

# Algebraic NC:

## Network Coding for Interference Networks

FINDING the capacity as well as the code construction for the multi-user wireless networks, unlike its wired counterparts, are generally open problems. Even the relatively simple relay network with one source, one sink, and one relay, has not been fully characterized. There are two sources of disturbances in multi-user wireless networks – channel noise and interference among users in the network. In order to better approximate the Gaussian multi-user wireless networks, [6][7] proposed a binary linear deterministic network model (known as the ADT model), which takes into account the multi-user interference but not the noise. A node within the network receives a bit if the signal is above the noise level; multiple bits that simultaneously arrive at a node are superposed.

References [6][7] showed that, for a multicast connection where a single source wishes to transmit the same data to a set of destinations, the achievable rate is equal to the minimal cut between the source and any of the destinations in an ADT network. As we shall discuss in Section 3.4, the min-cut of an ADT network may not equal to the graph theoretical cut value. In addition, [6][7] showed that the minimal cut between the source and a destination is equal to the minimal rank of incidence matrices of all cuts between the two nodes. This can be viewed as the equivalent of the Min-cut Max-flow criterion in the network coding for wired networks [3][67]. It has been shown that for several networks, the gap between the capacity of the deterministic ADT model and that of the corresponding Gaussian network

is bounded by a constant number of bits, which does not depend on the specific channel fading parameters [6][12][8].

In this chapter, we make a connection between the ADT network and network coding, in particular algebraic network coding introduced by Koetter and Médard [67]. Other approaches to operations in high SNR networks have been proposed [78], however, we do not compare these different approaches but build upon the given model proposed by [6][7]. We show that the ADT network problems, including that of computing the min-cut and constructing a code, can be captured by the algebraic network coding framework.

In the context of network coding, reference [67] showed that the solvability of the communication problem [3] is equivalent to ensuring that a certain polynomial does *not* evaluate to zero, *i.e.* avoid the roots of this certain polynomial. Furthermore, [67] showed that there are only *a fixed finite number of roots* of the polynomial; thus, with large enough field size, decodability can be guaranteed even under randomized coding schemes as shown in [41]. As we increase the field size $\mathbb{F}_q$, the space of feasible network codes increases exponentially; while the number of roots remain fixed.

We show that the solvability of ADT network problem can be characterized in a similar manner. The important difference between the algebraic network coding in [67] and the ADT network is that the broadcast constraints, as well as the interference constraints, are embedded in the ADT network. The interference constraint, represented by the additive multiple access channels (MAC), can be easily incorporated into the algebraic framework in [67] by pre-encoding at the transmitting nodes (*i.e.* MAC users). This is due to the fact that the MAC is modeled using a finite field additive channel; therefore, the operations performed by the MAC can be "canceled" by the transmitter appropriately pre-encoding the packets.

On the other hand, the broadcast constraint may seem more difficult to incorporate, as the same code affects the outputs of the broadcast channel simultaneously and the dependencies propagate through the network. Therefore, in essence, this chapter shows that this broadcast constraint is not problematic.

To briefly describe the intuition, consider an ADT network without the broadcast constraint, *i.e.* the broadcast edges do not need to carry the same information. Using this

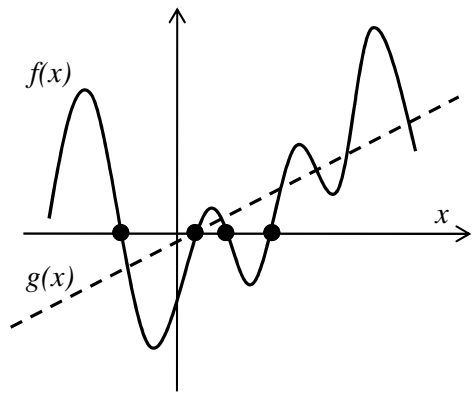



Figure 3-1: An interpretation of the broadcast constraint in ADT networks. A polynomial $f(x)$ and a hyperplane $g(x)$ with one variable $x \in \mathbb{R}$. The black dots represent the roots of $f(x)$. When considering the space where $f(x)$ intersects $g(x)$, the hyperplane $g(x)$ limits the space in which $f(x)$ operates in; however, $g(x)$ does not change the roots of $f(x)$. Some of the roots of $f(x)$ may no longer be "feasible" given the additional constraint; thus, this operation may reduce the number of roots that we have to consider.

"unconstrained" version of the ADT network, the algebraic framework in [67] can be applied directly; therefore, there are only a finite fixed number of roots that need to be avoided. Furthermore, as the field size increases, the probability of randomly selecting a root approaches zero. Now, we "re-apply" the broadcast constraints to this unconstrained ADT network. The broadcast constraint fixes the codes of the broadcast edges to be the same; this is equivalent to intersecting the space of network coding solutions with an hyperplane, which enforces the output ports of the broadcast to carry the same code. As shown in Figure 3-1, this operations does not change the polynomial whose roots we have to avoid, but changes the hyperspace we operate in. As a result, this operation does not affect the roots of the polynomial; there are still only a fixed finite number of roots that need to be avoided, and with high enough field sizes, the probability of randomly selecting a root approaches zero. Note that intersecting the space of network coding solutions with an hyperplane may even remove some roots of the polynomial from consideration; therefore, we may effectively have fewer roots to avoid.

By the same argument as in [67][41], we can then show that the solvability of an ADT network problem is equivalent to ensuring that a certain polynomial does not evaluate to zero within the space defined by the polynomial and the broadcast constraint hyperplane.

This enables us to describe the ADT network within the algebraic network coding framework and extend the random linear network coding results to the ADT networks.

Using this insight, we prove that the ADT network problem can be captured by a single matrix, called the *system matrix*. We show that the min-cut of an ADT network is the rank of the system matrix; thus, eliminating the need to optimize over exponential number of cuts between two nodes to compute the min-cut of an ADT network. We extend the capacity characterization for ADT networks to a more general set of connections, including single unicast or multicast connection and non-multicast connections such as multiple multicast, disjoint multicast, and two-level multicast. We also provide sufficiency conditions for achievability in ADT networks for any general connection set. Furthermore, we extend the results on ADT networks to those with random erasures and cycles (thus, allowing bidirectional links). We emphasize that these generalizations are possible precisely because we were able to take advantage of the algebraic structure within the ADT network.

We show that a direct consequence of this connection between ADT network problems and algebraic network coding is that random linear network coding, a randomized distributed algorithm for network code construction, achieves the capacity for the connections listed above. Note that Avestimehr *et al.*'s proposed code construction [6][7] is not guaranteed to be efficient and may potentially involve an infinite block length. Other coding schemes [5][86][37] have been proposed for the ADT networks. Many of these code construction algorithms require coordination among nodes and/or some knowledge of the global or local topology. We remove all these requirements from the algorithm by understanding that the ADT network problems can be translated to those of algebraic network coding framework, and present a randomized, distributed code construction for ADT networks.

The rest of this chapter is organized as follows. We provide a brief discussion on related work in Section 3.1. We present the network model in Section 3.2, and an algebraic formulation of the ADT network in Section 3.3. Using this algebraic formulation, we provide a definition of the min-cut in ADT networks in Section 3.4. In Sections 3.5, we restate the Min-cut Max-flow theorem using our algebraic formulation, and present new capacity characterizations for ADT networks to a more general set of traffic requirements in Section 3.6. The results in Section 3.6 show the optimality of linear operations for non-multicast

connections such as disjoint multicast and two-level multicast connections. In Section 3.7, we study ADT networks with link failures, and characterize the set of link failures such that the network solution is guaranteed to remain successful. Furthermore, in Sections 3.8, we extend the achievability results to ADT networks with delay. Finally, we conclude this chapter in Section 3.9.

## 3.1 Background

Avestimehr *et al.* introduced the ADT network model to better approximate wireless networks [6][7]. In the same work, they characterized the capacity of the ADT networks, and generalized the Min-cut Max-flow theorem for graphs to ADT networks for single unicast or multicast connection.

It has been shown that for several networks, the ADT network model approximates the capacity of the corresponding Gaussian network to within a constant number of bits. For instance, [6] considered the single relay channel and the diamond network, and showed that the gap between the capacity of the ADT model and that of Gaussian network is within 1 bit and 2 bits, respectively. Reference [12] considered many-to-one and one-to-many Gaussian interference networks. The networks in [12] are special cases of interference network with multiple users, where the interference is either experienced (many-to-one) or caused by (one-to-many) a single user. It was shown that in these cases, the gap between the capacity of the Gaussian interference channel and the corresponding deterministic interference channel is again bounded by a constant number of bits. The work in [12] provided an alternative proof to [29] on the existence of a scheme that can achieve a constant gap from the capacity for all values of channel parameters. In [8], the half-duplex butterfly network was considered. They showed that the deterministic model approximates the symmetric Gaussian butterfly network to within a constant.

As a result, there has been significant interest in finding an efficient code construction algorithm for the ADT network model. However, the achievability proof in [7] is not constructive, and involves information theoretical arguments. Therefore, the code construction is not guaranteed to be efficient and may potentially involve an infinite alphabet size or block

length. An important problem is to find an efficient code construction for the deterministic model of wireless multicast relay networks.

In the case of unicast communication, a number of previous code constructions have been proposed for the ADT networks. It is important to observe that in the code constructions for unicast communication, routing [86] or one-bit operations [5] are sufficient for achieving the capacity of the deterministic model. Reference [5] proposed an algorithm which can be viewed as an application of the Ford and Fulkerson flow construction to the deterministic model. The complexity of the algorithm was shown to be $O(|\mathcal{V}||\mathcal{E}|R^5)$, where $\mathcal{V}$ is the set of nodes in the network, $\mathcal{E}$ is the set of edges, and $R$ is the rate of the code. In [86], another algorithm for finding the flow for unicast networks was developed. The algorithm is based on an extension of the Rado-Hall transversal theorem for matroids and on Edmonds' theorem. The transmission scheme in [86] extracts at each relay node a subset of the input vectors and sets the outputs to the same values as that subset. In [37], it was shown that the deterministic model can be viewed as a special case of a more abstract flow model, called *linking network*, which is based on linking systems and matroids. Using this approach, [37] achieved a code complexity $O(\lambda N_{layer}^3 \log N_{layer})$, where $\lambda$ is the number of layers in the layered network, and $N_{layer}$ is the maximal number of nodes in a layer. Note that linear network coding is known to be matroidal [22]; thus, the fact that ADT networks are matroidal [37] is consistent with our result.

In the case for multicast communication, however, routing or one-bit operations may not be sufficient to achieve the capacity in the ADT model. This can be shown by considering the example in Figure 3-2, which is given in [30][84][32] for network coding. From the analysis for network coding, it follows that in the case of the deterministic model, the maximal rate of two can be achieved simultaneously for all sinks only with an alphabet size which is at least three. To see this, observe that to achieve a rate of two the source has to transmit at its outputs two statistically independent symbols $x_1$ and $x_2$. For node $v_i^2, 1 \leq i \leq 4$ at the second layer, the transmitted symbol is a certain function of the symbols $x_1$ and $x_2$, given by $y_i = f_i(x_1, x_2)$. Node $v_i^3$, $1 \leq i \leq 4$, at the third layer transmits at its outputs two functions of $y_i$, given by $f_i^1(y_i)$ and $f_i^2(y_i)$. Sink $t_i, 1 \leq i \leq 6$ receives at its two inputs symbols of the form $f_j^1(y_j)$ and $f_j^2(y_j) + f_k^1(y_k)$ for some $1 \leq j \leq 4, 1 \leq k \leq 4, j \neq k$.

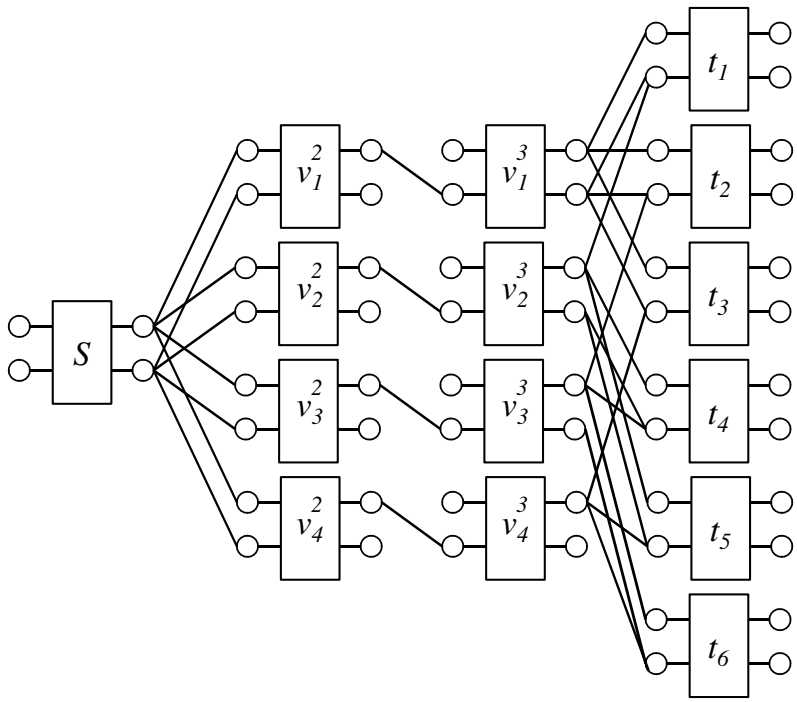



Figure 3-2: An example network for a non-binary code.

It follows that without rate loss, we can always assume $f_j^1(y_j) = y_j$ for each $1 \leq j \leq 4$. In that case, the sink $t_i$ receives $y_j$ at its upper input and can therefore find $f_j^2(y_j)$ and eliminate it from the second symbol it received. Therefore, it is equivalent to the situation in which the sink receives $y_j$ and $y_k$. This in turn is exactly the situation in [84] (Theorem 3.1) for network coding. Since the channels are all binary in the deterministic model, it follows that the minimal required alphabet size is in fact $2^2 = 4$, and therefore the minimal vector length is $\log_2(4) = 2$. Thus, for multicasting in ADT networks, we need to either operate in a higher field size, $\mathbb{F}_q$, $q \geq 2$, or use vector coding (or both).

References [23][24][28] proposed a polynomial time algorithm for multicasting in ADT networks. In particular, [24] extended the algebraic network coding result [67] to vector network coding, and showed that constructing a valid vector code is equivalent to certain algebraic conditions. This result [24] is supported by the result from [43]. Reference [43] introduced network codes, called *permute-and-add*, that only require bit-wise vector operations to take advantage of low-complexity operations in $\mathbb{F}_2$. In addition, [43] showed that

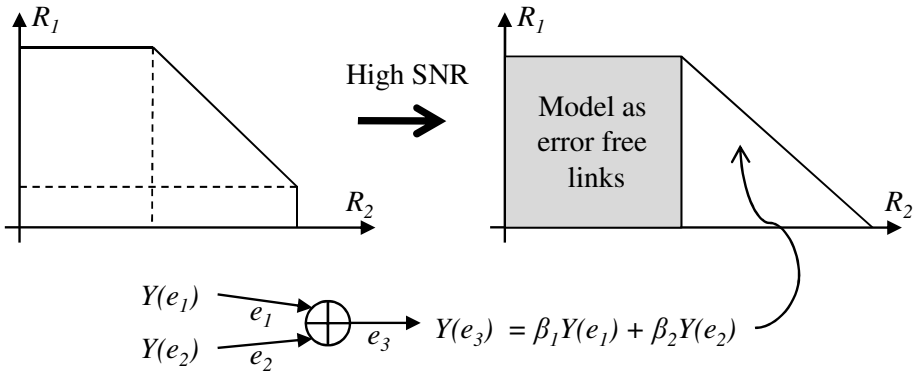



Figure 3-3: A model of MAC in the high SNR regime. Additive MAC with two users, and the corresponding rate region. The triangular region is modeled as a set of finite field additive MACs.

codes in higher field size $\mathbb{F}_q$ can be mapped to binary-vector codes without loss in performance. This insight, combined with that of [67], suggests that an algebraic property of a scalar code may translate into another algebraic property of the corresponding vector code.

## 3.2 Network Model

As in [6][7], we shall consider the high SNR regime, in which interference is the dominating factor. In high SNR, analog network coding, which allows and encourages strategic interference, is near optimal [78]. Analog network coding is a physical layer coding technique, introduced by [51], in which intermediate nodes amplify-and-forward the received signals without decoding. Thus, the nodes amplify not only the superposed signals from different transmitters but also the noise. Note that a network operating in high SNR regime is different from a network with high gain since a large gain amplifies the noise as well as the signal.

In the high SNR regime, the Cover-Wyner region may be well approximated by the combination of two regions, one square and one triangular, as in Figure 3-3. The square (shaded) part can be modeled as parallel links for the users, since they do not interfere. The triangular (unshaded) part can be considered as that of a time-division multiplexing (TDM), which is equivalent to using noiseless finite-field additive MAC [85]. This result

holds not only for binary field additive MAC, but also for higher field size additive MAC [85].

The ADT network model uses binary channels, and thus, binary additive MACs are used to model interference. Prior to [6][7], Effros *et al.* presented an additive MAC over a finite field $\mathbb{F}_q$ [25]. The Min-cut Max-flow theorem holds for all of the cases above. It may seem that the ADT network model differs greatly from that of [25] owing to the difference in field sizes used. In general, codes in $\mathbb{F}_q$ subsume binary codes, *i.e. binary-vector* codes in $(\mathbb{F}_2)^m$. However, for point-to-point links with memory (or equivalently by allowing nodes to code across time), we can convert a code in $(\mathbb{F}_2)^m$ to a code in higher field size $\mathbb{F}_q$ and vice versa by normalizing to an appropriate time unit. Note that ADT network model uses binary additive MACs and point-to-point links. Therefore, our work in part shows an equivalence of higher field size codes and binary-vector codes in ADT networks.

As noted in Section 3.1, reference [43] presented a method of converting between binary-vector codes and higher field size codes. We can achieve a higher field size in ADT networks by combining multiple binary channels. In other words, consider two nodes $V_1$ and $V_2$ with two binary channels connecting $V_1$ to $V_2$. Now, instead of considering them as two binary channels, we can "combine" the two channels as one with capacity of 2-bits. In this case, instead of using $\mathbb{F}_2$, we can use a larger field size of $\mathbb{F}_4$. Thus, selecting a larger field size $\mathbb{F}_q$, $q > 2$ in ADT network results in fewer but higher capacity parallel channels. Reference [43] also provides a conversion from a code in a higher field $\mathbb{F}_q$ to a binary-vector scheme in $\mathbb{F}_{2^m}$ where $q \leq 2^m$. Therefore, a solution in $\mathbb{F}_q$ may be converted back to a binary-vector scheme, which may be more appropriate for the original ADT model. Furthermore, it is known that to achieve capacity for multicast connections, $\mathbb{F}_2$ is not sufficient [28]; thus, we need to operate in a higher field size. Therefore, we shall not restrict ourselves to $\mathbb{F}_2$.

We now proceed to defining the network model precisely. A wireless network is modeled using a directed graph $G = (\mathcal{V}, \mathcal{E})$ with a *supernode* set $\mathcal{V}$ and an edge set $\mathcal{E}$, as shown in Figure 3-4. A supernode $V \in \mathcal{V}$ is a node of the original network. We use the term supernode to emphasize the fact that supernode $V$ consists of *input ports* $I(V)$ and *output ports* $O(V)$, as shown in Figure 3-5. Let $\mathcal{S}, \mathcal{T} \subseteq \mathcal{V}$ be the set of source and destination supernodes. An edge $(e_1, e_2)$ may exist from an output port $e_1 \in O(V_1)$ to an input port

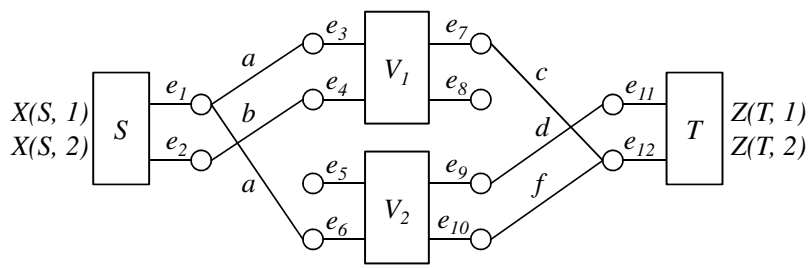



Figure 3-4: An example network. We omit $I(S)$ and $O(T)$ in this diagram as they do not participate in the communication.

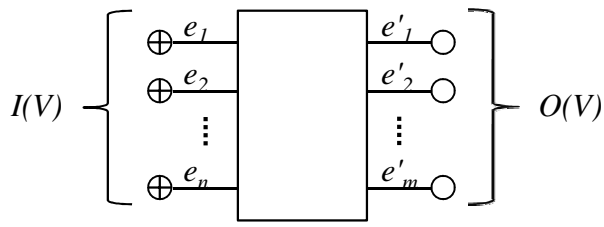



Figure 3-5: An illustration of a supernode $V$. A supernode $V$ consists of input ports $I(V)$ and output ports $O(V)$.

$e_2 \in I(V_2)$, for any $V_1, V_2 \in \mathcal{V}$. Let $\mathcal{E}(V_1, V_2)$ be the set of edges from $O(V_1)$ to $I(V_2)$. All edges are of unit capacity, where capacity is normalized with respect to the symbol size of $\mathbb{F}_q$.

Noise is embedded, or *hard-coded*, in the structure of the ADT network in the following way. Parallel links of $\mathcal{E}(V_1, V_2)$ deterministically model noise between $V_1$ and $V_2$. Let $SNR_{(V_i,V_j)}$ be the signal-to-noise ratio from supernode $V_i$ to supernode $V_j$. Then,

$$|\mathcal{E}(V_1, V_2)| = \left\lceil \frac{1}{2} \log SNR_{(V_i,V_j)} \right\rceil . \tag{3.1}$$

Thus, the number of edges between two supernodes $V_i$ and $V_j$ represents the channel quality (equivalently, the noise) between the two supernodes.

Given such a wireless network $G = (\mathcal{V}, \mathcal{E})$, let $\mathcal{S}$ be the set of sources. A source supernode $S \in \mathcal{S}$ has *independent* random processes $\mathcal{X}(S) = [X(S,1), X(S,2), \cdots, X(S, \mu(S))]$, $\mu(S) \leq |O(S)|$, which it wishes to communicate to a set of destination supernodes $\mathcal{T}(S) \subseteq \mathcal{T}$. In other words, we want $T \in \mathcal{T}(S)$ to replicate a subset of the random processes, denoted

$\mathcal{X}(S,T) \subseteq \mathcal{X}(S)$, by the means of the network. This algebraic formulation is not restricted to multicast connections; different sources may wish to communicate to different subsets of destinations.

We define a *connection* $c$ as a triple $(S, T, \mathcal{X}(S,T))$, and the rate of $c$ is defined as

$$R(c) = \sum_{X(S,i) \in \mathcal{X}(S,T)} H(X(S,i)) = |\mathcal{X}(S,T)| \text{ (symbols)}. \tag{3.2}$$

Information is transmitted through the network in the following manner. A supernode $V$ sends information through $e \in O(V)$ at a rate at most one symbol per time unit. Let $Y(e)$ denote the random process at port $e$. In general, $Y(e)$, $e \in O(V)$, is a function of $Y(e')$, $e' \in I(V)$. In this chapter, we consider only linear functions; therefore,

$$Y(e) = \sum_{e' \in I(V)} \beta_{(e',e)} Y(e'), \text{ for } e \in O(V). \tag{3.3}$$

For a source supernode $S$ and $e \in O(S)$,

$$Y(e) = \sum_{e' \in I(V)} \beta_{(e',e)} Y(e') + \sum_{X(S,i) \in \mathcal{X}(S)} \alpha_{(i,e)} X(S,i). \tag{3.4}$$

Finally, the destination $T$ receives a collection of input processes $Y(e')$, $e' \in I(T)$. Supernode $T$ generates a set of random processes $\mathcal{Z}(T) = [Z(T,1), Z(T,2), \cdots, Z(T,\nu(T))]$ where

$$Z(T,i) = \sum_{e' \in I(T)} \epsilon_{(e',(T,i))} Y(e'). \tag{3.5}$$

A connection $c = (S, T, \mathcal{X}(S,T))$ is established successfully if $\mathcal{X}(S) = \mathcal{Z}(T)$.

A supernode $V$ is said to *broadcast* to a set $\mathcal{V}' \subseteq \mathcal{V}$ if $\mathcal{E}(V, V') \neq \emptyset$ for all $V' \in \mathcal{V}'$. In Figure 3-4, $S$ broadcasts to supernodes $V_1$ and $V_2$. Superposition occurs at the input port $e' \in I(V)$; therefore,

$$Y(e') = \sum_{(e,e') \in \mathcal{E}} Y(e) \tag{3.6}$$

over a finite field $\mathbb{F}_q$. We say there is a $|\mathcal{V}'|$-user MAC channel if $\mathcal{E}(V', V) \neq \emptyset$ for all $V' \in \mathcal{V}'$. In Figure 3-4, supernodes $V_1$ and $V_2$ are users, and $T$ the receiver in a 2-user MAC.

$$
\begin{aligned}
Y(e_1) &= \alpha_{(1,e_1)}X(S,1) + \alpha_{(2,e_1)}X(S,2) \\
Y(e_2) &= \alpha_{(1,e_2)}X(S,1) + \alpha_{(2,e_2)}X(S,2) \\
Y(e_3) &= Y(e_6) = Y(e_1) \\
Y(e_4) &= Y(e_2) \\
Y(e_5) &= Y(e_8) = 0 \\
Y(e_7) &= \beta_{(e_3,e_7)}Y(e_3) + \beta_{(e_4,e_7)}Y(e_4) \\
Y(e_9) &= Y(e_{11}) = \beta_{(e_6,e_9)}Y(e_6) \\
Y(e_{10}) &= \beta_{(e_6,e_{10})}Y(e_6) \\
Y(e_{12}) &= Y(e_7) + Y(e_{10}) \\
Z(T,1) &= \epsilon_{(e_{11},(T,1))}Y(e_{11}) + \epsilon_{(e_{12},(T,1))}Y(e_{12}) \\
Z(T,2) &= \epsilon_{(e_{11},(T,2))}Y(e_{11}) + \epsilon_{(e_{12},(T,2))}Y(e_{12})
\end{aligned}
$$

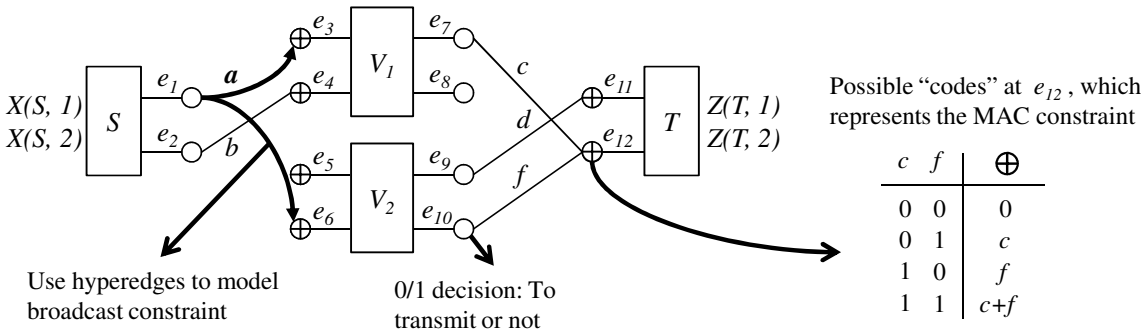
Figure 3-6: A set of linear equations relating the various processes from Figure 3-4.

For a given network $G$ and a set of connections $\mathcal{C}$, we say that $(G, \mathcal{C})$ is *solvable* if it is possible to establish successfully all connections $c \in \mathcal{C}$. The broadcast and MAC constraints are given by the network; however, we are free to choose the variables $\alpha_{(i,e)}$, $\beta_{(e',e)}$, and $\epsilon_{(e',i)}$ from $\mathbb{F}_q$. Hence, the problem of checking whether a given $(G, \mathcal{C})$ is solvable is equivalent to finding a feasible assignment to $\alpha_{(i,e)}, \beta_{(e',e)}$, and $\epsilon_{(e',(T,i))}$.

**Example 3.2.1** *The equations in Figure 3-6 relate the various processes in the example network in Figure 3-4. Note that in Figure 3-4, we have set* $Y(e_1) = a$, $Y(e_2) = b$, $Y(e_7) = c$, $Y(e_9) = d$, *and* $Y(e_{10}) = f$ *for notational simplicity.*

### 3.2.1 An Interpretation of the Network Model

The ADT network model uses multiple channels from an output port to model broadcast. In Figure 3-4, there are two edges from output port $e_1$ to input ports $e_3$ and $e_6$; however, due to the broadcast constraint, the two edges $(e_1, e_3)$ and $(e_1, e_6)$ carry the same information $a$. This introduces considerable complexity in constructing a network code as well as computing the min-cut of the network [6][7][5][37]. This is because multiple edges from a port do not capture the broadcast dependencies. Furthermore, the broadcast dependencies have to be propagated through the network.




| $c$ | $f$ | $\oplus$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | $c$ |
| 1 | 0 | $f$ |
| 1 | 1 | $c+f$ |

Figure 3-7: A new interpretation of the example network from Figure 3-4. The broadcast channel is modeled using an *hyperedge*. As a result, an output port's decision to transmit or not naturally affects all the input ports adjacent to it. Furthermore, interference is modeled using a finite field additive MAC, which provides a set of possible *binary codes* at the input ports.

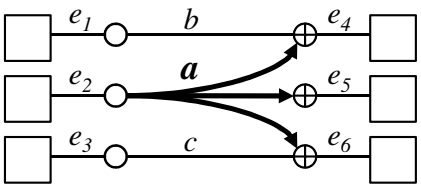



Figure 3-8: An example of finite field additive MAC.

In our approach, we remedy this by introducing the use of hyperedges, as shown in Figure 3-7 and further discussed in Section 3.3. An output port's decision to transmit affects the entire hyperedge; as a result, the output port transmits to all the input ports connected to the hyperedge simultaneously. This removes the aforementioned difficulties of computing the min-cut of ADT networks, as it naturally captures the broadcast dependencies. We shall redefine the min-cut of ADT networks using our approach in Section 3.4.

The finite field additive MAC model can be viewed as a set of codes that an input port may receive. As shown in Figure 3-7, input port $e_{12}$ receives one of the four possible codes. The code that $e_{12}$ receives depends on output ports $e_7$'s and $e_{10}$'s decision to transmit or not.

The difficulty in constructing a network code does not come from any single broadcast or MAC constraint. The difficulty in constructing a code is in satisfying multiple MAC and broadcast constraints simultaneously. For example, in Figure 3-8, the fact that $e_4$ may

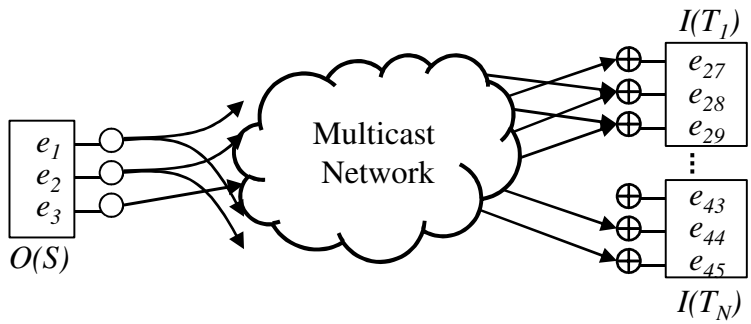



Figure 3-9: An example of a multicast network. Single multicast network with source $S$ and receivers $T_1, \cdots, T_N$.

receive $a + b$ does not constrain the choice of $a$ nor $b$. This is because we can choose any $a$ and $b$ such that $a + b \neq 0$, and ensure that both $a$ and $b$ are decoded as long as enough degrees of freedom are received by the destination node. The same argument applies to $e_6$ receiving $a + c$. However, the problem arises from the fact that a choice of value for $a$ at $e_2$ interacts both with $b$ and $c$. In such a case, we need to ensure that both $a + b \neq 0$ and $a + c \neq 0$. Therefore, our constraint is $(a + b)(a + c) \neq 0$. As the network grows in size, we will need to satisfy more constraints simultaneously. As we shall see in Section 3.4, we eliminate this difficulty by allowing the use of a larger field, $\mathbb{F}_q$.

## 3.3 Algebraic Network Coding Formulation

We provide an algebraic formulation for the ADT network problem $(G, \mathcal{C})$, and present an algebraic condition under which the system $(G, \mathcal{C})$ is solvable. We assume that $G$ is acyclic in this section; however, we shall extend the results in this section to ADT networks with cycles in Section 3.8. For simplicity, we describe the multicast problem with a single source $S$ and a set of destination supernodes $\mathcal{T}$, as in Figure 3-9. However, this formulation can be extended to multiple source $S_1, S_2, \cdots S_K$ by adding a super-source $S$ as in Figure 3-10.

We define a system matrix $M$ to describe the relationship between the source's random processes $\mathcal{X}(S)$ and the destinations' processes $\mathcal{Z} = [\mathcal{Z}(T_1), \mathcal{Z}(T_2), \cdots, \mathcal{Z}(T_{|\mathcal{T}|})]$. Thus, we want to characterize $M$ where

$$\mathcal{Z} = \mathcal{X}(S) \cdot M. \tag{3.7}$$

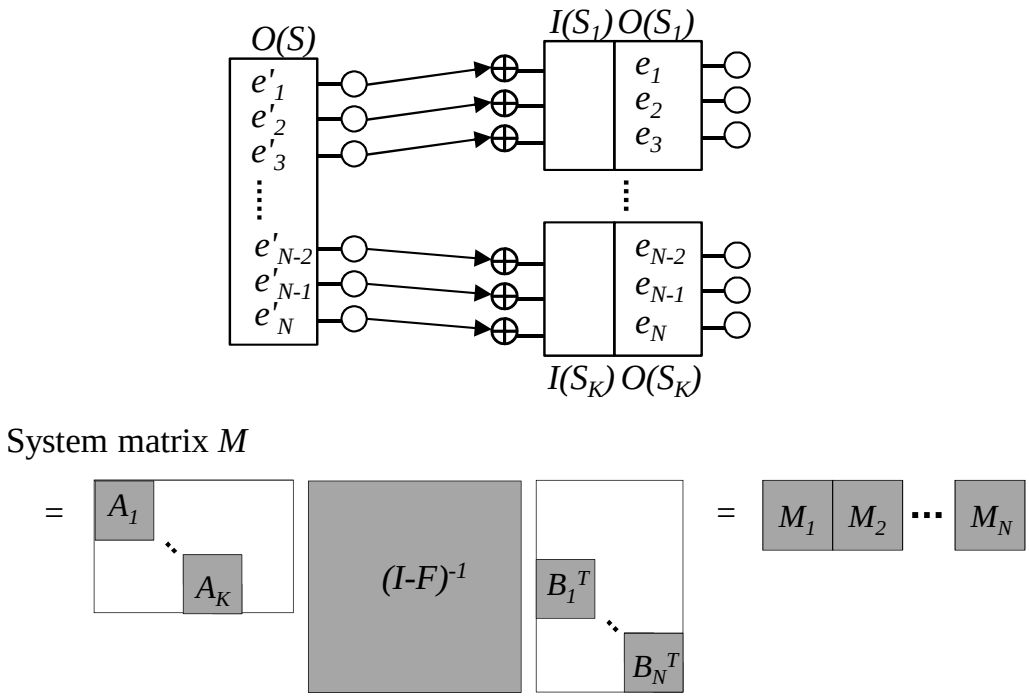



Figure 3-10: An example network with a supersource. A network with multiple sources $S_1, S_2, \cdots, S_K$ can be converted to a single source problem by adding a super-source $S$ with $|O(S)| = \sum_{i=1}^{K} |O(S_i)|$. Each $e'_j \in O(S)$ has a "one-to-one connection" to a $e_j \in O(S_i)$, for $i \in \{1, 2, \cdots, K\}$. Matrix $A_i$ represents the encoding matrix for source $S_i$, while $B_j$ is the decoding matrix at destination $T_j$. The white area represents the zero elements, and the shaded area represents the coding coefficients.

