Design and Analysis of Network Codes

Thesis by

Sidharth Jaggi

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2006

(Submitted October 28th, 2005)

But it's not who you are underneath, it's what you do that defines you.

– Rachel Dawes

To Mom, George, Michelle, and the good people at the Caltech Y.

Chapter 1 Acknowledgements

Pour undergraduate student in vat, ferment for five years, decant out a Ph.D. As with any reaction, this one required many ingredients, environmental controls, and catalysts. (Warning – do not try this at home.) Here's a list of some of the many people who deserve much of the credit but none of the blame.

Claude Elwood Shannon, who was there before anyone else. Michelle Effros, who showed me the way in more ways than one. Radhika Gowaikar and Chu-hsin Liang were there when I needed them, and how. Naveed Near-Ansari, John Lilley, and Michael Potter protected the world from my evil hacker-genius ways, and Linda Dozsa, Veronica Robles, and Shirley Beatty made sure the paper trail always led to President Nixon. My labmates-in-crime, Diego German Dugatkin, Hanying Feng, Qian Zhao, and Chaitanya Kumar Rao, educated me in the ways of graduate student life; in particular, I thank Michael Ian James Fleming for passing on the flame bearing the wisdom of old age, Wei-hsin Gu for accepting it, and Mayank Bakshi and Sukwon Kim for proving that life goes on. Jeremy Christopher Thorpe, Amir Farajidana, and Masoud Sharif, dammit, you were always right. Jehoshua (Shuki) Bruck was my fairy godfather, and his students, in particular Yuval Cassuto, Marc Riedel, Alex Sprintson, Anxiao Jiang, and Matthew Cook, were surrogate labmates. Peter Sanders, Sebastian Egner, and Ludo Tolhuizen deserve thanks for patiently bearing my bumbling efforts at being a token-bearer. Microsoft Research, Redmond, has been good to me – Philip Chou was the perfect mentor for a green summer intern, and set me on the path of what would eventually turn into the beast that is this document, and my mentor the next year, Kamal Jain, was my explorer-in-crime in strange realms of research. Abhinav Kumar and Yunnan Wu were my companions during my self-imposed exile, and Pradeep Shenoy was a willing sounding-board. Michael Langberg gently showed me how to simultaneously be a good collaborator and friend. Daniel Jerome Katz's love of what he does is infectious though luckily not fatal – I highly recommend trying to contract the disease. Tracey Ho proved, amongst many other things, that great minds can also think like lesser ones. Muriel Médard, Ralf Koetter, and Steven Low have been generous in many matters – most of all with their time, which they have donated at important check-points in my academic career. Robert J. McEliece, Jr., Babak Hassibi, and Leonard Schulman showed how the right candidacy committee at the right time can make a significant difference. The Braun house tower of oracles bears special mention – Mandyam Aji Srinivas, Aamod Khandekar, and Ravi Palanki poured down learning from on high. Sony John Akkarakaran's presence provided some of the simple pleasures of life. The Patils opened their home and family to a brash young DCBA; paying such generosity of spirit forward in equal measure will be hard.

Chapter 2 Abstract

The information theoretic aspects of large networks with many terminals present several interesting and non-intuitive phenomena. One such crucial phenomenon was first explored in a detailed manner in the excellent work [1]. It compared two paradigms for operating a network – one in which interior nodes were restricted to only copying and forwarding incoming messages on outgoing links, and another in which internal nodes were allowed to perform non-trivial arithmetic operations on information on incoming links to generate information on outgoing links. It showed that the latter approach could substantially improve throughput compared to the more traditional scenario. Further work by various authors showed how to design codes (called *network codes*) to transmit under this new paradigm and also demonstrated exciting new properties of these codes such as distributed design, increased security, and robustness against network failures.

In this work, we consider the low-complexity design and analysis of network codes, with a focus on codes for *multicasting* information. We examine both centralized and decentralized design of such codes, and also both randomized and deterministic design algorithms. We compare different notions of linearity and show the interplay between these notions in the design of linear network codes. We determine bounds on the complexity of network codes. We also consider the problem of error-correction and secrecy for network codes when a malicious adversary controls some subset of the network resources.

Contents

List of Figures

Chapter 3 Introduction

3.1 Background

Once in a while, a simple observation can have far-reaching consequences. Shannon's seminal results [59] forming the basis of information theory relied on the underlying ideas that data storage and transmission systems could be modeled stochastically, and that almost all codes are "good." Yet, efficient design and implementation of codes that achieve the rate region for these problems is not always easy; for many problems only the existence of good codes are known, and polynomial-time constructions, encoding and decoding is not known. Further, generalizing most point-to-point communication results to general networks turns out, for many problems, to be much harder. Much further work by many researchers led to results on networks with a "few" nodes or with some simple structure, but for many classical information theoretic problems even a tight characterization of the rate-region is not known.

Against this backdrop of unknown rate regions, computational intractability in code designs, and a lack of analytical tools to attack network information flow problems, the results of the field of network coding seem especially remarkable and exciting. The work in [1] examines the class of *multicast problems*, i.e., information flow problems where one source wishes to transmit all of its information over a network to a set of prespecified sinks, each of which wishes to receive all of the information. The classical paradigm for flow of information over a network involves intermediate nodes being passive copiers and forwarders of information on incoming links to outgoing links. Under this restricted class of operations, even computing the rate-region of multicast problems within a constant multiplicative factor is computationally intractable [35].

In contrast, as stated on the network coding home-page [48], "the core notion of network coding is to allow and encourage mixing of data at intermediate network nodes." The work in [1] gives a tight characterization of the rate region, such that the simple min-cut upper bounds is matched by random codes in which each intermediate node performs a random operation on its incoming messages to produce outgoing messages. Further, it can be shown [33] that the throughput achievable by network codes can be arbitrarily larger than the throughput achievable by routing-only schemes.

Recently, there has been a steady trend towards ever simpler designs and implementations of network codes. Work by [45] shows that the same rate region remains achievable even when all operations in the network are restricted to be linear over an appropriate field, and the work of [37] shows that such codes can be designed over appropriate finite fields and gives explicit (though exponential-time) algorithms to design such codes. Linear codes are important for three reasons. First, as shown by [37] and [45], restricting oneself to the class of linear codes does not reduce the capacity region for an important class of network coding problems that includes multicasting. Second, the complexity of implementation of such codes is polynomial in the blocklength n, which is attractive from an implementation point of view. Lastly, prior known results in linear algebra give us guidance in designing linear network codes with provably good performance; such guarantees are difficult to provide for a larger class of codes.

Independent work by [57] and [30] (combined in [33]) gives the first polynomial-time design algorithms for network codes. This sets the stage for the design of network codes that are not only low complexity in encoding and decoding, but also in design.

Concurrent and independent work by three groups [30], [58] (unpublished), and [25] examines the low-complexity distributed random design of network multicast codes. This set of results is particularly interesting from a network practitioner's point of view; they indicate a means of operating networks in a decentralized manner, and yet simultaneously attaining theoretically optimal throughputs. Such random distributed codes are provably robust against failure of network resources such as links and edges, and their throughput degrades gracefully with successively more serious network failures [7]. An excellent survey on random coding techniques and results can be found in [23].

Some other applications of network coding include [11], which considers the problem of quickly disseminating information from multiple sources to all nodes in a network, and [62], which shows how using network coding ideas in ad hoc wireless networks can reduce the average energy required per transmission. The work in [34], [18] considers secrecy issues for networks and shows how using network codes can help improve network security.

The interested reader is encouraged to visit the network coding home-page [48] to access more references.

3.2 Contributions

The central contributions of this thesis are in the areas of low-complexity deterministic and randomized designs of network codes for multicast problems, classification of types of linear network codes and analysis of their complexity, and of design of network multicast transmission protocols in the presence of a malicious hidden eavesdropping and jamming adversary.

Chapter 5 examines the low-complexity design of network codes for multicast problems. The first polynomial-time centralized code designs for both deterministic and randomized code design algorithms are demonstrated; these design algorithms make designing multicast network codes computationally tractable. A decentralized code design is also presented, which at the cost of an asymptotically negligible error probability allows for very low-complexity code design, resulting in codes that require minimal network management, and are robust against failures of network nodes and links. Also presented are some deterministic decentralized code designs that guarantee the correctness of the designed code, at the cost of greater complexity in either code design or code implementation.

In Chapter 6, we examine three different notions of linearity – algebraic, block, and convolutional linear network codes. For some of these reductions that convert codes designed under one notion of linearity into codes that are linear under another notion of linearity are shown. This allows for a single code design mechanism for all three types of linear codes, and also indicates methods for reconciling different types of linear operations in different parts of the network. For some other notions of linearity, it is shown that no such reductions can exist. A distinction is made between reductions for multicast network codes and those for general network coding problems. We also distinguish between reductions that are local, and can therefore be implemented in a decentralized manner, and those that are global, and therefore require a central controlling authority. These reductions show the advantages and limitations of each kind of linear network code, depending on the particular type of network coding problem at hand.

We analyze different notions of the complexity of implementation of network codes in Chapter 7. One notion, the delay complexity, considers the minimal alphabet size required for the network code to achieve optimal throughput. Upper and lower alphabet size bounds for the case of multicast network codes are presented; these bounds match up to a multiplicative factor of 2. It is shown that using convolutional codes can further reduce the required field-size by a factor of two. Another notion of complexity, the number of bit operations required for each encoding operation to generate a single decodable bit at the sink, is also examined. In particular, we design a class of randomized block codes we call *permute-and-add* codes and show that they require a number of bit operations that is the lowest possible – only as many as are required by routing-only network codes.

Lastly, we consider in Chapter 8 error-correction and secrecy for network codes when a malicious adversary controls some network resources. The motivation behind considering this problem is the scenario where a rogue hidden network component, either passively or actively malicious, injects fake information into the network; since interior nodes in a network code mix all information coming on incoming links to generate messages on outgoing links, this is potentially catastrophic, since *all* the information in a network could be corrupted by a single bad node. We design codes to transmit information in this scenario. The computationally unbounded, hidden adversary knows the message to be transmitted and can observe and change information over the part of the network he controls. The network nodes do not share resources such as shared randomness or a private key. We demonstrate that if the adversary controls a fraction $p < 0.5$ of the $|\mathcal{E}|$ edges, the maximal throughput equals $(1-p)|\mathcal{E}|$, otherwise it equals 0. We describe low-complexity design and implementation codes that achieve this rate region. We then extend these results to investigate more general multicast problems in networks with adversaries.

Chapter 4 Definitions

4.1 Graphs

Let $\mathcal{V}$ be a set of vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times \mathbb{Z}$ be a set of unit-capacity directed edges, where $e = (v, v', i) \in \mathcal{E}$ denotes the ith edge from v to v'. An edge of the form $e = (v, v, i)$ is called a *self-loop*. The tuple $(\mathcal{V}, \mathcal{E})$ defines a directed graph $\mathcal{G}$.

For a node $v \in \mathcal{V}$, let $\Gamma_O(v)$ denote the set of edges (v, v', i) outgoing from v and $\Gamma_I(v)$ denote the set of edges (v'', v, i) entering v. An edge $e = (v, v', i)$ is said to have *tail* v, denoted by $v = v_t(e)$, and *head* v', denoted by $v' = v_h(e)$.

An ordered set $\{u_1, i_1, u_2, i_2, \ldots, i_{n-1}, u_n\}$ is said to be a *path* $P(u_1, u_n)$ *from* u_1 *to* u_n *in* $\mathcal{G}$ if $(u_j, u_{j+1}, i_j) \in \mathcal{E}$ for all $j \in \{1, \ldots, n-1\}$. Two paths P and P' are said to be *edge-disjoint* if they do not share edges in common. A path $P(u, v)$ is said to be a *cycle* if $u = v$. A graph $\mathcal{G}$ is said to be *acyclic* if it contains no cycles.

For any $S \subseteq \mathcal{V}$, a *cut* $Cut(S) \subseteq \mathcal{E}$ is the set of all edges (v, v', i) such that $v \in S$, $v' \in \mathcal{V} \setminus S$. The *value of* $Cut(S)$, $|Cut(S)|$ equals the size of $Cut(S)$. A *min-cut from* v *to* v' $Mincut(v, v')$ is any $Cut(S)$ of minimum size such that $v \in S$ and $v' \notin S$. The *value of* $Mincut(v, v')$, $|Mincut(v, v')|$, equals the size of $Mincut(v, v')$.

4.2 Network Codes

The graph $\mathcal{G}$ contains pre-specified sets $\mathcal{S}$ of *source vertices* and $\mathcal{T}$ of *sink vertices*. For each $s \in \mathcal{S}$, $R(s) \in \mathbb{Z}$ is called the *source rate at* s. Time is discrete and indexed by non-negative integers. Let $\mathcal{X} = \{0, 1\}$. At time i, each $s \in \mathcal{S}$ generates Bernoulli-(1/2) random bits $X_i^s = \{X_{i,j}^s\}_{j=1}^{R(s)} \in \mathcal{X}^{R(s)}$. All X_i^s are independent and identically distributed.

A *connection* $\chi(s,t)$ is a triple $(s,t,X_i^{s,t}) \in \mathcal{S} \times \mathcal{T} \times \mathcal{X}_i^s$. The random variables $X_i^{s,t}$ comprise the *message from* s *to* t. The *rate from* s *to* t, $R(s,t)$ is defined as $|X_i^{s,t}|$. A *network coding problem* $\mathcal{P}(\mathcal{G})$ is a set $\{\chi(s,t)\}$ of connections in $\mathcal{G}$.

Network coding problems of particular interest are *multicast network coding problems.* In such problems, there is a single $s \in \mathcal{S}$ with source rate R. For each $t \in \mathcal{T}$, $X_i^{s,t} = X_i$.

Each $s \in \mathcal{S}$ possesses a *source encoder* Xavier$_s$. Each $t \in \mathcal{T}$ possesses a *sink decoder* Yvonne$_t$. Every other node in $\mathcal{V}$ possesses an *internal encoder.* A *network code* $\mathcal{C}$ is defined by its source encoders, internal encoders, and decoders at receiver nodes. Permissible arithmetic operations in $\mathcal{C}$ are now described.

Let *alphabet size* q be a design parameter for $\mathcal{C}$, and let $q = 2^b$ for some positive integer b. The *alphabet* $\{0, 1, \ldots, q-1\}$ *of* $\mathcal{C}$ is the finite field $\mathbb{F}_q$. For each $s \in \mathcal{S}$ source bits are blocked into b-dimensional vectors that are treated as elements of the finite field $\mathbb{F}_q$. In particular, $(X^s_{ib+1}, \ldots, X^s_{ib+b})$ becomes $X^s(i) \in \mathbb{F}_q$ and $(X^{s,t}_{ib+1}, \ldots, X^{s,t}_{ib+b})$ becomes $X^{s,t}(i) \in \mathbb{F}_q$.

We first consider *block network codes* for acyclic graphs. A design parameter for block network codes is the *blocklength* n. Let Y^e be the length-n vector transmitted across edge e, defined inductively as follows.

For each $s \in \mathcal{S}$ the source encoder Xavier$_s$ comprises a collection of functions $\{f^{s,e}\}_{e \in \Gamma_O(s)}$. For each $e \in \Gamma_O(s)$, $f^{s,e} : (\mathbb{F}_q)^{nR(s)} \rightarrow (\mathbb{F}_q)^n$ maps source vector X^s to the vector Y^e transmitted across edge e.