The matrix $M$ is composed of three matrices, $A$, $F$, and $B$.

### 3.3.1 Adjacency Matrix $F$

Given $G$, we define the adjacency matrix $F$ as follows:

$$F_{i,j} = \begin{cases} 1 & \text{if } (e_i, e_j) \in \mathcal{E}, \\ \beta_{(e_i,e_j)} & \text{if } e_i \in I(V),\ e_j \in O(V) \text{ for } V \in \mathcal{V}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

Matrix $F$ is defined on the ports, rather than on the supernodes. This is because, in the ADT model, each port is the basic receiver or transmitter unit. Each entry $F_{i,j}$ represents the input-output relationships of the ports. A zero entry indicates that the ports are not directly connected, while an entry of one represents that they are connected.

The adjacency matrix $F$ naturally captures the physical structure of the ADT network. A row of $F$ with multiple entries of one represents the broadcast hyperedge; while a column with multiple entries of one represents the MAC constraint. Note that the 0-1 entries of $F$ represent the *fixed* network topology as well as the broadcast and MAC constraints. On the other hand, $\beta_{(e_i,e_j)}$ are free variables, representing the coding coefficients used at $V$ to map the input port processes to the output port processes. This is the key difference between the work presented here and in [67]. The adjacency matrix $F$ is partially fixed in the ADT network model due to network topology, the broadcast constraints, and the MAC constraints, while in [67], only the network topology affects $F$.

In [6][7], the supernodes are allowed to perform any internal operations; while in [5][37], only permutation matrices (*i.e.* routing) are allowed. Note that [5][37] only consider a single unicast traffic, in which routing is known to be sufficient. References [6][7] showed that linear operations are sufficient to achieve the capacity in ADT networks for a single multicast traffic. We consider a general setup in which $\beta_{(e_i,e_j)} \in \mathbb{F}_q$. Therefore, our setup allows any matrix operation, as in [6][7].

Observe that $F^k$, the $k$-th power of an adjacency matrix of a graph $G$, shows the existence of paths of length $k$ between any two nodes in $G$. Therefore, the series $I + F +$

$F^2 + F^3 + \cdots$ represents the connectivity of the network. It can be verified that $F$ is nilpotent, which means that there exists a $k$ such that $F^k$ is a zero matrix. As a result, $I+F+F^2+F^3+\cdots$ can be written as $(I-F)^{-1}$, and $(I-F)^{-1}$ represents the connectivity of the networks; thus, the impulse response of the network.

From our discussions above, the existence of $(I-F)^{-1}$ or the invertibility of $(I-F)$ may be desirable for us. We show in the following lemma that $(I-F)$ is indeed invertible for all acyclic networks.

**Lemma 3.3.1** *Let $F$ be the adjacency matrix of acyclic network $G = (\mathcal{V}, \mathcal{E})$. Then, $I - F$ is invertible.*

*Proof:* Since $G$ is cycle-free, there is a partial topological ordering of the nodes in $\mathcal{V}$. Since each port is associated with a node in $\mathcal{V}$, there also is an ordering of the ports. Therefore, $F$, which is defined over the ports, is a strict upper-triangle matrix. This shows that $I - F$ is an upper-triangle matrix with all diagonal entries equal to one. Since the determinant of a triangular matrix is the product of its diagonal entries, we have $\det(I - F) = 1$. ■

**Example 3.3.2** *In Figure 3-11, we provide the $12 \times 12$ adjacency matrix $F$ for the example network in Figures 3-4 and 3-7. The first row (with two entries of one) represents the broadcast hyperedge, $e_1$ connected to both $e_3$ and $e_6$. The last column with two entries equal to one represents the MAC constraint, both $e_7$ and $e_{10}$ transmitting to $e_{12}$. The highlighted elements in $F$ represent the coding variables, $\beta_{(e',e)}$, of $V_1$ and $V_2$ in Figure 3-7. For some $(e', e)$, $\beta_{(e',e)} = 0$ since these ports of $V_1$ and $V_2$ are not used.*

### 3.3.2 Encoding Matrix $A$

Matrix $A$ represents the encoding operations performed at $S$. We define a $|\mathcal{X}(S)| \times |\mathcal{E}|$ encoding matrix $A$ as follows:

$$A_{i,j} = \begin{cases} \alpha_{(i,e_j)} & \text{if } e_j \in O(S) \text{ and } X(S,i) \in \mathcal{X}(S), \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

$$\begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \beta_{(e_3,e_7)} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \beta_{(e_4,e_7)} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_{(e_6,e_9)} & \beta_{(e_6,e_{10})} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Figure 3-11: The $12 \times 12$ adjacency matrix $F$ for the example network in Figure 3-4.

**Example 3.3.3** *We provide the $2 \times 12$ encoding matrix $A$ for the network in Figure 3-4.*

$$A = \begin{pmatrix} \alpha_{1,e_1} & \alpha_{1,e_2} & 0 & \cdots & 0 \\ \alpha_{2,e_1} & \alpha_{2,e_2} & 0 & \cdots & 0 \end{pmatrix}. \tag{3.10}$$

### 3.3.3 Decoding Matrix $B$

Matrix $B$ represents the decoding operations performed at the destination $T \in \mathcal{T}$. Since there are $|\mathcal{T}|$ destination nodes, $B$ is a matrix of size $|\mathcal{Z}| \times |\mathcal{E}|$ where $\mathcal{Z}$ is the set of random processes derived at the destination supernodes. We define the decoding matrix $B$ as follows:

$$B_{i,(T_j,k)} = \begin{cases} \epsilon_{(e_i,(T_j,k))} & \text{if } e_i \in I(T_j), Z(T_j,k) \in \mathcal{Z}(T_j), \\ 0 & \text{otherwise.} \end{cases} \tag{3.11}$$

**Example 3.3.4** *We provide the $2 \times 12$ decoding matrix $B$ for the example network in Figure 3-4.*

$$B = \begin{pmatrix} 0 & \cdots & 0 & \epsilon_{(e_{11},(T,1))} & \epsilon_{(e_{12},(T,1))} \\ 0 & \cdots & 0 & \epsilon_{(e_{11},(T,2))} & \epsilon_{(e_{12},(T,2))} \end{pmatrix}. \tag{3.12}$$

### 3.3.4 System Matrix $M = A(I-F)^{-1}B^T$

**Theorem 3.3.5** *Given a network $G = (\mathcal{V}, \mathcal{E})$, let $A$, $B$, and $F$ be the encoding, decoding, and adjacency matrices, respectively. Then, the system matrix $M$ is given by*

$$M = A(I-F)^{-1}B^T. \tag{3.13}$$

*Proof:* The proof of this theorem is similar to that of Theorem 3 in [67]. As previously shown in Lemma 3.3.1, $(I-F)^{-1} = I + F + F^2 + \cdots$ always exists for an acyclic network $G$. ■

The algebraic framework shows a clear separation between the given physical constraints (fixed 0-1 entries of $F$ showing the topology, the broadcast constraints, and the MAC constraints), and the coding decisions. As mentioned previously, we can freely choose the coding variables $\alpha_{(i,e_j)}$, $\epsilon_{(e_i,(T_j,k))}$, and $\beta_{(e_i,e_j)}$. Thus, solvability of $(G, \mathcal{C})$ is equivalent to assigning values to $\alpha_{(i,e_j)}$, $\epsilon_{(e_i,(T_j,k))}$, and $\beta_{(e_i,e_j)}$ such that each receiver $T \in \mathcal{T}$ is able to decode the data it is intended to receive.

**Example 3.3.6** *We can combine the matrices $F$, $A$, and $B$ from Examples 3.3.2, 3.3.3, and 3.3.4 respectively to obtain the system matrix $M = A(I-F)^{-1}B^T$ for the network in Figure 3-4. We show a schematic of the system matrix $M$ in Figure 3-12.*

## 3.4 Definition of Min-cut

Consider a source $S$ and a destination $T$. Reference [6] proves the maximal achievable rate to be the minimum value of all $S$-$T$ cuts, denoted $mincut(S,T)$, which we reproduce below in Definition 3.4.1.

**Definition 3.4.1 (Min-cut of ADT networks [6][7])** *A cut $\Omega$ between a source $S$ and a destination $T$ is a partition of the supernodes into two disjoint sets $\Omega$ and $\Omega^c$ such that $S \in \Omega$ and $T \in \Omega^c$. For any cut, $G_\Omega$ is the incidence matrix associated with the bipartite graph with ports in $\Omega$ and $\Omega^c$. Then, the capacity of the given ADT network, equivalently*

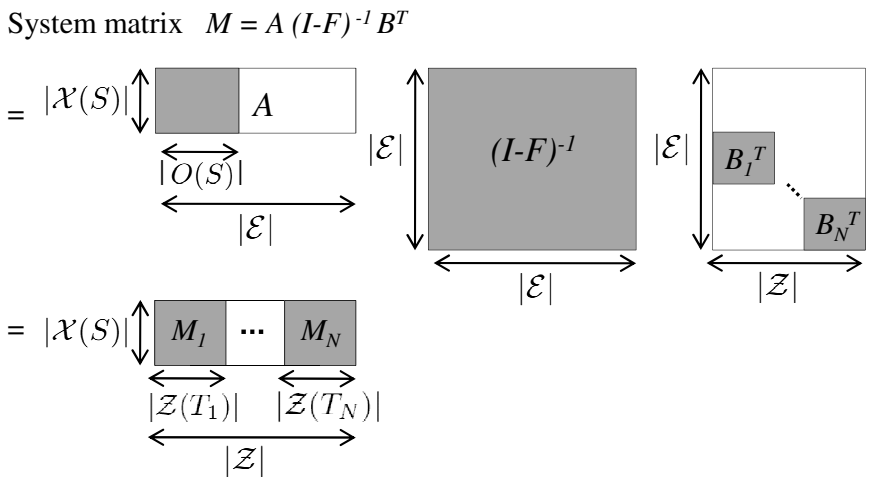



Figure 3-12: An example of the system matrix $M$ for a multicast connection. The system matrix $M$ has components $A$, $(I-F)^{-1}$, and $B$ for the single multicast connection with source $S$ and destinations $T_i$, $i \in \{1, 2, \cdots, N\}$.

*$mincut(S,T)$, is defined as*

$$mincut(S,T) = \min_{\Omega} rank(G_{\Omega}). \tag{3.14}$$

*This rate of $mincut(S,T)$ can be achieved using linear operations for a single unicast/multicast connection.*

In order to compute $mincut(S,T)$ using Definition 3.4.1, we need to optimize over all cuts between $S$ and $T$. In addition, the proof of achievability in [6] is not constructive, as it assumes infinite block length and does not consider the details of internal operations at the supernodes.

We introduce a new algebraic definition of the min-cut, and show that it is equivalent to that of Definition 3.4.1.

**Theorem 3.4.2** *The capacity of the given ADT, equivalently the minimum value of all*

*$S-T$ cuts $mincut(S,T)$, is*

$$mincut(S,T) = \min_{\Omega} rank(G_\Omega) \tag{3.15}$$

$$= \max_{\alpha_{(i,e)},\beta_{(e',e)},\epsilon_{(e',i)}} rank(M). \tag{3.16}$$

*Proof:* By [6] and Definition 3.4.1, we know that $mincut(S,T) = \min_\Omega \text{rank}(G_\Omega)$. Therefore, we show that $\max_{\alpha,\beta,\epsilon} \text{rank}(M)$ is equivalent to the maximal achievable rate in an ADT network.

First, we show that $mincut(S,T) \geq \max_{\alpha,\beta,\epsilon} \text{rank}(M)$. In our algebraic formulation, $\mathcal{Z}(T) = \mathcal{X}(S)M$. Therefore, the rank of $M$ represents the rate achieved. Let $R = \max_{\alpha,\beta,\epsilon} \text{rank}(M)$. Then, there exists an assignment of $\alpha_{(i,e)}, \beta_{(e',e)}$, and $\epsilon_{(e',i)}$ such that the network achieves a rate of $R$. By the definition of min-cut, it must be the case that $mincut(S,T) \geq R = \max_{\alpha,\beta,\epsilon} \text{rank}(M)$.

Second, we show that $mincut(S,T) \leq \max_{\alpha,\beta,\epsilon} \text{rank}(M)$. Assume that $R = mincut(S,T)$. Then, by [6][7], there exists a linear configuration of the network such that we can achieve a rate of $R$ such that the destination $T$ is able to reproduce $\mathcal{X}(S,T)$. This configuration of the network provides a linear relationship of the source-destination processes. Actually, the resulting system matrix is an identity matrix, as the destination reproduced $\mathcal{X}(S,T)$. This implies that there is an assignment of the variables $\alpha_{(i,e)}, \beta_{(e',e)}$, and $\epsilon_{(e',i)}$ for our algebraic framework such that the system matrix has rank $R$. We denote $M'$ to be the system matrix corresponding to this assignment. Note that, by the definition, $M'$ is an $R \times R$ matrix with a rank of $R$. Therefore, $\max_{\alpha,\beta,\epsilon} \text{rank}(M) \geq \text{rank}(M') = mincut(S,T)$. ■

The system matrix $M$ (thus, the network and decodability at the destinations) depends not only on the structure of the ADT network, but also on the field size used, supernodes' internal operations, transmission rate, and connectivity. For example, the network topology may change with a choice of larger field size, since larger field sizes result in fewer parallel edges or channels. Furthermore, if we adjust the rate such that $|\mathcal{X}(S)| \leq mincut(S,T)$, then $M$ is full rank. However, if $|\mathcal{X}(S)| > mincut(S,T)$, then $M$ may have rank of $mincut(S,T)$ but not be full-rank. It is important to note that the cut value in the ADT network may not equal the graph theoretical cut value as shown in Figure 2 in [5].

## 3.5 Min-cut Max-flow Theorem

In this section, we provide an algebraic interpretation of the Min-cut Max-flow theorem for a single unicast connection and a single multicast connection [6][7]. This result is a direct consequence of [67] when applied to the algebraic formulation for the ADT network. We also show that a distributed randomized coding scheme achieves the capacity for these connections.

**Theorem 3.5.1 (Min-cut Max-flow Theorem)** *Given an acyclic network $G$ with a single connection $c = (S, T, \mathcal{X}(S,T))$ of rate $R(c) = |\mathcal{X}(S,T)|$, the following are equivalent:*

1. *A unicast connection $c$ is feasible.*
2. *$mincut(S,T) \geq R(c)$.*
3. *There exists an assignment of $\alpha_{(i,e_j)}$, $\epsilon_{(e_i,(T_j,k))}$, and $\beta_{(e_i,e_j)}$ such that the $R(c) \times R(c)$ system matrix $M$ is invertible in $\mathbb{F}_q$,* i.e. $\det(M) \neq 0$.

*Proof:* Statements 1 and 2 have been shown to be equivalent in ADT networks [6][5][37]. We now show that statements 1 and 3 are equivalent. Assume that there exists an assignment such that $\det(M) \neq 0$ in $\mathbb{F}_q$. Then, the system matrix $M$ is invertible; there exists $M^{-1}$ such that $\mathcal{X}(S) = \mathcal{Z}M^{-1}$ and a connection of rate $R(c) = |\mathcal{X}(S,T)|$ is established. Conversely, if connection $c$ is feasible, there exists a solution to the ADT network $G$ that achieves a rate of $R(c)$. When using this ADT network solution, the destination $T$ is able to reproduce $\mathcal{X}(S,T)$; thus the resulting system matrix is an identity matrix, $M = I$. Therefore, $M$ is invertible. ■

**Corollary 3.5.2 (Random Coding for Unicast)** *Consider an ADT network problem with a single connection $c = (S, T, \mathcal{X}(S,T))$ of rate $R(c) = |\mathcal{X}(S,T)| \leq mincut(S,T)$. Then, random linear network coding, where some or all code variables $\alpha_{(i,e_j)}$, $\epsilon_{(e_i,(T_j,k))}$, and $\beta_{(e_i,e_j)}$ are chosen independently and uniformly over all elements of $\mathbb{F}_q$, guarantees decodability at destination $T$ with high probability at least $(1 - \frac{1}{q})^{\eta}$, where $\eta$ is the number of links carrying random combinations of the source processes.*

*Proof:* From Theorem 3.5.1, there exists an assignment of $\alpha_{(i,e_j)}$, $\epsilon_{(e_i,(T_j,k))}$, and $\beta_{(e_i,e_j)}$ such that $\det(M) \neq 0$, which gives a capacity-achieving network code for the given $(G, \mathcal{C})$. Thus, this connection $c$ is feasible for the given network. Reference [41] proves that random linear network coding is capacity-achieving and guarantees decodability with high probability $(1 - \frac{1}{q})^\eta$ for such a feasible unicast connection $c$. ■

**Theorem 3.5.3 (Single Multicast Theorem)** *Given an acyclic network $G$ and connections $\mathcal{C} = \{(S, T_1, \mathcal{X}(S)), (S, T_2, \mathcal{X}(S)), \cdots, (S, T_N, \mathcal{X}(S))\}$, $(G, \mathcal{C})$ is solvable if and only if $mincut(S, T_i) \geq |\mathcal{X}(S)|$ for all $i$.*

*Proof:* If $(G, \mathcal{C})$ is solvable, then $mincut(S, T_i) \geq |\mathcal{X}(S)|$. Therefore, we only have to show the converse. Assume $mincut(S, T_i) \geq |\mathcal{X}(S)|$ for all $i \in [1, N]$. The system matrix $M = \{M_i\}$ is a concatenation of $|\mathcal{X}(S)| \times |\mathcal{X}(S)|$ matrices where $\mathcal{Z}(T_i) = \mathcal{X}(S)M_i$, as shown in Figure 3-12. We can write

$$M = [M_1, M_2, \cdots, M_N] = A(I - F)^{-1}B^T = A(I - F)^{-1}[B_1, B_2, \cdots, B_N]. \tag{3.17}$$

Thus, $M_i = A(I - F)^{-1}B_i$. Note that $A$ and $B_i$'s do not substantially contribute to the system matrix $M_i$ since $A$ and $B_i$ only perform linear encoding and decoding at the source and destinations, respectively.

By Theorem 3.5.1, there exists an assignment of $\alpha_{(i,e_j)}$, $\epsilon_{(e_i,(T_j,k))}$, and $\beta_{(e_i,e_j)}$ such that each individual system submatrix $M_i$ is invertible, *i.e.* $\det(M_i) \neq 0$. However, an assignment that makes $\det(M_i) \neq 0$ may lead to $\det(M_j) = 0$ for $i \neq j$. Therefore, we need to show that it is possible to achieve *simultaneously* $\det(M_i) \neq 0$ for all $i$. In other words, we need to show that

$$\prod_i \det(M_i) \neq 0. \tag{3.18}$$

By [41], we know that if the field size is larger than the number of receivers ($q > N$), then there exists an assignment of $\alpha_{(i,e_j)}$, $\epsilon_{(e_i,(T_j,k))}$, and $\beta_{(e_i,e_j)}$ such that $\det(M_i) \neq 0$ for all $i$. ■

**Corollary 3.5.4 (Random Coding for Multicast)** *Consider an ADT network problem with a single multicast connection $\mathcal{C} = \{(S, T_1, \mathcal{X}(S)), (S, T_2, \mathcal{X}(S)), \cdots, (S, T_N, \mathcal{X}(S))\}$ with*

*$mincut(S, T_i) \geq |\mathcal{X}(S)|$ for all $i$. Then, random linear network coding, where some or all code variables $\alpha_{(i,e_j)}$, $\epsilon_{(e_i,(T_j,k))}$, and $\beta_{(e_i,e_j)}$ are chosen independently and uniformly over all elements of $\mathbb{F}_q$, guarantees decodability at destination $T_i$ for all $i$ simultaneously with high probability at least $(1 - \frac{N}{q})^\eta$, where $\eta$ is the number of links carrying random combinations of the source processes; thus, $\eta \leq |\mathcal{E}|$.*

*Proof:* Theorem 3.5.3 shows that the multicast connection $\mathcal{C}$ is feasible. Given that the multicast connection is feasible, reference [41] shows that random linear network coding achieves the capacity for multicast connections, and allows all destination supernodes to decode the source processes $\mathcal{X}(S)$ with high probability of at least $(1 - \frac{N}{q})^\eta$. ■

Theorem 3.5.1 and Theorem 3.5.3 provide an alternate proof of sufficiency of linear operations for unicast and multicast in ADT networks, which was first shown in [6].

## 3.6 Extensions to Other Connections

In this section, we extend the ADT network results to a more general set of traffic requirements. We use the algebraic formulation and the results from [67] to characterize the feasibility conditions for a given problem $(G, \mathcal{C})$.

### 3.6.1 Multiple Multicast

**Theorem 3.6.1 (Multiple Multicast Theorem)** *Given a network $G$ and a set of connections $\mathcal{C} = \{(S_i, T_j, \mathcal{X}(S_i)) \mid S_i \in \mathcal{S}, T_j \in \mathcal{T}\}$, $(G, \mathcal{C})$ is solvable if and only if Min-cut Max-flow bound is satisfied for any cut that separates the set of source supernodes $\mathcal{S}$ and a destination $T_j$, for all $T_j \in \mathcal{T}$.*

*Proof:* We first introduce a super-source $S$ with $|O(S)| = \sum_{S_i \in \mathcal{S}} |O(S_i)|$, and connect each $e'_j \in O(S)$ to an input of $S_i$ such that $e_j \in O(S_i)$ as shown in Figure 3-10. Then, we apply Theorem 3.5.3, which proves the statement. ■

**Corollary 3.6.2 (Random Coding for Multiple Multicast)** *Consider an ADT network problem with multiple multicast connections $\mathcal{C} = \{(S_i, T_j, \mathcal{X}(S_i)) | S_i \in \mathcal{S}, T_j \in \mathcal{T}\}$ with $mincut(\mathcal{S}, T_j) \geq \sum_i |\mathcal{X}(S_i)|$ for all $i$. Then, random linear network coding, where some*

*or all code variables* $\alpha_{(i,e_j)}$, $\epsilon_{(e_i,(T_j,k))}$, *and* $\beta_{(e_i,e_j)}$ *are chosen independently and uniformly over all elements of* $\mathbb{F}_q$, *guarantees decodability at destination* $T_i$ *for all* $i$ *simultaneously with high probability at least* $(1-\frac{N}{q})^{\eta}$, *where* $\eta$ *is the number of links carrying random combinations of the source processes; thus,* $\eta \leq |\mathcal{E}|$.

The optimality of random coding in Corollary 3.6.2 comes from the fact that we allow coding across multicast connections $(S_i, T_j, \mathcal{X}(S_i))$'s. In other words, the source supernodes and the intermediate supernodes can randomly and uniformly select the coding coefficients. Thus, intermediate nodes within the network do not distinguish the flow from source $S_i$ from that of $S_j$, and are allowed to encode them together randomly.

### 3.6.2 Disjoint Multicast

**Theorem 3.6.3 (Disjoint Multicast Theorem)** *Given an acyclic network* $G$ *with a set of connections* $\mathcal{C} = \{(S, T_i,\ \mathcal{X}(S,T_i)) \ \mid i = 1, 2, \cdots, K\}$ *is called a* disjoint multicast *if*

$$\mathcal{X}(S,T_i) \cap \mathcal{X}(S,T_j) = \emptyset \tag{3.19}$$

*for all* $i \neq j$. *Then,* $(G, \mathcal{C})$ *is solvable if and only if the min-cut between* $S$ *and any subset of destinations* $\mathcal{T}' \subseteq \mathcal{T}$ *is at least* $\sum_{T_i \in \mathcal{T}'} |\mathcal{X}(S,T_i)|$, i.e. $mincut(S,\mathcal{T}') \geq \sum_{T_i \in \mathcal{T}'} |\mathcal{X}(S,T_i)|$ *for any* $\mathcal{T}' \subseteq \mathcal{T}$.

*Proof:* Create a super-destination supernode $T$ with $|I(T)| = \sum_{i=1}^{K} |I(T_i)|$ and an edge $(e, e')$ from $e \in O(T_i)$, $i \in [1, K]$ to $e' \in I(T)$, as in Figure 3-13. This converts the problem of disjoint multicast to a single-source $S$, single-destination $T$ problem with rate $\mathcal{X}(S,T) = \sum_{T' \in \mathcal{T}} |\mathcal{X}(S,T)|$. The $mincut(S,T) \geq |\mathcal{X}(S,T)|$; so, Theorem 3.5.1 applies. Therefore, it is possible to achieve a communication of rate $\mathcal{X}(S,T)$ between $S$ and $T$.

Now, we have to guarantee that the receiver $T_i$ is able to receive the exact subset of processes $\mathcal{X}(S,T_i)$. Since the system matrix to $T$ is full rank, it is possible to carefully choose the encoding matrix $A$ such that the system matrix $M$ at super-destination supernode $T$ is an identity matrix. This implies that for each edge from the output ports of $T_i$ (for all $i$) to input ports of $T$ is carrying a distinct symbol, disjoint from all the other symbols carried by those edges from output ports of $T_j$, for all $i \neq j$. By appropriately permuting the symbols

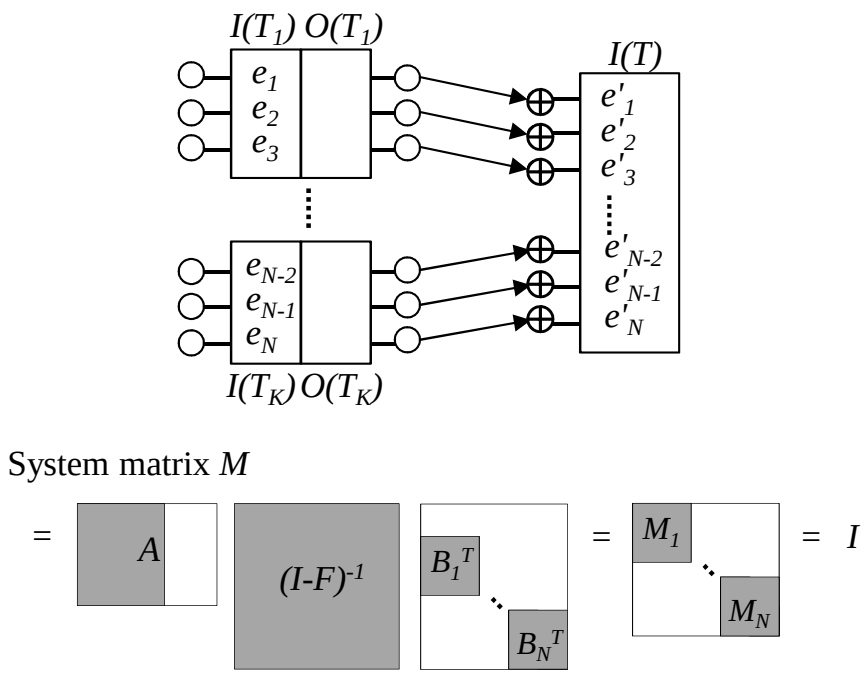



Figure 3-13: An example of the system matrix for a disjoint multicast connection. Disjoint multicast problem can be converted into a single destination problem by adding a super-destination $T$. The system matrix $M$ for the disjoint multicast problem is shown as well. Note that, unlike the multicast problem in Figure 3-12, the system matrix $M$ for the disjoint multicast is a diagonal concatenation of $M_i$'s.

at the source, $S$ can deliver the desired processes to the intended $T_i$ as shown in Figure 3-13. ■

Random linear network coding with a minor modification achieves the capacity for disjoint multicast. We note that only the source's encoding matrix $A$ needs to be modified. The intermediate supernodes can randomly and uniformly select coding coefficients $\epsilon_{(e_i,(T_j,k))}$ and $\beta_{(e_i,e_j)}$ over all elements of $\mathbb{F}_q$. Once these coding coefficients at the intermediate supernodes are selected, $S$ carefully chooses the encoding matrix $A$ such that the system matrix corresponding to the receivers of the disjoint multicast is an identity matrix, as shown in Figure 3-13. To be more precise, when $\epsilon_{(e_i,(T_j,k))}$ and $\beta_{(e_i,e_j)}$ are randomly selected over elements of $\mathbb{F}_q$, with high probability, $(I-F)^{-1}B^T$ is full rank. Therefore, there exists a matrix $A$ such that $A(I-F)^{-1}B^T$ is an identity matrix $I$. Note that $A(I-F)^{-1}B^T$ does not need to be an identity matrix. It only needs to have a diagonal structure as shown

in Figure 3-13; however, being an identity matrix is sufficient for proof of optimality.

We note another subtlety here. Theorem 3.6.3 holds precisely because we allow the intermediate nodes to code across all source processes, even if they are destined for different receivers. This takes advantage of the fact that the single source can cleverly pre-code the data.

### 3.6.3 Two-level Multicast

**Theorem 3.6.4 (Two-level Multicast Theorem)** *Given an acyclic network $G$ with a set of connections $\mathcal{C} = \mathcal{C}_d \cup \mathcal{C}_m$ where*

$$\mathcal{C}_d = \{(S, T_i, \mathcal{X}(S, T_i)) | \mathcal{X}(S, T_i) \cap \mathcal{X}(S, T_j) = \emptyset, i \neq j, i, j \in [1, K]\} \tag{3.20}$$

*is a disjoint multicast connection, and*

$$\mathcal{C}_m = \{(S, T_i, \mathcal{X}(S)) \mid i \in [K+1, N]\} \tag{3.21}$$

*is a single source multicast connection. Then, $(G, \mathcal{C})$ is solvable if and only if the min-cut between $S$ and any $\mathcal{T}' \subseteq \{T_1, \cdots, T_K\}$ is at least $\sum_{T_i \in \mathcal{T}'} |\mathcal{X}(S, T_i)|$, and the min-cut between $S$ and $T_j$ is at least $|\mathcal{X}(S)|$ for $j \in [K+1, N]$.*

*Proof:* We create a super-destination $T$ for the disjoint multicast destinations as in the proof for Theorem 3.6.3. Then, we have a single multicast problem with receivers $T$ and $\{T_i \mid i \in [K+1, N]\}$ and Theorem 3.5.3 applies. By choosing the appropriate matrix $A$, $S$ can satisfy both the disjoint multicast and the single multicast requirements, as shown in Figure 3-14. ■

As in the disjoint multicast case, random linear network coding with a minor modification at the source achieves the capacity for two-level multicast. Note that, receivers $T_i$, $i \in [K+1, N]$ are of no concern; the source $S$ can randomly choose coding coefficients $\alpha_{(i,e_j)}$ to achieve a full-rank system matrix $M_i$. Thus, $S$ needs to carefully choose the encoding matrix $A$ to satisfy the disjoint multicast constraint, which can be done as shown in Section 3.6.2.

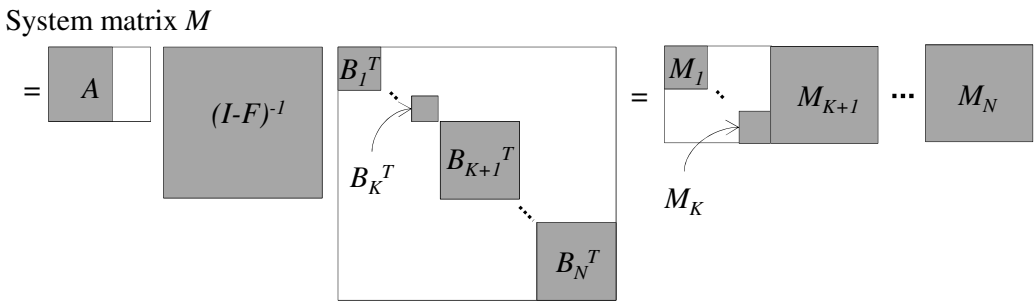



Figure 3-14: An example of the system matrix $M$ for a two-level multicast connection. The structure of the system matrix $M$ is a "concatenation" of the disjoint multicast problem (shown in Figure 3-13) and the single multicast problem (shown in Figure 3-12).

Theorem 3.6.4 does not extend to a three-level multicast. Three-level multicast, in its simplest form, consists of connections $\{(S, T_i, \mathcal{X}(S, T_i))|\ i \in [1, 3]\}$ where $\mathcal{X}(S, T_1) \subset \mathcal{X}(S, T_2) \subset \mathcal{X}(S, T_3)$.

### 3.6.4 General Connection Set

In the theorem below, we present sufficient conditions for solvability of a general connection set. This theorem does not provide necessary conditions as shown in [21].

**Theorem 3.6.5 (Generalized Min-cut Max-flow Theorem)** *Given an acyclic network $G$ with a connection set $\mathcal{C}$, let $M = \{M_{i,j}\}$ where $M_{i,j}$ is the system matrix for source processes $\mathcal{X}(S_i)$ to destination processes $\mathcal{Z}(T_j)$. Then, $(G, \mathcal{C})$ is solvable if there exists an assignment of $\alpha_{(i,e_j)}$, $\epsilon_{(e_i,(T_j,k))}$, and $\beta_{(e_i,e_j)}$ such that*

1. *$M_{i,j} = 0$ for all $(S_i, T_j, \mathcal{X}(S_i, T_j)) \notin \mathcal{C}$,*

2. *Let $(S_{\sigma(i)}, T_j, \mathcal{X}(S_{\sigma(i)}, T_j)) \in \mathcal{C}$ for $i \in [1, K(j)]$. Thus, this is the set of connections with $T_j$ as a receiver. Then, $[M^T_{\sigma(1),j}, M^T_{\sigma(2),j}, \cdots, M^T_{\sigma(K_j),j}]$ is a $|\mathcal{Z}(T_j)| \times |\mathcal{Z}(T_j)|$ is a* nonsingular *system matrix.*

*Proof:* Note that $[M^T_{\sigma(1),j}, M^T_{\sigma(2),j}, \cdots, M^T_{\sigma(K_j),j}]$ is a system matrix for source processes $\mathcal{X}(S_{\sigma(i)})$, $i \in [1, K(j)]$, to destination processes $\mathcal{Z}(T_j)$.

Condition 2 states the Min-cut Max-flow condition; therefore, Condition 2 is necessary to establish the connections. Condition 1 states that the destination supernode $T_j$ should be

able to distinguish the information it is intended to receive from the information that may have been mixed into the flow it receives. These two conditions are sufficient to establish all connections in $\mathcal{C}$. The proof is similar to that of Theorem 6 in [67]. ■

## 3.7 Network with Random Erasures

We consider the algebraic ADT problem where links may fail randomly and cause erasures. Wireless networks are stochastic in nature, and random erasures occur dynamically over time. However, the original ADT network models noise deterministically with parallel noise-free bit-pipes. As a result, the min-cut defined in Definition 3.4.1 have to be recomputed every time the network changes. Since the network codes introduced in [5][37][28] depend on the hard-coded representation of noise, these network codes also have to be recomputed every time the network changes.