For each $s \in \mathcal{S}$ and each $e \notin \Gamma_O(s)$ the internal encoder is a function $f^{s,e} : (\mathbb{F}_q)^{n|\Gamma_I(v_t(e))|} \rightarrow (\mathbb{F}_q)^n$ that maps messages $Y^{e'}$ on all links e' incoming to $v_t(e)$ to the vector Y^e transmitted across edge e. In multicast networks with only one source, we abbreviate $f^{s,e}$ as f^e.

The *channel* g^e for each edge $e \in \mathcal{E}$ is an identity function unless explicitly defined otherwise.

For each $t \in \mathcal{T}$, the sink decoder Yvonne$_t$ comprises a collection of functions $\{h^{s,t}\}_{s \in \mathcal{S}}$. For each $s \in \mathcal{S}$, $h^{s,t} : (\mathbb{F}_q)^{n|\Gamma_I(t)|} \rightarrow (\mathbb{F}_q)^{nR(s,t)}$ maps the collection $Y^t = (Y^e)_{e \in \Gamma_I(t)}$ of received channel outputs to a reconstruction $\hat{X}^{s,t}$ of message $X^{s,t}$. For multicast network codes we denote the decoders as h^t.

The *error probability* is defined as

$$P_e^{(n)} = \mathbf{Pr}\{\exists \chi(s,t) \in \mathcal{P}(\mathcal{G}) \text{ such that } \hat{X}^{s,t}(Y^t) \neq X^{s,t}\}.$$

The rate-vector $R(\mathcal{P}(\mathcal{G})) = (R(s,t))_{\chi(s,t) \in \mathcal{P}(\mathcal{G})}$ is *achievable* if for any $\epsilon > 0$ and n sufficiently large there exists a blocklength-n network code $\mathcal{C}$ with $P_e^{(n)} < \epsilon$. The *capacity-region* C of the network $\mathcal{G}$ equals the convex hull of the achievable rates.

The network code $\mathcal{C}$ is said to *solve with zero error* the network coding problem $\mathcal{P}(\mathcal{G})$ if $\hat{X}^{s,t} = X^{s,t}$ for every $\chi(s,t)$ in $\mathcal{P}(\mathcal{G})$. The rate-vector $R_0(\mathcal{P}(\mathcal{G})) = (R(s,t))_{\chi(s,t) \in \mathcal{P}(\mathcal{G})}$ is *achievable without error* if for n sufficiently large there exists a blocklength-n network code $\mathcal{C}$ that solves $\mathcal{P}(\mathcal{G})$ with zero error. The *zero error capacity-region* C_0 of the network $\mathcal{G}$ equals the convex hull of the rates achievable without error. For multicast network codes C and C_0 are scalars.

Block network codes are not well-defined for networks with cycles. In such cases (and some other cases to be specified later) *sliding window network codes* $\mathcal{C}_{SW}$ are useful. Sliding window network codes differ from block network codes in their encoders and decoder definitions.

For each $s \in \mathcal{S}$ the source encoder Xavier$_s$ comprises a collection of functions $\{f^{s,e,SW}\}_{e \in \Gamma_O(s)}$. For each $e \in \Gamma_O(s)$ and for each time i, $f^{s,e,SW} : (\mathbb{F}_q)^{n(R(s)+|\Gamma_O(s)|)} \rightarrow \mathbb{F}_q$ maps $\{X^s(j), \{Y^e(j)\}_{e \in \Gamma_O(s)}\}_{j=i-n}^{i}$ to the ith symbol $Y^e(i)$ transmitted across edge e.

For each $s \in \mathcal{S}$ and all edges $e \notin \Gamma_O(s)$ and for each time i the internal encoder is a function $f^{e,SW} : (\mathbb{F}_q)^{n(|\Gamma_I(v_t(e))|+|\Gamma_O(v_t(e))|)} \to \mathbb{F}_q$ that at time i maps the n previous symbols $\{\{Y^e(j)\}_{e\in\Gamma_I(v_t(e))\cup\Gamma_O(v_t(e))}\}_{j=i-n}^{i}$ on all links e' incoming to and outgoing from $v_t(e)$ to the ith symbol $Y^e(i)$ transmitted across edge e. For multicast network codes, we denote the encoders as $f^{e,SW}$.

For each $t \in \mathcal{T}$, the sink decoder Yvonne$_t$ comprises a collection of functions $\{h^{s,t,SW}\}_{s\in\Gamma_O(s)}$. For each $s \in \Gamma_O(s)$, $h^{s,t,SW} : (\mathbb{F}_q)^{n(|\Gamma_I(t)|+|\Gamma_O(t)|)} \longrightarrow (\mathbb{F}_q)^{R(s,t)}$ maps the collection $Y^t = \{\{Y^e(j)\}_{e\in\Gamma_I(t)\cup\Gamma_O(t)}\}_{j=i-n}^{i}$ of n previous received channel outputs to a reconstruction symbol $\{\hat{X}^{s,t}(i)\}$ of message $\{X^{s,t}(i)\}$. For multicast network codes we denote the decoders as $h^{t,SW}$.

The error probability, capacity region and zero error capacity region for sliding window network codes are defined in a manner similar to that of block network codes.

4.3 Linear Network Codes

We define three important classes of linear network codes.

An $\mathbb{F}_{2^m}$*-network code* is a block network code such that the alphabet $q = 2^m$, the blocklength $n = 1$, each encoder $f^{s,e}$ and f^e is of the form $\Sigma_{e'\in\Gamma_I(v_t(e))}\beta^{e',e}Y^{e'}$ and each decoder $h^{s,t}$ is of the form $(\Sigma_{e'\in\Gamma_I(t)}\beta^{e',e}Y^{e'})_{e\in\Gamma_O(t)}$. Here all $\beta^{e',e}$ and $Y^{e'}$ are elements of $\mathbb{F}_{2^m}$ and all operations are linear over $\mathbb{F}_q$. Another name for $\mathbb{F}_{2^m}$ network codes is *algebraic linear network codes* [37].

An $(\mathbb{F}_2)^m$*-network code* is a block network code such that $q = 2$, $n = m$, each encoder $f^{s,e}$ and f^e is of the form $\Sigma_{e'\in\Gamma_I(v_t(e))}[\beta^{e',e}]\vec{Y}^{e'}$ and each decoder $h^{s,t}$ is of the form $(\Sigma_{e'\in\Gamma_I(t)}[\beta^{e',e}]\vec{Y}^{e'})_{e\in\Gamma_O(t)}$. Here all $[\beta^{e',e}]$ are $m \times m$ matrices over $\mathbb{F}_2$, $\vec{Y}^{e'}$ are length-m vectors over $\mathbb{F}_2$, and all operations are linear over $\mathbb{F}_2$. Another name for $(\mathbb{F}_2)^m$ network codes is *block linear network codes* [30].

A *degree-m $\mathbb{F}_2(z)$-network code* is a sliding window network code such that $q = 2$, $n = m$, each internal encoder $f^{e,SW}$ is of the form $\Sigma_{e'\in\Gamma_I(v_t(e))}\beta^{e',e}(z)Y^{e'}(z)$, and each decoder $h^{s,t,SW}$ is of the form $(\Sigma_{e'\in\Gamma_I(t)}\beta^{e',e}(z)Y^{e'}(z))_{e\in\Gamma_O(t)}$. Here all $\beta^{e',e}(z)$ are rational functions in z over $\mathbb{F}_2$ with degree of numerator and denominator at most m, $Y^{e'}(z) = \Sigma Y^{e'}(i)z^i$ are polynomials in z over $\mathbb{F}_2$, and all operations are linear over $\mathbb{F}_2(z)$. Another name for degree-m $\mathbb{F}_2(z)$ network codes is *degree-m convolutional linear network codes* [16]. If all $\beta^{e,e'}(z)$ are polynomials of degree at most m in z, then the network codes are said to be *FIR* (Finite Impulse Response) degree-m $\mathbb{F}_2(z)$-network codes. Otherwise, each $\beta^{e,e'}(z)$ is a ratio of polynomials of degree at most m in z, and such network codes are said to be *IIR* (Infinite Impulse Response) degree-m $\mathbb{F}_2(z)$-network codes.

For all three types of linear network codes, the *global coding measure for edge* $e \in \mathcal{E}$. describes the linear transformation of the information from the sources that traverses edge e. In particular, for algebraic network codes, if e carries the symbol $\beta_1 X(1)+\cdots+\beta_C X(C)$, then the *global coding vector* β^e equals $[\beta_1, \ldots, \beta_C]^T$. For block network codes, if e carries the length-n vector $[\beta_1]X(1)+\cdots+[\beta_C]X(C)$, then the *global coding matrix* $[\beta^e]$ equals the matrix $[[\beta_1]\ldots[\beta_C]]^T$. For convolutional network codes, if e carries the bit-stream whose z-transform equals $\beta_1(z)X_1(z) + \cdots + \beta_C(z)X_C(z)$, then the *global coding filter* $\beta^e(z)$ equals the vector $[\beta_1(z)\ldots\beta_C(z)]^T$.

We note that the above three types of linear networks codes are not the most general possible. For instance, the case of block linear network codes with infinite blocklength is not considered.

Chapter 5 Design of Multicast Network Codes

In this chapter we consider the design of linear network multicast codes. We consider both centralized and decentralized designs. In centralized design, a central authority with knowledge of the entire network's topology is in charge of designing the network code. This leads to efficient design of network multicast codes with guarantees of correctness. Some such codes were first considered in [37], but no polynomial-time design algorithms were known before [33].

The centralized design paradigm assumes the existence of a network management authority that not only has a large amount of knowledge about a possibly dynamic network, but also is able to dictate to individual network nodes the particular coding operations they must perform. Since centralized design is not always possible, we also consider a decentralized code design paradigm wherein each interior node of the network, chooses its own coding operations without knowledge of the code at other nodes in the network. The overall effect of these decisions can be percolated down the network using small headers, which enables the receivers to decode the messages received on incoming links. Similar results were concurrently shown in [25].

For both the centralized and decentralized paradigms we consider both deterministic and randomized code designs, which result in algorithms with varying properties in terms of design and implementation complexities.

We consider only designs for directed acyclic graphs. For networks with cycles, extensions of the arguments presented below appear in [16], where a centralized design is discussed, and in [27], where a distributed randomized design is proposed. The case of undirected networks is examined in [46].

5.1 Centralized Design

We first present a deterministic centralized algorithm, which provides intuition about the underlying algebraic formulation of the design problem. We begin with an informal outline, which describes the underlying principles in designing an $\mathbb{F}_{2^m}$*-network code.*

Our algorithm consists of two stages. In the first stage, a flow algorithm (see for instance [2]) is run to find, for each sink $t \in \mathcal{T}$, a *flow to sink* t, i.e., a set $F^t = \{P_j(s,t)\}_{j=1}^{t}$ of C edge-disjoint paths $P_j(s,t)$ from the source s to sink t. Only the edges in the union of these flows over all sinks are considered in the second stage of the algorithm.

The second stage is a greedy algorithm that visits each edge in turn and designs the linear code employed on that edge. The order for visiting the edges is chosen so that the encoding for edge e is designed after the encodings for all edges leading to e. The goal in designing the encoding for e is to choose a linear combination of the inputs to node $v_t(e)$ that ensures that all sinks that lie downstream from e obtain C linearly independent combinations of the C original source symbols $X = (X(1), \ldots, X(C))$. When designing a linear encoding function f^e for any edge e, the algorithm maintains a set $D^{t,e} \subset \mathcal{E}$, and a $C \times C$ matrix $B^{t,e}$ for each sink t. Set $D^{t,e}$, called the *frontier edge-set* of t, describes the most recently processed edge e in each of the C edge-disjoint paths in F^t. The C columns of $B^{t,e}$ correspond to the C edges in $D^{t,e}$, and the column for edge $e \in D^{t,e}$, denoted β^e, describes the linear combination of $X(1), \ldots, X(C)$ that traverses edge e. This is the *global coding vector for edge* e. That is, if e carries the symbol $\beta_1 X(1) + \cdots + \beta_C(C)$, then the corresponding column of $B^{t,e}$ is $\beta^e = [\beta_1, \ldots, \beta_C]^T$. The algorithm maintains the invariant that the matrix

$B^{t,e}$ of global coding vectors is invertible for every t and e, thereby insuring that the copy of $X(1), \ldots, X(C)$ intended for sink t remains retrievable with every new code choice.

The following lemma will help us prove the existence of encoders $\{f^e\}$ with the required properties. Let k be a fixed positive integer, and for each $i \in \{1, 2, \ldots, k\}$ let M_i be an arbitrary non-singular $n \times n$ matrix over the finite field $\mathbb{F}_q$. For all $i \in \{1, 2, \ldots, k\}$, let v_i be some row vector of M_i. Let $\mathbf{L}$ be any subspace of $(\mathbb{F}_q)^n$ containing the span of $\{v_1, \ldots, v_k\}$. For all $i \in \{1, 2, \ldots, k\}$, let $\mathbf{S}_i$ be the linear span of all the row vectors of M_i except v_i, i.e., $\mathbf{S}_i = span\{rows(M_i) \backslash v_i\}$. Let $\oplus$ represent the direct sum of two vector spaces. We are interested in the satisfiability of the following condition

$$\exists \text{ vector } v \text{ such that } \mathbf{S}_i \oplus v = (\mathbb{F}_q)^n \ \forall i \in \{1, 2, \ldots, k\}. \tag{5.1}$$

Lemma 1 *For $q \geq k$, (5.1) is always satisfiable. Further, the probability that a v chosen uniformly and at random in $\mathbf{L}$ satisfies (5.1) is greater than $1 - k/q$.*

Proof: We note that $\mathbf{L} - \cup_{i=1}^{k} \mathbf{S}_i$ equals $\mathbf{L} - \cup_{i=1}^{k} (\mathbf{L} \cap \mathbf{S}_i)$, and by using a counting argument on the set $\mathbf{L} - \cup_{i=1}^{k} (\mathbf{L} \cap \mathbf{S}_i)$ we shall get the required result. Let $d(\mathbf{L})$ be the dimension of the vector space $\mathbf{L}$. Then the number of vectors in $\mathbf{L}$ equals $q^{d(\mathbf{L})}$. For each $i \in \{1, 2, \ldots, k\}$, the dimension of the vector space $(\mathbf{L} \cap \mathbf{S}_i)$ is strictly less than $d(\mathbf{L})$, since $\mathbf{L}$ contains at least one vector, v_i, that is not contained in $\mathbf{S}_i$. Hence for each $i \in \{1, 2, \ldots, k\}$ the number of vectors in $\mathbf{L} \cap \mathbf{S}_i$ equals at most $q^{d(\mathbf{L})-1}$. Also, each $\mathbf{L} \cap \mathbf{S}_i$ contains the zero vector, and therefore the total number of vectors in $\cup_{i=1}^{k} (\mathbf{L} \cap \mathbf{S}_i)$ is no more than $k(q^{d(\mathbf{L})-1} - 1) + 1$. Since $k \leq q$, the probability that a randomly chosen $v \in \mathbf{L}$ satisfies $v \in \mathbf{S}_i$ for any $i \in \{1, 2, \ldots, k\}$ is at most k/q. $\square$

Our main result for centralized design is given in Theorem 2 below. A similar algorithm with asymptotically better running-time was independently presented in [57], and a combined version published in [33].