We show that the algebraic framework for the ADT network is robust against random erasures and failures. First, we show that for some set of link failures, the network code remains successful. This translates to whether the system matrix $M$ preserves its full rank even after a subset of variables $\alpha_{(i,e_j)}, \epsilon_{(e_i,(D_j,k))}$, and $\beta_{(e_i,e_j)}$ associated with the failed links is set to zero. Second, we show that the specific instance of the system matrix $M$ and its rank are not as important as the *average* rank$(M)$ when computing the time average min-cut. Note that the original min-cut defined in Definition 3.4.1 requires an optimization over an exponential number of cuts for every time step to find the average min-cut. We shall use the results from [77] to show that random linear network coding achieves the time-average min-cut; thus, is capacity-achieving.

We assume that any link within the network may fail. Given an ADT network $G$ and a set of link failures $f$, we denote $G_f$ to be the network $G$ experiencing failures $f$. We also denote $B(f)$ to be the set of coding variables associated with the failing links $f$. Representing failures $f$ in the algebraic framework may be achieved by deleting the failing links from $G$, which is equivalent to setting the coding variables in $B(f)$ to zero.

We denote $M$ be the system matrix for network $G$. Let $M_f$ be the system matrix for the network $G_f$. We do not assume that the link failures are static. Therefore, we can

consider a static link failure pattern, a distribution over link failures patterns, or a sequence $f_1, f_2, f_3 \cdots$ of link failures.

### 3.7.1 Robust against Random Erasures

Given an ADT network problem $(G, \mathcal{C})$, let $\mathcal{F}$ be the set of *all* link failures such that, for any $f \in \mathcal{F}$, the problem $(G_f, \mathcal{C})$ is solvable. The solvability of a given $(G_f, \mathcal{C})$ can be verified using results in Sections 3.5 and 3.6.

We are interested in static solutions, where the network is oblivious of $f$. In other words, we are interested in finding the set of link failures such that the network code is still successful in delivering the source processes to the destinations. For a multicast connection, we show the following surprising result.

**Theorem 3.7.1 (Static Solution for Random Erasures)** *Given an ADT network problem $(G, \mathcal{C})$ with a multicast connection $\mathcal{C} = \{(S, T_1, \mathcal{X}(S)), (S, T_2, \mathcal{X}(S)), \cdots, (S, T_N, \mathcal{X}(S))\}$, there exists a* static *solution to the problem $(G_f, \mathcal{C})$ for all $f \in \mathcal{F}$. In other words, there exists a* fixed *network code that achieves the multicast rate despite any failures $f \in \mathcal{F}$.*

*Proof:* By Theorem 3.5.3, we know that for any given $f \in \mathcal{F}$, the problem $(G_f, \mathcal{C})$ is solvable. Therefore, there exists a code such that $\det(M_f) \neq 0$. Now, we need to show that there exists a code such that $\det(M_f) \neq 0$ for all $f \in \mathcal{F}$ simultaneously. This is equivalent to finding a non-zero solution to the following polynomial:

$$\prod_{f \in \mathcal{F}} \det(M_f) \neq 0. \tag{3.22}$$

Reference [41] showed that if the field size is large enough (*i.e.* $q > |\mathcal{F}||\mathcal{T}| = |\mathcal{F}|N$), then there exists an assignment of $\alpha_{(i,e_j)}, \epsilon_{(e_i,(D_j,k))}$, and $\beta_{(e_i,e_j)}$ such that $\det(M_f) \neq 0$ for all $f \in \mathcal{F}$ simultaneously. ■

**Corollary 3.7.2 (Random Coding against Random Erasures)** *Consider an ADT network problem with a multicast connection $\mathcal{C} = \{(S, T_1, \mathcal{X}(S)), (S, T_2, \mathcal{X}(S)), \cdots, (S, T_N, \mathcal{X}(S))\}$, which is solvable under link failures $f$, for all $f \in \mathcal{F}$. Then, random linear network coding, where some or all code variables $\alpha_{(i,e_j)}, \epsilon_{(e_i,(D_j,k))}$, and $\beta_{(e_i,e_j)}$ are chosen independently and*

*uniformly over all elements of $\mathbb{F}_q$ guarantees decodability at destination supernodes $T_i$ for all $i$ simultaneously and remains successful regardless of the failure pattern $f \in \mathcal{F}$ with high probability at least $(1-\frac{N|\mathcal{F}|}{q})^\eta$, where $\eta$ is the number of links carrying random combinations of the source processes.*

*Proof:* Given a multicast connection that is feasible under any link failures $f \in \mathcal{F}$, reference [41] showed that random linear network coding achieves the capacity for multicast connections, and is robust against any failures $f \in \mathcal{F}$ with high probability $(1-\frac{N|\mathcal{F}|}{q})^\eta$. ■

It is unclear whether this can be extended to the non-multicast connections, as noted in [67]. Reference [67] shows a simple example network with a non-multicast connection in which no static solution is available for a set of feasible failure patterns.

### 3.7.2 Time-average Min-cut

In this section, we study the time-average behavior of the ADT network given random erasures. We use techniques from [77], which studies reliable communication over lossy networks with network coding.

Consider an ADT network $G$. In order to study the time-average steady state behavior, we introduce erasure distributions. Let $\mathcal{F}'$ be a set of link failure patterns in $G$. A set of link failures $f \in \mathcal{F}'$ may occur with probability $p_f$.

**Theorem 3.7.3 (Min-cut for Time-varying Network)** *Assume an ADT network $G$ in which link failure pattern $f \in \mathcal{F}'$ occurs with probability $p_f$. Then, the average min-cut between two supernodes $S$ and $T$ in $G$, $mincut_{\mathcal{F}'}(S,T)$ is*

$$mincut_{\mathcal{F}'}(S,T) = \sum_{f\in\mathcal{F}'} p_f \left( \max_{\alpha_{(i,e)},\beta_{(e',e)},\epsilon_{(e',i)}} rank(M_f) \right). \tag{3.23}$$

*Proof:* By Theorem 3.4.2, we know that at any given time instance with failure pattern $f$, the min-cut between $S$ and $T$ is given by

$$\max_{\alpha_{(i,e)},\beta_{(e',e)},\epsilon_{(e',i)}} \text{rank}(M_f). \tag{3.24}$$

Then, the above statement follows naturally by taking a time average of the min-cut values between $S$ and $T$. ■

The key difference between Theorems 3.7.1 and 3.7.3 is in the effect of the link failures. In Theorem 3.7.1, any $f \in \mathcal{F}$ may change the network topology as well as min-cut; however, $mincut(S,T) \geq |\mathcal{X}(S)|$ holds for all $f \in \mathcal{F}$ (*i.e.* $(G_f, \mathcal{C})$ is assumed to be solvable for all $f \in \mathcal{F}$). However, in Theorem 3.7.3, we make no such assumption about the connection as we are evaluating the average value of the min-cut.

Unlike the case of static ADT networks, with random erasures, it may be necessary to maintain a queue at each supernode in the ADT network. This is because, if a link fails when a supernode has data to transmit on it, then it will have to wait until the link recovers. In addition, a transmitting supernode needs to be able to learn whether a packet has been received by the next hop supernode, and whether it was innovative. This can be achieved using channel estimation, feedback and/or redundancy. In the original ADT network, the issue of feedback was removed by assuming that the links are noiseless bit-pipes.

Therefore, in the remainder of this section, we assume the existence of queues and feedback in ADT networks. We present the following corollaries under these assumptions.

**Corollary 3.7.4 (Multicast in Time-varying Network)** *Consider an ADT network $G$ and a multicast connection $\mathcal{C} = \{(S, T_1, \mathcal{X}(S)), \cdots, (S, T_N, \mathcal{X}(S))\}$. Assume that failures occur where failure patten $f \in \mathcal{F}'$ occurs with probability $p_f$. Then, the multicast connection is feasible if and only if $mincut_{\mathcal{F}'}(S, T_i) \geq |\mathcal{X}(S)|$ for all $i$.*

*Proof:* Reference [77] shows that the multicast connection is feasible if and only $mincut_{\mathcal{F}'}(S, T_i) \geq |\mathcal{X}(S)|$ for all $i$. The proof in [77] relies on the fact that every supernode behaves like a stable $M/M/1$ queuing system in steady-state, and thus, the queues (or the number of innovative packets to be sent to the next hop supernode) has a finite mean if the network is run for sufficiently long period of time. ■

**Corollary 3.7.5 (Random Coding for Time-varying Network)** *Consider $(G, \mathcal{C})$ where $\mathcal{C}$ is a multicast connection. Assume failure pattern $f \in \mathcal{F}'$ occurs with probability $p_f$. Then, random linear network coding, where some or all code variables $\alpha_{(i,e_j)}, \beta_{(e_i,e_j)}, \epsilon_{(e_i,(D_j,k))}$ are*

*chosen randomly over all elements of $\mathbb{F}_q$, guarantees decodability at destination nodes $T_i$ for all $i$ simultaneously with arbitrary small error probability.*

*Proof:* This is a direct consequence of Corollary 3.7.4 and results in [41][77]. ■

## 3.8 Network with Cycles

ADT networks are acyclic, with links directed from the source supernodes to the destination supernodes. However, wireless networks intrinsically have cycles as wireless links are bi-directional by nature. In this section, we extend the ADT network model to networks with cycles. In order to incorporate cycles, we need to introduce the notion of time since, without the notion of time, the network with cycles may not be causal. To do so, we introduce delay on the links. As in [67], we model each link to have the same delay, and express the network random processes in the delay variable $D$.

We define $X_t(S,i)$ and $Z_t(T,j)$ to be the $i$-th and $j$-th binary random process generated at source $S$ and received at destination $T$ at time $t$, for $t = 1, 2, \cdots$. We define $Y_t(e)$ to be the process on edge $e$ at time $t = 1, 2, \cdots$, respectively. We express the source processes as a power series in $D$, $\mathcal{X}(S,D) = [X(S,1,D), X(S,2,D), \cdots, X(S,\mu(S),D)]$ where

$$X(S,i,D) = \sum_{t=0}^{\infty} X_t(S,i)D^t. \tag{3.25}$$

Similarly, we express the destination random processes $\mathcal{Z}(T,D) = [Z(T,1,D), \cdots, Z(T,\nu(Z),D)]$ where

$$Z(T,i,D) = \sum_{t=0}^{\infty} Z_t(T,i)D^t. \tag{3.26}$$

In addition, we express the edge random processes as

$$Y_t(e,D) = \sum_{t=0}^{\infty} Y_t(e)D^t. \tag{3.27}$$

Then, we can rewrite Equations (3.3) and (3.4) as

$$Y_{t+1}(e) = \sum_{e' \in I(V)} \beta_{(e',e)} Y_t(e') + \sum_{X_t(S,i) \in \mathcal{X}(S)} \alpha_{(i,e)} X_t(S,i). \tag{3.28}$$

Furthermore, the output processes $Z_t(T,i)$ can be rewritten as

$$Z_{t+1}(T,i) = \sum_{e' \in I(T)} \epsilon_{e',(T,i)} Y_t(e'). \tag{3.29}$$

Using this formulation, we can extend the results from [67] to ADT networks with cycles. We show that a system matrix $M(D)$ captures the input-output relationships of the ADT networks with delay and/or cycles.

**Theorem 3.8.1** *Given a network $G = (\mathcal{V}, \mathcal{E})$, let $A(D)$, $B(D)$, and $F$ be the encoding, decoding, and adjacency matrices, as defined here:*

$$A_{i,j} = \begin{cases} \alpha_{(i,e_j)}(D) & \text{if } e_j \in O(S) \text{ and } X(S,i) \in \mathcal{X}(S), \\ 0 & \text{otherwise.} \end{cases} \tag{3.30}$$

$$B_{i,(T_j,k)} = \begin{cases} \epsilon_{(e_i,(T_j,k))}(D) & \text{if } e_i \in I(T_j),\ Z(T_j,k) \in \mathcal{Z}(T_j), \\ 0 & \text{otherwise.} \end{cases} \tag{3.31}$$

*and $F$ as in Equation (3.8). The variables $\alpha_{(i,e_j)}(D)$ and $\epsilon_{(e_i,(T_j,k))}(D)$ can either be constants or rational functions in $D$. Then, the system matrix of the ADT network with delay, which now may include cycles, is given by*

$$M(D) = A(D) \cdot (I - DF)^{-1} \cdot B(D)^T. \tag{3.32}$$

*Proof:* The proof for this is similar to that of Theorem 3.3.5. ■

Similar to Section 3.3, $(I - DF)^{-1}$ represents the impulse response of the network with delay. This is because the series $I + DF + D^2F^2 + D^3F^3 + \cdots$ represents the connectivity of the network while taking delay into account. For example, $F^k$ has a non-zero entry if there exists a path of length $k$ between two ports. Now, since we want to represent the time associated with traversing from port $e_i$ to $e_j$, we use $D^kF^k$, where $D^k$ signifies that the path is of length $k$. Thus, $(I - DF)^{-1} = I + DF + D^2F^2 + D^3F^3 + \cdots$ is the impulse response of the network with delay. An example of $(I - DF)^{-1}$ for the example network in Figure 3-7 is shown in Figure 3-15.

$$\begin{pmatrix}
1 & 0 & D & 0 & 0 & D & D^2\beta_{(e_3,e_7)} & 0 & D^2\beta_{(e_6,e_9)} & D^2\beta_{(e_6,e_{10})} & D^3\beta_{(e_6,e_9)} & D^3\beta_{(e_3,e_7)} + D^3\beta_{(e_6,e_{10})} \\
0 & 1 & 0 & D & 0 & 0 & D^2\beta_{(e_4,e_7)} & 0 & 0 & 0 & 0 & D^3\beta_{(e_4,e_7)} \\
0 & 0 & 1 & 0 & 0 & 0 & D\beta_{(e_3,e_7)} & 0 & 0 & 0 & 0 & D^2\beta_{(e_3,e_7)} \\
0 & 0 & 0 & 1 & 0 & 0 & D\beta_{(e_4,e_7)} & 0 & 0 & 0 & 0 & D^2\beta_{(e_4,e_7)} \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & D\beta_{(e_6,e_9)} & D\beta_{(e_6,e_{10})} & D^2\beta_{(e_6,e_9)} & D^2\beta_{(e_6,e_{10})} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & D \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & D & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & D \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Figure 3-15: The $12 \times 12$ matrix $(I - DF)^{-1}$ for the example network in Figure 3-4. The matrix $F$ can be found in Figure 3-11.

Using the system matrix $M(D)$ from Theorem 3.8.1, we can extend Theorem 3.5.1, Theorem 3.5.3, Theorem 3.6.1, Theorem 3.6.3, and Theorem 3.6.4 to ADT networks with cycles and/or delay. However, there is a minor technical change. We now operate in a different field. Instead of having coding coefficients from the finite field $\mathbb{F}_q$, the coding coefficients $\alpha_{(i,e_j)}(D)$ and $\epsilon_{((e_i,(T_j,k)))}(D)$ are now from $\mathbb{F}_q(D)$, the field of rational functions of $D$. We shall not discuss the proofs in detail; however, this is a direct application of the results in [67].

## 3.9 Conclusions

ADT networks [6][7] have drawn considerable attention for their potential to approximate the capacity of wireless relay networks. ADT networks take into account the wireless interference among users, but not the ambient noise. Thus, ADT networks approximate wireless networks in high SNR regime where interference is the dominating factor. In this chapter, we showed that the ADT network can be described well within the algebraic network coding framework [67]. This connection between ADT network and algebraic network coding allows the use of results on network coding to understand better the ADT networks.

In this chapter, we derived an algebraic definition of min-cut for the ADT networks, and provided an algebraic interpretation of the Min-cut Max-flow theorem for a single unicast or a multicast connection in ADT networks. Furthermore, by taking advantage of the algebraic structure, we have shown feasibility conditions for a variety of sets of connections $\mathcal{C}$, such as

multiple multicast, disjoint multicast, and two-level multicast. Furthermore, we extended the capacity characterization to networks with cycles, random erasures, and failures. We proved the optimality of linear operations for multicast connections in ADT networks with cycles.

We further showed that a randomized, distributed network code can achieve capacity in ADT networks. We first showed optimality of linear operations for the connections listed above in the ADT networks, and then showed that random linear network coding achieves the capacity. By incorporating random erasures into the ADT network model, we showed that random linear network coding is robust against failures and erasures.

# Chapter 4

# TCP/NC:

## Network Coding for Erasure Networks

The Transmission Control Protocol (TCP) is one of the core protocols of today's Internet Protocol Suite. Many Internet applications, such as the World Wide Web, email, file transfer, peer-to-peer file sharing, and multimedia streaming, rely on TCP's promise to correctly deliver the data stream without losses or duplications. Furthermore, TCP plays an essential role in end-to-end flow control and congestion control, which are important in enabling a diverse set of devices to share the network resources. TCP uses several mechanisms to detect congestion in the network, and controls its rate to avoid congestion collapse and allow fair sharing of network resources among many flows.

TCP was designed for reliable transmission over wired networks, in which losses are generally an indication of congestion. This is not the case in wireless networks, where losses are often due to fading, interference, and other physical phenomena. In wireless networks, TCP often incorrectly assumes that there is congestion within the network and unnecessarily reduces its transmission rate, when it should have actually transmitted continuously to overcome the lossy links. Consequently, TCP's performance in wireless networks is poor when compared to wired counterparts as shown *e.g.* in [13][81].

There has been extensive research to combat these harmful effects of erasures and failures; however, TCP even with modifications does not achieve significant improvement. For example, there have been suggestions to allow the TCP sender to maintain a large trans-

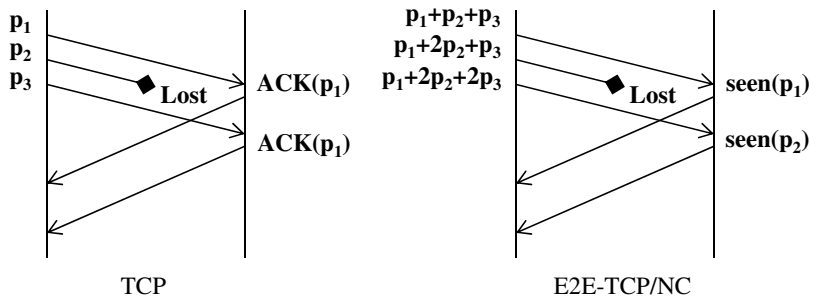



Figure 4-1: An example of TCP's and TCP/NC's behavior in lossy networks. In the case of TCP, the TCP sender receives duplicate ACKs for packet $\mathbf{p_1}$, which may wrongly indicate congestion. However, for TCP/NC, the TCP/NC sender receives ACKs for packets $\mathbf{p_1}$ and $\mathbf{p_2}$; thus, the TCP/NC sender perceives a longer round-trip time (RTT) but does not mistake the loss to be congestion.

mission window to overcome the random losses within the network. However, as we shall show in this chapter, just naïvely keeping the window open does not lead to improvements in TCP's performance. Even if the transmission window is kept open, the sender can not transmit additional packets into the network without receiving acknowledgments. Eventually, this leads to TCP's performance degradation. References [9][93] give an overview and a comparison of various TCP versions over wireless links.

Some relief may come from network coding. In order to combine the benefits of TCP and network coding, reference [91] proposes a new protocol called TCP/NC. TCP/NC modifies TCP's acknowledgment (ACK) scheme such that it acknowledges *degrees of freedom* instead of individual packets, as shown in Figure 4-1. This is done so by using the concept of "seen" packets, in which the number of degrees of freedom received is translated to the number of consecutive packets received.

In this chapter, we present a performance evaluation of TCP as well as TCP/NC in lossy networks. We adopt the same TCP model as in [81], *i.e.* we consider standard TCP with Go-Back-N pipelining. Therefore, the standard TCP discards packets that are out-of-order. We analytically show the throughput gains of TCP/NC over standard TCP, and present simulations results that support this analysis. We characterize the steady state throughput behavior of both TCP and TCP/NC as a function of erasure rate, round-trip time (RTT), and maximum window size. Our work thus extends the work of [81] for TCP

and TCP/NC in lossy wireless networks. Furthermore, we use NS-2 (Network Simulator [1]) to verify our analytical results for TCP and TCP/NC. Our analysis and simulation results show very close concordance and support that TCP/NC is robust against erasures and failures. TCP/NC is not only able to increase its window size faster but also maintain a large window size despite the random losses, whereas TCP experiences window closing because losses are mistakenly attributed to congestion. Note that network coding only masks random erasures, and allows TCP/NC to react to congestion; as a result, when there are correlated losses, TCP/NC also closes its window. Therefore, TCP/NC is well suited for reliable communication in lossy networks.

TCP/NC's key achievement is in recognizing the stochastic nature of wireless channels, and adapting the transport protocol to not overreact to random losses in wireless networks. This is done by inserting a coding shim between TCP and IP layers, allowing the use of network coding to mask random losses from the congestion control mechanisms and significantly improve throughput in wireless. This chapter studies how this network coding layer interacts with TCP's congestion control mechanism, and the various effects parameters such as redundancy factor and round-trip time have on the performance of TCP/NC.

There has been extensive research on modeling and analyzing TCP's performance [76][75][4][16][33][74]. Our goal is to present an analysis for TCP/NC, and to provide a comparison of TCP and TCP/NC in a lossy wireless environment. We adopt Padhye *et al.*'s model [81] as their model provides a simple yet good model to predict the performance of TCP. It would be interesting to extend and analyze TCP/NC in other TCP models in the literature.

The chapter is organized as follows. In Section 4.1, we provide a brief overview of TCP/NC. In Section 4.2, we introduce our communication model. In Section 4.3, we provide the intuition behind the benefit of using network coding with TCP. Then, we provide throughput analysis for TCP and TCP/NC in Sections 4.4 and 4.5, respectively. In Section 4.6, we provide simulation results to verify our analytical results in Sections 4.4 and 4.5. Finally, we conclude in Section 4.7.

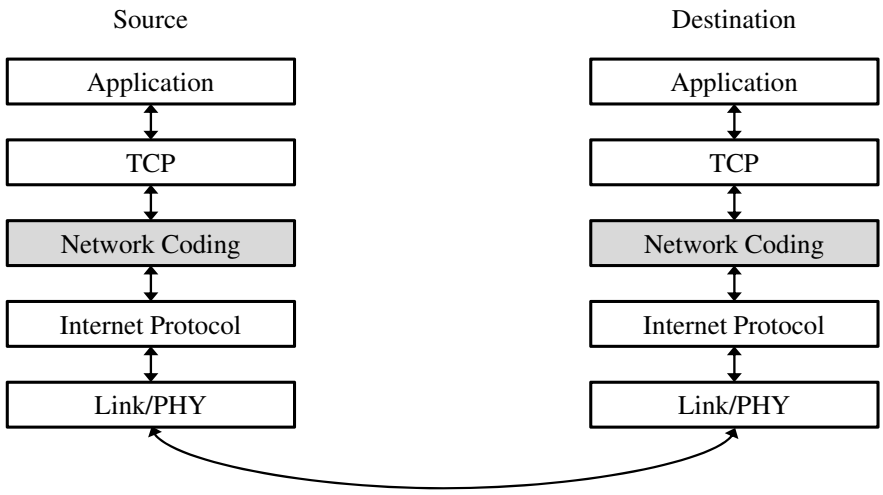



Figure 4-2: The new network protocol stack with network coding layer. The network coding layer sits between the transport layer and Internet Protocol layer. This new layer introduces coding, which enables robustness against erasures.

## 4.1 Overview of TCP/NC

Reference [91] introduces a new *network coding* layer between the TCP and IP in the protocol stack as shown in Figure 4-2. The network coding layer intercepts and modifies TCP's acknowledgment (ACK) scheme such that random erasures do not affect the transport layer's performance. To do so, the *encoder*, the network coding unit under the sender TCP, transmits $R$ random linear combinations of the buffered packets for every transmitted packet from TCP sender. The parameter $R$ is the *redundancy factor*. Redundancy factor helps TCP/NC to recover from random losses; however, it cannot mask correlated losses, which are usually due to congestion. The *decoder*, the network coding unit under the receiver TCP, acknowledges *degrees of freedom* instead of individual packets, as shown in Figure 4-1. This uses the concept of "seen" packets, in which the number of degrees of freedom received is translated to the number of consecutive packets received. Once enough degrees of freedoms are received at the decoder, the decoder solves the set of linear equations to decode the original data transmitted by the TCP sender, and delivers the data to the TCP receiver.

We briefly discuss the overhead associated with network coding. The main overhead

associated with network coding can be considered in two parts: 1) the coding vector (or the coding coefficients) that has to be included in the header; 2) the encoding and the decoding complexity. For receiver to decode a network coded packet, the packet needs to indicate the coding coefficients used to generate the linear combination of the original data packets. The overhead associated with the coefficients depend on the field size used for coding as well as the number of original packets combined. It has been shown that even a very small field size of $\mathbb{F}_{256}$ (*i.e.* 8 bits = 1 byte per coefficient) can provide a good performance [91][89]. Therefore, even if we combine 50 original packets, the coding coefficients amount to 50 bytes overall. Since a packet is typically around 1500 bytes, the overhead associated with coding vector is not substantial.

The second overhead associated with network coding is the encoding and the decoding complexity, and the delay associated with such coding operations. However, to affect TCP's performance, the decoding and the encoding operations must be substantial enough to affect the round-trip time estimate of the TCP sender and receiver. We note that the delay caused the coding operations is negligible compared to the network round-trip time. For example, the network round-trip time is often in milliseconds (if not in hundreds of milliseconds). On the other hand, the encoding or decoding operations involve a matrix multiplication and inversion in $\mathbb{F}_{256}$, which can be performed in a few microseconds.

In [91], the authors present two versions of TCP/NC – one that adheres to the end-to-end philosophy of TCP, in which coding operations are only performed at the source and destination; another that takes advantage of network coding even further by allowing any subset of intermediate nodes to re-encode. Re-encoding at the intermediate nodes is an optional feature, and is not required for TCP/NC to work. Here, we focus on TCP/NC with end-to-end network coding, which we denote E2E-TCP/NC or in short E2E. However, a similar analysis applies to TCP/NC with re-encoding.

## 4.2 A Model for Congestion Control

We focus on TCP's congestion avoidance mechanism, where the congestion control window size $W$ is incremented by $1/W$ each time an ACK is received. Therefore, when every packet

in the congestion control window is ACKed, the window size $W$ is increased to $W + 1$. On the other hand, the window size $W$ is reduced whenever an erasure or congestion is detected.

We model TCP's behavior in terms of *rounds* as in [81]. We denote $W_i$ to be the size of TCP's congestion control window size at the beginning of round $i$. The sender transmit $W_i$ packets in its congestion window at the start of round $i$, and once all $W_i$ packets have been sent, it defers transmitting any other packets until at least one ACK for the $W_i$ packets are received. The ACK reception ends the current round, and starts round $i + 1$.

For simplicity, we assume that the duration of each round is equal to a round trip time ($RTT$), independent of $W_i$. This assumes that the time needed to transmit a packet is much smaller than the round trip time. This implies the following sequence of events for each round $i$:

1. $W_i$ packets are transmitted. Some packets may be lost.

2. The receiver transmits ACKs for the received packets. TCP uses cumulative ACKs. Therefore, if the packets $1, 2, 3, 5, 6$ arrive at the receiver in sequence, then the receiver ACKs packets $1, 2, 3, 3, 3$. This signals that it has not yet received packet 4. Some of the ACKs may also be lost.

3. Once the sender receives the ACKs, it updates its window size. Assume that $a_i$ packets are acknowledged in round $i$. Then,

$$W_{i+1} \leftarrow W_i + a_i/W_i. \tag{4.1}$$

   Once the sender receives any ACK for the packets transmitted in the beginning of the round, the current round terminates and the next round starts.

TCP reduces the window size for congestion control using the following two methods.

1. *Triple-duplicate (TD):* When the sender receives four ACKs with the same sequence number, then

$$W_{i+1} \leftarrow \frac{1}{2} W_i. \tag{4.2}$$

2. *Time-out (TO):* If the sender does not hear from the receiver for a predefined time period, called the "time-out" period (which is $T_o$ rounds long), then the sender closes its transmission window,

$$W_{i+1} \leftarrow 1. \tag{4.3}$$

At this point, the sender updates its TO period to $2T_o$ rounds and transmits one packet. For any subsequent TO events, the sender transmits the one packet within its window, and doubles its TO period until $64T_o$ is reached, after which the TO period is fixed to $64T_o$. Once the sender receives an ACK from the receiver, it resets its TO period to $T_o$ and increments its window according to the congestion avoidance mechanism. During time-out, the throughput of both TCP and E2E-TCP/NC is zero.

Finally, in practice, the TCP receiver sends a single cumulative ACK after receiving $\beta$ number of packets, where $\beta = 2$ typically. However, we assume that $\beta = 1$ for simplicity. Extending the analysis to $\beta \geq 1$ is straightforward.

There are several variants to the traditional TCP congestion control. For example, STCP [53], Compound TCP [92], and CUBIC TCP [38] modify the congestion control mechanism for networks with high bandwidth-delay products, such as the high speed wide area networks. Other variants of TCP include those with selective acknowledgment schemes (SACK) [31]. It may be interesting to compare the performance of the TCP variants with that of TCP/NC. However, we focus on traditional TCP here.

### 4.2.1 Maximum Window Size

In general, TCP cannot increase its window size unboundedly; there is a maximum window size $W_{\max}$. The TCP sender uses a congestion avoidance mechanism to increment the window size until $W_{\max}$, at which the window size remains $W_{\max}$ until a TD or a TO event.

### 4.2.2 Erasures

We assume that there are random erasures within the network. We denote $p$ to be the probability that a packet is lost at any given time. We assume that packet losses are independent. Our erasure model is different from that of [81] where losses are correlated within

a round, *i.e.* bursty erasures. Correlated erasures model well bursty traffic and congestion in wired networks. In our case, however, we are aiming to model wireless networks, thus we shall use random independent erasures.

We do not model congestion or correlated losses within this framework, but show by simulation that when there are correlated losses, both TCP and E2E-TCP/NC close their window; thus, E2E-TCP/NC is able to react to congestion.

### 4.2.3 Performance Metric

We analyze the performance of TCP and E2E-TCP/NC in terms of two metrics: the average throughput $\mathcal{T}$ and the expected window evolution $E[W]$, where $\mathcal{T}$ represents the total average throughput while window evolution $E[W]$ reflects the perceived throughput at a given time. We define $\mathcal{N}_{[t_1,t_2]}$ to be the number of packets received by the receiver during the interval $[t_1, t_2]$. The total average throughput is defined as:

$$\mathcal{T} = \lim_{\Delta \to \infty} \frac{\mathcal{N}_{[t,t+\Delta]}}{\Delta}. \tag{4.4}$$

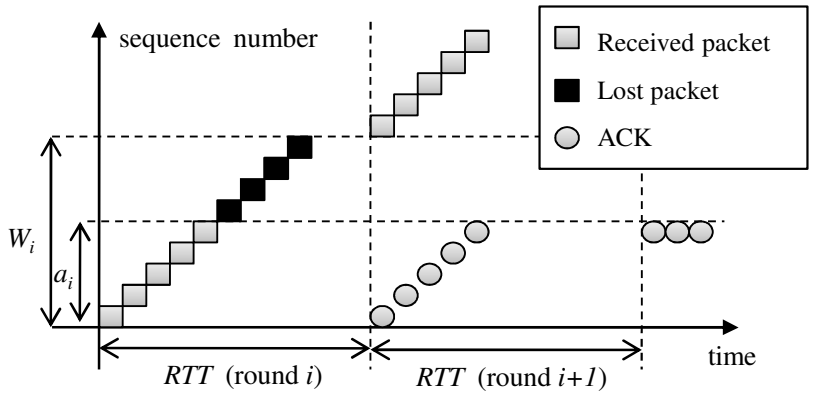
We denote $\mathcal{T}_{tcp}$ and $\mathcal{T}_{e2e}$ to be the average throughput for TCP and E2E-TCP/NC, respectively.

## 4.3 Intuition

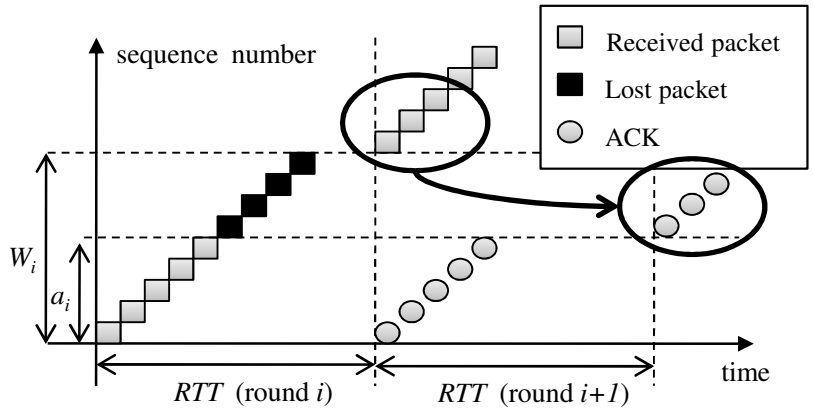
For traditional TCP, random erasures in the network can lead to triple-duplicate ACKs. For example, in Figure 4-3a, the sender transmits $W_i$ packets in round $i$; however, only $a_i$ of them arrive at the receiver. As a result, the receiver ACKs the $a_i$ packets and waits for packet $a_i + 1$. When the sender receives the ACKs, round $i + 1$ starts. The sender updates its window

$$W_{i+1} \leftarrow W_i + \frac{a_i}{W_i}, \tag{4.5}$$

and starts transmitting the new packets in the window. However, since the receiver is still waiting for packet $a_i + 1$, any other packets cause the receiver to request for packet $a_i + 1$.




(a) TCP




(b) E2E-TCP/NC

Figure 4-3: The effect of erasures on TCP and TCP/NC. TCP experiences triple-duplicate ACKs, and results in $W_{i+2} \leftarrow W_{i+1}/2$. However, E2E-TCP/NC masks the erasures using network coding, which allows E2E-TCP/NC to advance its window. This figure depicts the sender's perspective, therefore, it indicates the time at which the sender transmits the packet or receives the ACK.

This results in a triple-duplicate ACKs event and the TCP sender closes its window:

$$W_{i+2} \leftarrow \frac{1}{2}W_{i+1} = \frac{1}{2}(W_i + a_i/W_i). \tag{4.6}$$

Notice that, as illustrated in Figure 4-3b, this window closing due to TD does not occur when using E2E-TCP/NC. With network coding, any linearly independent packet delivers new information. Any subsequent packet (in Figure 4-3b, the first packet sent in round $i+1$) can be viewed as packet $a_i + 1$. As a result, the receiver is able to increment its ACK and the sender continues transmitting data. It follows that network coding masks the losses within the network from TCP, and prevents it from closing its window by misjudging link losses as congestion. ***Network coding translates random losses as longer RTT***, thus slowing down the transmission rate to adjust for losses without closing down the window in a drastic fashion.

Network coding does not mask correlated (or bursty) losses due to congestion. Network coding is able to translate random losses as longer RTT; however, with enough correlated losses, network coding cannot correct for all the losses present in the network. As a result, E2E-TCP/NC's transmission rate will be adjusted according to standard TCP's congestion control mechanism when E2E-TCP/NC detects correlated losses. As a result, network coding allows TCP to maintain a high throughput connection in a lossy environment; at the same time, allows TCP to react to congestion. Therefore, network coding naturally distinguishes congestion from random losses.

## 4.4 Throughput Analysis for TCP

We consider the effect of losses for TCP. The throughput analysis for TCP is similar to that of [81]. However, the model has been modified from that of [81] to account for independent losses and allow a fair comparison with network coded TCP. TCP can experience a TD or a TO event from random losses. As in [81], we first consider TD events, and then incorporate TO events.