Theorem 2 ([30]) *For any $\mathcal{G}$ there exists an $\mathbb{F}_{2^m}$-network code $\mathcal{C}$ that solves the multicast network coding problem for any $R \leq C$ and $m = \lceil \log_2 |\mathcal{T}| \rceil$. The randomized complexity of designing $\mathcal{C}$ is $\mathcal{O}(|\mathcal{E}||\mathcal{T}|(C)^3)$, and the deterministic complexity of designing $\mathcal{C}$ is $\mathcal{O}(C|\mathcal{E}||\mathcal{T}|(C+|\mathcal{T}|)^2 + |\mathcal{E}||\mathcal{T}|^4)$.*

Proof: As shown in [1], the rate region is defined by $R \leq C$. We describe a design algorithm to produce codes at $R = C$.
Algorithm Inputs: The 4-tuple $(\mathcal{G}, s, \mathcal{T}, R)$.
Algorithm Outputs: The encoders f^e for all $e \in \mathcal{E}$, decoders $\{h^t\}_{t\in\mathcal{T}}$.
Preprocessing: A low time complexity maximum flow algorithm (for instance [2]) is run $|\mathcal{T}|$ times to obtain a set of edge-disjoint paths between s and each $t \in \mathcal{T}$. We define the *network flow $F^{\mathcal{T}}$ between s and $\mathcal{T}$* as the union of the edges in the flows of rate C to each t_i, i.e., $F^{\mathcal{T}} = \cup_{t\in\mathcal{T}} F^t = \{e \in \mathcal{E} : \exists j, t \text{ such that } e \in P_j(s,t)\}$. Our network only uses edges in $F^{\mathcal{T}}$, rather than possibly all edges in $\mathcal{E}$. The edges in $F^{\mathcal{T}}$ are numbered from 1 to $|F^{\mathcal{T}}|$ in a manner consistent with the partial order on $e \in \mathcal{E}$ (i.e., for any edges e and e', $v_h(e) = v_t(e')$ implies $e < e'$). Given the encoding functions f^e at each node, one can compute the global coding vectors β^e for each $e \in \mathcal{E}$ in a consistent manner by inductively computing $\beta^e = \sum_{e'\in\Gamma_I(v_t(e))} \beta^{e',e} Y^{e'}$. The set $\{\beta^{e',e} : e' \in \Gamma_I(v_t(e)\}$ defines the linear encoding function f^e.

Also, a *step-counter a* that keeps track of the edge encoding function is being designed is initialized to 1. Henceforth, $m = \lceil \log_2 |\mathcal{T}| \rceil$.

To make definitions easier, append edges $\{e(-1), e(-2), \ldots, e(-C)\}$ to the network such that $v_h(e(-j)) = s$ for all $j \in \{1, 2, \ldots, C\}$. The message $Y^{e(-j)}$ transmitted over edge $e(-j)$ is the symbol $X(j)$. For each $j \in \{1, 2, \ldots, C\}$, the global coding vector $\beta^{e(-j)}$ corresponding to $e(-j)$ is initialized as the length-C vector with 1 in the j^{th} position and 0 everywhere else. Thus the global coding vector matrices $B^{t,e}$

are initialized to the identity matrices for all sinks $t \in \mathcal{T}$.

Calculating f^e's: At each step $a \in \{1, \dots, |F^{\mathcal{T}}|\}$, the frontier edge set for sink t at step a is defined as an ordered set $D^{t,e(a)}$ containing C edges with the following properties. The j^{th} edge in $D^{t,e(a)}$ is the edge $e(a')$ in $P_j(s,t)$ with the largest value of a' less than or equal to a. The frontier edge set *matrix* for sink t at step a is the $C \times C$ matrix $B^{t,e(a)}$ whose rows are, respectively, the global coding vectors $\beta^{e(a)}$ for the edges in $D^{t,e(a)}$.

Thus for each a, there are C distinct edges in each frontier edge set $D^{t,e(a)}$. The edges in each $D^{t,e(a)}$ form a subset of a cut from s to t. At step a the design algorithm, for each $t \in \mathcal{T}$, $j \in \{1, \dots, C\}$, checks whether the j^{th} element of $D^{t,e(a)}$ is the immediate predecessor of $e(a)$ in $P_j(s,t)$. If so, this edge is replaced by $e(a)$ to obtain the updated frontier edge set $D^{t,e(a)}$. The algorithm then calculates the encoder $f^{e(step)}$ for $e(a)$, and therefore the corresponding global coding vector $\beta^{e(a)}$. The global coding vector matrices $\{B^{t,e(a-1)}\}_{t \in \mathcal{T}}$ are also updated by replacing the vectors corresponding to immediate predecessors of $e(a)$ with $\beta^{e(a)}$. In particular, $\beta^{e(a)}$ is chosen to be a global coding vector satisfying

$$\forall t \in \mathcal{T}, \ \operatorname{rank}(B^{t,e(a)}) = C. \tag{5.2}$$

By Lemma 1, this is always possible. In particular, set $n = C$ and $q = 2^m$. Let edge $e(t,a)$ be the edge in the frontier edge set $D^{t,e(a-1)}$ that is replaced at step a by $e(a)$, and M_i in Lemma 1 be $B^{t,e(a-1)}$. Finally, let $\mathbf{L}$ be the span of β^e for all $e \in \Gamma_I(e(a))$. Then a direct application of Lemma 1 proves that an appropriate $\beta^{e(a)}$ can always be found over a large enough field. We outline in the next two subsections two subroutines (one randomized and the other deterministic) that compute appropriate β^e and the corresponding f^e.

The step-counter a is then incremented by 1 and this procedure repeats until

$a = |F^{\mathcal{T}}|$. After the above procedure terminates, for each $t \in \mathcal{T}$ frontier edge set $D^{t,e(|F^{\mathcal{T}}|)}$ consists only of edges e such that $v_h(e = t$. At this stage, all β^es have been determined, and to terminate the algorithm we need to find the decoders $\{h^t\}_{t\in\mathcal{T}}$.

Calculating h^t: For each $t \in \mathcal{T}$, h^t is defined via matrix multiplication acting on a subset of Y^t. Let $\pi(Y^t)$ be a length-C column vector with j^{th} component equal to Y^e, where e is the last edge on the j^{th} path $P_j(s,t)$ between s and t. Then $\hat{X}^t = h^t(Y^t) = (B^{t,e(|F^{\mathcal{T}}|)})^{-1}\pi(Y^t)$; that is, $\pi(Y^t)$ multiplied by the inverse of the last global coding vector matrix for t. This operation is well-defined since by assumption the global coding vector matrices are invertible. Therefore, by the definition of the global coding vector matrices, for all $t \in \mathcal{T}$, $\hat{X}^t = (B^{t,e(|F^{\mathcal{T}}|)})^{-1}\pi(Y^t) = (B^{t,e(|F^{\mathcal{T}}|)})^{-1}(B^{t,e(|F^{\mathcal{T}}|)})X = X$. □

Note on field size: It is interesting to note that Lemma 1 is tight in the sense that for any $n > 1$ and q it is possible to construct $q+1$ subspaces $\mathbf{S}_i$ of dimension $n-1$ such that $\cup_{i=1}^{q+1}\mathbf{S}_i = (\mathbb{F}_q)^n$. Hence for such subspaces there is no v that satisfies (5.1). One particular example of such subspaces is the set of subspaces $\{S_i\}_{i\in\{0,1,\ldots,q\}}$, where S_i consists of all the vectors v such that $v(1)$ and $v(2)$ (respectively the first and second scalar elements of the length-n vector v) satisfy condition (5.3)

$$\begin{aligned} v(1) + iv(2) &= 0 \text{ if } i \neq q, \\ v(2) &= 0 \text{ if } i = q. \end{aligned} \tag{5.3}$$

It can be seen that any vector v must satisfy (5.3) for at least one value of i. However, this is a purely linear-algebraic condition; it is possible that there are no networks that actually require such a large field-size. Indeed, the best known lower bound on the field size is $q \geq \mathcal{O}\sqrt{|\mathcal{T}|}$ ([30],[44]), and it is conjectured that the minimum field-size required for any multicast problem is of this order of magnitude. To obtain an estimate better than $q \geq |\mathcal{T}|$ on the minimum field-size, one needs stronger tools than linear algebra; the topology of the network must be examined. One possible topological bound is the minimum in-degree of any node in the network.