We note that, despite independent packet erasures, a single packet loss may affect subsequent packet reception. This is due to the fact that TCP requires in-order reception. A

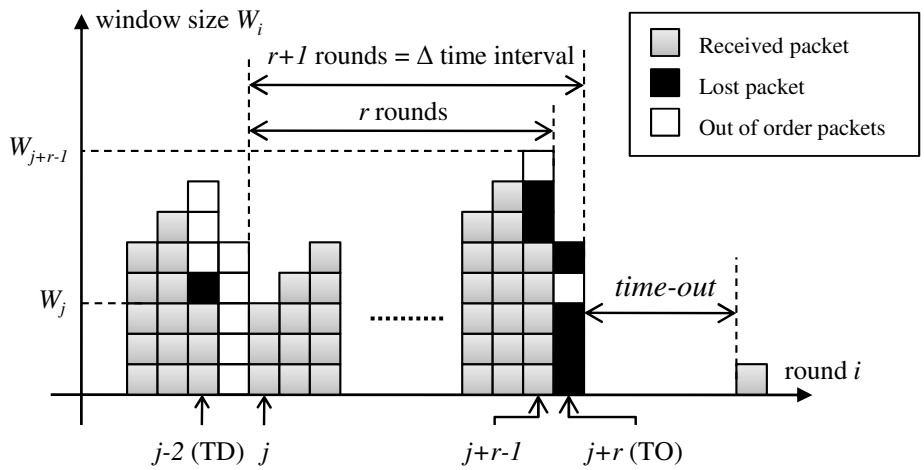



Figure 4-4: The evolution of TCP's window size when losses occur. Losses result in TD events or TO events, which decreases TCP's window size significantly. In round $j-2$, losses occur resulting in triple-duplicate ACKs. On the other hand, in round $j+r-1$, losses occur; however, in the following round $j+r$ losses occur such that the TCP sender only receives two-duplicate ACKs. As a result, TCP experiences time-out.

single packet loss within a transmission window forces all subsequent packets in the window to be out of order. Therefore, they are discarded by the TCP receiver. As a result, standard TCP's throughput behavior with independent losses is similar to that of [81], where losses are correlated within one round.

### 4.4.1 Triple-duplicate for TCP

We consider the expected throughput between consecutive TD events, as shown in Figure 4-4. Assume that the TD events occurred at time $t_1$ and $t_2 = t_1 + \Delta$, $\Delta > 0$. Assume that round $j$ begins immediately after time $t_1$, and that packet loss occurs in the $r$-th round, *i.e.* round $j + r - 1$.

First, we calculate $E[\mathcal{N}_{[t_1,t_2]}]$. During the time interval $[t_1, t_2]$, there are no packet losses. Given that the probability of a packet loss is $p$, the expected number of consecutive packets that are successfully sent from sender to receiver is

$$E\left[\mathcal{N}_{[t_1,t_2]}\right] = \left(\sum_{k=1}^{\infty} k(1-p)^{k-1}p\right) - 1 = \frac{1-p}{p}. \tag{4.7}$$

The packets (in white in Figure 4-4) sent after the lost packets (in black in Figure 4-4) are out of order, and will not be accepted by the standard TCP receiver. Thus, Equation (4.7) does not take into account the packets sent in round $j-1$ or $j+r$.

We calculate the expected time period between two TD events, $E[\Delta]$. As in Figure 4-4, after the packet losses in round $j$, there is an additional round for the loss feedback from the receiver to reach the sender. Therefore, there are $r+1$ rounds within the time interval $[t_1, t_2]$, and $\Delta = RTT(r+1)$. Thus,

$$E[\Delta] = RTT(E[r]+1). \tag{4.8}$$

We now derive $E[r]$. To derive $E[r]$, we note that

$$W_{j+r-1} = W_j + r - 1, \tag{4.9}$$

assuming that there are no loss events between round $j+r-1$ and $j$. This is because, at every round, the window is incremented by one when all packets are acknowledged, as discussed in Section 4.2.

When there are losses, however, TCP experiences a TD event and closes it's window. Following the Figure 4-4, we assume that losses occur before round $j$. Then,

$$W_j = \frac{1}{2}W_{j-1} = \frac{1}{2}\left(W_{j-2} + \frac{a_{j-2}}{W_{j-2}}\right). \tag{4.10}$$

Equation (4.10) is due to TCP's congestion control. TCP interprets the losses in round $j-2$ as congestion, and as a result halves its window.

Assuming that, in the long run, $E[W_{j+r-1}] = E[W_{j-2}]$ (*i.e.* the window size before a loss event is expected to be the same) and that $a_{j-2}$ is uniformly distributed between $[0, W_{j-2}]$, we can derive an expression for $E[W_{j+r-1}]$ and $E[W_j]$ in terms of $E[r]$.

$$W_{j+r-1} = W_j + r - 1 \tag{4.11}$$

$$= \frac{1}{2}\left(W_{j-2} + \frac{a_{j-2}}{W_{j-2}}\right) + r - 1. \tag{4.12}$$

Taking the expectation,

$$E[W_{j+r-1}] = \frac{1}{2}\left(E[W_{j-2}] + \frac{E[a_{j-2}]}{E[W_{j-2}]}\right) + E[r] - 1. \tag{4.13}$$

Using the assumption that $E[W_{j+r-1}] = E[W_{j-2}]$ and that $E[a_{j-2}] = \frac{1}{2}E[W_{j-2}]$, we can derive that

$$E[W_{j+r-1}] = 2\left(E[r] - \frac{3}{4}\right) \tag{4.14}$$

and

$$E[W_j] = E[r] - \frac{1}{2}. \tag{4.15}$$

During these $r$ rounds, we expect to successfully transmit $\frac{1-p}{p}$ packets as noted in Equation (4.7). This results in:

$$\frac{1-p}{p} = \sum_{k=0}^{r-1} a_{j+k} \tag{4.16}$$

$$= \left(\sum_{k=0}^{r-2} W_{j+k}\right) + a_{j+r-1} \tag{4.17}$$

$$= \sum_{k=0}^{r-2} (W_j + k) + a_{j+r-1} \tag{4.18}$$

$$= (r-1)W_j + \frac{(r-1)(r-2)}{2} + a_{j+r-1}. \tag{4.19}$$

Taking the expectation of Equation (4.19) and using Equation (4.15),

$$\frac{1-p}{p} = (E[r]-1)\,E[W_j] + \frac{(E[r]-1)(E[r]-2)}{2} + E[a_{j+r-1}] \tag{4.20}$$

$$= (E[r]-1)\left(E[r] - \frac{1}{2}\right) + \frac{(E[r]-1)(E[r]-2)}{2} + E[a_{j+r-1}] \tag{4.21}$$

$$= \frac{3}{2}(E[r]-1)^2 + E[a_{j+r-1}]. \tag{4.22}$$

Note that $a_{j+r-1}$ is assumed to be uniformly distributed across $[0, W_{j+r-1}]$. Therefore, by Equation (4.14),

$$E[a_{j+r-1}] = \frac{E[W_{j+r-1}]}{2} = E[r] - \frac{3}{4}. \tag{4.23}$$

Finally, by substituting $E[a_{j+r-1}]$ in Equation (4.22), we solve for $E[r]$. Solving the quadratic equation (Equation (4.22)) for $E[r]$, we find:

$$E[r] = \frac{2}{3} + \sqrt{-\frac{1}{18} + \frac{2}{3}\frac{1-p}{p}}. \tag{4.24}$$

Note that $E[r] \approx \sqrt{\frac{2}{3p}} + o\left(\frac{1}{\sqrt{p}}\right)$ for small $p$. For large $p$, $E[r] < 1$. This is consistent with our intuition that for $p \to 1$, we expect the erasures to occur every round, resulting in $E[r] < 1$.

The steady state average window size $E[W]$ is the average window size over two consecutive TD events. This provides an expression of steady state average window size for TCP. We use Equations (4.14) and (4.15) to find the average window size between two consecutive TD events, $E[W]$:

$$E[W] = \frac{E[W_j] + E[W_{j+r-1}]}{2} = \frac{3}{2}E[r] - 1. \tag{4.25}$$

Then, the average throughput can be expressed as

$$\mathcal{T}'_{tcp} = \frac{E[\mathcal{N}_{[t_1,t_2]}]}{E[\Delta]} = \frac{1-p}{p}\frac{1}{RTT(E[r]+1)}. \tag{4.26}$$

Therefore, if we only consider TD events as loss indications, the long-term steady state throughput is equal to that in Equation (4.26).

The analysis above assumes that the window size can grow unboundedly; however, this is not the case. To take maximum window size $W_{\max}$ into account, we make a following approximation:

$$\mathcal{T}_{tcp} = \min\left(\frac{W_{\max}}{RTT}, \mathcal{T}'_{tcp}\right). \tag{4.27}$$

We briefly discuss the behavior of TCP as described by Equation (4.26). For small $p$,

$$\mathcal{T}'_{tcp} \approx \frac{1}{RTT}\sqrt{\frac{3}{2p}} + o(\frac{1}{\sqrt{p}}); \tag{4.28}$$

and for large $p$,

$$\mathcal{T}'_{tcp} \approx \frac{1}{RTT}\frac{1-p}{p}. \tag{4.29}$$

For small $p$, this result in Equation (4.28) coincides with the results in [81]. This shows that TCP is not well-suited for wireless networks. In [81], erasures were correlated as they were modeling congestion. Here, we assume erasures are independent, modeling random losses in wireless networks. However, TCP's throughput behavior is the same in both cases as captured by Equation (4.28).

### 4.4.2 Time-out for TCP

If there are enough losses within two consecutive rounds, TCP may experience a TO event as shown in Figure 4-4. A TO event occurs if two consecutive rounds have losses with two or fewer out-of-order packets transmitted in the latter round. Therefore, $\mathbf{P}(\text{TO}|W)$, the probability of a TO event given a window size of $W$, is given by

$$\mathbf{P}(\text{TO}|W) = \begin{cases} 1 & \text{if } W < 3; \\ \sum_{i=0}^{2} \binom{W}{i} p^{W-i}(1-p)^i & \text{if } W \geq 3. \end{cases} \tag{4.30}$$

When the window is small ($W < 3$), then losses result in TO events. For example, assume $W = 2$ with packets $\mathbf{p_1}$ and $\mathbf{p_2}$ in its window. Assume that $\mathbf{p_2}$ is lost. Then, the TCP sender may send another packet $\mathbf{p_3}$ in the subsequent round since the acknowledgment for $\mathbf{p_1}$ allows it to transmit a new packet. However, this would generate a single duplicate ACK with no further packets in the pipeline, and TCP sender waits for ACKs until it times out.

We approximate $W$ in above Equation (4.30) with the expected window size $E[W]$ from Equation (4.25). Therefore, for the remaining of this chapter, we estimate the probability of time out to be

$$\mathbf{P}(\text{TO} \mid E[W]) = \mathbf{P}\left(\text{TO} \;\middle|\; \frac{3}{2}E[r] - 1\right). \tag{4.31}$$

The length of the TO event depends on the duration of the loss events. We denote the duration of a TO event as $r_{TO}$ rounds. Therefore, the expected duration of TO period,

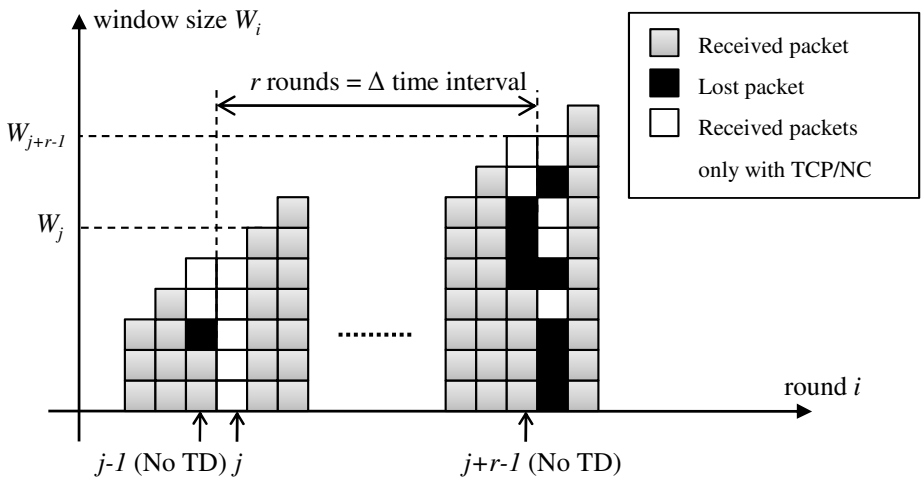



Figure 4-5: The evolution of E2E-TCP/NC's window size when losses occur. E2E-TCP/NC's window size does not decrease with erasures, which would have led to a triple-duplicate ACKs event when using standard TCP. Note that unlike TCP, the window size is non-decreasing.

$E[r_{\text{TO event}}]$, is given as:

$$E[r_{\text{TO event}}] \tag{4.32}$$

$$= (1-p)\left[T_o p + 3T_o p^2 + 7T_o p^3 + 15T_o p^4 + 31T_o p^5 + \sum_{i=0}^{\infty}(63 + i \cdot 64)T_o p^{6+i}\right] \tag{4.33}$$

$$= (1-p)\left[T_o p + 3T_o p^2 + 7T_o p^3 + 15T_o p^4 + 31T_o p^5 + 63T_o \frac{p^6}{1-p} + 64T_o \frac{p^7}{(1-p)^2}\right]. \tag{4.34}$$

Finally, by combining the results in Equations (4.27), (4.30), and (4.34), we get an expression for the average throughput of TCP, $\mathcal{T}_{tcp}$, as:

$$\mathcal{T}_{tcp} = \min\left(\frac{W_{\max}}{RTT}, \frac{1-p}{p}\frac{1}{RTT\,(1 + E[r] + \mathbf{P}(\text{TO}|E[W])E[r_{\text{TO event}}])}\right). \tag{4.35}$$

## 4.5 Throughput Analysis for E2E-TCP/NC

We consider the expected throughput for E2E-TCP/NC. The erasure patterns that result in TD event or TO event under TCP may not yield the same result under E2E-TCP/NC, as illustrated in Section 4.3. We emphasize again that this is due to the fact that any linearly

independent packet conveys a new degree of freedom to the receiver. Figure 4-5 illustrates this effect. The packets (in white) sent after the lost packets (in black) are acknowledged by the receivers, thus allowing E2E-TCP/NC to advance its window. This implies that E2E-TCP/NC does not experience window closing owing to random losses often.

### 4.5.1 E2E-TCP/NC Window Evolution

From Figure 4-5, we observe that E2E-TCP/NC is able to maintain its window size despite experiencing losses. This is partially because E2E-TCP/NC receiver is able to receive packets that would be considered out of order by TCP receiver. As a result, E2E-TCP/NC's window evolves differently from that of TCP, and can be characterized by a simple recursive relationship as

$$E[W_i] = E[W_{i-1}] + \frac{E[a_{i-1}]}{E[W_{i-1}]} = E[W_{i-1}] + \min\{1, R(1-p)\}. \tag{4.36}$$

The recursive relationship captures the fact that every packet that is linearly independent of previously received packets is considered to be *innovative* and is therefore acknowledged. Consequently, any arrival at the receiver is acknowledged with high probability; therefore, we expect $E[a_{i-1}]$ packets to be acknowledged and the window to be incremented by $\frac{E[a_{i-1}]}{E[W_{i-1}]}$. Note that $E[a_{i-1}] = (1-p) \cdot R \cdot E[W_{i-1}]$ since the encoder transmits on average $R$ linear combinations for every packet transmitted by the TCP sender.

Once we take the maximum window size $W_{\max}$ into account, we have the following expression for E2E-TCP/NC's expected window size:

$$E[W_i] = \min(W_{\max}, E[W_1] + i \min\{1, R(1-p)\}), \tag{4.37}$$

where $i$ is the round number. $E[W_1]$ is the initial window size, and we set $E[W_1] = W_1 = 1$. Figure 4-6 shows an example of the evolution of the E2E-TCP/NC window using Equation (4.37).

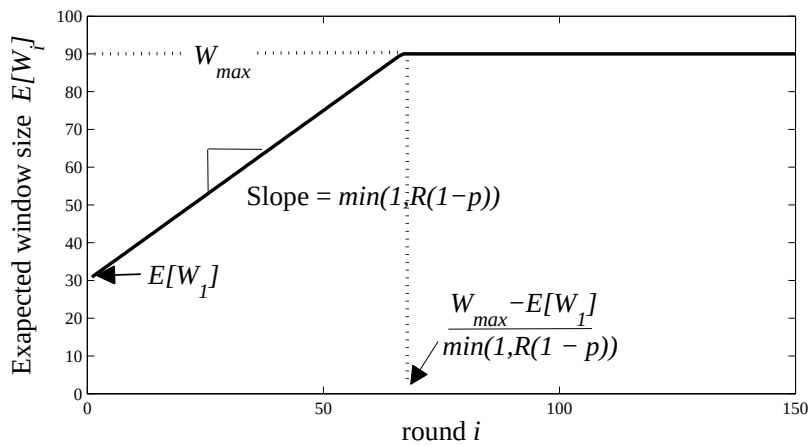



Figure 4-6: The predicted E2E-TCP/NC's window size when random losses are present in the network. The figure presents the expected window size for E2E-TCP/NC where $W_{\max} = 90$, $E[W_1] = 30$. We usually assume $W_1 = 1$; here we use $W_1 = 30$ to exemplify the effect of $W_1$.

### Markov Chain Model

The above analysis describes the expected behavior of E2E-TCP/NC's window size. We can also describe the window size behavior using a Markov chain. Figure 4-7 presents the Markov chain representing E2E-TCP/NC's window evolution. The states of this Markov chain represent the instantaneous window size, and are not specific to a round. Therefore, in this subsection, we briefly abandon the notion of rounds.

A transition occurs whenever a packet is transmitted. We denote $S(W)$ to be the state representing the window size of $W$. Assume that we are at state $S(W)$. If a transmitted packet is received by the E2E-TCP/NC receiver and acknowledged, the window is incremented by $\frac{1}{W}$; thus, we end up in state $S(W + \frac{1}{W})$. This occurs with probability $(1 - p)$. On the other hand, if the packet is lost, then we stay at $S(W)$. This occurs with probability $p$. Therefore, the Markov chain states represent the window size, and the transitions correspond to packet transmissions.

State $S(W_{\max})$ is an absorbing state of the Markov chain. As noted in Section 4.3, E2E-TCP/NC does not often experience a window shutdown unless there are correlated or heavy losses. We remind the reader again that we assume that losses are random, and we do not model correlated losses. As a result, E2E-TCP/NC's window size is non-decreasing as

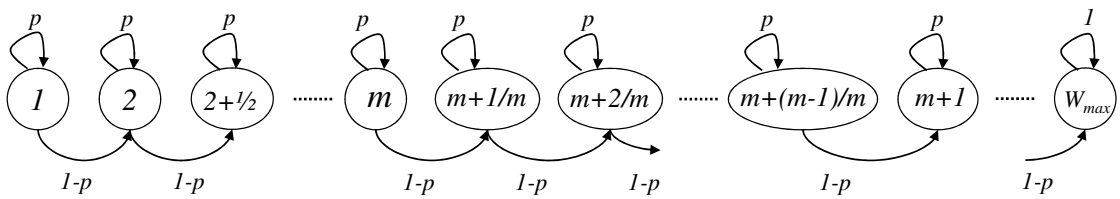



Figure 4-7: A Markov chain showing the E2E-TCP/NC's window evolution. Each state corresponds to the size of the congestion window.

shown in Figure 4-7. Therefore, given enough time, E2E-TCP/NC reaches state $S(W_{\max})$ with probability equal to 1. We analyze the expected number of packet transmissions needed for absorption.

The *transition matrix* $P$ of the Markov chain is

$$P = \begin{pmatrix} p & 1-p & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & p & 1-p & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & p & 1-p & 0 & \cdots & 0 & 0 \\ \vdots & & & \ddots & \ddots & \cdots & & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & p & 1-p \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.38}$$

The shaded part of the matrix in Equation (4.38) is denoted $Q$.

Matrix $N = (I-Q)^{-1}$ is the *fundamental matrix* of the Markov chain, and can be used to compute the expected rounds until the absorption state. Therefore, the fundamental matrix $N$ plays an important role in determining the number of packet transmissions needed for absorption. The entry $N(S_1, S_2)$ represents the expected number of visits to state $S_2$ before absorption, *i.e.* we reach state $S(W_{\max})$, when we start from state $S_1$. Our objective is to find the expected number of packets transmitted to reach $S(W_{\max})$ starting from state $S(E[W_1])$ where $E[W_1] = 1$. Therefore, the partial sum of the first row entries of $N$ gives the expected number of packets transmitted until we reach the window size $W$.

We now derive the expression for the first row of $N$. First, we note that

$$I - Q = \begin{pmatrix} 1-p & -(1-p) & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1-p & -(1-p) & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1-p & -(1-p) & 0 & \cdots & 0 & 0 \\ \vdots & & & \ddots & \ddots & \cdots & & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1-p & -(1-p) \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1-p \end{pmatrix}. \quad (4.39)$$

We compute the inverse of $N = (I-Q)^{-1}$ using matrix of cofactors, known as the *adjugate matrix*. However, as noted above, we do not need to compute the inverse of $(I-Q)$ entirely. We are only interested in computing the first row of $N$. To compute the first row of $N = (I-Q)^{-1}$, we need the determinant of $I-Q$ and a subset of its cofactors.

First, since $I-Q$ is an upper triangular matrix, the determinant is the product of its diagonal entries,

$$\det(I-Q) = (1-p)^{\mathcal{S}}, \quad (4.40)$$

where $\mathcal{S}$ is the number of states in the Markov chain minus the absorbing state $S(W_{\max})$.

To compute $\mathcal{S}$, we observe that for the congestion window to increase from $W$ to $W+1$, we have to traverse states $S(W)$, $S(W+\frac{1}{W})$, $S(W+\frac{2}{W})$, $\cdots$, $S(W+\frac{W-1}{W})$. Therefore, there are $W$ states from state $S(W)$ to $S(W-1)$. This leads to

$$\mathcal{S} = \sum_{m=1}^{W_{\max}} m - 1 = \frac{W_{\max}(W_{\max}-1)}{2}. \quad (4.41)$$

Second, we compute the cofactors $C_{m1}$ for $m \in [1, \mathcal{S}]$ as these cofactors form the first row of the adjugate matrix and $(I-Q)^{-1}$. As shown in Figure 4-8,

$$C_{m1} = (1-p)^{\mathcal{S}-1} \quad (4.42)$$

for all $m \in [1, \mathcal{S}]$. Now, we are ready to compute the first row of $N = (I - Q)^{-1}$:

$$N(1,:) = \frac{1}{\det(I-Q)} \left[C_{11}, C_{21}, \cdots, C_{\mathcal{S}1}\right] \tag{4.43}$$

$$= \frac{1}{(1-p)^{\mathcal{S}}} \left[(1-p)^{\mathcal{S}-1}, (1-p)^{\mathcal{S}-1}, \cdots, (1-p)^{\mathcal{S}-1}\right] \tag{4.44}$$

$$= \left[\frac{1}{1-p}, \frac{1}{1-p}, \cdots, \frac{1}{1-p}\right]. \tag{4.45}$$

The sum of the elements in the first row of $N$ represents $T_p(W_{\max})$, the number of packet transmissions needed to reach a window size of $W_{\max}$ starting from initial window size of $W_1 = 1$. Therefore, we can now compute $T_p(W_{\max})$:

$$T_p(W_{\max}) = \sum_{m=1}^{\mathcal{S}} N(1,m) \tag{4.46}$$

$$= \sum_{m=1}^{\mathcal{S}} \frac{1}{1-p} \tag{4.47}$$

$$= \frac{W_{\max}(W_{\max}-1)}{2(1-p)}. \tag{4.48}$$

Equation (4.48) is derived using Equation (4.41).

$T_p(W_{\max})$ is the number of packets we expect to transmit given the erasure probability $p$. If we set $p = 0$, then $T_0(W_{\max}) = \frac{W_{\max}(W_{\max}-1)}{2}$. Therefore, $\frac{W_{\max}(W_{\max}-1)}{2}$ is the minimal number of transmission needed to achieve $W$ since $p = 0$ assumes no packets are lost.

The ratio

$$\frac{T_p(W_{\max})}{T_0(W_{\max})} = \frac{1}{1-p} \tag{4.49}$$

represents a lower bound on cost when losses are introduced, *i.e.* to combat random erasures, the sender on average has to send at least $\frac{1}{1-p}$ packets for each packet it wishes to send. This is exactly the definition of redundancy factor $R$. This analysis indicates that we should set

$$R \geq \frac{T_p(W_{\max})}{T_0(W_{\max})}. \tag{4.50}$$

Furthermore, $T_0(W_{\max})$ is equal to the area under the curve for rounds $i \in [0, \frac{W_{\max}-E[W_1]}{\min\{1, R\cdot(1-p)\}}]$ in Figure 4-6 if we set $R \geq \frac{1}{1-p}$. This is consistent with our intuition since Figure 4-6 repre-

$$C_{11} = (-1)^2 \begin{vmatrix} 1-p & -(1-p) & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1-p & -(1-p) & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1-p & -(1-p) & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1-p & -(1-p) & \cdots & 0 & 0 \\ \vdots & & & \ddots & \ddots & \cdots & & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1-p & -(1-p) \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1-p \end{vmatrix} = (1-p)^{\mathcal{S}-1}$$

$$C_{21} = (-1)^3 \begin{vmatrix} -(1-p) & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1-p & -(1-p) & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1-p & -(1-p) & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1-p & -(1-p) & \cdots & 0 & 0 \\ \vdots & & & \ddots & \ddots & \cdots & & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1-p & -(1-p) \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1-p \end{vmatrix} = (1-p)^{\mathcal{S}-1}$$

$$C_{31} = (-1)^4 \begin{vmatrix} -(1-p) & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -(1-p) & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1-p & -(1-p) & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1-p & -(1-p) & \cdots & 0 & 0 \\ \vdots & & & \ddots & \ddots & \cdots & & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1-p & -(1-p) \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1-p \end{vmatrix} = (1-p)^{\mathcal{S}-1}$$

$$C_{41} = (-1)^5 \begin{vmatrix} -(1-p) & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -(1-p) & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & -(1-p) & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1-p & -(1-p) & \cdots & 0 & 0 \\ \vdots & & & \ddots & \ddots & \cdots & & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1-p & -(1-p) \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1-p \end{vmatrix} = (1-p)^{\mathcal{S}-1}$$

$$\vdots$$

Figure 4-8: The cofactors of matrix $I - Q$. These cofactors $C_{m1}$, $m \in [1, \mathcal{S}]$, form the first row of $N = (I - Q)^{-1}$. The structure of $I - Q$ lends itself to a simple characterization of $C_{m1} = (1-p)^{\mathcal{S}-1}$ for all $m \in [1, \mathcal{S}]$.

sents the evolution of window size, and the area under the curve represents the number of packets transmitted. A more detailed discussion on the effect of $R$ can be found in Section 4.5.3.

### 4.5.2 E2E-TCP/NC Average Throughput

Using the results in Section 4.5.1, we derive an expression for the throughput. Once we have the expected value of the window size for any given round $i$, the long term average throughput is straight forward to derive.

The throughput of round $i$, denoted $\mathcal{T}_i$, the number packets received or acknowledged by the receiver in that round over the round trip time. Therefore, $\mathcal{T}_i$ is directly proportional to $E[W_i]$, the window size, and $1-p$, the probability of a successful transmission. Thus, we get

$$\mathcal{T}_i = \frac{E[W_i]}{SRTT} \min\{1, R(1-p)\} \text{ packets per second,} \tag{4.51}$$

where $R$ is the redundancy factor of E2E-TCP/NC, and $SRTT$ is the round trip time estimate.

We note that $\mathcal{T}_i \propto (1-p) \cdot R \cdot E[W_i]$. At any given round $i$, E2E-TCP/NC sender transmits $R \cdot E[W_i]$ coded packets, and we expect $pR \cdot E[W_i]$ packets to be lost. As a result, E2E-TCP/NC receiver only receives and acknowledges $(1-p) \cdot R \cdot E[W_i]$ degrees of freedom, which results in the sender incrementing its window by $\min\{1, (1-p)R\}$. Therefore, $\mathcal{T}_i \propto (1-p) \cdot R \cdot E[W_i]$ coincides with our intuition.

Taking Equation (4.51), we can average the throughput over $n$ rounds to obtain the average throughput for E2E-TCP/NC.

$$\begin{aligned}
\mathcal{T}_{e2e} &= \frac{1}{n} \sum_{i=1}^{n} \frac{E[W_i]}{SRTT} \min\{1, R(1-p)\} && (4.52)\\
&= \frac{\sum_{i=1}^{n} \min(W_{\max}, E[W_1] + i \min\{1, R(1-p)\})}{n \cdot SRTT} && (4.53)\\
&= \frac{\sum_{i=1}^{n} \min(W_{\max}, E[W_1] + i)}{n \cdot SRTT} \qquad \text{since } R \geq \frac{1}{1-p} && (4.54)\\
&= \frac{1}{n \cdot SRTT} \cdot f(n), && (4.55)
\end{aligned}$$

where

$$f(n) = \begin{cases} nE[W_1] + \frac{n(n+1)}{2} & \text{for } n \leq r^* \\ nW_{\max} - r^*(W_{\max} - E[W_1]) + \frac{r^*(r^*-1)}{2} & \text{for } n > r^* \end{cases} \tag{4.56}$$

$$r^* = W_{\max} - E[W_1]. \tag{4.57}$$

Note that as $n \to \infty$, the average throughput $\mathcal{T}_{e2e} \to \frac{W_{\max}}{SRTT}$.

An important aspect of TCP is congestion control mechanism. This analysis may suggest that network coding no longer allows for TCP to react to congestion. We emphasize that the above analysis assumes that there are only random losses with probability $p$, and that there are no correlated losses. The erasure correcting power of network coding is limited by the redundancy factor $R$. If the network experiences packet erasures at a rate less than $p$, then network coding can correct those erasures with redundancy $R \geq \frac{1}{1-p}$. However, if there are enough losses (e.g., losses caused by congestion), network coding cannot mask all the erasures from TCP. This will eventually lead E2E-TCP/NC to experience a TD or TO event, depending on the variants of TCP used. In Section 4.6.3, we present simulation results that show that TCP's congestion control mechanism still applies to E2E-TCP/NC when appropriate.

As shown in Equation (4.55), the $RTT$ and its estimate $SRTT$ play an important role in E2E-TCP/NC. In addition, the choice of $R$ can have significant effect on the behavior of E2E-TCP/NC. We shall formally define and discuss the effect of $R$ and $SRTT$ in the following sections.

### 4.5.3 The Effect of Redundancy Factor $R$

The redundancy factor $R \geq 1$ is the ratio between the average rate at which linear combinations are sent to the receiver and the rate at which TCP's window progresses. For example, if the TCP sender has 10 packets in its window, then the encoder transmits $10R$ linear combinations. If $R$ is large enough, the receiver will receive at least 10 linear combinations to decode the original 10 packets even in the presence of losses. This redundancy is necessary to 1) compensate for the losses within the network, and 2) match TCP's sending rate to

the rate at which data is actually received at the receiver.

As shown in Equation (4.51), setting $R < \frac{1}{1-p}$ can cause performance degradation since network coding may no longer be able to compensate for the losses and this may lead to window closing for E2E-TCP/NC. To maximize throughput, an appropriate value of $R \geq \frac{1}{1-p}$ should be chosen.

References [91][89] introduce the redundancy factor with TCP/NC, and show that $R \geq \frac{1}{1-p}$ is necessary. This coincides with our analysis in Section 4.5.1. Therefore, by references [91][89] and Equation (4.51), it may seem that setting $R = \frac{1}{1-p}$ would maximize throughput and minimize redundancy; making $R = \frac{1}{1-p}$ seem like the optimal value of $R$.

However, the redundancy factor $R$ should be chosen with some care. Setting $R \gg \frac{1}{1-p}$ may over-compensate for the losses within the network; thus, introducing more redundant packets than necessary. On the other hand, matching $R$ to exactly $\frac{1}{1-p}$ may not be desirable for two reasons: 1) The exact value of $\frac{1}{1-p}$ may not be available or difficult to obtain in real applications; 2) As $R \to \frac{1}{1-p}$, it becomes more likely that E2E-TCP/NC is unable to *fully* recover from losses in any given round. By *fully* recover, we mean that E2E-TCP/NC decoder is able to acknowledge all packet transmitted in that round.

To analytically understand this intuition, consider an E2E-TCP/NC connection with congestion window size $W$. Assume that loss probability is $p$, and the redundancy factor is $R$. For E2E-TCP/NC to fully recover from the losses in a given round, the E2E-TCP/NC decoder has to receive $W$ out of $\lceil RW \rceil$ coded packets sent. Therefore, the probability that the E2E-TCP/NC decoder will fully recover from losses is

$$\mathbf{P}(\text{E2E-TCP/NC decoder receives at least } W \text{ out of } \lceil RW \rceil \text{ coded packets}) \tag{4.58}$$

$$= \mathbf{P}(\text{At most } (R-1)W \text{ out of } \lceil WR \rceil \text{ coded packets are lost}) \tag{4.59}$$

$$= \sum_{i=0}^{(R-1)W} \binom{\lceil RW \rceil}{i} p^i \, (1-p)^{\lceil RW \rceil - i}. \tag{4.60}$$

Figure 4-9 shows the probability that E2E-TCP/NC will fully decode, as shown in Equation (4.60), for various values of $p$ and $R$.

E2E-TCP/NC can maintain a fairly high throughput with just partial acknowledgment (in each round, only a subset of the packets are acknowledged). In Section 4.6, we show

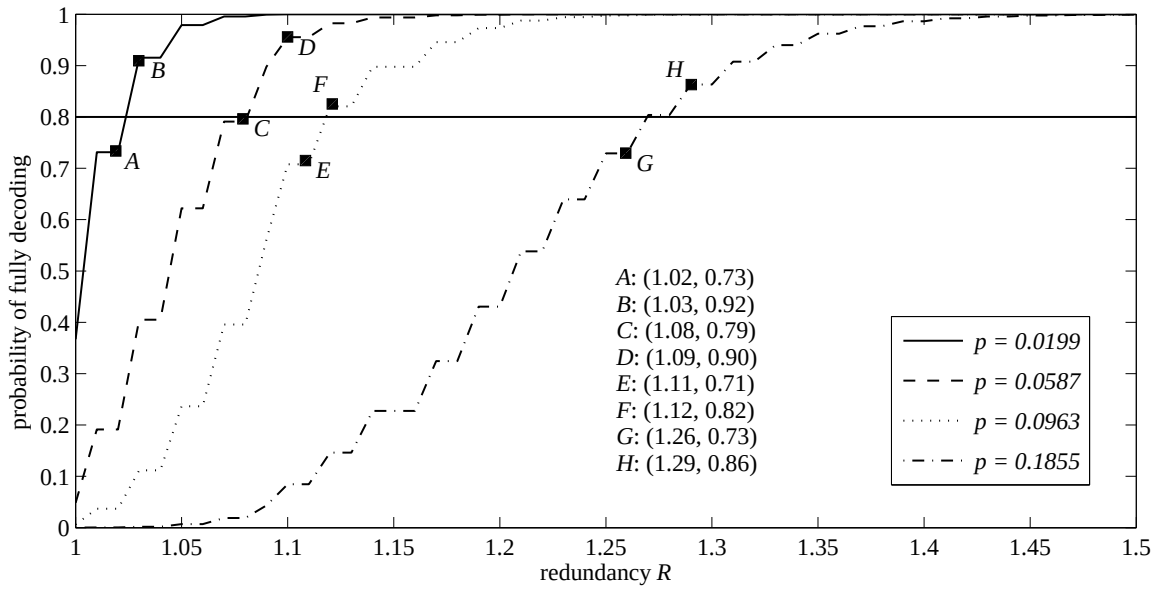



Figure 4-9: The probability of E2E-TCP/NC fully decoding. The figure provides the probability that the decoder is able to receive enough degrees of freedom to fully decode in a given round as the redundancy factor $R$ varies. As we increase $R$, the probability increases. In addition, given a redundancy factor $R$, the probability of fully decoding increases as the loss rate $p$ decreases.

empirically that setting the redundancy factor $R$ such that

$$\sum_{i=0}^{W(R-1)} \binom{\lceil RW \rceil}{i} p^i \, (1-p)^{\lceil RW \rceil - i} \geq 0.8 \tag{4.61}$$

is sufficient for TCP/NC to overcome the losses and maintain high throughput.