5.1.1 Random Design of β^e

We present in this subsection a randomized subroutine to find an appropriate value for β^e for each $e \in \mathcal{E}$.

Randomized subroutine: Without loss of generality, let β^e for all $e \in \Gamma_I(v_t(e(a)))$ be linearly independent. We choose $f^{e(a)}$ uniformly and at random over all length-$|\Gamma_I(v_t(e(a)))|$ vectors. We test to see if the resulting $\beta^{e(a)}$ satisfies 5.1. If so, the subroutine exits successfully; otherwise, it loops back.

By Lemma 1 the probability of failure for any choice of $f^{e(a)}$ is at most $1 - |\mathcal{T}|/q$. Therefore the expected number of attempts to find a single $f^{e(a)}$ is $1/(1-|\mathcal{T}|/q)$, which is $\mathcal{O}(1)$ for large enough q. For each choice of $f^{e(a)}$, the computational cost of checking the invertibility of a single $D^{t,e(a)}$ equals the cost of doing Gaussian elimination on a $C \times C$ matrix. Therefore the expected cost of checking that the chosen $f^{e(a)}$ satisfies Equation 5.1 is $\mathcal{O}(C^3|\mathcal{T}|)$, and the expected cost of randomized design of a multicast network code is $\mathcal{O}(C^3|\mathcal{E}||\mathcal{T}|)$.

Note: If fast matrix-multiplication techniques are used, then the complexity can actually be reduced to $\mathcal{O}(C^\omega|\mathcal{E}||\mathcal{T}|)$, where ω is the best exponent for matrix inversion [8]. Similar reductions in time-complexity can be achieved in the following deterministic sub-routine.

5.1.2 Deterministic Design of β^e

We now present a deterministic algorithm for finding $f^{e(a)}$ for any edge $e(a)$.

Deterministic subroutine: Let v_i, S_i, $\mathbf{L}$, $d(\mathbf{L})$, $\mathbf{S}_i$, and k be as defined in Lemma 1. Let L be a $d(\mathbf{L}) \times C$ matrix whose rows form a basis for $\mathbf{L}$. First, all vectors are written in the basis L. Then, for each $i \in \{1, 2, \ldots, k\}$, row operations are performed on L and S_i to obtain the $(d(\mathbf{L})-1)\times d(\mathbf{L})$ matrix S'_i, whose rows form a basis for $\mathbf{L}\cap\mathbf{S}_i$ in the coordinate system given by the rows of L. The complexity of using Gaussian elimination to perform this operation is $\mathcal{O}(C(C+|\mathcal{T}|)^2)$ for each S'_i, for a total complexity of $\mathcal{O}(C|\mathcal{T}|(C+|\mathcal{T}|)^2)$ for this step.

Since each S'_i represents a $(d(\mathbf{L})-1)$-dimensional subspace of the $d(\mathbf{L})$-vector space S, the null-space of the transform is of dimension 1. Hence there exists a column vector s_i such that for any vector y' in the span of the rows of S'_i, the dot-product $y's_i^T$ equals 0. To obtain each such s_i, row operations are performed on S'_i until it contains the $(d(\mathbf{L})-1)\times(d(\mathbf{L})-1)$ identity matrix and a $(d(\mathbf{L})-1)$-length column vector s'_i. The vector s_i then equals the row vector obtained by adjoining the element 1 to $-s'_i$, that is, $s_i=(-s_i'^T,1)$. This gives us a compact representation of each S'_i. The time-complexity of this operation is $\mathcal{O}(|\mathcal{T}|^3)$ for each S'_i, for an overall complexity of $\mathcal{O}(|\mathcal{T}|^4)$ for this step.

Finding a v such that $v\in\mathbf{L}$ but $v\notin\mathbf{S_i}$ for any i is then equivalent to finding a length-$d(\mathbf{L})$ row vector y such that the dot-products ys_i satisfy

$$ys_i\neq 0,\ \text{for any } i\in\{1,2,\ldots,k\}. \tag{5.4}$$

The result is the length-C vector $v=yL$. A vector $y=(y_1,y_2,\ldots,y_{d(\mathbf{L})})$ satisfying (5.4) can be obtained via the greedy sub-subroutine described below, which requires $\mathcal{O}(|\mathcal{T}|^3)$ steps. Therefore the overall time-complexity is bounded by the complexity of finding S'_i and s_i, giving a total of $\mathcal{O}(|\mathcal{E}||\mathcal{T}|C(C+|\mathcal{T}|)^2+|\mathcal{E}||\mathcal{T}|^4)$.

Lastly, to obtain β^e, we note that the components of the vector v correspond to the elements $\{\beta^{e',e}\}_{e'\in\Gamma_I(v_t(e))}$ of the encoder $f^{e(a)}$.

Greedy Sub-subroutine: Our sub-subroutine computes elements of a length-k column vector c one at a time, such that any length-$d(\mathbf{L})$ vector y satisfying $Zy^T=c$ satisfies (5.4). We now show that this can be done *greedily.* This is, one can choose elements of $c = \{c_i\}_{i=1}^{k}$ one at a time, such that the choice of any c_i only depends on $\{c_j\}_{j<i}$, and the resulting C and corresponding y satisfy (5.4).

Denote the $k \times d(\mathbf{L})$ matrix whose i^{th} row vector is s_i^T by Z. The rank of this matrix is in general less than both k and $d(\mathbf{L})$; we partition it into two parts – the first a full-rank subset of rows, and the second a set of rows linearly dependent on the full-rank subset. Without loss of generality, let the first rank(Z) rows of Z be linearly independent, and denote this rank$(Z) \times d(\mathbf{L})$ sub-matrix by Z_I. Denote the matrix consisting of the remaining rows of Z by Z_D. Matrix Z_D can be written as the matrix product TZ_I for some $(k - \text{rank}(Z)) \times \text{rank}(Z)$ matrix T.

Our sub-subroutine greedily chooses elements of a length-rank(Z) column vector $c_I = \{c_i\}_{i \le \text{rank}(Z)}$. This choice induces the values $c_D = \{c_j\}_{j > \text{rank}(Z)}$, and also fixes the vector y that generates c as Zy^T.

Since all of the row vectors of Z are non-zero, the rank of Z is necessarily greater than zero. In the first step of the greedy sub-subroutine, some arbitrary value $c_1 \neq 0$ is chosen. Now there are two possibilities – either rank(Z) equals 1, or it is greater than 1. If rank(Z) equals 1, then c_I is assigned the value (1), a corresponding vector y is computed, and the sub-subroutine terminates. This choice of c works since all rows of Z are non-zero multiples of the first row, and therefore each c_i for $i > 1$ is also a corresponding non-zero multiple of 1. If rank$(Z) > 1$, more coefficients of c need to be calculated. We perform this computation inductively as follows. Let T_i and $T_{i,j}$ be the i^{th} row vector and the $(i,j)^{th}$ element of the matrix T, respectively. Consider all row vectors T_i of T that have non-zero elements only in the first b positions, and denote them by the superscript b, as T_i^b. Also, initialize counter b to 2. We proceed inductively in b. For each b we compute a non-zero constant, $c_b \in \mathbb{F}_q$, such that

$(Z_I)_b y^T$ equals c_b. To choose c_b, we note that

$$\begin{aligned} (Z_I)_i^b y^T \neq 0 \quad \Leftrightarrow \quad (Z_I)_b y \;&\neq\; -(T)_{i,b}^{-1}(\textstyle\sum_{j=1}^{b-1}(T)_{i,j}(Z_I)_j)\mathbf{y}^T \\ \Leftrightarrow \quad (Z_I)_b \mathbf{y} \;&\neq\; -(T)_{i,b}^{-1}\textstyle\sum_{j=1}^{b-1}(T)_{i,j}c_j = d_{i,b}, \end{aligned} \tag{5.5}$$

where each $d_{i,b}$ is a constant in $\mathbb{F}_q$. Since $\text{rank}(Z) > 1$, the number of rows in Z_D is at most $k-2$. Then $k \leq q = 2^m$ implies that there are at most $2^m - 2$ different values of $d_{i,b}$ for a fixed value of b. Hence we can always find a non-zero value of c_b for which $(Z_I)_b \mathbf{y}^T = c_b$ does not contradict (5.5). The counter b is incremented and the process repeated until $b = \text{rank}(Z)$. The entire greedy sub-subroutine requires $\mathcal{O}(|\mathcal{T}|^3)$ steps.

5.2 Decentralized Design

Often, the network topology is unknown to the code designer, source, or sinks. Only local information (such as number of immediately upstream and downstream nodes, and capacities on incoming and outgoing links) is available at nodes. In such a situation, decentralized design of network codes would considerably reduce the overhead required to first determine the network topology.

Decentralized design is also potentially useful in highly dynamic networks. Ideally, network codes should be robust against minor changes in network topology, such as a few nodes or links failing, or new internal nodes and sinks joining. A decentralized design can lend such robustness when network changes do not drastically alter the network's fundamental properties, (e.g., its multicast capacity).

The multicast capacity is the one global network parameter that must be known prior to multicast network code design; if one designs codes at a rate higher than the multicast capacity, it is possible that the sinks will not be able to retrieve even a single bit of information. We therefore assume that this information is known by the source prior to code design. In fact, our randomized code design suggests a simple decentralized algorithm to get a good estimate of the min-cut to any receiver.

We consider the distributed design of linear network multicast codes for directed acyclic networks.

5.2.1 Random Code Design

We examine the use of block linear network codes. The source s is assumed to have good bounds on the multicast capacity C and the network parameters $|\mathcal{E}|$ and $|\mathcal{T}|$. It then chooses a blocklength n and forwards the value of n to all downstream nodes, which in turn do the same until every node knows the value.

At this point, every node v independently selects a random linear encoding function f^e for each outgoing edge $e \in \Gamma_O(v)$. The global coding matrix for each $e \in \Gamma_O(v)$ is then computed and percolated down the network. This enables the sinks to compute the matrix inverse required to decode the messages coming in to them.

This paradigm was developed concurrently and independently by three groups [30], [58] (unpublished), and [25] and [27] (which consider algebraic network codes designed in this distributed randomized manner).

We examine a version of the distributed design of block linear network codes that was presented in Theorem 3 [30]. This design was proposed in [30] as a means of obtaining robustness against failures of network components, corruption by malicious adversaries, security against eavesdroppers, and complexity of implementation.

We now present our code design scheme, followed by the proof of correctness.

Randomized Code Design: Each node v, for each $e' \in \Gamma_I(v)$, $e \in \Gamma_O(v)$, chooses a $n \times n$ coefficient matrix $[\beta^{e',e}]$ such that every element of $[\beta^{e',e}]$ is chosen independently from every other, equals 1 with probability p, and is 0 otherwise. Probability p is a design parameter of the problem and can take any value in $[\log(n)/n, 0.5]$. The resulting random block network code is denoted by $\mathcal{C}_R(p)$.

We define *network failure patterns* as follows. Let $\mathcal{E}_F \subset \mathcal{E}$ be any subset of edges that fail, thereby passing only an infinite string of 0s. If edge e fails, then node $v_h(e)$ percolates news of this failure down the network to the receiver and treats the input from e as the all zeroes vector in performing its linear encoding. Effectively, this is equivalent to $v_h(e)$ ignoring the input on e. Let the multicast capacity of the resulting network be denoted by C_F. The failure of any node v can also be treated in this framework, by assuming instead the failure of all edges in $\Gamma_O(v)$.

The following theorem on random matrices is crucial in proving our result.

Theorem 3 *([3, Corollary 2.4]) There exists a universal constant A such that for any $p \in [(\log(n))/n, 0.5]$, any $n > 1$, and $\epsilon > 0$, the probability that an $n \times n$ binary matrix with entries independently chosen to be 1 with probability p and 0 otherwise has rank less than $n\epsilon$ is at most $A2^{-n\epsilon}$.*

This result shows that, with high probability, even very sparse random matrices are close to full rank. This enables us to show that with high probability, block network codes composed of sparse linear transformations still achieve rates close to capacity. We prove several closely related results.

Theorem 4

1. *There exists a universal constant A such that for any $\epsilon > 0$ and $n > 1$, the probability that the achievable zero error multicast rate using $\mathcal{C}_R(p)$ is at least $C - (|\mathcal{E}| + 1)\epsilon$ is at least $1 - 2^{-n\epsilon + \log|\mathcal{E}||\mathcal{T}|}$.*

2. *There exists a universal constant A such that for any $\epsilon > 0$, $n > 1$, and any failure pattern $\mathcal{E}_F$, the probability that the achievable zero error multicast rate using $\mathcal{C}_R(p)$ is at least $C_F - (|\mathcal{E}| + 1)\epsilon$ is at least $1 - A2^{-n\epsilon + \log|\mathcal{T}| + |\mathcal{E}|}$.*

3. *There exists a universal constant A such that for any $\epsilon > 0$, $n > 1$, and any set of source nodes $\mathcal{S}$ including a prespecified source node s, the probability that the*

achievable zero error multicast rate using $\mathcal{C}_R(p)$ is at least $C - (|\mathcal{E}|+1)\epsilon$. is at least $1 - A2^{-n\epsilon+\log|\mathcal{T}|+|\mathcal{V}|}$.

Proof:

1. As in the centralized design, we impose a partial order on the edges in $\mathcal{E}$. We also define for each $t \in \mathcal{T}$ and each $a \in \{1, \ldots, |\mathcal{E}|\}$ a frontier edge-set $D^{t,e(a)}$ and corresponding global coding vector matrix $B^{t,e(a)}$, which represents the bits on each edge as a linear combination of the source bits.

   As our inductive hypothesis, we assume that the rank of $B^{t,e(a)}$ is $nC - a\epsilon$. We then show that, with high probability, the rank of $B^{t,e(a+1)}$ is $nC - (a+1)\epsilon$.