In general, the value of $R$ which satisfies Equation (4.61) is slightly larger than $\frac{1}{1-p}$. This is because $R = \frac{1}{1-p}$ is the average redundancy factor needed to overcome losses, and does not account for the variance of loss patterns. Therefore, if $R = \frac{1}{1-p}$, E2E-TCP/NC is unable to recover consistently from losses, and we observe large fluctuations in performance round to round. This results in performance degradation.

There are two points in Figure 4-9. First set of points corresponds to the value of maximum value of $R$ such that the probability of fully decoding is less than 0.8 (points $A$, $C$, $E$, and $G$). Second set of points corresponds to the value of minimal value of $R$ such that the probability of fully decoding is greater than 0.8 (points $B$, $D$, $F$, and $H$). This second set of points corresponds to the points that satisfy the heuristic from Equation (4.61). As we shall show in Section 4.6.2, choosing the redundancy factor $R$ according to this heuristic is effective in overcoming losses and achieving large throughput.

### 4.5.4 The Effect of Smoothed Round Trip Time $SRTT$

$SRTT$ is the round trip time estimate that TCP maintains by sampling the behavior of packets sent over the connection. It is denoted $SRTT$ because it is often referred to as "smoothed" round trip time as it is obtained by averaging the time for a packet to be acknowledged after the packet has been sent. In Equation (4.51), we use $SRTT$ instead of $RTT$ because $SRTT$ is the "effective" round trip time E2E-TCP/NC experiences.

In lossy networks, E2E-TCP/NC's $SRTT$ is often greater than $RTT$. This can be seen in Figure 4-1. The first coded packet $\mathbf{p_1} + \mathbf{p_2} + \mathbf{p_3}$ is received and acknowledged by $\mathbf{seen}(\mathbf{p_1})$. Therefore, the sender is able to estimate the round trip time correctly; resulting in $SRTT = RTT$. However, the second packet $\mathbf{p_1} + \mathbf{2p_2} + \mathbf{p_3}$ is lost. As a result, the third packet $\mathbf{p_1} + \mathbf{2p_2} + \mathbf{2p_3}$ is used to acknowledge the second degree of freedom $\mathbf{seen}(\mathbf{p_2})$. This increases the round trip time estimate $SRTT > RTT$. In our model,

we assume for simplicity that the time needed to transmit a packet is much smaller than RTT; therefore, despite the losses, our model would result in $SRTT \approx RTT$. However, in practice, depending on the size of the packets, the transmission time may not be negligible.

This results in a longer round trip time estimate, which can be characterized as described below. We define $t_p$ to be the time to transmit a packet. Then, the sender expects to receive an ACK of a packet after $SRTT$ time units, where

$$SRTT = \sum_{i=0}^{\infty}(RTT + i \cdot t_p)p^i(1-p) \tag{4.62}$$

$$= RTT + t_p\frac{p}{1-p}. \tag{4.63}$$

For simplicity, Equation (4.63) does not take into account the "edge effect" of packets that are waiting to be acknowledged across rounds. As the window size grows, the edge effect can safely be ignored.

## 4.6 Simulation Results

We use simulations to verify that our analysis captures the behavior of both TCP and E2E-TCP/NC. We use NS-2 (Network Simulator [1]) to simulate TCP and E2E-TCP/NC, where we use the implementation of E2E-TCP/NC from [89]. Two FTP applications (ftp0, ftp1) wish to communicate from the source (src0, src1) to sink (sink0, sink1), respectively. There is no limit to the file size. The sources generate packets continuously until the end of the simulation. The two FTP applications use either TCP or E2E-TCP/NC. We denote TCP0, TCP1 to be the two FTP applications when using TCP; and we denote NC0, NC1 to be the two FTP applications when using E2E-TCP/NC.

The network topology for the simulation is shown in Figure 4-10. All links, in both forward and backward paths, are assumed to have a bandwidth of $C$ megabits per second (Mbps), a propagation delay of 100 ms, a buffer size of 200, and a erasure rate of $q$. Since there are in total four links in the path from node 0 to node 4, the probability of packet erasure is $p = 1 - (1-q)^4$. Each packet transmitted is assumed to be 8000 bits (1000 bytes). We set $W_{\max} = 50$ packets for all simulations. In addition, time-out period is

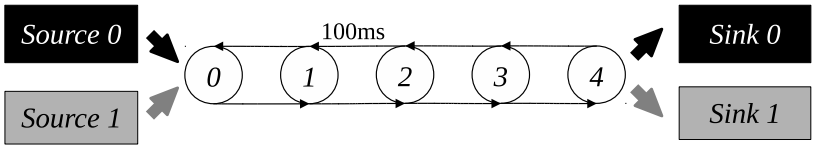



Figure 4-10: Network topology for the simulations. All links, in both forward and backward paths, are assumed to have a bandwidth of $C$ megabits per second (Mbps), a propagation delay of 100 ms, a buffer size of 200, and an erasure rate of $q$. Since there are in total four links in the path from node 0 to node 4, the probability of end-to-end packet erasure is $p = 1 - (1 - q)^4$.

$T_o = \frac{3}{RTT} = 3.75$ rounds long (3 seconds). Therefore, our variables for the simulations are:

- $p = 1 - (1 - q)^4$: End-to-end erasure rate,
- $R$: Redundancy factor,
- $C$: Capacity of the links in megabits per second (Mbps).

We study the effect these variables have on the following:

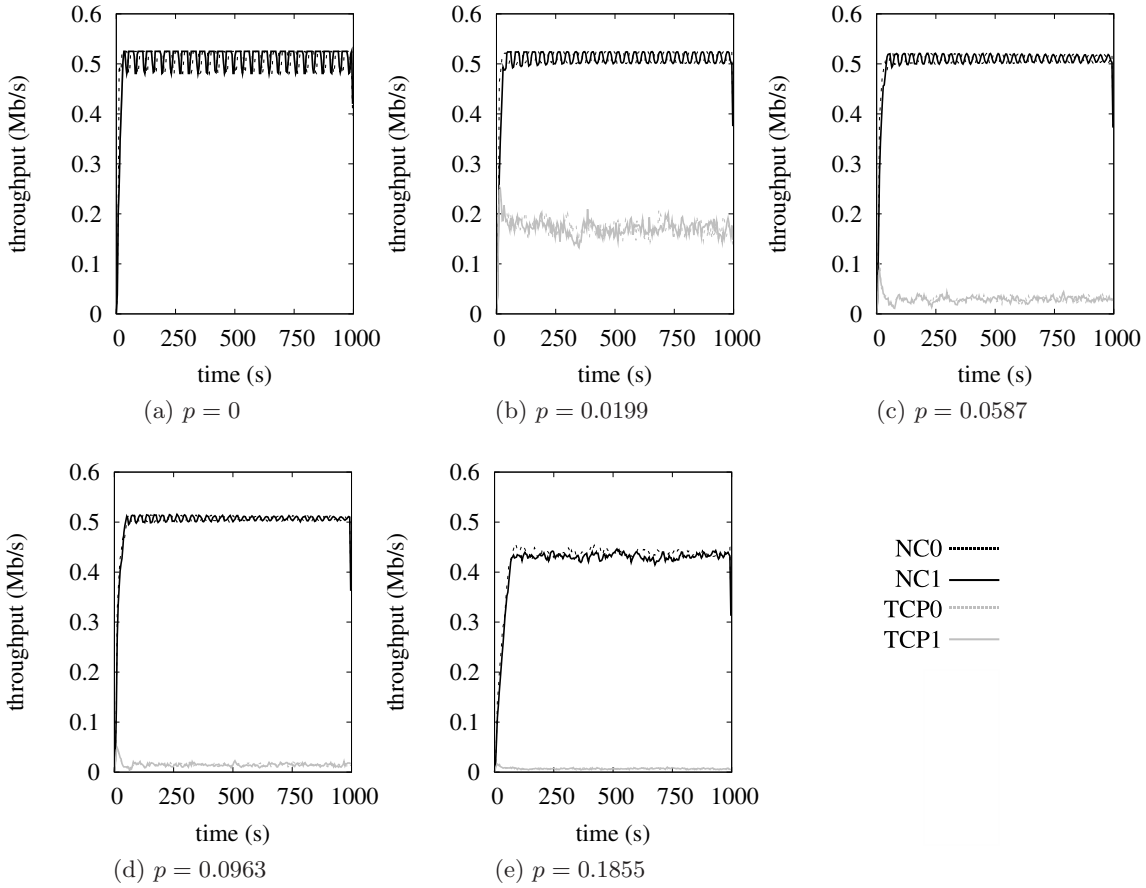
- $\mathcal{T}$: Throughput of TCP or E2E-TCP/NC,
- $E[W]$: Average window size of TCP or E2E-TCP/NC,
- $SRTT$: Round-trip estimate.

For each data point, we average the performance over 100 independent runs of the simulation, each of which is 1000 seconds long.

### 4.6.1 Probability of Erasure $p$

In this section, we set $C = 2$ Mbps and $R = 1.25$ regardless of the value of $p$. We vary $q$ to be 0, 0.005, 0.015, 0.025, and 0.05. The corresponding $p$ values are 0, 0.0199, 0.0587, 0.0963, and 0.1855. The simulation results are shown in Figures 4-11, 4-12, and 4-13.

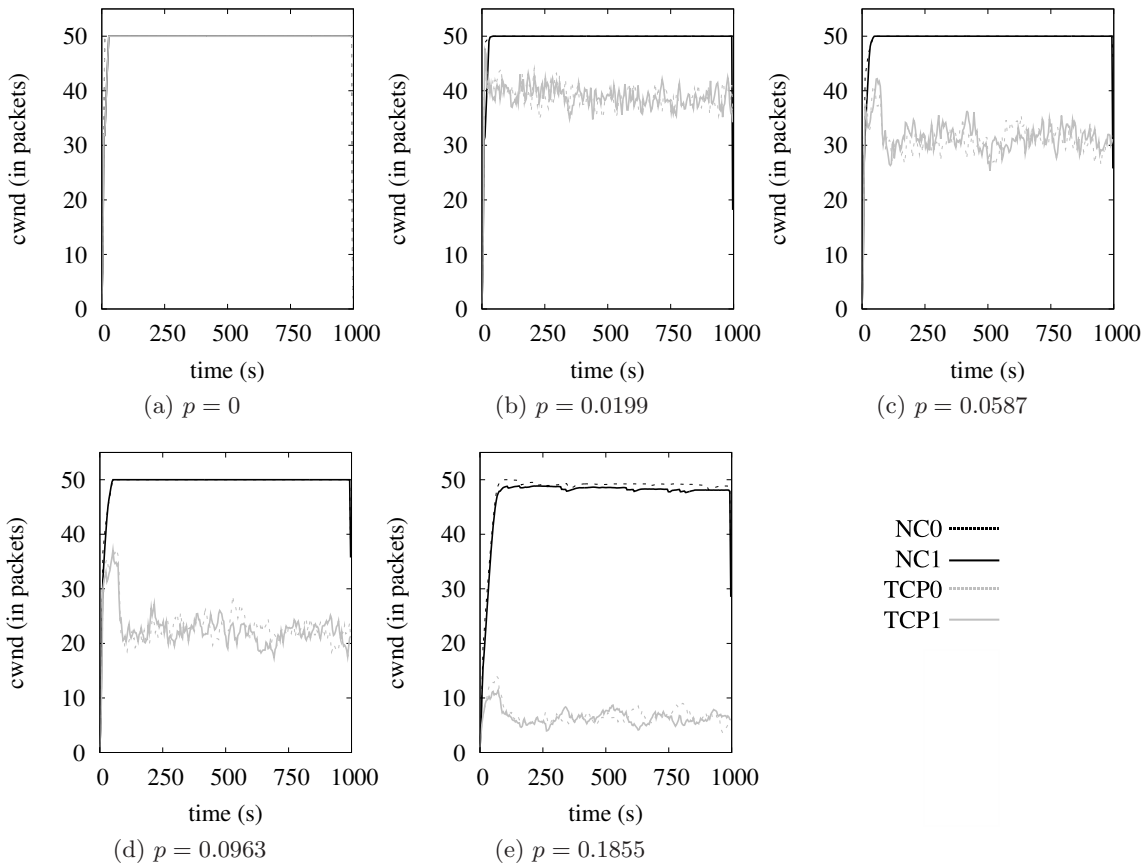
Firstly, we show that when there are no random erasures ($p = 0$), then E2E-TCP/NC and TCP behave similarly as shown in Figures 4-11a, 4-12a, and 4-13a. Without any random losses and congestion, all of the flows (NC0, NC1, TCP0, TCP1) achieve the maximal throughput, $\frac{W_{\max}}{RTT} \cdot \frac{8}{10^6} = 0.5$ Mbps.




(a) $p = 0$ (b) $p = 0.0199$ (c) $p = 0.0587$

(d) $p = 0.0963$ (e) $p = 0.1855$

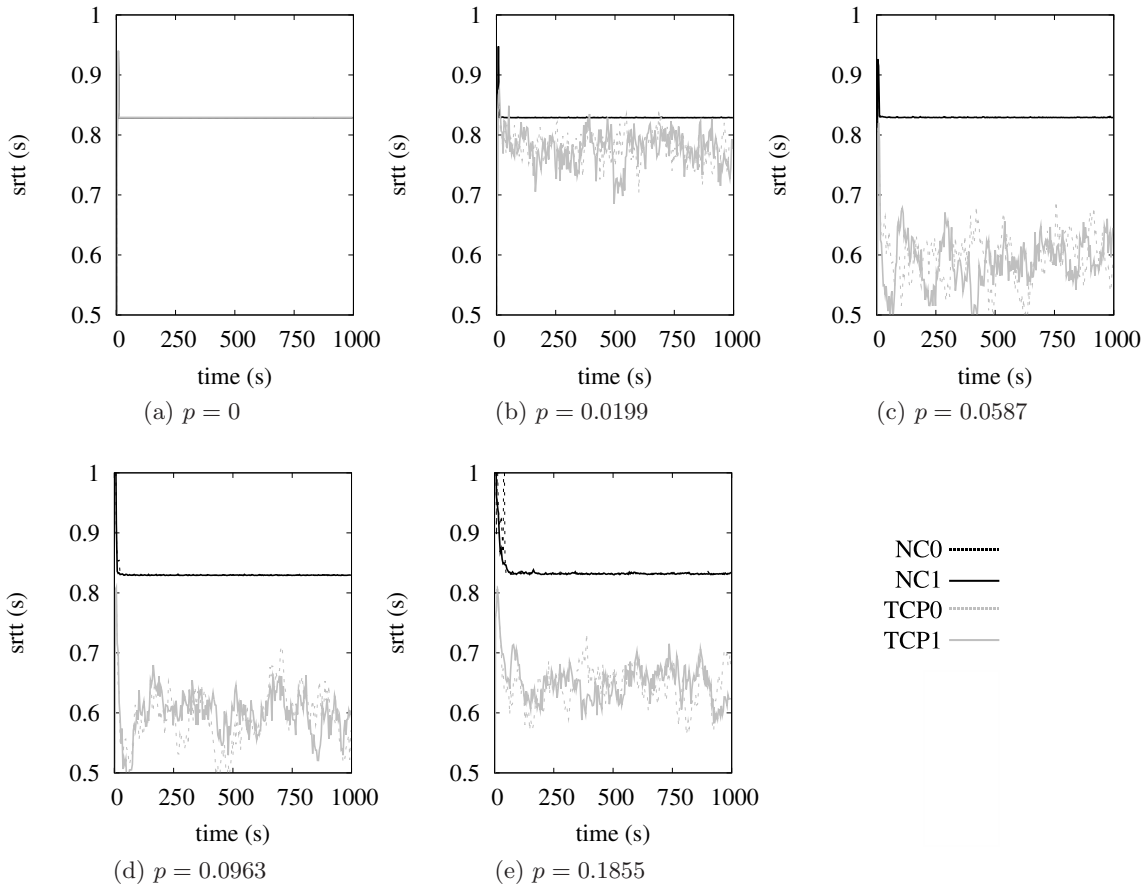
Figure 4-11: Throughput of E2E-TCP/NC and TCP with varying end-to-end erasure probability $p$. The figure show that E2E-TCP/NC can maintain high throughput even in very lossy networks, while TCP is unable to do so as soon as the loss rates exceed a few percent. For each data point, we average the performance of 100 independent runs of the simulations, each of which is 1000 seconds long.




(a) $p = 0$ (b) $p = 0.0199$ (c) $p = 0.0587$

(d) $p = 0.0963$ (e) $p = 0.1855$

Figure 4-12: The congestion window size of E2E-TCP/NC and TCP with varying end-to-end erasure probability $p$. The figure shows that E2E-TCP/NC maintains a large window size despite losses in the network. An interesting observation is TCP's window size. TCP maintains a moderately large window size, but TCP's throughput is much smaller than the corresponding window size. For each data point, we average the performance of 100 independent runs of the simulations, each of which is 1000 seconds long.




(a) $p = 0$ (b) $p = 0.0199$ (c) $p = 0.0587$

(d) $p = 0.0963$ (e) $p = 0.1855$

Figure 4-13: The round trip time estimate (SRTT) of E2E-TCP/NC and TCP with varying end-to-end erasure probability $p$. The figure shows that E2E-TCP/NC maintains a stable SRTT, which reflects the loss rate $p$ present in the network. On the other hand, TCP's estimate varies significantly over time, reflecting TCP's performance degradation and fluctuations in throughput. For each data point, we average the performance of 100 independent runs of the simulations, each of which is 1000 seconds long.

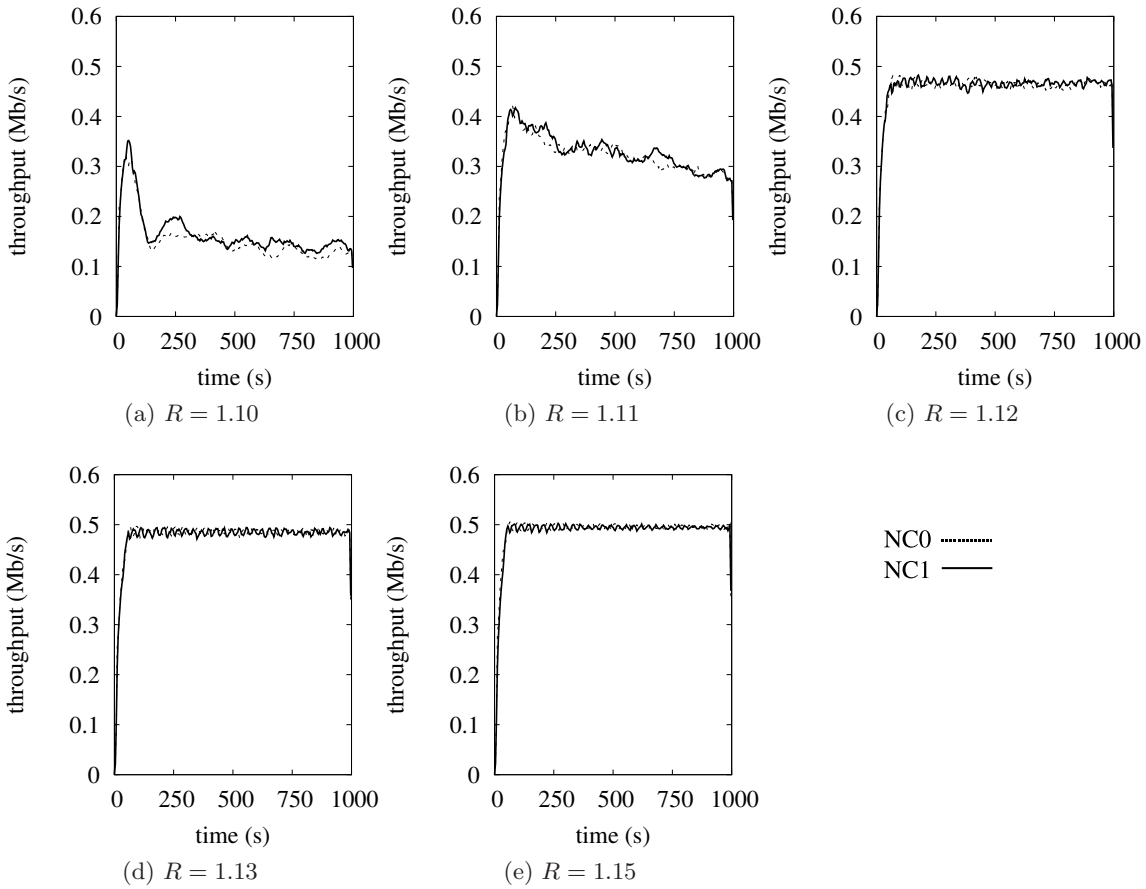
The more interesting result is when $p > 0$ as shown in Figures 4-11 and 4-12. As our analysis predicts, E2E-TCP/NC sustains its high throughput despite the random erasures in the network. We observe that TCP may close its window due to triple-duplicates ACKs or timeouts; however, E2E-TCP/NC is more resilient to such erasure patterns. Therefore, E2E-TCP/NC is able to increment its window consistently, and *maintain* the window size of 50 even under lossy conditions when standard TCP is unable to (resulting in the window fluctuation in Figure 4-12).

An interesting observation is that, TCP achieves a moderate average window size although the throughput (Mbps) is much lower as shown in Figures 4-11 and 4-12. For example, TCP's throughput is near zero for $p = 0.0587$ and $p = 0.0963$; however, TCP's congestion window size is approximately 30 and 20, respectively. This shows that naïvely keeping the transmission window open is not sufficient to overcome the random losses within the network, and does not lead to improvements in TCP's performance. Even if the transmission window is kept open (e.g. during timeout period), the sender can not transmit additional packets into the network without receiving ACKs. Eventually, this leads to a TD or TO event.

TCP modifies its RTT estimation depending on the ACKs received. However, due to random erasures, TCP's RTT estimate fluctuates significantly. On the other hand, E2E-TCP/NC is able to maintain a consistent estimate of the RTT; however, is slightly above the actual 800 ms. As described in Sections 4.3 and 4.5.4, E2E-TCP/NC masks errors by translating losses as longer RTT. For E2E-TCP/NC, if a specific packet is lost, the next subsequent packet received can "replace" the lost packet; thus, allowing the receiver to send an ACK. Therefore, the longer RTT estimate takes into account the delay associated with waiting for the next subsequent packet at the receiver. In Figure 4-13, we verify that this is indeed true.

### 4.6.2 Redundancy Factor $R$

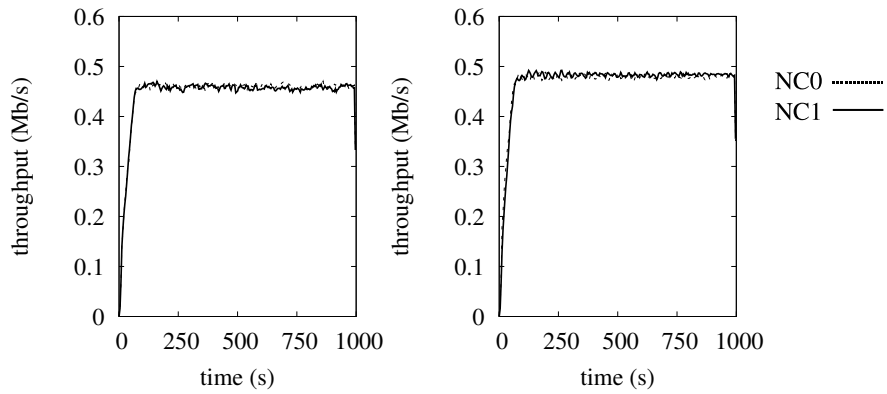
We set $C = 2$ Mbps. We vary the value of $p$ and $R$ to understand the relationship between $p$ and $R$. In Section 4.5.3, we noted that $R \geq \frac{1}{1-p}$ is necessary to mask random erasures from TCP. However, as $R \to \frac{1}{1-p}$, the probability that the erasures are completely masked




(a) $R = 1.10$ (b) $R = 1.11$ (c) $R = 1.12$

(d) $R = 1.13$ (e) $R = 1.15$

Figure 4-14: The effect of redundancy factor $R$ on E2E-TCP/NC's throughput. The figure presents the throughput of E2E-TCP/NC for $p = 0.0963$ with varying redundancy factor $R$. Note that $\frac{1}{1-p} = 1.107$.




(a) $R = 1.26$ (b) $R = 1.28$

Figure 4-15: The effect of redundancy factor $R$ on E2E-TCP/NC's throughput. The figure presents the throughput of E2E-TCP/NC for $p = 0.1855$ with varying redundancy factor $R$. Note that $\frac{1}{1-p} = 1.228$.

decreases. This may suggest that we need $R \gg \frac{1}{1-p}$ for E2E-TCP/NC to maintain its high throughput. However, we shall show by simulations that $R$ need not be much larger than $\frac{1}{1-p}$ for E2E-TCP/NC to achieve its maximal throughput.

In Figure 4-14, we present E2E-TCP/NC throughput behavior with $p = 0.0963$ and varying $R$. Note that $\frac{1}{1-p} = 1.107$ for $p = 0.0963$. There is a dramatic change in throughput behavior as we increase $R$ from 1.11 to 1.12. Note that $R = 1.12$ is only 1% additional redundancy than the theoretical minimum, *i.e.* $\frac{1.12}{1/(1-p)} \approx 1.01$. Another interesting observation is that, even with $R = 1.10$ or $R = 1.11$, E2E-TCP/NC achieves a significantly higher throughput than TCP (in Figure 4-11d) for $p = 0.0963$.

Figure 4-11e shows that, for $p = 0.1855$, E2E-TCP/NC throughput is not as steady and does not achieve the maximal throughput of 0.5 Mbps. This is because $\frac{1}{1-p} = 1.23$ is very close to $R = 1.25$. As a result, $R = 1.25$ is not sufficient to mask erasures with high probability. In Figure 4-15, we show that E2E-TCP/NC achieves an average throughput of 0.5 Mbps once $R \geq 1.28$ for $p = 0.1855$. Note that $R = 1.28$ is only 4% additional redundancy than the theoretical minimum, *i.e.* $\frac{1.28}{1/(1-p)} \approx 1.04$.

Similar behavior can be observed for $p = 0.0199$ and 0.0587, and setting $R$ to be slightly above $\frac{1}{1-p}$ is sufficient to achieve maximal throughput with E2E-TCP/NC.

As mentioned in Section 4.5.2, a good heuristic to use in setting $R$ is the following. Given

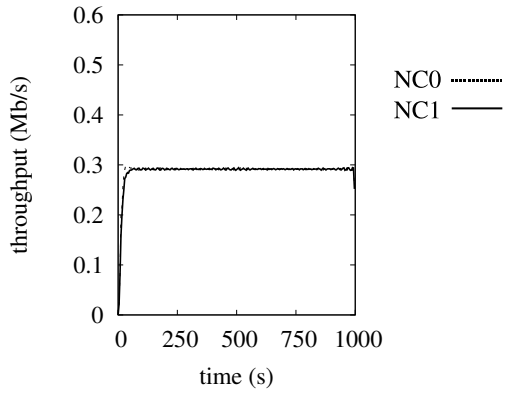



Figure 4-16: Fairness of E2E-TCP/NC. The figure presents the throughput of E2E-TCP/NC for $p = 0.0963$ and $C = 0.7$ Mbps. The two E2E-TCP/NC flows share the $C = 0.7$ Mbps fairly, each achieving 0.3162 Mbps.

a probability of erasure $p$ and window size $W$, the probability that losses in any given round is completely masked is upper bounded by $\sum_{x=0}^{W(R-1)} \binom{RW}{x} p^x (1-p)^{RW-x}$. Ensuring that this probability is at least 0.8 has proven to be a good heuristic to use in finding the appropriate value of $R$. This is equivalent to ensuring that there are no more than $W(R-1)$ losses in at least 80% of the rounds.

### 4.6.3 Congestion Control

We showed that E2E-TCP/NC achieves a good performance in lossy environment. This may raise concerns about masking correlated losses from TCP, which would disable TCP's congestion control mechanism. We show that the network coding layer masks random losses only, and allows TCP's congestion control to take affect when necessary.

Given a capacity $C$ and erasure rate $p$, the available bandwidth is $C(1-p)$ Mbps. Therefore, given two flows, a fair allocation of bandwidth should be $\frac{C(1-p)}{2}$ Mbps per flow. This is the *available* bandwidth, not the *achieved* bandwidth.

With E2E-TCP/NC flows, there is another parameter we need to consider: the redundancy factor $R$. Since E2E-TCP/NC sends $R$ coded packets for each data packet, the achievable bandwidth is $\min\{C(1-p), \frac{C}{R}\}$ Mbps; if shared among two flows fairly, we expect $\frac{1}{2}\min\{C(1-p), \frac{C}{R}\}$ Mbps per coded flow. Note that, if $R$ is chosen appropriately (*i.e.* slightly above $\frac{1}{1-p}$), then E2E-TCP/NC can achieve rate close to $C(1-p)$, which is optimal.

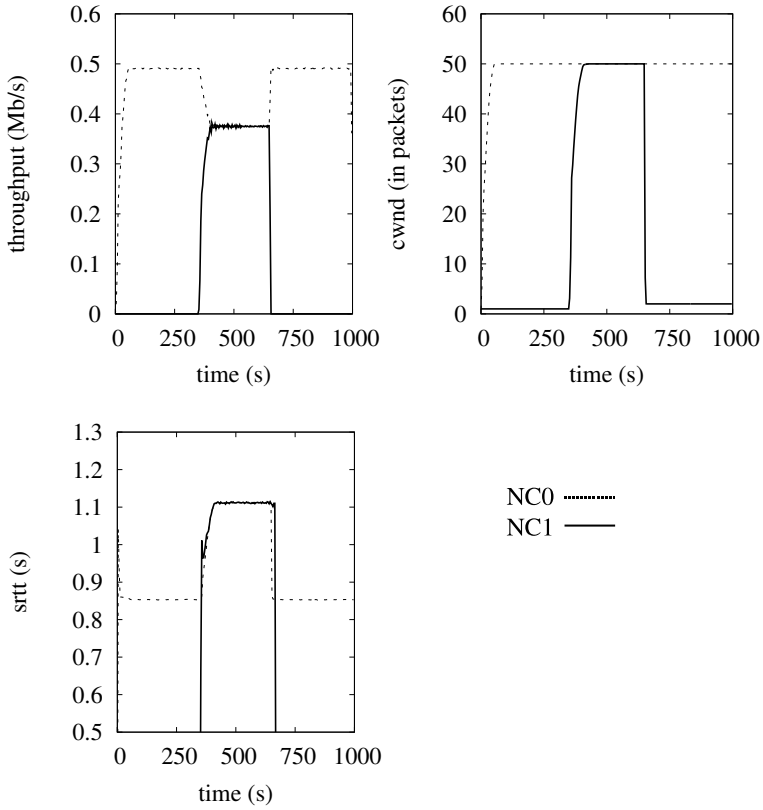



Figure 4-17: Fairness and congestion control of E2E-TCP/NC. The figure presents the throughput of E2E-TCP/NC for $p = 0.0963$ with congestion ($C = 0.9$ Mbps, $R = 1.2$, $W_{\max} = 50$). Before NC1 joins the network, NC0 achieves the maximum throughput of 0.5 Mbps. As the second flow of E2E-TCP/NC (NC1) joins the network, both NC0 and NC1 share the bandwidth, each achieving 0.37 Mbps.

As we have seen, if $p > 0$, TCP may not be able to use fully the available bandwidth. On the other hand, E2E-TCP/NC is able to use the available bandwidth efficiently. We now show that multiple E2E-TCP/NC flows share the bandwidth fairly.

We consider two flows (NC0, NC1) with $W_{\max} = 50$, $R = 1.2$, and $p = 0.0963$. If there is no congestion, each flow would achieve approximately 0.5 Mbps. However, we set $C = 0.7$ Mbps. The two flows should achieve $\frac{1}{2}\min\{0.7(1-0.0963), \frac{0.7}{1.2}\} = 0.2917$ Mbps. We observe in Figure 4-16 that NC0 and NC1 achieve 0.2878 Mbps and 0.2868 Mbps, respectively. Note that $\frac{C(1-p)}{2} = 0.3162$; thus, NC0 and NC1 is near optimal even though $R = 1.2 > \frac{1}{1-p} = 1.106$.

For our next simulations, we set $C = 0.9$ Mbps, $W_{\max} = 50$, $p = 0.0963$, and $R = 1.2$. Furthermore, we assume that NC0 starts at 0s and runs for 1000s, while NC1 starts at time 350s and ends at time 650s. Each connection, without congestion, can achieve a throughput of 0.5 Mbps. Therefore, before NC1 enters, NC0 should be able to achieve a throughput of 0.5 Mbps; however, when NC1 starts its connection, there is congestion, and both NC0 and NC1 have to react to this. Figure 4-17 shows that indeed this is true. We observe that when NC1 starts its connection, both NC0 and NC1 share the bandwidth equally (0.3700 and 0.3669 Mbps, respectively). The achievable bandwidth predicted by $\min\{C(1-p), \frac{C}{R}\}$ is 0.75 Mbps (or 0.375 Mbps per flow). Notice that both NC0 and NC1 maintains its maximum window size of 50. Instead, NC0 and NC1 experience a longer RTT, which naturally translates to a lower throughput given the same $W_{\max}$.

### 4.6.4 Comparison to the Analytical Model

Finally, we examine the accuracy of our analytical model in predicting the behavior of TCP and E2E-TCP/NC. First, note that our analytical model of window evolution, which is shown in Equation (4.37) and Figure 4-6, demonstrates the same trend as that of the E2E-TCP/NC window evolution of NS-2 simulations shown in Figure 4-12.

Second, we compare the actual NS-2 simulation performance to the analytical model. This is shown in Table 4.1. We observe that Equations (4.51) and (4.37) predict well the trend of E2E-TCP/NC's throughput and window evolution, and provides a good estimate of E2E-TCP/NC's performance. Furthermore, our analysis predicts the average TCP behavior

Table 4.1: The performance of TCP and E2E-TCP/NC. The average simulated or predicted long-term throughput of TCP and E2E-TCP/NC in megabits per second (Mbps). 'NC0', 'NC1', 'TCP0', 'TCP1' are average throughput achieved in the NS-2 simulations with the corresponding redundancy factor '$R$'. 'E2E analysis' is calculated using Equation (4.55) with $\lfloor n \cdot SRTT \rfloor = 1000$. 'TCP analysis' is computed using Equation (4.35).

| $p$ | E2E $SRTT$ | $R$ | NC0 | NC1 | E2E analysis | TCP0 | TCP1 | TCP analysis |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.8256 | 1 | 0.5080 | 0.5057 | 0.4819 | 0.5080 | 0.5057 | 0.5000 |
| 0.0199 | 0.8260 | 1.03 | 0.4952 | 0.4932 | 0.4817 | 0.1716 | 0.1711 | 0.0667 |
| 0.0587 | 0.8264 | 1.09 | 0.4926 | 0.4909 | 0.4814 | 0.0297 | 0.0298 | 0.0325 |
| 0.0963 | 0.8281 | 1.13 | 0.4758 | 0.4738 | 0.4804 | 0.0149 | 0.0149 | 0.0220 |
| 0.1855 | 0.8347 | 1.29 | 0.4716 | 0.4782 | 0.4766 | 0.0070 | 0.0070 | 0.0098 |

well. In Table 4.1, we see that Equation (4.35) is consistent with the NS-2 simulation results even for large values of $p$. Therefore, both simulations as well as analysis support that E2E-TCP/NC is resilient to erasures; thus, better suited for reliable transmission over unreliable networks, such as wireless networks.

## 4.7 Conclusions

We have presented an analytical study and compared the performance of TCP and E2E-TCP/NC. Our analysis characterizes the throughput of TCP and E2E-TCP/NC as a function of erasure rate, round-trip time, maximum window size, and the duration of the connection. We showed that network coding, which is robust against erasures and failures, can prevent TCP's performance degradation often observed in lossy networks. Our analytical model shows that TCP with network coding has significant throughput gains over TCP. E2E-TCP/NC is not only able to increase its window size faster but also maintain a large window size despite losses within the network; on the other hand, TCP experiences window closing as losses are mistaken to be congestion. Furthermore, NS-2 simulations verify our analysis on TCP's and E2E-TCP/NC's performance. Our analysis and simulation results both support that E2E-TCP/NC is robust against erasures and failures. Therefore, E2E-TCP/NC is well suited for reliable communication in lossy wireless networks.

# Chapter 5

# Algebraic Watchdog:

## Network Coding for Secure Communications

There have been numerous contributions to secure wireless networks, including key management, secure routing, Byzantine detection, and various protocols (for a general survey on this topic, see [42][97][20][48][83][14][69][82]). Countering these types of threats is particularly important in military communications and networking, which are highly dynamic in nature and must not fail when adversaries succeed in compromising some of the nodes in the network. We consider the problem of Byzantine detection. The traditional approach is *receiver-based* – *i.e.* the receiver of the corrupted data detects the presence of an upstream adversary. However, this detection may come too late as the adversary is partially successful in disrupting the network even if it is detected. It has wasted network bandwidth, while the source is still unaware of the need for retransmission.

Reference [79] introduces a protocol for routing wireless networks, called the *watchdog and pathrater*. Upstream nodes police their downstream neighbors nodes using *promiscuous monitoring*, where a node $v$ within range of node $v'$ overhears communication to and from $v'$ even if those communication do not directly involve $v$. This scheme successfully detects adversaries and removes adversaries by dynamically adjusting the routing paths. However, the protocol requires a significant overhead (12% to 24%) owing to increased control traffic and numerous cryptographic messages [79].

Our goal is to design and analyze a watchdog-inspired protocol for wireless networks

using network coding. We propose a new scheme called the *Algebraic Watchdog*, in which nodes can detect malicious behaviors probabilistically by taking advantage of the broadcast nature of the wireless medium. Although we focus on detecting malicious or misbehaving nodes, the same approach can be applied to faulty or failing nodes. Our ultimate goal is a robust *self-checking network*.

The key difference between our work [62] and that of [79] is that we allow network coding. Network coding [3][67] is advantageous as it increases throughput, robustness against failures and erasures, and resilience in dynamic and unstable networks where state information may change rapidly or may be hard to obtain.

The key challenge in Algebraic Watchdog is that, by incorporating network coding, we can no longer recognize packets individually. In [79], $v$ can monitor its downstream neighbor $v'$ by checking that the packet sent by $v'$ is a copy of what $v$ sent to $v'$. However, with network coding, this is no longer possible as transmitted packets are a function of the received packets. Furthermore, $v$ may not have full information regarding the packets received at $v'$; thus, node $v$ is faced with the challenge of inferring the packets received at $v'$ and ensuring that $v'$ is transmitting a valid function of the received packets. We note that [72] combines source coding with watchdog; thus, [72] does not face the same problem as the Algebraic Watchdog.

Algebraic Watchdog is one of many approaches to detecting malicious behavior in coded networks. For example, there are end-to-end error corrections [94], packet signature schemes [96], and generation-based detection schemes [40]. These schemes, including Algebraic Watchdog, are effective in detecting or correcting different degrees of malicious behavior. Depending on the severity of adversarial errors, different approaches should be employed as shown in [55].

The chapter is organized as follows. In Section 5.1, we discuss the intuition behind Algebraic Watchdog. In Section 5.2, we present the related material. We present the main body of the Algebraic Watchdog in Sections 5.3 to 5.8. In Section 5.3, we introduce our problem statement and network model. In Section 5.4, we analyze the protocol for a simple two-hop network, first graphically in Section 5.4.1 and then algebraically in Section 5.4.2. In Section 5.5, we extend the analysis for Algebraic Watchdog to a more general two-

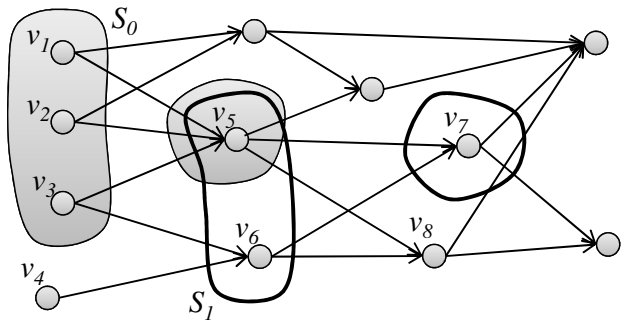



Figure 5-1: An example network showing how nodes may monitor their downstream nodes.

hop network, and in Section 5.7, we present an Algebraic Watchdog protocol for a multi-hop network. We present simulation results in Section 5.8, which show that an adversary within the network can be detected probabilistically by upstream nodes. In Section 5.9, we summarize our contribution.

## 5.1 Intuition

Assume that nodes in $S_0 = \{v_1, v_2, v_3\}$ and $v_4$ are well-behaving sources in Figure 5-1. Nodes in $S_0$ can monitor $v_5$ collectively or independently. Furthermore, $v_3$ and $v_4$ can monitor $v_6$. This enforces $v_5$ and $v_6$ to send valid data. We do not make any assumption on whether $v_5$ and $v_6$ are malicious or not. Nodes $v_5$ and $v_6$ are forced to send valid information regardless of their true nature.

If $v_5$ and $v_6$ are well-behaving, then $v_5$ or $v_6$ can check $v_7$'s behavior. Thus, propagating *trust* within the network. Now, what if $v_5$ or $v_6$ are malicious? If both $v_5$ and $v_6$ are malicious, all flows to $v_7$ are controlled by malicious nodes, *i.e.* flows through $v_7$ are completely compromised. Therefore, even if $v_1, v_2, v_3$, and $v_4$ can detect that $v_5$ and $v_6$ are misbehaving, there is nothing that $v_7$ or $v_1, v_2, v_3, v_4$ can do to protect the flow through $v_7$. The only solution in this case would be to physically remove $v_5$ and $v_6$ from the network or to construct a new path to $v_7$.

The intuition is that as long as the min-cut to any node is not dominated by adversaries, then the remaining well-behaving nodes can check its neighborhood and enforce that the

information flow is delivered correctly to the destination. For example, assume that only $v_6$ is malicious and $v_5$ is well-behaving in Figure 5-1. Since $v_3$ and $v_4$ monitor $v_6$, despite $v_6$ being malicious, $v_6$ is forced to send valid data. Then, $v_7$ receives two valid flows, which it is now responsible of forwarding. If $v_7$ is well-behaving, we do not have any problem. If $v_7$ is malicious, it may wish to inject errors to the flow. In this case, $v_7$ is only liable to $v_5$; but it is liable to at least one well-behaving node $v_5$. Thus, it is not completely free to inject any error it chooses; it has to ensure that $v_5$ cannot detect its misbehavior, which may be difficult to accomplish.

In this chapter, we show that this is indeed the case. We first start by studying a two-hop network, which would be equivalent to focusing on the operations performed by nodes in $S_0$ to check $v_5$. Then, we discuss how we can propagate this two-hop policing strategy to general network topologies.

## 5.2 Background

### 5.2.1 Secure Network Coding

Network coding allows algebraic mixing of information in the intermediate nodes. This mixing has been shown to have numerous performance benefits. It is known that network coding maximizes throughput for multicast [3] and increases robustness against failures [67] and erasures [77]. However, a major concern for network coded systems is their vulnerability to Byzantine adversaries. A single corrupted packet generated by a Byzantine adversary can contaminate all the information to a destination, and propagate to other destinations quickly. For example, in random linear network coding [77], one corrupted packet in a generation (a fixed set of packets) can prevent a receiver from decoding any data from that generation even if all the other packets received are valid.

There has been significant work to address this problem. One approach is to correct the errors injected by the Byzantine adversaries using *end-to-end network error correction* [94]. Reference [94] bounds the maximum achievable rate in an adversarial setting, and generalizes the Hamming, Gilbert-Varshamov, and Singleton bounds. Jaggi *et al.* [45] propose a distributed, rate-optimal, network coding scheme for multicast network that is resilient in

the presence of Byzantine adversaries for sufficiently large field and packet size. Reference [66][87] generalizes [45] to provide correction guarantees against adversarial errors for any given field and packet size.

Another approach of Byzantine detection in coded networks is to use *generation-based* detection schemes. Ho *et al.* [40] introduce an information-theoretic approach for detecting Byzantine adversaries, which only assumes that the adversary did not see all linear combinations received by the receivers. Their detection probability varies with the length of the hash, field size, and the amount of information unknown to the adversary. A polynomial hash is added to each packet in the generation. Once the destination node receives enough packets to decode a generation, it can probabilistically detect errors. The intuition behind this scheme is that if a packet is valid, then its data and hash are consistent with its coding vector, and a linear combination of valid packets is also valid.

Finally, *packet-based* detection schemes allow for finer granularity of Byzantine detection. There are several signature schemes that allow detection on per-packet basis. For instance, [17] proposes a signature scheme for network coding based on Weil pairing on elliptic curves. Elliptic curves are hard to analyze and are known to be computationally expensive [65]. The experimental results in [95] show that this scheme is indeed costly and time-consuming. Reference [68] uses homomorphic hash functions to verify packets in P2P systems, and [36] extends this approach to secure network coded P2P systems against Byzantine attacks. However, [36] requires a secure channel to transmit the hashes to all receivers before data is delivered. In this chapter, we assume that no such secure channel is available.

Reference [95] proposes a homomorphic signature scheme with RSA encryption and decryption to allow authentication and verification of data. Unfortunately, the scheme is incorrect[1]. This homomorphic property does not hold due to an error in the second to last equation in (12) of [95]; that is:

$$(a \bmod p) \times (b \bmod p) \bmod r \neq (ab \bmod p) \bmod r. \tag{5.1}$$

[1]This fact has been communicated to the authors of [95] by Anthony E. Kim, Raluca Ada Popa, and Muriel Médard, and acknowledged by the authors.

In [96], Fang *et al.* propose a signature scheme for network coding, which makes use of the linearity property of the packets in a coded system. This scheme does not require intermediate nodes to decode coded packets to check the validity of a packet; therefore, it is efficient in terms of computational cost as well as delay. Taking advantage of the fact that in linear network coding, any valid packet transmitted should belong to the subspace spanned by the original set of vectors, Fang *et al.* design a signature that can be used to easily check the membership of a received vector in the given subspace, while making it hard to generate a fake signature that is not in the subspace but passes the check.

Kim *et al.* [58] compare the cost and benefit associated with the various detection schemes described above. The cost and benefit of a scheme is measured in terms of transmitted bits by allowing nodes to employ the detection schemes to drop polluted data. In [58], it is shown that, with enough attackers present in the network, Byzantine detection schemes may improve the overall throughput of the system by choosing to forward only reliable or valid information. When the probability of attack is high, a packet-based detection scheme is most bandwidth efficient; however, when the probability of attack is low, the overhead involved with signing each packet becomes costly, and the generation-based scheme may be preferred.

### 5.2.2 Secure Routing Protocol: Watchdog and Pathrater

The problem of securing networks in the presence of Byzantine adversaries has been studied extensively [83][14][69][82]. The *watchdog and pathrater* [79] are two extensions to the Dynamic Source Routing [47] protocol that attempt to detect and mitigate the effects of routing misbehavior. The watchdog detects misbehavior based on promiscuous monitoring of the downstream node's transmissions to confirm if the relay correctly forwards the packets it receives. If a node fails to forward a packet within a certain period of time, the watchdog increments a failure rating for that node. A node is deemed to be misbehaving when this failure rating exceeds a certain threshold. The pathrater then uses the gathered information to determine the best possible routes by avoiding misbehaving nodes. This mechanism, which does not punish these nodes (it actually relieves them from forwarding operations), provides an increase in the throughput of networks with misbehaving nodes [79].

### 5.2.3 Hypothesis Testing

Hypothesis testing is a method of deciding which of the two hypotheses, denoted $H_0$ and $H_1$, is true given an observation denoted as $U$. In this chapter, $H_0$ is the hypothesis that $v$ is well-behaving, $H_1$ is that $v$ is malicious, and $U$ is the information gathered from overhearing. The observation $U$ is distributed differently depending whether $H_0$ or $H_1$ is true, and these distributions are denoted as $P_{U|H_0}$ and $P_{U|H_1}$, respectively.

An algorithm is used to choose between the hypotheses given the observation $U$. There are two types of error associated with the decision process, as shown below.

- *Type 1 error, False detection*: Accepting $H_1$ when $H_0$ is true (*i.e.* considering a well-behaving $v$ to be malicious), and the probability of this event is denoted $\gamma$.
- *Type 2 error, Misdetection*: Accepting $H_0$ when $H_1$ is true (*i.e.* considering a malicious $v$ to be well-behaving), and the probability of this event is denoted $\beta$.

The Neyman-Pearson theorem gives the optimal decision rule: Given the maximal tolerable $\beta$, we can minimize $\gamma$ by accepting $H_0$ if and only if $\log \frac{P_{U|H_0}}{P_{U|H_1}} \geq t$ for some threshold $t$ dependent on $\gamma$. For more thorough survey on hypothesis testing in the context of authentication, see [80].

## 5.3 Problem Statement

We shall use elements from a field (in italic font, *e.g.* $x$), and their bit-representation (in bold font, *e.g.* $\mathbf{x}$). We use underscore bold font (*e.g.* $\underline{\mathbf{x}}$) for vectors. For arithmetic operations in the field, we shall use the conventional notation $(+, -, \cdot)$. For bit-operation, we shall use $\oplus$ for addition, and $\otimes$ for multiplication.

We use polynomial hash functions defined as follows [18].

**Definition 5.3.1 (*Polynomial hash functions*)** *For a finite field* $\mathbf{F}$ *and* $d \geq 1$*, the class of polynomial hash functions on* $\mathbf{F}$ *is defined as* $\mathcal{H}^d(\mathbf{F}) = \{h_a | a = \langle a_0, ..., a_d \rangle \in \mathbf{F}^{d+1}\}$*, where* $h_a(x) = \sum_{i=0}^{d} a_i x^i$ *for* $x \in \mathbf{F}$*.*

We model a wireless network with a hypergraph $G = (V, E_1, E_2)$, where $V$ is the set of the nodes in the network, $E_1$ is the set of hyperedges representing the connectivity (wireless

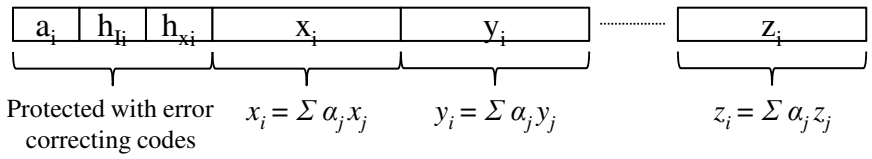



Figure 5-2: Packet structure for Algebraic Watchdog. The figure presents the structure of a valid packet sent by well-behaving $v_i$.

links), and $E_2$ is the set of hyperedges representing the interference. We assume that there are a set of sources and a set of destinations, which are well-behaved. We use the hypergraph to capture the broadcast nature of the wireless medium. If $(v_1, v_2) \in E_1$ and $(v_1, v_3) \in E_2$ where $v_1, v_2, v_3 \in V$, then there is an intended transmission from $v_1$ to $v_2$, and $v_3$ can overhear this transmission (possibly incorrectly).

A node $v_i \in V$ transmits coded information $x_i$ by transmitting a packet $\underline{\mathbf{p_i}}$, where $\underline{\mathbf{p_i}} = [\mathbf{a_i}, \mathbf{h_{I_i}}, \mathbf{h_{x_i}}, \mathbf{x_i}]$ is a $\{0, 1\}$-vector. A valid packet $\underline{\mathbf{p_i}}$ is defined as below:

- $\mathbf{a_i}$ corresponds to the coding coefficients $\alpha_j$, $j \in I_i$, where $I_i \subseteq V$ is the set of nodes adjacent to $v_i$ in $E_1$,
- $\mathbf{h_{I_i}}$ corresponds to the hash $h(x_j)$, $v_j \in I_i$ where $h(\cdot)$ is a $\delta$-bit polynomial hash function,
- $\mathbf{h_{x_i}}$ corresponds to the polynomial hash $h(x_i)$,
- $\mathbf{x_i}$ is the $n$-bit representation of $x_i = \sum_{j \in I} \alpha_j x_j \in \mathbb{F}_{2^n}$.

Figure 5-2 illustrates the structure of a valid packet. For simplicity, we assume the payload to be a single symbol. However, the protocol and the analysis extends to packets with multiple symbols. For example, in Figure 5-2, the payload consists of multiple symbols. In order to apply the Algebraic Watchdog protocol to the entire payload, we can call on the same protocol on $\mathbf{x_i}$, $\mathbf{y_i}$, ..., and $\mathbf{z_i}$ separately. Therefore, the computational complexity grows linearly with the payload. Furthermore, as we shall discuss in Section 5.5, separate calls to Algebraic Watchdog can use the same precomputed tables and data structures; therefore, the protocol can scale efficiently with the payload length.

The payload $\mathbf{x_i}$ may be coded with a $(n, k_i)$-code $\mathcal{C}_i$ with minimum distance $d_i$. Code $\mathcal{C}_i$ is an error-correcting code of rate $R_i = \frac{k_i}{n} = 1 - \frac{d_i}{n}$, and is tailored for the forward communication. For instance, $v_1$ uses code $\mathcal{C}_1$ chosen appropriately for the channel $(v_1, v_j) \in E_1$, to transmit $\mathbf{x_1}$.

As we shall show in Section 5.8, the Algebraic Watchdog does not need the hash function to work (*i.e.* $\delta$ can be 0). It can be observed that the larger the hash function, the better the performance of Algebraic Watchdog. We assume that the hash function $h(\cdot)$ is known to all nodes, including the adversary.

We assume that $\mathbf{a_i}$, $\mathbf{h_{I_i}}$ and $\mathbf{h_{x_i}}$ are part of the header information, and are sufficiently coded such that the previous hop nodes can correctly receive them even under noisy channel conditions. For example, in Figure 5-3a, nodes $v_1$ to $v_m$ should be able to decode the header from $v_{m+1}$. In practice, a node just needs to obtain $\mathbf{a_{m+1}}$, $\mathbf{h_{I_{m+1}}}$ and $\mathbf{h_{x_{m+1}}}$ from any nodes; however, for simplicity, we assume that a node overhears them correctly from its next hop node. Protecting the header sufficiently to allow the parent nodes to receive it correctly will induce some overhead, but the assumption remains a reasonable one to make. First, the header is smaller than the message itself. Second, even in the routing case, the header and the state information need to be coded sufficiently. Third, the hashes $\mathbf{h_{I_i}}$ and $\mathbf{h_{x_i}}$ are contained within one hop. A node that receives $\underline{\mathbf{p_i}} = [\mathbf{a_i}, \mathbf{h_{I_i}}, \mathbf{h_{x_i}}, \mathbf{x_i}]$ does not need to repeat $\mathbf{h_{I_i}}$, only $\mathbf{h_{x_i}}$. Therefore, the overhead associated with the hashes is proportional to the in-degree of a node, and does not accumulate with the routing path length.

There are certain transition probabilities associated with the interference channels known to the nodes, which applies to all packets (payload and header) except for the header between two consecutive hop nodes as previously mentioned. They are modeled as binary channels, $BSC(p_{ij})$ for $(v_i, v_j) \in E_2$. Therefore, $p_{ij} \in [0, 1]$ represents the bit-error rate of the overhearing channel. If $p_{ij} = 0$, then $v_j$ can hear $v_i$'s transmission noiselessly; if $p_{ij} = \frac{1}{2}$, then $v_j$ cannot overhear anything from $v_i$. Our formulation does not require that each node overhear all its neighbors; however, the more a node can overhear, the better the performance of the protocol.

Assume that $v_i$ transmits $\underline{\mathbf{p_i}} = [\mathbf{a_i}, \mathbf{h_{I_i}}, \mathbf{h_{x_i}}, \hat{\mathbf{x}}_\mathbf{i}]$, where $\hat{\mathbf{x}}_\mathbf{i} = \mathbf{x_i} \oplus \mathbf{e}$, $\mathbf{e} \in \{0, 1\}^n$. If $v_i$ is misbehaving, then $\mathbf{e} \neq 0$. Our goal is to detect with high probability when $\mathbf{e} \neq 0$.

Even if $|\mathbf{e}|$ is small (*i.e.* the hamming distance between $\hat{\mathbf{x}}_\mathbf{i}$ and $\mathbf{x}_\mathbf{i}$ is small), the algebraic interpretation of $\hat{\mathbf{x}}_\mathbf{i}$ and $\mathbf{x}_\mathbf{i}$ may differ significantly. For example, consider $n = 4$, $\hat{\mathbf{x}}_\mathbf{i} = [0000]$, and $\mathbf{x}_\mathbf{i} = [1000]$. Then, $\mathbf{e} = [1000]$ and $|\mathbf{e}| = 1$. However, the algebraic interpretations of $\hat{\mathbf{x}}_\mathbf{i}$ and $\mathbf{x}_\mathbf{i}$ are 0 and 8, respectively. Thus, even a single bit flip can alter the message significantly.

### 5.3.1 Threat Model

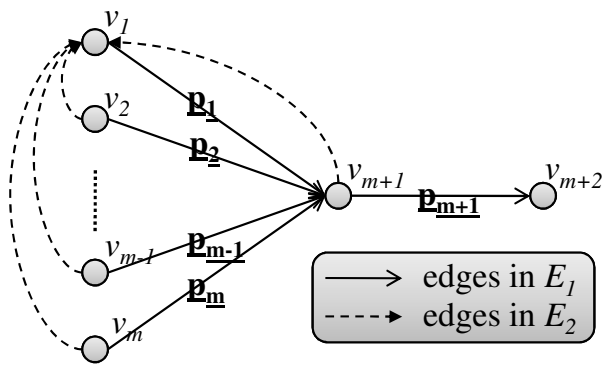
We assume powerful adversaries, who can eavesdrop their neighbor's transmissions, have the power to inject or corrupt packets, and are computationally unbounded. Therefore, the adversary will find $\hat{\mathbf{x}}_\mathbf{i}$ that will allow its misbehavior to be undetected, if there is any such $\hat{\mathbf{x}}_\mathbf{i}$. However, the adversary does not know the specific realization of the random errors introduced by the channels. We denote the rate at which an adversary injects error (*i.e.* performs bit flips to the payload) to be $p_{adv}$. The adversaries' objective is to corrupt the information flow without being detected by other nodes.

Our goal is to detect probabilistically a malicious behavior that is beyond the channel noise, represented by $BSC(p_{ik})$. The Algebraic Watchdog does not completely eliminate errors introduced by the adversaries; its objective is to limit the adversarial errors to be at most that of the channel. Channel errors (or adversaries errors below the channel noise level) can be corrected using appropriate error correction schemes, which will be necessary even without Byzantine adversaries in the network.

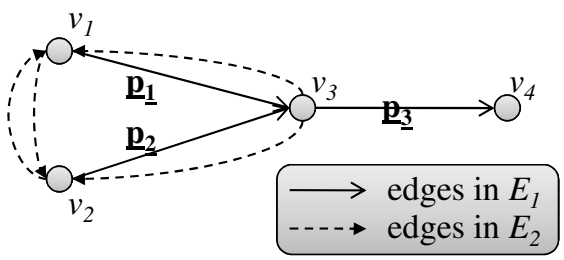
The notion that adversarial errors should sometimes be treated as channel noise has been introduced previously in [58]. Under heavy attack (where the adversary can corrupt significant portion of the data traffic), attacks should be treated with special attention; while under light attack, the attacks can be treated as noise and corrected using error-correction schemes. The results in this chapter partially reiterate this idea.

## 5.4 Two-hop network: An Example

Consider a network or a small neighborhood as in Figure 5-3a. Nodes $v_i$, $i \in [1, m]$, want to send $x_i$ to $v_{m+2}$ via $v_{m+1}$. A single node $v_i$, $i \in [1, m]$, cannot monitor $v_{m+1}$ even if $v_i$




(a) General $m$




(b) $m = 2$

Figure 5-3: A small neighborhood of a wireless network with $v_1$. The dotted arrows represent the overhearing channels, and the solid arrows represent the transmissions.

overhears $\mathbf{x_{m+1}}$, since without any information about $x_j$ for $j \in [1, m]$, $x_{m+1}$ is completely random to $v_i$. In contrast, if $v_i$ knows $x_j$ for all $j \in [1, m+1]$, then $v_i$ can verify that $v_{m+1}$ is behaving with certainty; however, this requires at least $m - 1$ additional reliable transmissions to $v_i$.

In Figure 5-3a, we use the solid lines to represent the intended channels $E_1$, and dotted lines for the interference channels $E_2$. Each node checks whether its neighbors are transmitting values that are consistent with the gathered information. If a node detects that its neighbor is misbehaving, then it can alert other nodes in the network and isolate the misbehaving node.

In the next subsections, we use an example with $m = 2$, as shown Figure 5-3b. We introduce the graphical model which explains how a node $v_i$ checks its neighbor's behavior. Then, we use an algebraic approach to analyze and compute $\gamma$ and $\beta$ for this example network with $m = 2$. We assume for simplicity that nodes do not code the payload – *i.e.*

error-correcting code has rate $R_i = 1$.

A malicious $v_3$ would not inject errors in $\mathbf{h_{x_3}}$ only, as the destination $v_4$ can easily verify if $\mathbf{h_{x_3}}$ is equal to $h(\mathbf{x_3})$. Therefore, a malicious $v_3$ is forced to transmit a packet such that $\mathbf{h_{x_3}} = h(\mathbf{x_3})$. In addition, $v_3$ would not inject errors in $\mathbf{h_{x_j}}$, $j \in I_3$, as each $v_j$ can verify the hash of its message. However, a malicious $v_3$ can inject errors in $\mathbf{a_3}$, forcing $v_4$ to receive incorrect coefficients $\tilde{\alpha}_j$'s instead of $\alpha_j$'s. Notice that any error introduced in $\mathbf{a_3}$ can be translated to errors in $\mathbf{x_3}$ by assuming that $\tilde{\alpha}_j$'s are the correct coding coefficients. Therefore, we are concerned only with the case in which $v_3$ introduces errors in $\mathbf{x_3}$ (and in $\mathbf{h_{x_3}}$ such that $\mathbf{h_{x_3}} = h(\mathbf{x_3})$).

### 5.4.1 Graphical Model Approach

We present a graphical approach to model the problem for $m = 2$ systematically, and to explain how a node may check its neighbors. This approach may be advantageous as it lends easily to already existing graphical model algorithms as well as some approximation algorithms.

We shall consider the problem from $v_1$'s perspective. We denote $\tilde{x_i}$ to be what $v_1$ overhears $v_i$ transmitting. Note that $\tilde{x_i}$ may have bit errors introduced by the interference channel $BSC(p_{i1})$. As shown in Figure 5-4, the graphical model has four layers:

- *Layer 1* contains $2^{n+h}$ vertices, each representing a bit-representation of $[\tilde{\mathbf{x}}_\mathbf{2}, \mathbf{h}(\mathbf{x_2})]$;
- *Layer 2* contains $2^n$ vertices, each representing a bit-representation of $\mathbf{x_2}$;
- *Layer 3* contains $2^n$ vertices corresponding to $\mathbf{x_3}$; and
- *Layer 4* contains $2^{n+h}$ vertices corresponding to $[\tilde{\mathbf{x}}_\mathbf{3}, \mathbf{h}(\mathbf{x_3})]$.

Node $v_1$ overhears the transmissions from $v_2$ to $v_3$ and from $v_3$ to $v_4$; therefore, it receives $[\tilde{\mathbf{x}}_\mathbf{2}, \mathbf{h}(\mathbf{x_2})]$ and $[\tilde{\mathbf{x}}_\mathbf{3}, \mathbf{h}(\mathbf{x_3})]$, corresponding to the *starting point* in Layer 1 and the *destination point* in Layer 4 respectively. We add edges to the graphical model in a manner such that the sum of the product of the weights of all possible paths between the starting and the destination points is equal to the probability that $v_3$ is consistent with the information gathered by $v_1$. Edges exist between adjacent layers as follows.

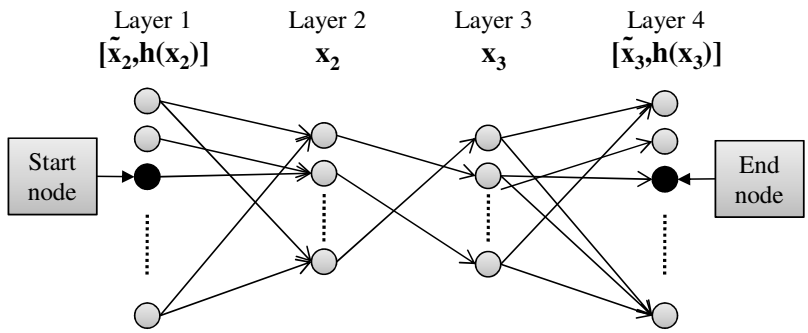



Figure 5-4: A graphical model of the inference process from $v_1$'s perspective for the two-hop example network.

- *Layer 1 to Layer 2:* An edge exists between a vertex $[\mathbf{v}, \mathbf{u}]$ in Layer 1 and a vertex $\mathbf{w}$ in Layer 2 if and only if $\mathbf{h}(\mathbf{w}) = \mathbf{u}$. The edge weight is normalized such that the total weight of edges leaving $[\mathbf{v}, \mathbf{u}]$ is 1, and the weight is proportional to:

$$\mathbf{P}(\mathbf{v}| \text{ Channel statistics and } \mathbf{w} \text{ is the original message}),$$

  which is the probability that the inference channel outputs message $\mathbf{v}$ given an input message $\mathbf{w}$.

- *Layer 2 to Layer 3:* The edges represent a permutation. A vertex $\mathbf{v}$ in Layer 2 is adjacent to a vertex $\mathbf{w}$ in Layer 3 if and only if $w = c + \alpha_2 v$, where $c = \alpha_1 x_1$ is a constant, $\mathbf{v}$ and $\mathbf{w}$ are the bit-representation of $v$ and $w$, respectively. The edge weights are all 1.

- *Layer 3 to Layer 4:* An edge exists between a vertex $\mathbf{v}$ in Layer 3 and a vertex $[\mathbf{w}, \mathbf{u}]$ in Layer 4 if and only if $\mathbf{h}(\mathbf{v}) = \mathbf{u}$. The edge weight is normalized such that the total weight leaving $\mathbf{v}$ is 1, and is proportional to:

$$\mathbf{P}(\mathbf{w}| \text{ Channel statistics and } \mathbf{v} \text{ is the original message}).$$

This graphical model illustrates sequentially and visually the inference process $v_1$ exe-

cutes. Furthermore, by using approximation algorithms and pruning algorithms, we may be able to simplify the computation as well as the structure of the graph. In addition, the graphical approach may be extend to larger networks, as we shall discuss in Section 5.5.

### 5.4.2 Algebraic Approach

We explain the inference process described above using the graphical model introduced in Section 5.4.1. Consider $v_1$. By assumption, $v_1$ correctly receives $\mathbf{a_3}$, $\mathbf{h_{I_3}}$, and $\mathbf{h_{x_3}}$ from $v_3$, which contain $\alpha_1, \alpha_2, h(x_2)$, and $h(x_3)$. In addition, $v_1$ receives $\tilde{\mathbf{x}}_\mathbf{2} = \mathbf{x_2} + \mathbf{e}'$ and $\tilde{\mathbf{x}}_\mathbf{3} = \mathbf{x_3} + \mathbf{e}''$, where $\mathbf{e}'$ and $\mathbf{e}''$ are outcomes of the interference channels. Given $\tilde{\mathbf{x}}_\mathbf{j}$ for $j = \{2, 3\}$ and the transition probabilities, $v_1$ computes $r_{j\to 1}$ such that the sum of the probability that the interference channel from $v_j$ and $v_1$ outputs $\tilde{\mathbf{x}}_\mathbf{j}$ given $\mathbf{x} \in B(\tilde{\mathbf{x}}_\mathbf{j}, r_{j\to 1})$ is greater or equal to $1 - \epsilon$ where $\epsilon$ is a constant, and $B(\mathbf{x}, r)$ is a $n$-dimensional ball of radius $r$ centered at $\mathbf{x}$. Now, $v_1$ computes $\tilde{X}_j = \{\mathbf{x} \mid h(x) = h(x_j)\} \cap B(\tilde{\mathbf{x}}_\mathbf{j}, r_{j\to 1})$ for $j = \{2, 3\}$. Then, $v_1$ computes $\alpha_1 x_1 + \alpha_2 \hat{x}$ for all $\hat{\mathbf{x}} \in \tilde{X}_2$. Then, $v_1$ intersects $\tilde{X}_3$ and the computed $\alpha_1 x_1 + \alpha_2 \hat{x}$'s. If the intersection is empty, then $v_1$ claims that $R$ is misbehaving.

The set $\{\mathbf{x} \mid h(x) = h(x_2)\}$ represents the Layer 2 vertices reachable from the starting point $[\tilde{\mathbf{x}}_\mathbf{2}, \mathbf{h}(\mathbf{x_2})]$ in Layer 1, and $\tilde{X}_2$ is a subset of the reachable Layer 2 vertices such that the total edge weight (corresponding to the transition probability) from the starting point is greater than $1-\epsilon$. Then, computing $\alpha_1 x_1 + \alpha_2 \hat{x}$ represents the permutation from Layers 2 to 3. Finally, the intersection with $\tilde{X}_3$ results in a set of Layer 3 vertices such that they are adjacent to the destination point $[\tilde{\mathbf{x}}_\mathbf{3}, \mathbf{h}(\mathbf{x_3})]$ in Layer 4 and their total transition probability to $[\tilde{\mathbf{x}}_\mathbf{3}, \mathbf{h}(\mathbf{x_3})]$ is greater than $1 - \epsilon$.

**Lemma 5.4.1** *For $n$ sufficiently large, the probability of false detection, $\gamma \leq \epsilon$ for any arbitrary small constant $\epsilon$.*

*Proof:* Assume that $v_3$ is not malicious. Then, for $n$ sufficiently large, $v_1$ can choose $r_{2\to 1}$ and $r_{3\to 1}$ such that the probability that the bit representation of $x_3 = \alpha_1 x_1 + \alpha_2 x_2$ is in $\tilde{X}_3$ and the probability that $\mathbf{x_2} \in \tilde{X}_2$ are greater than $1 - \epsilon$. Therefore, $\tilde{X}_3 \cap \{\alpha_1 x_1 + \alpha_2 \hat{x} \mid \forall \hat{\mathbf{x}} \in \tilde{X}_2\} \neq \emptyset$ with probability arbitrary close to 1. Thus, a well-behaving $v_3$ passes $v_1$'s check with probability at least $1 - \epsilon$. This shows that $\gamma \leq \epsilon$. ■

**Lemma 5.4.2** *The probability that a malicious $v_3$ is undetected from $v_1$'s perspective is given by*

$$\min\left\{1, \frac{\sum_{k=0}^{r_{1\to 2}} \binom{n}{k}}{2^{(\delta+n)}} \cdot \frac{\sum_{k=0}^{r_{2\to 1}} \binom{n}{k}}{2^{(\delta+n)}} \cdot \frac{\sum_{k=0}^{r_{3\to 1}} \binom{n}{k}}{2^{\delta}}\right\}. \tag{5.2}$$

*Proof:* Assume that $v_3$ is malicious and injects errors into $\mathbf{x_3}$. Consider an element $\mathbf{z} \in \tilde{X}_3$, where $z = \alpha_1 x_1 + \alpha_2 x_2 + e = \alpha_1 x_1 + \alpha_2(x_2 + e_2)$ for some $e$ and $e_2$. Note that, since we are using a field of size $2^n$, multiplying an element from the field by a randomly chosen constant has the effect of randomizing the product. Here, we consider two cases:

- *Case 1:* If $x_2 + e_2 \notin \tilde{X}_2$, then $v_3$ fails $v_1$'s check.
- *Case 2:* If $x_2 + e_2 \in \tilde{X}_2$, then $v_3$ passes $v_1$'s check, but $v_3$ is unlikely to pass $v_2$'s check. Note that $\alpha_1 x_1 + \alpha_2(x_2 + e_2) = \alpha_1(x_1 + e_1) + \alpha_2 x_2$ for some $e_1$. For uniformly random $\alpha_1$ and $\alpha_2$, $e_1$ is also uniformly random. Therefore, $v_3$ will pass if the random vector $x_1 + e_1$ belongs to $\tilde{X}_1 = \{x \mid h(x) = h(x_1)\} \cap B(\tilde{\mathbf{x}}_\mathbf{1}, r_{1\to 2})$. Therefore,

$$\mathbf{P}(\text{A malicious } v_3 \text{ passes } v_2\text{'s check}) = \mathbf{P}(x_1 + e_1 \in \tilde{X}_1) \tag{5.3}$$

$$= \frac{Vol(\tilde{X}_1)}{2^n}, \tag{5.4}$$

where $Vol(\cdot)$ is the number of $\{0,1\}$-vectors in the given set. Since $Vol(B(x,r)) = \sum_{k=0}^{r} \binom{n}{k} \leq 2^n$ and the probability that $h(x)$ equals a given value is $\frac{1}{2^\delta}$, $Vol(\tilde{X}_1)$ is given as follows:

$$Vol(\tilde{X}_1) = \frac{Vol(B(\tilde{x}_1, r_{1\to 2}))}{2^\delta} = \frac{\sum_{k=0}^{r_1\to 2} \binom{n}{k}}{2^\delta}. \tag{5.5}$$

Hence, putting the observations from the above two cases, the probability that a $\mathbf{z} \in \tilde{X}_3$ passes the checks from $v_1$'s perspective is

$$\mathbf{P}(\mathbf{z} \text{ passes check}) = 0 \cdot \mathbf{P}(x_2 + e_2 \notin \tilde{X}_2) + \frac{\sum_{k=0}^{r_{1\to 2}} \binom{n}{k}}{2^{(\delta+n)}} \cdot \mathbf{P}(x_2 + e_2 \in \tilde{X}_2). \tag{5.6}$$

Similarly,

$$\mathbf{P}(x_2 + e_2 \in \tilde{X}_2) = \frac{\sum_{k=0}^{r_{2\to 1}} \binom{n}{k}}{2^{(\delta+n)}}, \tag{5.7}$$

and

$$Vol(\tilde{X}_3) = \frac{\sum_{k=0}^{r_{3\to1}} \binom{n}{k}}{2^{\delta}}. \tag{5.8}$$

Then, the probability that $v_3$ is undetected from $v_1$'s perspective is the probability that *at least one* $\mathbf{z} \in \tilde{X}_3$ passes the check, which is equal to

$$\begin{aligned}
&\mathbf{P}(\text{A malicious } v_3 \text{ is undetected from } v_1\text{'s perspective}) && (5.9)\\
&= \min\{1, \mathbf{P}(\mathbf{z} \text{ passes check}) \cdot Vol(\tilde{X}_3)\} && (5.10)\\
&= \min\left\{1, \frac{\sum_{k=0}^{r_{1\to2}} \binom{n}{k}}{2^{(\delta+n)}} \cdot \frac{\sum_{k=0}^{r_{2\to1}} \binom{n}{k}}{2^{(\delta+n)}} \cdot \frac{\sum_{k=0}^{r_{3\to1}} \binom{n}{k}}{2^{\delta}}\right\}. && (5.11)
\end{aligned}$$

Note that $\mathbf{P}(\mathbf{z} \text{ passes check}) \cdot Vol(\tilde{X}_3)$ is the expected number of $\mathbf{z} \in \tilde{X}_3$ that passes the check; thus, given a high enough $\mathbf{P}(\mathbf{z} \text{ passes check})$, would exceed 1. Therefore, we take the minimum of 1 and $\mathbf{P}(\mathbf{z} \text{ passes check}) \cdot Vol(\tilde{X}_3)\}$ to get a valid probability. This proves the statement. ■

**Lemma 5.4.3** *The probability that a malicious $v_3$ is undetected from $v_2$'s perspective is given by*

$$\min\left\{1, \frac{\sum_{k=0}^{r_{1\to2}} \binom{n}{k}}{2^{(\delta+n)}} \cdot \frac{\sum_{k=0}^{r_{2\to1}} \binom{n}{k}}{2^{(\delta+n)}} \cdot \frac{\sum_{k=0}^{r_{3\to2}} \binom{n}{k}}{2^{\delta}}\right\}. \tag{5.12}$$

*where $v_2$ overhears $\tilde{\mathbf{x}}_3$ from $v_3$, and the probability that the interference channel from $v_3$ to $v_2$ outputs $\tilde{\mathbf{x}}_3$ given $\mathbf{x} \in B(\tilde{\mathbf{x}}_3, r_{3\to2})$ is greater than $1-\epsilon$.*

*Proof:* This statement can be proven by an analysis similar to in the proof for Lemma 5.4.2. ■

**Theorem 5.4.4** *The probability of misdetection, $\beta$, is*

$$\beta = \min\left\{1, \frac{\sum_{k=0}^{r_{1\to2}} \binom{n}{k}}{2^{(\delta+n)}} \cdot \frac{\sum_{k=0}^{r_{2\to1}} \binom{n}{k}}{2^{(\delta+n)}} \cdot \frac{1}{2^{\delta}} \sum_{k=0}^{r} \binom{n}{k}\right\}, \tag{5.13}$$

*where $r = \min\{r_{3\to1}, r_{3\to2}\}$.*

*Proof:* The probability of misdetection is the minimum of the probability that $v_1$ and $v_2$ do not detect a malicious $v_3$. Therefore, by Lemma 5.4.2 and 5.4.3, the statement is true. ■

Theorem 5.4.4 shows that the probability of misdetection $\beta$ decreases with the hash size $\delta$, as $\delta$ restricts the space of consistent codewords. Since $r_{i\to j}$ represents the uncertainty introduced by the interference channels, $\beta$ increases with it. Interestingly, $\beta$ decreases with $n$, since $\sum_{k=0}^{r}\binom{n}{k} < 2^n$ for $r < n$. This is because network coding randomizes the messages over a field whose size increases exponentially with $n$. This makes it difficult for an adversary to introduce errors without introducing inconsistencies.

We can apply Theorem 5.4.4 even when $v_1$ and $v_2$ cannot overhear each other. Then, $r_{1\to 2} = r_{2\to 1} = n$, giving

$$\beta = \min\left\{1, \frac{\sum_{k=0}^{r}\binom{n}{k}}{8^{\delta}}\right\} \tag{5.14}$$

where $r = \min\{r_{3\to 1}, r_{3\to 2}\}$. Here, $\beta$ highly depends on $\delta$, as $v_1$ and $v_2$ are only using their own messages and the overheard hashes from $v_3$.

The algebraic approach results in an analysis with exact formulae for $\gamma$ and $\beta$. These formulae are conditional probabilities; as a result, they hold regardless of a priori knowledge of whether $v_3$ is malicious or not. However, performing algebraic analysis is not very extensible with growing $m$, the number of nodes in the network. Therefore, as we extend the Algebraic Watchdog to a more general network topology, we shall focus on the graphical model.

## 5.5 Algebraic Watchdog for Two-hop Network

We extend the Algebraic Watchdog to a more general two-hop network as in Figure 5-3a. We develop upon the trellis introduced in Section 5.4, and formally present a graphical representation of the inference process performed by a node performing Algebraic Watchdog on its downstream neighbor.

There are three steps in performing the Algebraic Watchdog. First, we infer the original messages from the overheard data, which is captured by the transition matrix in Section 5.5.1. The second step consists of forming an opinion regarding what the next-hop node $v_{m+1}$ should be sending, which is inferred using a trellis structure in Section 5.5.2 and a Viterbi-like algorithm in Section 5.5.3. Finally, we combine the inferred information with what we overhear from $v_{m+1}$ to make a decision on $v_{m+1}$'s behavior as discussed in Section

5.5.4. Figures 5-5, 5-6, and 5-7 illustrate these three steps.

### 5.5.1 Transition Matrix

We define a *transition matrix* $T_i$ to be a $2^{n(1-H(\frac{d_i}{n}))+\delta} \times 2^{n(1-H(\frac{d_i}{n}))}$ matrix, where $H(\cdot)$ is the entropy function.

$$T_i(\tilde{x}_i, y) = \begin{cases} \frac{p_i(\tilde{x}_i, y)}{\mathcal{N}} & \text{if } h(y) = h(x_i), \\ 0 & \text{otherwise,} \end{cases} \tag{5.15}$$

where

$$p_i(\tilde{x}_i, y) = p_{i1}^{\Delta(\tilde{\mathbf{x}}_\mathbf{i}, \mathbf{y})}(1 - p_{i1})^{n - \Delta(\tilde{\mathbf{x}}_\mathbf{i}, \mathbf{y})}, \tag{5.16}$$

$$\mathcal{N} = \sum_{\{y | h(y) = h(x_i)\}} p_i(\tilde{x}_i, y), \tag{5.17}$$

$$\Delta(\mathbf{x}, \mathbf{y}) \text{ gives the Hamming distance between codewords } \mathbf{x} \text{ and } \mathbf{y}. \tag{5.18}$$

In other words, $v_1$ computes $\tilde{X}_i = \{x | h(x) = h(x_i)\}$ to be the list of *candidates* of $x_i$. For any overheard pair $[\tilde{\mathbf{x}}_\mathbf{i}, \mathbf{h}(\mathbf{x}_\mathbf{i})]$, there are multiple candidates of $x_i$ (*i.e.* $|\tilde{X}_i|$) although the probabilities associated with each candidates are different. This is because there are uncertainties associated with the overhearing channel, represented by $BSC(p_{i1})$.

For each $x \in \tilde{X}_i$, $p_i(\tilde{x}_i, x)$ gives the probability of $x$ being the original codeword sent by node $v_i$ given that $v_1$ overheard $\tilde{x}_i$ under $BSC(p_{i1})$. Since we are only considering $x \in \tilde{X}_i$, we normalize the probabilities using $\mathcal{N}$ to get the *transition probability* $T_i(\tilde{x}_i, x)$. Note $T_i(\tilde{x}_i, y) = 0$ if $h(y) \neq h(x_i)$.

The structure of $T_i$ heavily depends on the collisions of the hash function $h(\cdot)$ in use. The structure of $T_i$ is independent of $i$, and therefore, a single transition matrix $T$ can be precomputed for all $i \in [1, m]$ given the hash function $h(\cdot)$. A graphical representation of $T$ is shown in Figure 5-5. For simplicity of notation, we represent $T$ as a matrix; however, the transition probabilities can be computed efficiently using hash collision lists as well.

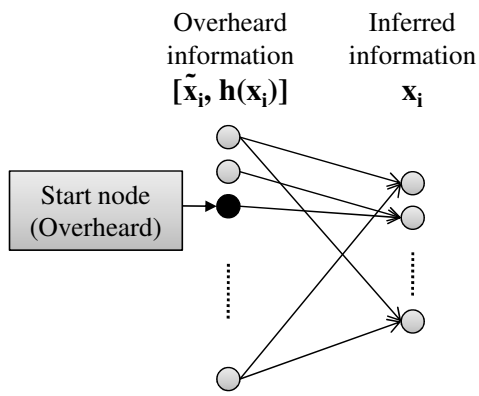



Figure 5-5: An example of the transition matrix $T(\tilde{x}_i, y)$. A graphical representation of the inference process at a node which overhears node $v_i$'s transmission. The overheard information is $[\tilde{x}_i, h(x_i)]$, from which the node infers what $x_i$ may be.

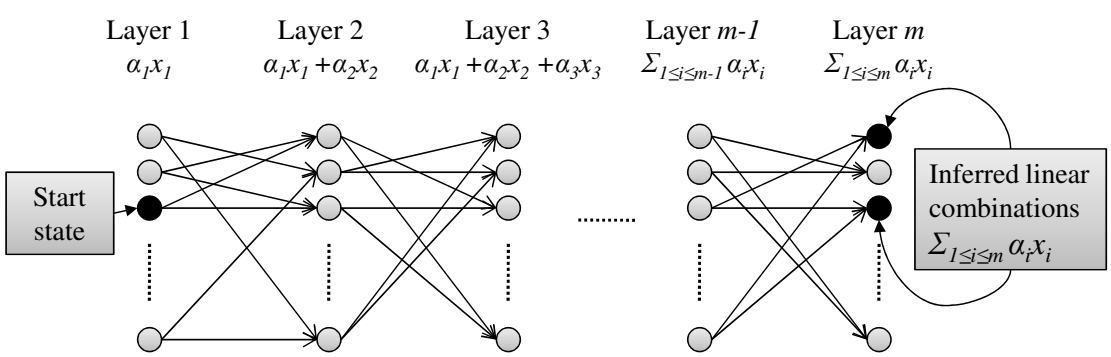



Figure 5-6: An example of the trellis for Algebraic Watchdog. In the trellis, the transition probability from Layer $i-1$ to Layer $i$ is given by $T_i(\tilde{x}_i, x_i)$, which is shown in Figure 5-5.

### 5.5.2 Watchdog Trellis

Node $v_1$ uses the information gathered to generate a trellis, which is used to infer the valid linear combination that $v_{m+1}$ should transmit to $v_{m+2}$. As shown in Figure 5-6, the trellis has $m$ layers: each layer may contain up to $2^n$ states, each representing the inferred linear combination so far. For example, Layer $i$ consist of all possible values of $\sum_{j=1}^{i} \alpha_j x_j$.

The matrices $T_i, i \in [2, m]$, defines the connectivity of the trellis. Let $s_1$ and $s_2$ be states in Layer $i-1$ and Layer $i$, respectively. Then, an edge $(s_1, s_2)$ exists between $s_1$ and $s_2$ if and only if

$$\exists\ x \text{ such that } s_1 + \alpha_i x = s_2,\ T_i(\tilde{x}_i, x) \neq 0. \tag{5.19}$$

We denote $w_e(\cdot, \cdot)$ to be the edge weight, where $w_e(s_1, s_2) = T_i(\tilde{x}_i, x)$ if edge $(s_1, s_2)$ exists, and zero otherwise.

### 5.5.3 Viterbi-like Algorithm

We denote $w(s, i)$ to be the weight of state $s$ in Layer $i$. Node $v_1$ selects a *start state* in Layer 1 corresponding to $\alpha_1 x_1$, as in Figure 5-6. The weight of Layer 1 is $w(s, 1) = 1$ if $s = \alpha_1 x_1$, zero otherwise. For the subsequent layers, multiple paths can lead to a given state, and the algorithm keeps the aggregate probability of reaching that state. To be more precise,

$$w(s, i) = \sum_{\forall s' \in \text{Layer } i-1} w(s', i-1) \cdot w_e(s', s). \tag{5.20}$$

By definition, $w(s, i)$ is equal to the total probability of $s = \sum_{j=1}^{i} \alpha_j x_j$ given the overheard information. Therefore, $w(s, m)$ gives the probability that $s$ is the valid linear combination that $v_{m+1}$ should transmit to $v_{m+2}$. It is important to note that $w(s, m)$ is dependent on the channel statistics, as well as the overheard information. For some states $s$, $w(s, m) = 0$, which indicates that state $s$ can not be a valid linear combination; only those states $s$ with $w(s, m) > 0$ are the *inferred candidate linear combinations*.

The algorithm introduced above is a dynamic program, and is similar to the Viterbi algorithm. Therefore, tools developed for dynamic programming or Viterbi algorithm can be used to compute the probabilities efficiently.

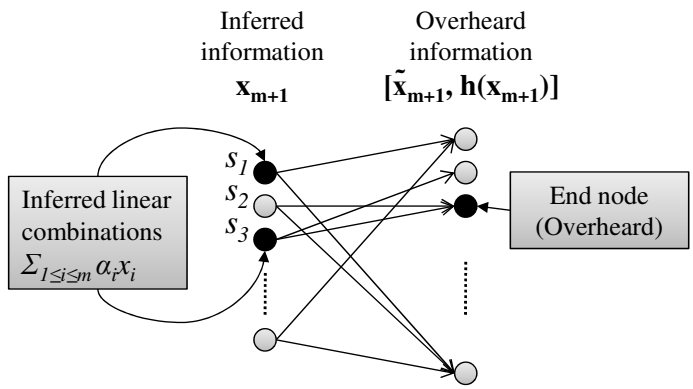



Figure 5-7: An example of the inverse transition matrix $T^{-1}(y, \tilde{x}_{m+1})$. A graphical representation of the inference process at a node which overhears relay $v_{m+1}$'s transmission. The overheard information is $[\tilde{x}_{m+1}, h(x_{m+1})]$, to which the node compares what it has inferred.

### 5.5.4 Decision Making

Node $v_1$ computes the probability that the overheard $\tilde{x}_{m+1}$ and $h(x_{m+1})$ are consistent with the inferred $w(\cdot, m)$ to make a decision regarding $v_{m+1}$'s behavior. To do so, $v_1$ constructs an *inverse transition matrix* $T^{-1}$, which is a $2^{n(1-\frac{d_{m+1}}{n})} \times 2^{n(1-\frac{d_{m+1}}{n})+\delta}$ matrix whose elements are defined as follows:

$$T^{-1}(y,\tilde{x}_{m+1}) = \begin{cases} \frac{p_{m+1}(\tilde{x}_{m+1},y)}{\mathcal{M}}, & \text{if } h(y) = h(x_{m+1}) \\ 0, & \text{otherwise} \end{cases}, \tag{5.21}$$

$$\text{where } \mathcal{M} = \sum_{\{y|h(y)=h(x_{m+1})\}} p_{m+1}(\tilde{x}_{m+1}, y). \tag{5.22}$$

Unlike $T$ introduced in Section 5.5.1, $T^{-1}(x, \tilde{x}_{m+1})$ gives the probability of overhearing $[\tilde{x}_{m+1}, h(x_{m+1})]$ given that $x \in \{y|h(y) = h(x_{m+1})\}$ is the original codeword sent by $v_{m+1}$ and the channel statistics. The inverse transition matrix $T^{-1}$ is identical to the transition matrix $T$ except for the normalizing factor $\mathcal{M}$. A graphical representation of $T^{-1}$ is shown in Figure 5-7.

In Figure 5-7, $s_1$ and $s_3$ are the inferred candidate linear combinations, *i.e.* $w(s_1, m) \neq 0$

and $w(s_3, m) \neq 0$; the *end node* indicates what $v_1$ has overheard from $v_{m+1}$. Observe that although $s_1$ is one of the inferred linear combinations, $s_1$ is not connected to the end node. This is because $h(s_1) \neq h(x_{m+1})$. On the other hand, $h(s_2) = h(x_{m+1})$. As a result, $s_2$ is connected to the end node although $w(s_2, m) = 0$. We define an inferred linear combination $s$ as *matched* if $w(s, m) > 0$ and $h(s) = h(x_{m+1})$.

Node $v_1$ uses $T^{-1}$ to compute the total probability $p^*$ of hearing $[\tilde{x}_{m+1}, h(x_{m+1})]$ given the inferred linear combinations by computing the following equation:

$$p^* = \sum_{\forall s} w(s, m) \cdot T^{-1}(s, \tilde{x}_{m+1}). \tag{5.23}$$

Probability $p^*$ is the probability of overhearing $\tilde{x}_{m+1}$ given the channel statistics; therefore, measures the likelihood that $v_{m+1}$ is consistent with the information gathered by $v_1$. Node $v_1$ can use $p^*$ to make a decision on $v_{m+1}$'s behavior. For example, $v_1$ can use a threshold decision rule to decide whether $v_{m+1}$ is misbehaving or not. Node $v_1$ claims that $v_{m+1}$ is malicious if $p^* \leq t$ where $t$ is a threshold value determined by the given channel statistics $BSC(p_{i1})$ for $i \in [2, m+1]$; otherwise, $v_1$ claims $v_{m+1}$ is well-behaving.

Depending on the decision policy used, we can use the hypothesis testing framework to analyze the probability of false positive and false negative. Section 5.4 provides such analysis for the simple two-hop network with a simple decision policy: if the set of matched inferred linear combinations is not empty, we declare the node well-behaving. However, the main purpose of this chapter is to propose a method in which we can compute $p^*$, which can be used to establish trust within a network. It would be worthwhile to look into specific decision policies and their performance, including the false positive and the false negative probabilities, as in Section 5.4.2.

## 5.6 Analysis for Two-hop Network

We provide an analysis for the performance of Algebraic Watchdog for two-hop network.

**Theorem 5.6.1** *Consider a two-hop network as shown in Figure 5-3a. Consider $v_j$, $j \in [1, m]$ where $p_{ij} \leq \frac{1}{2}$, and $p_{ij}n \geq 1$. Then, the expected number of* matched *codewords is*

*less than or equal to*

$$2^{n\left[\sum_{i\neq j, i\in[1,m+1]}\left(H(p_{ij})-H(\frac{d_i}{n})\right)-1\right]-m\delta}. \tag{5.24}$$

*This bound becomes tighter as $n$ increases.*

*Proof:* Without loss of generality, we consider $v_1$. The proof uses concepts and techniques developed for list-decoding [26]. We first consider the overhearing of $v_k$'s transmission, $k \in [2, m]$. Node $v_1$ overhears $\tilde{x}_k$ from $v_k$. The noise introduced by the overhearing channel is characterized by $BSC(p_{k1})$; therefore, $E[\Delta(\mathbf{x_k}, \tilde{\mathbf{x}}_\mathbf{k})] = np_{k1}$.

Now, we consider the number of codewords that are within $B(\tilde{x}_k, np_{k1})$, the Hamming ball of radius $np_{k1}$ centered at $\tilde{x}_k$. The size of $B(\tilde{x}_k, np_{k1})$ is bounded by

$$|B(\tilde{x}_k, np_{k1})| \leq 2^{n(H(p_{k1})-H(\frac{d_k}{n}))}. \tag{5.25}$$

Node $v_1$ overhears the hash $h(x_k)$; thus, the number of codewords that $v_1$ considers is reduced to

$$\frac{|B(\tilde{x}_k, np_{k1})|}{2^\delta} = 2^{n(H(p_{k1})-H(\frac{d_k}{n}))-\delta}. \tag{5.26}$$

Using this information, $v_1$ computes the set of inferred linear combinations, *i.e.* $s$ where $w(s, m) > 0$. Note that $v_1$ knows precisely the values of $x_1$. Therefore, the number of inferred linear combinations is upper bounded by

$$\prod_{k\in[2,m]} 2^{n\left(H(p_{k1})-H(\frac{d_k}{n})\right)-\delta} = 2^{n\left[\sum_{k\in[2,m]}\left(H(p_{k1})-H(\frac{d_k}{n})\right)\right]-(m-1)\delta}. \tag{5.27}$$

Due to the finite field operations, these inferred linear combinations are randomly distributed over the space $\{0, 1\}^n$.

Now, we consider the overheard information, $\tilde{x}_{m+1}$ from the downstream node $v_{m+1}$. By similar analysis as above, we can derive that there are at most

$$2^{n(H(p_{m+1,1})-H(\frac{d_{m+1}}{n}))-\delta} \tag{5.28}$$

codewords in the hamming ball $B(\tilde{x}_{m+1}, np_{m+1,1})$ with hash value $h(x_{m+1})$. Thus, the probability that a randomly chosen codeword in the space of $\{0, 1\}^n$ is in $B(\tilde{x}_{m+1}, np_{m+1,1}) \cap$

$\{x|h(x) = h(x_{m+1})\}$ is at most

$$2^{n(H(p_{m+1,1})-H(\frac{d_{m+1}}{n}))-\delta-n}. \tag{5.29}$$

Then, the expected number of *matched* codewords is bounded by the product of Equations (5.27) and (5.29). ■

If we assume that the hash is of length $\delta = \varepsilon n$, then the statement in Theorem 5.6.1 is equal to

$$2^{n\left[\sum_{i\neq j, i\in[1,m+1]} H(p_{ij})-\left(\sum_{i\neq j, i\in[1,m+1]} H(\frac{d_i}{n})+1+m\varepsilon\right)\right]}. \tag{5.30}$$

This highlights the trade-off between the quality of overhearing channel and the redundancy (introduced by the error-correcting code $\mathcal{C}_i$'s and the hash $h$). If there is enough redundancy, then $\mathcal{C}_i$ and $h$ together form an error-correcting code for the overhearing channels; therefore, allows exact decoding to a single matched codeword.

The analysis also shows how adversarial errors can be interpreted. Assume that $v_{m+1}$ wants to inject errors at rate $p_{adv}$. Then, node $v_1$, although has an overhearing $BSC(p_{m+1,1})$, effectively experiences an error rate of $p_{adv} + p_{m+1,1} - p_{adv} \cdot p_{m+1,1}$. This does not change the set of the inferred linear combinations but it affects $\tilde{x}_{m+1}$. Therefore, overall, adversarial errors affect the set of matched codewords and the distribution of $p^*$. As we shall see in Section 5.8, the difference in distribution of $p^*$ between a well-behaving relay and adversarial relay can be used to detect malicious

## 5.7 Protocol for Algebraic Watchdog

We use the two-hop Algebraic Watchdog (Section 5.5) in a hop-by-hop manner to ensure a globally secure network. We extend the two-hop Algebraic Watchdog from Section 5.5 to a protocol that can be applied to general network topologies. For example, consider Figure 5-1. Nodes $v_1$, $v_2$, and $v_3$ monitor $v_5$ using the two-hop Algebraic Watchdog; $v_3$ and $v_4$ can do the same for $v_6$; $v_5$ and $v_6$ can watch over $v_7$, and so on.

In Algorithm 1, we present a distributed algorithm for nodes to secure the their local neighborhood. Each node $v$ transmits and receives data as scheduled; however, node $v$

```
foreach node v do
    According to the schedule, transmit and receive data;
    if v decides to check its neighborhood then
        Listen to neighbors' transmissions;
        foreach downstream neighbor v' do
            Perform Two-hop Algebraic Watchdog on v';
        end
    end
end
```

**Algorithm 1**: Distributed protocol for Algebraic Watchdog at $v$.

randomly chooses to check its neighborhood, at which point node $v$ listens to neighbors transmissions to perform the two-hop Algebraic Watchdog from Section 5.5.

The two-hop Algebraic Watchdog can be applied to specific subsets of the network as needed. The protocol does not depend on every nodes executing the two-hop Algebraic Watchdog. If some nodes are assumed to never misbehave, then the upstream nodes of these nodes do not need to monitor them. However, whenever the need arises (*i.e.* a new node joins the neighborhood, network is unstable, etc.), a node can employ the two-hop Algebraic Watchdog.

**Corollary 5.7.1** *Consider $v_{m+1}$ as in Figure 5-3a. Assume that the downstream node $v_{m+2}$ is well-behaving, and thus, forces $\mathbf{h_{x_{m+1}}}$ to be consistent with $x_{m+1}$,* i.e. $\mathbf{h_{x_{m+1}}} = h(x_{m+1})$. *Let $\underline{\mathbf{p_i}}$ be the packet received by $v_{m+1}$ from parent node $v_i \in P(v_{m+1})$. Then, if there exists at least one well-behaving parent $v_j \in P(v_{m+1})$, $v_{m+1}$ cannot inject errors beyond the overhearing channel noise ($p_{m+1,j}$) without being detected.*

Section 5.6 noted that presence of adversarial error (at a rate above the channel noise) can be detected by a change in distribution of $p^*$. Corollary 5.7.1 does not make any assumptions on whether packets $\underline{\mathbf{p_i}}$'s are valid or not. Instead, the claim states that $v_{m+1}$ transmits a valid packet *given* the packets $\underline{\mathbf{p_i}}$ it has received.

**Corollary 5.7.2** *Node $v$ can inject errors beyond the channel noise only if either of the two conditions are satisfied:*

1. *All its parent nodes $P(v) = \{u|(u,v) \in E_1\}$ are colluding Byzantine nodes;*

2. *All its downstream nodes,* i.e. *receivers of the transmission* $\underline{\mathbf{p_i}}$*, are colluding Byzantine nodes.*

*Remark:* In Case 1), $v$ is not responsible to any well-behaving nodes. Node $v$ can transmit any packet without the risk of being detected by any well-behaving parent node. However, then, the min-cut to $v$ is dominated by adversaries, and the information flow through $v$ is completely compromised, regardless of whether $v$ is malicious or not.

In Case 2), $v$ can generate any hash value since its downstream nodes are colluding adversaries. Therefore, it is not liable to transmit a consistent hash, which is necessary for $v$'s parent nodes to monitor $v$. However, note that $v$ is not responsible in delivering any data to a well-behaving node. Even if $v$ were well-behaving, it cannot reach any well-behaving node without going through a malicious node in the next hop. Thus, the information flow through $v$ is again completely compromised.

Corollary 5.7.2 states that we cannot ensure the validity of the information that is generated or forwarded by node $v$ if either of the two conditions are met. However, this does not imply that the entire network is compromised. For example, any flow that does not go through $v$ can still be protected, even if this flow is coded with packets from the contaminated flow. At the destination, the contaminated flow may result in unusable data; the remaining flows can be decoded correctly.

Therefore, Corollary 5.7.2 shows that the Algebraic Watchdog can aid in ensuring correct delivery of data when the following assumption holds: for every intermediate node $v$ in the path between source to destination, $v$ has at least one well-behaving parent and at least one well-behaving child, *i.e.* there exists at least a path of well-behaving nodes. This is not a trivial result as we are not only considering a single-path network, but also multi-hop, multi-path network.

## 5.8 Simulations

We present MATLAB simulation results that show the difference in distribution of $p^*$ between the well-behaving and adversarial relay. We consider a setup in Figure 5-3a. We set all $p_{i1}$, $i \in [2, m]$ to be equal, and we denote this probability as $p_s = p_{i1}$ for all $i$. We

denote $p_{adv}$ to be the probability at which the adversary injects error; thus, the effective error that $v_1$ observes from an adversarial relay is combined effect of $p_{m+1,1}$ and $p_{adv}$. The hash function $h(x) = ax + b \mod 2^\delta$ is randomly chosen over $a, b \in \mathbb{F}_{2^\delta}$.

We set $n = 10$ (field size of $2^{10}$). A typical packet can have a few hundreds to tens of thousand bits. Thus, a network coded packet with $n = 10$ could have a few thousand symbols over which to perform Algebraic Watchdog. It may be desirable to randomize which symbols or when to perform Algebraic Watchdog on. This choice depends not only on the security requirement, but also on the computational and energy budget of the node.

For each set of parameters, we randomly generate symbols from $\mathbb{F}_{2^{10}}$ and run Algebraic Watchdog. For each symbol, under a non-adversarial setting, we assume that only channel randomly injects bit errors to the symbol; under adversarial setting, both the channel and the adversary randomly inject bit errors to the symbol. For each set of parameters, we run the Algebraic Watchdog 1000 times. Recall that the Algebraic Watchdog operates on per-symbol basis (Figure 5-2); therefore, operating the Algebraic Watchdog a thousand times is equivalent to operating over a 1000 symbols, which could be within a single packet or across multiple packets. Therefore, this is equivalent to running the Algebraic Watchdog on a moderately-sized packet (10,000 bits) or over several smaller packets, which are coded in $\mathbb{F}_{2^{10}}$.

For simplicity, nodes in the simulation do not use error-correcting codes; thus, $d_i = 0$ for all $i$. This limits the power of the Algebraic Watchdog; thus, the results shown can be further improved by using error correcting codes $\mathcal{C}_i$.

We denote $p^*_{adv}$ and $p^*_{relay}$ as the value of $p^*$ when the relay is adversarial and is well-behaving, respectively. We denote $var_{adv}$ and $var_{relay}$ to be the variance of $p^*_{adv}$ and $p^*_{relay}$. We shall show results that show the difference in distribution of $p^*_{adv}$ and $p^*_{relay}$ from $v_1$'s perspective. This illustrates that only *one* good parent node (*i.e.* $v_1$), is sufficient to notice the difference in distribution of $p^*_{adv}$ and $p^*_{relay}$. Therefore, confirming our analysis in Section 5.7. With more parent nodes performing the check independently, we can improve the probability of detection.

Our simulation results coincide with our analysis and intuition. Figure 5-8 shows that adversarial above the channel noise can be detected. First of all, for all values of $p_{adv} > 0$,

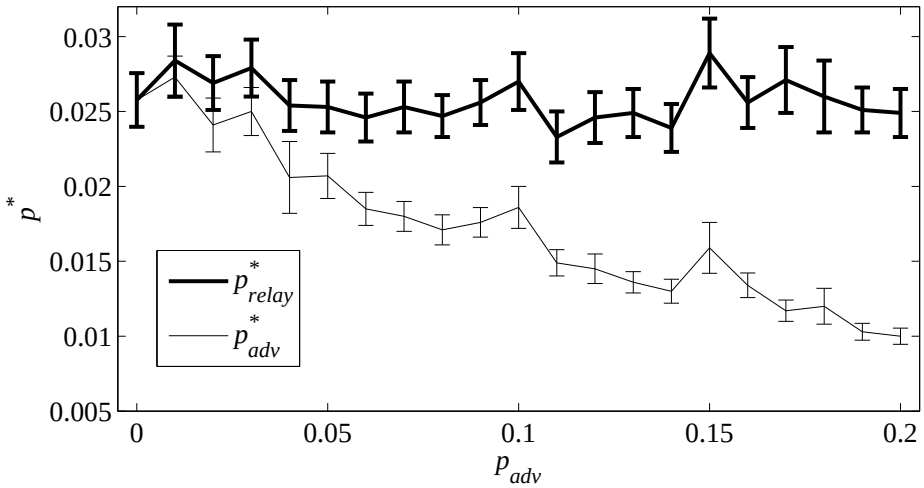
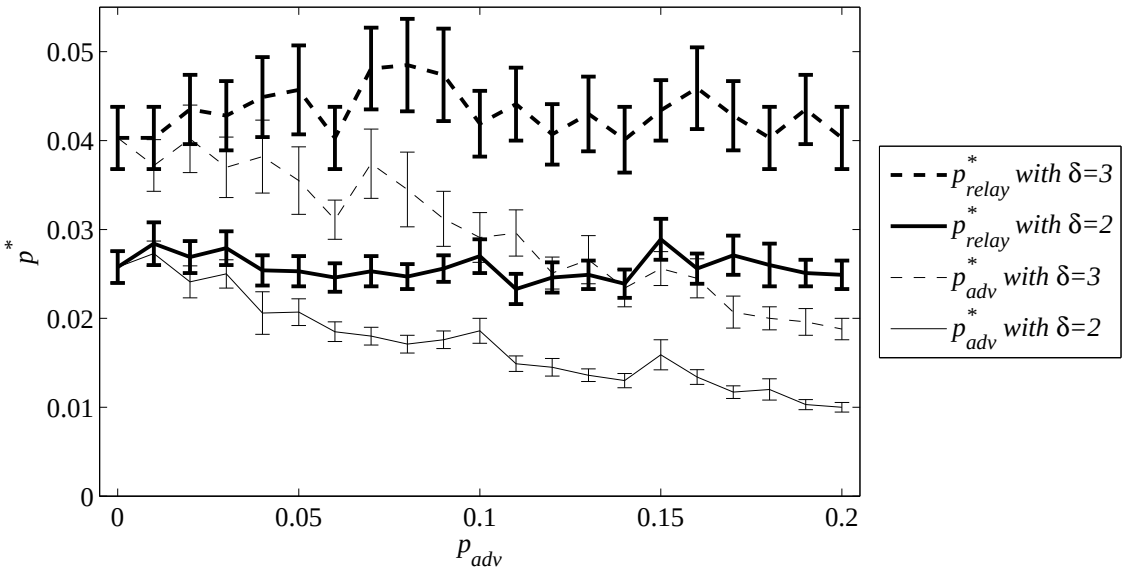



Figure 5-8: Simulation results showing that adversarial noise above the channel noise can be detected. The average value $p^*_{relay}$ and $p^*_{adv}$ over 1000 random iterations. The error bars represent the variance, $var_{relay}$ and $var_{adv}$. We set $m = 3$, $n = 10$, $\delta = 2$, and $p_s = p_{m+1,1} = 10\%$. We vary $p_{adv}$, the adversary's error injection rate.

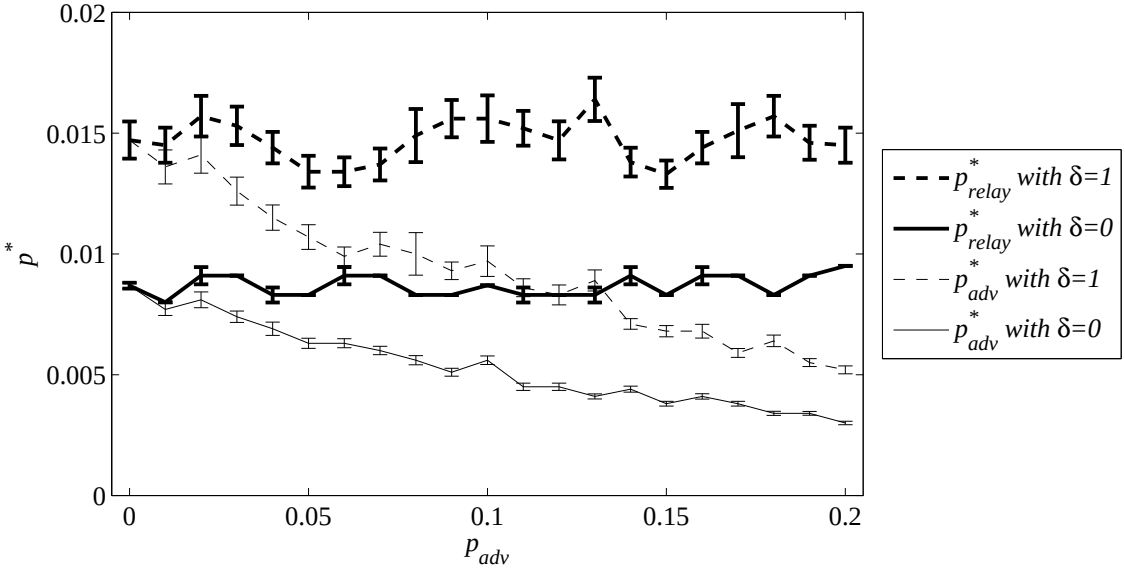
$p^*_{adv} < p^*_{relay}$; thus, showing that adversarial errors can be detected. Furthermore, the larger the adversarial error injection rate, the bigger the difference in the distributions of $p^*_{adv}$ and $p^*_{relay}$. When adversarial error rate is small, then the effective error $v_1$ sees in the packet can easily be construed as that of the channel noise. As a result, the values and the distributions of $p^*_{relay}$ and $p^*_{adv}$ are similar. Note that in such scenarios where $p_{adv}$ is small, these adversarial errors can be corrected by the downstream node if appropriate channel error correcting code is used.

As the adversarial error rate increases, there is a divergence between the two distributions of $p^*_{adv}$ and $p^*_{relay}$. The difference in the distributions of $p^*_{relay}$ and $p^*_{adv}$ is not only in the average value. The variance $var_{relay}$ is relatively constant throughout ($var_{relay}$ is approximately 0.0018 throughout). On the other hand, $var_{adv}$ generally decreases with increase in $p_{adv}$. For small $p_{adv}$, the variance $var_{adv}$ is approximately 0.0018; while for large $p_{adv}$, the variance $var_{adv}$ is approximately 0.0008. This trend intuitively shows that, with increase in $p_{adv}$, not only do we detect that the adversarial relay more often (since the average value of $p^*_{adv}$ decreases), but we are more confident of the decision.

Figure 5-9 shows the affect of the size of the hash. With increase in redundancy (by using

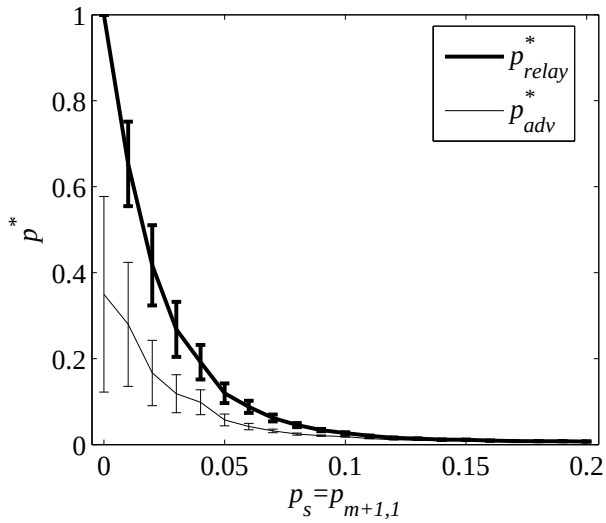



(a) For $\delta = \{2, 3\}$

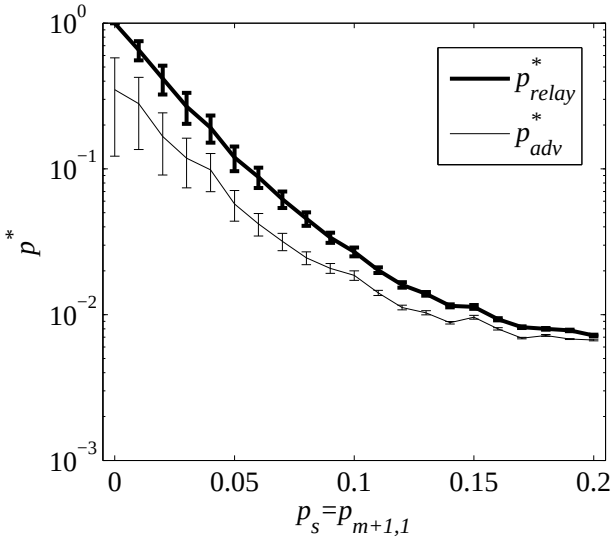



(b) For $\delta = \{0, 1\}$

Figure 5-9: Simulation results showing the effect of the hash length $\delta$. The average value of $p^*_{relay}$ and $p^*_{adv}$ over 1000 random iterations. We vary the hash length, $\delta$, and adversarial error rate, $p_{adv}$. The error bars represent the variance, $var_{relay}$ and $var_{adv}$. We set $m = 3$, $n = 10$, and $p_s = p_{m+1,1} = 10\%$.

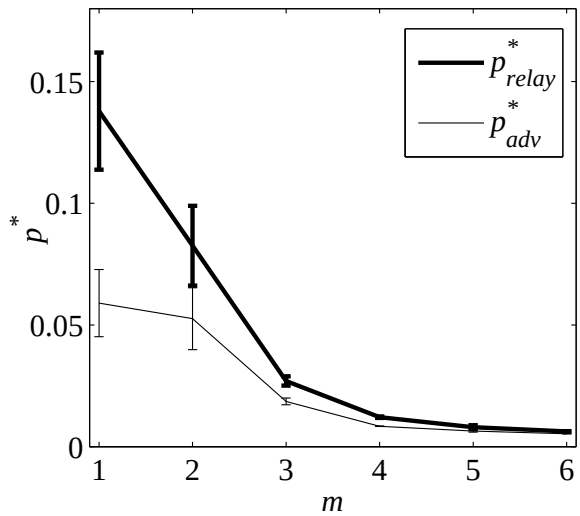



(a) Linear scale

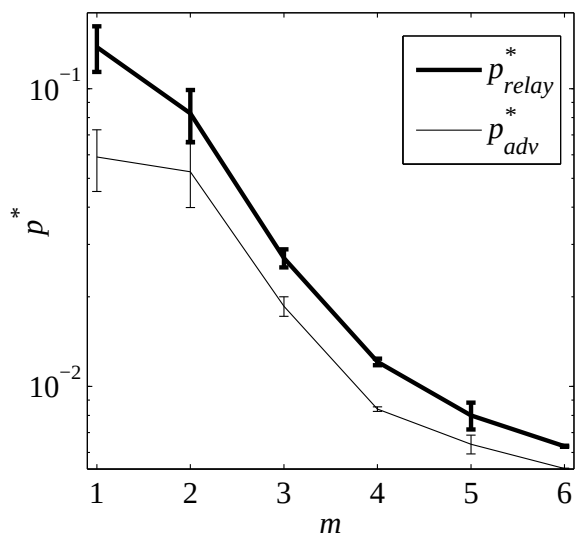



(b) Log scale

Figure 5-10: Simulation results showing the effect of the overhearing channel, $p_s$. The average value of $p^*_{relay}$ and $p^*_{adv}$ over 1000 random iterations. We vary the value of $p_s = p_{m+1,1}$, the quality of overhearing channels. The error bars represent the variance, $var_{relay}$ and $var_{adv}$. We set $m = 3$, $n = 10$, and $p_{adv} = 10\%$.




(a) Linear scale




(b) Log scale

Figure 5-11: Simulation results showing the effect of the number of neighbors, $m$. The average value of $p^*_{relay}$ and $p^*_{adv}$ over 1000 random iterations. We vary the value of $m$, the number of nodes using $v_{m+1}$ as a relay. The error bars represent the variance, $var_{relay}$ and $var_{adv}$. We set $m = 3$, $n = 10$, and $p_s = p_{m+1,1} = p_{adv} = 10\%$.

hash functions of length $\delta$), $v_1$ can detect malicious behavior better. This is true regardless of whether the relay is well-behaving or not. Node $v_1$'s ability to judge its downstream node increases with $\delta$. Therefore, $p^*_{adv}$ for $\delta$ is generally higher than that of $\delta'$ where $\delta > \delta'$. This holds for $p^*_{relay}$ as well. However, for any fixed $\delta$, node $v_1$ can see a distinction between $p^*_{relay}$ and $p^*_{adv}$ as shown in Figure 5-9. A similar trend to that of Figure 5-8 can be seen for the distributions of $p^*_{adv}$ and $p^*_{relay}$ for each value of $\delta$. The interesting result is that even for $\delta = 0$, *i.e.* we include no redundancy or hash, node $v_1$ is able to distinguish an adversarial relay from a well-behaving relay.

Results in Figure 5-10 confirms our intuition that the better $v_1$'s ability to collect information from $v_i$'s, where $i \neq 1$, the better its detection ability. If node $v_1$ is able to infer better or overhear $x_i$ with little or no errors, the better its inference on what the relay node should be transmitting. As overhearing channel progressively worsens ($p_s$ increases), $v_1$'s ability to detect malicious behavior deteriorates; unable to distinguish between a malicious and a well-behaving relay.

Finally, we note the effect of $m$, the number of nodes in the network, in Figure 5-11. Node $v_1$'s ability to check $v_{m+1}$ is reduced with $m$. When $m$ increases, the number of messages $v_1$ has to infer increases, which increases the uncertainty within the system. However, it is important to note that as $m$ increases, there are more nodes $v_i$'s, $i \in [1, m]$ that can independently perform checks on $v_{m+1}$. This affect is not captured by the results shown in Figure 5-11.

## 5.9 Conclusions

We proposed the *Algebraic Watchdog*, in which nodes can verify their neighbors probabilistically and police them locally by the means of overheard messages in a coded network. Using the Algebraic Watchdog scheme, nodes can compute a probability of consistency, $p^*$, which can be used to detect malicious behavior. Once a node has been identified as malicious, this node can be punished, eliminated, or excluded from the network using reputation based schemes such as [42][79].

We first presented a graphical model and an analysis of the Algebraic Watchdog for

two-hop networks. We then extended the Algebraic Watchdog to multi-hop, multi-source networks. We provided a trellis-like graphical model for the detection inference, and an algorithm that may be used to compute the probability that a downstream node is consistent with the overheard information. We analytically showed how the size of the hash function, minimum distance of the error-correcting code, and the quality of the overhearing channel can affect the probability of detection. Finally, we presented simulation results that support our analysis and intuition.

# Chapter 6

# Conclusions

NETWORKS and communications, in particular wireless networks, have great potential to improve access to services and information. For this reason, networks have been adopted and deployed rapidly, and users have come to expect and rely on the networks to not only transfer but also store their information. However, it has been shown that our current routing networks are not suited to meet the requirements of all types of networks, in particular wireless networks.

Our current wireless networks are not well equipped to meet the growth in demands as they are mainly based on the assumptions of wired networks with unicast traffic. Wireless networks have several challenging properties, which are not as prominent in wired networks. As a result, applying techniques that we have developed and implemented for wired networks directly to wireless networks can be inefficient.

As our current wireless networks struggle to meet the demands of high bandwidth applications, we have to rethink and rebuild our current networks. We believe that recognizing that wireless networks are fundamentally different from wired networks, and building systems and networks appropriate for the wireless medium is key to solving many of the problems we face today. This dissertation strives towards this goal, and proposes network coding as a versatile and powerful tool that can aid us in achieving this goal.

In this dissertation, we demonstrated how network coding can be applied to provide robust wireless networks. We focused on harnessing the broadcast nature of wireless medium, and designing algorithms and protocols that can overcome the challenges of operating in

wireless networks, such as interference, erasures, and attacks. The first step in designing such robust systems comes from understanding the effects the broadcast and stochastic nature of wireless network have on our current network designs. Using these insights, we provided designs of network coded systems that show significant performance improvements over the existing systems.

The main body of the dissertation, contained in Chapters 3, 4, and 5, presented designs of network coding that address interference, erasures, and attacks, respectively, in wireless networks.

- *Algebraic NC* was introduced for capacity achieving codes in multi-user wireless networks with interference. We used the algebraic network coding framework to model and understand the effect of interference in multi-user wireless networks. Our algebraic framework allowed us to reduce the problem of capacity in interference dominated wireless networks to that of determining the rank of a single matrix. This understanding enabled us to show that a distributed, randomized network code can achieve capacity in multicast connections as well as some non-multicast connections.

- *TCP/NC* was introduced for efficient and reliable transport of data over faulty and lossy networks. We studied the inability of TCP to overcome or compensate for the stochastic nature of wireless networks. With this understanding, we combined network coding's erasure correction capability with TCP's congestion control mechanism. We showed that network coding can provide significant throughput gains in lossy network scenarios where TCP suffered before, while keeping intact TCP's congestion control mechanism.

- *Algebraic Watchdog* was introduced for secure self-checking network in wireless networks with adversarial attackers. By taking advantage of the broadcast nature of wireless medium, nodes monitor their neighbors locally in a distributed manner. By monitoring their neighborhood independently, the nodes together provide a globally secure network. Network coding allows for a more efficient operation of the wireless network, and the algebraic transformation performed by network coding was harnessed to provide secure communication.

In each of these chapters, we presented analytical understanding of the problem posed and offered an algorithm that may serve the proposed problem. This dissertation made an effort to design simple and practical algorithms, so that the analysis is tractable and the algorithms are easily deployable. Furthermore, we supported our algorithms and analysis with simulation results that demonstrate the performance improvements presented by our algorithms.

This dissertation made a case for network coding as a new paradigm to operate wireless networks. Network coding promises a more efficient network with higher throughput and reliability, and we have demonstrated that network coding indeed lives up to this promise. This dissertation has shown that network coding, if used properly, can achieve significant gains in efficiency, performance, and robustness.

## 6.1 Future Work

There are several avenues of future work for the problems considered in this dissertation. We discuss possible extensions for each chapter sequentially.

The use of network coding in ADT networks has brought a simpler algebraic characterization of the ADT networks. However, the purpose of ADT network is to approximate wireless multi-user Gaussian networks. Therefore, it is important to understand how the results in ADT network can translate to results in wireless multi-user Gaussian networks. Reference [6][7] showed that, for a few simple networks, the capacity of an ADT network is a constant bit away from the capacity of the wireless multi-user Gaussian network. An equivalent result for the random linear network coding would be of interest.

TCP/NC, the subject of Chapter 4, has great practical potential. There are various modifications that can be considered: TCP/NC with multiple sources, multiple paths, and re-encoding at intermediate nodes. In addition, further understanding of TCP/NC's impact and applicability are needed. For instance, this dissertation focused on understanding the throughput performance gain of using network coding with TCP. TCP/NC may bring forth more than just throughput gains, such as energy efficiency, decrease in delay, and reduction in infrastructure needed to operate the network.

In designing Algebraic Watchdog, our ultimate goal is to design a network in which the participants check their neighborhood locally to enable a secure global network. This dissertation is a step towards realizing this goal. Possible future work on Algebraic Watchdog includes developing inference methods and approximation algorithms to make local decisions efficiently and to aggregate local trust information to a global trust state. These problems are particularly interesting, especially in a network where malicious nodes may inject incorrect or false local trust information.

This dissertation aimed to provide some guidance in designing network coded systems and make network coding more practical. We chose a few selected area of applications, in particular protocols and algorithms for robust wireless networks. Of course, there are many more applications that may benefit from network coding. Finding such applications and designing appropriate network coded systems are problems we as a community have to address and answer.

## 6.2 Final Remarks

Routing solutions are undeniably powerful. It is how our current networks operate, and it has proven to be scalable and efficient. However, we should not be satisfied with what is currently available and deployed. Routing solutions have brought us fast, affordable, and ubiquitous network access. Now, it is time to build upon the success of routing networks and create better networks.

Network coding promises a fundamentally new way to operate networks. In an essence, network coding questions the fundamental assumptions in our network designs. Currently deployed networks are built using architecture rooted in wired unicast networks, which ultimately limits performance. Network coding brings into light these aspects of our current architecture, and beckons us to rethink networking overall. Perhaps, this is what makes network coding so powerful.

This dissertation advocates that network coding, if used appropriately, can overcome the limitations of routing networks and provide higher throughput and reliability. By recognizing that our networks are no longer just wired, network coding allows us to harness

the broadcast nature of wireless networks. By understanding that we need more than just unicast connections, network coding allows more efficient delivery of data to all users involved. By learning the algebraic nature of data, network coding becomes a powerful tool for designing more efficient and robust networks.

# Bibliography


[1] Network simulator (ns-2). `http://www.isi.edu/nsnam/ns/`.

[2] S. Acedański, S. Deb, M. Médard, and R. Koetter. How good is random linear coding based distributed network storage? In *Proceedings of IEEE International Symposium on Network Coding (NetCod)*, Riva del Garda, Italy, April 2005.

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

[4] E. Altman, T. Jiménez, and R. Núñez Queija. Analysis of two competing TCP/IP connections. *Performance Evaluation*, 49(1-4):43–55, September 2002.

[5] A. Amaudruz and C. Fragouli. Combinatorial algorithms for wireless information flow. In *Proceedings of the Annual ACM -SIAM Symposium on Discrete Algorithms (SODA)*, pages 555–564, January 2009.

[6] A. S. Avestimehr, S. N. Diggavi, and D. N. C. Tse. A deterministic approach to wireless relay networks. In *Proceedings of Allerton Conference on Communication, Control, and Computing*, September 2007.

[7] A. S. Avestimehr, S. N. Diggavi, and D. N. C. Tse. Wireless network information flow. In *Proceedings of Allerton Conference on Communication, Control, and Computing*, September 2007.

[8] A. S. Avestimehr and T. Ho. Approximate capacity of the symmetric half-duplex Gaussian butterfly network. In *Proceedings of IEEE Information Theory Workshop (ITW)*, pages 311–315, June 2009.

[9] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 5(6):756–769, December 1997.

[10] J. Barros, R. A. Costa, D. Munaretto, and J. Widmer. Effective delay control for online network coding. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 208–216, April 2009.

[11] K. Bhattad, N. Ratnakar, R. Koetter, and K. R. Narayanan. Minimal network coding for multicast. In *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pages 1730–1734, September 2005.

[12] G. Bresler, A. Parekh, and D. Tse. The approximate capacity of the many-to-one and one-to-many Gaussian interference channels. *Submitted to Transactions on Information Theory*, September 2008.

[13] R. Cáceres and L. Iftode. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE Journal on Selected Areas in Communications*, 13(5):850–857, June 1995.

[14] M. Castro and B. Liskov. Practical Byzantine fault tolerance. In *Proceedings of Symposium on Operating Systems Design and Implementation (OSDI)*, pages 173–186, February 1999.

[15] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *Proceedings of ACM SIGCOMM*, pages 169–180, August 2007.

[16] A. Chaintreau, F. Baccelli, and C. Diot. Impact of TCP-like congestion control on the throughput of multicast groups. *IEEE/ACM Transactions on Networking*, 10(4):500–512, August 2002.

[17] D. Charles, K. Jain, and K. Lauter. Signatures for network coding. In *Proceedings of IEEE Conference on Information Sciences and Systems*, pages 857–863, March 2006.

[18] M. Dietzfelbinger, J. Gil, Y. Matias, and N. Pippenger. Polynomial hash functions are reliable. In *Automata, Languages and Programming*, volume 623 of *Lecture Notes in Computer Science*, pages 235–246. Springer Berlin / Heidelberg, 1992.

[19] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 2000–2008, May 2007.

[20] J. Douceur. The sybil attack. In *Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer Berlin, 2002.

[21] R. Dougherty, C. Freiling, and K. Zeger. Insufficiency of linear coding in network information flow. *IEEE Transaction on Information Theory*, 51(8):2745–2759, August 2005.

[22] R. Dougherty, C. Freiling, and K. Zeger. Networks, matroids, and non-shannon information inequalities. *IEEE Transactions on Information Theory*, 53(6):1949–1969, June 2007.

[23] J. Ebrahimi and C. Fragouli. Multicasting algorithms for deterministic networks. In *Proceedings of IEEE Information Theory Workshop (ITW)*, pages 1–5, January 2010.

[24] J. Ebrahimi and C. Fragouli. Vector network coding. Technical report, EPFL, February 2010.

[25] M. Effros, M. Médard, T. Ho, S. Ray, D. Karger, and R. Koetter. Linear network codes: A unified framework for source channel, and network coding. In *Proceedings of the DIMACS workshop on network information theory (Invited paper)*, pages 197–216, March 2003.

[26] P. Elias. List decoding for noisy channels. Technical report, Research Laboratory of Electronics, Massachusetts Institute of Technology, 1957. Technical Report 335.

[27] P. Erdos and A. Renyi. On a classical problem of probability theory. *Magyar Tud. Akad. Mat Kutató Int. Közl*, 6:215–220, 1961.

[28] E. Erez, Y. Xu, and E. M. Yeh. Coding for the deterministic network model. In *Proceedings of Allerton Conference on Communication, Control and Computing*, pages 1534–1541, September 2010.

[29] R. Etkin, D. Tse, and H. Wang. Gaussian interference channel capacity to within one bit. *IEEE Transactions on Information Theory*, 54(12):5534–5562, December 2008.

[30] M. Feder, D. Ron, and A. Tavory. Bounds on linear codes for network multicast. *Electronic Colloquium on Computational Complexity (ECCC)*, 10(33), May 2003.

[31] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. An extension to the selective acknowledgement (SACK) option for TCP. *RFC 2883*, July 2000.

[32] C. Fragouli, E. Soljanin, and A. Shokrollahi. Network coding as a coloring problem. *Annual Conference on Information Sciences and Systems (CISS)*, Princeton, NJ, March 2004.

[33] M. Garetto, R. L. Cigno, M. Meo, and M. A. Marsan. Modeling short-lived TCP connections with open multiclass queuing networks. *Computer Networks*, 44(2):153–176, February 2004.

[34] C. Gkantsidis, J. Miller, and P. Rodriguez. Comprehensive view of a live network coding P2P system. In *Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference*, pages 177–188, Rio de Janeiro, Brazil, October 2006.

[35] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 2235–2245, Miami, FL, March 2005.

[36] C. Gkantsidis and P. Rodriguez. Cooperative security for network coding file distribution. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 1–13, April 2006.

[37] M. X. Goemans, S. Iwata, and R. Zenklusen. An algorithmic framework for wireless information flow. In *Proceedings of Allerton Conference on Communication, Control, and Computing*, pages 294–300, September 2009.

[38] S. Ha, I. Rhee, and L. Xu. CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review*, 42(5):64–74, July 2008.

[39] B. Haeupler, M. Kim, and M. Médard. Optimality of network coding in packet networks. In *Proceedings of IEEE Information Theory Workshop (ITW) Paraty*, pages 1–5, October 2011.

[40] T. Ho, B. Leong, R. Koetter, M. Médard, and M. Effros. Byzantine modification detection in multicast networks using randomized network coding. In *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pages 144–148, June 2004.

[41] T. Ho, M. Médard, R. Koetter, M. Effros, J. Shi, and D. R. Karger. A random linear coding approach to mutlicast. *IEEE Transaction on Information Theory*, 52(10):4413–4430, Octoboer 2006.

[42] J.-P. Hubaux, L. Buttyán, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proceedings of the ACM international Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 146–155, October 2001.

[43] S. Jaggi, Y. Cassuto, and M. Effros. Low complexity encoding for network codes. In *Proceedings of International Symposium on Information Theory (ISIT)*, pages 40–44, July 2006.

[44] S. Jaggi, P. A. Chou, and K. Jain. Low complexity algebraic multicast network codes. In *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, page 368, June 2003.

[45] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard. Resilient network coding in the presence of Byzantine adversaries. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 616 – 624, March 2007.

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

[47] D. B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 158–163, December 1994.

[48] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. In *Proceedings of IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, May 2003.

[49] S. Katti, S. Gollakota, and D. Katabi. Embracing wireless interference: Analog network coding. In *Proceedings of ACM SIGCOMM*, pages 397–408, August 2007.

[50] S. Katti, D. Katabi, H. Balakrishnan, and M. Médard. Symbol-level network coding for wireless mesh networks. In *Proceedings of ACM SIGCOMM*, pages 401–412, August 2008.

[51] S. Katti, I. Maric, A. Goldsmith, D. Katabi, and M. Médard. Joint relaying and network coding in wireless networks. In *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pages 1101 –1105, June 2007.

[52] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. XORs in the air: Practical wireless network coding. In *Proceedings of ACM SIGCOMM*, September 2006.

[53] T. Kelly. Scalable TCP: improving performance in highspeed wide area networks. *ACM SIGCOMM Computer Communication Review*, 33(2):83–91, April 2003.

[54] M. Kim, E. Erez, E. M. Yeh, and M. Médard. Deterministic network model revisited: An algebraic network coding approach. *Submitted to IEEE Transactions on Information Theory*, March 2011.

[55] M. Kim, L. Lima, F. Zhao, J. Barros, M. Médard, R. Koetter, T. Kalker, and K. Han. On counteracting Byzantine attacks in network coded peer-to-peer networks. *IEEE Journal on Selected Areas in Communications (JSAC) Mission Critical Networking*, 28(5):692–702, June 2010.

[56] M. Kim, D. Lucani, X. Shi, F. Zhao, and M. Médard. Network coding for multi-resolution multicast. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 1–9, March 2010.

[57] M. Kim and M. Médard. Algebraic network coding approach to deterministic wireless relay networks. In *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*, pages 1518–1525, September 2010.

[58] M. Kim, M. Médard, and J. Barros. Counteracting Byzantine adversaries with network coding: An overhead analysis. In *Proceedings of IEEE Conference on Military Communications (MILCOM)*, pages 1–7, November 2008.

[59] M. Kim, M. Médard, and J. Barros. A multi-hop multi-source algebraic watchdog. In *Proceedings of IEEE Information Theory Workshop (ITW) Dublin (invited paper)*, pages 1–5, August 2010.

[60] M. Kim, M. Médard, and J. Barros. Algebraic watchdog: Mitigating misbehavior in wireless network coding. *IEEE Journal on Selected Areas in Communications (JSAC) Advances in Military Networking and Communications*, 29(10):1–11, December 2011.

[61] M. Kim, M. Médard, and J. Barros. Modeling network coded TCP throughput: A simple model and its validation. In *Proceedings of International ICST/ACM Conference on Performance Evaluation Methodologies and Tools (Valuetools)*, May 2011.

[62] M. Kim, M. Médard, J. Barros, and R. Koetter. An algebraic watchdog for wireless network coding. In *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pages 1159–1163, June 2009.

[63] M. Kim, J. K. Sundararajan, and M. Médard. Network coding for speedup in switches. In *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, 2007.

[64] M. Kim, J. K. Sundararajan, M. Médard, A. Eryilmaz, and R. Koetter. Network coding in a multicast switch. *IEEE Transactions on Information Theory*, 57(1):436–460, January 2011.

[65] N. Koblitz, A. Menezes, and S. Vanstone. The state of elliptic curve cryptography. *Designs, Codes and Cryptography*, 19(2-3):173–193, March 2000.

[66] R. Koetter and F. R. Kschischang. Coding for errors and erasures in random network coding. *IEEE Transactions on Information Theory*, 54(8):3579–3591, August 2008.

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

[68] M. Krohn, M. Freedman, and D. Mazières. On-the-fly verification of rateless erasure codes for efficient content distribution. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 226–240, May 2004.

[69] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[70] M. Langberg, A. Sprintson, and J. Bruck. The encoding complexity of network coding. *IEEE Transactions on Information Theory*, 52(6):2386–2397, June 2006.

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

[72] G. Liang, R. Agarwal, and N. Vaidya. When watchdog meets coding. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 1–9, March 2010.

[73] Y. Lin, B. Li, and B. Liang. CodeOr: Opportunistic routing in wireless mesh networks with segmented network coding. In *Proceedings of IEEE International Conference on Network Protocols*, pages 13–22, October 2008.

[74] S. Liu, T. Başar, and R. Srikant. Exponential-RED: a stabilizing AQM scheme for low- and high-speed TCP protocols. *IEEE/ACM Transactions on Networking*, 13(5):1068–1081, October 2005.

[75] S. H. Low, F. Paganini, and J. C. Doyle. Internet congestion control. *Proceedings of IEEE Control Systems Magazine*, 22(1):28–43, February 2002.

[76] S. H. Low, L. Peterson, and L. Wang. Understanding TCP Vegas: a duality model. *Journal of the ACM*, 49(2):207–235, March 2002.

[77] D. Lun, M. Médard, R. Koetter, and M. Effros. On coding for reliable communication over packet networks. *Physical Communication*, 1(1):3–20, March 2008.

[78] I. Maric, A. Goldsmith, and M. Médard. Analog network coding in the high-SNR regime. In *Proceedings of the IEEE Conference on Wireless Network Coding (WiNC)*, pages 1–6, June 2010.

[79] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiHoc)*, pages 255–265, August 2000.

[80] U. M. Maurer. Authentication theory and hypothesis testing. *IEEE Transaction on Information Theory*, 46(4):1350–1356, July 2000.

[81] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of the ACM SIGCOMM*, pages 303–314, October 1998.

[82] P. Papadimitratos and Z. J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Communication Networks and Disbributed Systems Modeling and Simulation Conference*, pages 193–204, January 2002.

[83] R. Perlman. *Network layer protocols with Byzantine robustness*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, October 1988.

[84] A. Rasala-Lehman and E. Lehman. Complexity classification of network information flow problems. *Proceedings of the Annual ACM-SIAM Symposium on Discrete algorithms*, pages 142–150, January 2004.

[85] S. Ray, M. Médard, and J. Abounadi. Random coding in noise-free multiple access networks over finite fields. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1898–1902, December 2003.

[86] S. M. Sadegh Tabatabaei-Yazdi and S. A. Savari. A combinatorial study of linear deterministic relay networks. In *Proceedings of IEEE Information Theory Workshop (ITW) Cairo*, pages 1–5, January 2010.

[87] D. Silva and F. Kschischang. Adversarial error correction for network coding: Models and metrics. In *Proceedings of Annual Allerton Conference on Communications, Control, and Computing*, pages 1246–1253, Monticello, IL, September 2008.

[88] J. K. Sundararajan, S. Deb, and M. Médard. Extending the Birkhoff-von Neumann switching strategy to multicast switches. In *Proceedings of the International IFIP-TC6 Networking Conference*, May 2005.

[89] J. K. Sundararajan, S. Jakubczak, M. Médard, M. Mitzenmacher, and J. Barros. Interfacing network coding with TCP: an implementation. Technical report, ArXiv, August 2009. `http://arxiv.org/abs/0908.1564`.

[90] J. K. Sundararajan, M. Médard, M. Kim, A. Eryilmaz, D. Shah, and R. Koetter. Network coding in a multicast switch. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 1145–1153, May 2007.

[91] J. K. Sundararajan, D. Shah, M. Médard, M. Mitzenmacher, and J. Barros. Network coding meets TCP. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 280–288, April 2009.

[92] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A compound TCP approach for high-speed and long distance networks. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 1–12, April 2006.

[93] Y. Tian, K. Xu, and N. Ansari. TCP in wireless environments: Problems and solutions. *IEEE Communications Magazine*, 43(3):S27–S32, March 2005.

[94] R. W. Yeung and N. Cai. Network error correction. *Communications in Information and Systems*, 6(1):19–54, 2006.

[95] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan. An efficient signature-based scheme for securing network coding against pollution attacks. In *Proceedings of IEEE Conference*

*on Computer Communications (INFOCOM)*, pages 1409–1417, Pheonix, AZ, April 2008.

[96] F. Zhao, T. Kalker, M. Médard, and K. J. Han. Signatures for content distribution with network coding. In *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pages 556–560, June 2007.

[97] L. Zhou and Z. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24 –30, November/December 1999.