   The linear transform $f^{e(a)}$, which takes $B^{t,e(a)}$ to $B^{t,e(a+1)}$, leaves all but C rows of $B^{t,e(a)}$ unchanged. Let e' denote the corresponding edge in $\Gamma_I(v_t(e(a)))$. We use $B_1^{t,e(a)}$ to denote the $n(C-1) \times nC$ submatrix of $B^{t,e(a)}$ that remains unchanged and $B_2^{t,e(a)}$ to denote the remaining $n \times nC$ submatrix of $B^{t,e(a)}$. Edge-set $\Gamma_I(e(a))$ may contain both edges that are in $D^{t,e(a)}$ and edges that are not elements of $D^{t,e(a)}$. Therefore to obtain the new global coding matrix $B^{t,e(a+1)}$, $f^{e(a)}$ replaces the submatrix $B_2^{t,e(a)}$ by the submatrix

$$B_2^{t,e(a+1)} = \sum_{e' \in \Gamma_I(v_t(e(a))) \cap D^{t,e(a)}} [\beta^{e',e(a)}] B_2^{t,e(a)} + \sum_{e'' \in \Gamma_I(v_t(e(a))) \setminus D^{t,e(a)}} [\beta^{e'',e(a)}] L B^{t,e(a)}.$$

   The matrix L captures the linear combinations in predecessor edges of e that are not in frontier edge-set $D^{t,e(a)}$. To prove the inductive step, we need to show that the rows of $B_2^{t,e(a+1)}$ are nearly linearly independent of those of $B_1^{t,e(a)}$ (i.e., the dimension of the subspace of intersection is at most $n\epsilon$), and further that the rank of $B_2^{t,e(a+1)}$ is with high probability at least $n(1-\epsilon)$. By examining $B^{t,e(a+1)}$ in a basis composed of rows from $B^{t,e(a)}$ and performing Gaussian elimination on the rows of $B^{t,e(a+1)}$, one can see that both of these conditions are satisfied if and only if the $n \times n$ times submatrix of $B^{t,e(a+1)}$ corresponding to $e(a)$ has rank $n(1-\epsilon)$. This submatrix, denoted $B(t, e', e(a))$, equals $[\beta^{e',e(a)}] + L'$, where L' equals the appropriate sub-matrix of $\sum_{e'' \in \Gamma_I(v_t(e(a))) \setminus D^{t,e(a)}} [\beta^{e'',e(a)}] L B^{t,e(a+1)}$. Suppose the rank of this matrix is less than $n(1-\epsilon)$. Then there are at least $n(1-\epsilon)+1$ linearly independent vectors of length C, say $y_1, \ldots, y_{n(1-\epsilon)+1}$, for which $[\beta^{e',e(a)}]y_i = L''y_i$. By Theorem 3, the probability that this occurs is less than $A2^{n\epsilon}$. Since there are $|\mathcal{T}|$ global coding matrices to consider, the probability of a rank-loss greater than $n\epsilon$ for any a is at most $1 - |\mathcal{T}|2^{n\epsilon}$ by the union bound. Therefore, the corresponding probability after $|\mathcal{E}|$ steps is at most $1 - |\mathcal{T}||\mathcal{E}|2^{n\epsilon}$. Assuming that the rank-loss at each step a is at most $n\epsilon$, the overall rank-loss after at most $|\mathcal{E}|$ steps, equals at most $|\mathcal{E}|n\epsilon$, leading to an overall rank-loss $|\mathcal{E}|\epsilon$, which is asymptotically negligible in n.

2. The proof is very similar to that in the previous part of the theorem, except here we need to take the union bound over all possible failure patterns as well. Since there are at most $2^{|\mathcal{E}|}$ possible failure patterns, we have the required result.

3. The proof is again similar to that of the first part of the theorem. The difference here is that the union bound also needs to be taken over all possible sets of source terminals for $\mathcal{G}$. Since this is at most the power-set of $\mathcal{V}$, the union bound has at most $2^{|\mathcal{V}|}$ terms. In each case, the multicast capacity is still at least C since the min-cut from a set of vertices including s to a sink t is at least as large as the min-cut from s to t.

□.

Distributed min-cut estimation: As a corollary to the above code design, we have a simple means of coming up with an estimate of the min-cut from s to any t. The source transmits random vectors of length $|n\Gamma_O(s)|$, with n distinct random vectors on each outgoing edge from s. Each intermediate node chooses uniformly at random among all linear combinations of vectors on incoming edges, producing n vectors on each outgoing edge. Each sink estimates its corresponding min-cut by calculating the rank of the collection of length-$|n\Gamma_O(s)|$ vectors it receives and normalizing by n. The result is a lower bound on the min-cut. By the above theorem, it is also within $|\mathcal{E}|\epsilon$ of the true value with probability exponentially close to 1 in n. Choosing ϵ such that $|\mathcal{E}|\epsilon < 1$ implies that the estimate is the true value of the min-cut (since it must be an integer). Further, while the above estimate may underestimate the min-cut with exponentially small probability, as a lower bound it is always correct.

5.2.2 Deterministic Code Design

We now discuss various multicast code design algorithms that require only limited global topological information. These algorithms guarantee codes with no errors, rather than codes with asymptotically negligible error probabilities. This guarantee is met without centralized design.

In our model,

1. Each node only knows information about the network that can be percolated down to it from upstream nodes.
2. The source node has a good estimate of the mincut.
3. Each node has an *identification number, u*. Each identification number is assigned to only one node in the network. Let the maximal such number be U.
4. Nodes have good upper bounds on $|\mathcal{V}|$. This last assumption is not critical. We also devise more complicated codes which do not require this information.

We call any code designed with just the above information a *distributed network code*. This relatively small amount of global information enables distributed zero error network code design. In comparison, the centralized design of multicast network codes requires information corresponding to an entire cutset at each node.

We first prove a result for network multicast problems where the transmission rate is 2. This case has also been examined in [4], in which code design is only somewhat decentralized. We begin by stating some definitions and known results about binary polynomials.

An *irreducible polynomial over* $\mathbb{F}_q$ is a polynomial with coefficients in $\mathbb{F}_q$ that cannot be written as the product of polynomials of strictly smaller degree.

Lemma 5 *The number of irreducible binary polynomials of degree m is $\mathcal{O}(m^{-1}2^m)$.*

Lemma 5 is a direct corollary of well-known bounds (see, for example, [40]) on the number of irreducible polynomials of degree m.

Theorem 6 *For any $\mathcal{G}$ with multicast capacity at least 2, and any multicast network coding problem $\mathcal{P}(\mathcal{G})$, a degree-$\mathcal{O}(\log(|\mathcal{V}|U))$ $\mathbb{F}_2(z)$ zero error distributed network code with rate 2 can be designed.*

Proof: We first give a code design that results in encoders with higher than necessary degree. After design of its encoding functions, each node percolates the global coding filter used on an edge down that edge. The encoding functions are as follows. Each node v receives on incoming edges $\Gamma_I(v)$ either one or two linearly independent combinations of the source information vector $(X_1(z), X_2(z))$. If it receives just one linearly independent combination, then it transmits this information unchanged down all outgoing edges. If it receives two linearly independent combinations of $(X_1(z), X_2(z))$, this enables it to reconstruct both $X_1(z)$ and $X_2(z)$ [16]. On the jth outgoing link from v, it then transmits $X_1(z) + p_u^j(z)X_2(z)$, where $p_u^j(z)$ is the j^{th} power of the u^{th} irreducible polynomial (according to the natural lexicographic ordering on binary polynomials). By [40], the u^{th} polynomial is of degree $\mathcal{O}(\log(u))$. Each node is connected to each of at most $|\mathcal{V}|$ other nodes by at most two links (any extra links can be removed since the rate of transmission is at most two). Therefore the maximum degree of any polynomial in the global coding vectors of the network code is $\mathcal{O}(|\mathcal{V}|\log(U))$.

The previous design is distributed and requires no knowledge $|\mathcal{V}|$. The code yields a rate-2 network code by the following argument. We first show that each global coding filter on each edge is of the form $(1, q^e(z))$, where $q^e(z)$ is a binary polynomial with the property that for any pair of edges e and e', $q^e(z) = q^{e'}(z)$ if and only if the min-cut between s and $\{v_t(e), v_t(e')\}$ equals 1. This is true since if there are no edge-disjoint paths from s to $\{v_t(e), v_t(e')\}$, then e and e' cannot possibly carry linearly independent information. Conversely, if there are two edge-disjoint paths from s to $\{v_t(e), v_t(e')\}$, then the two edges leading out of s on each of these paths carry different global coding filters, and, inductively at each stage, the global coding vectors must be different. (By the definition of irreducible polynomials, for any two distinct pairs (u, j) and (u', j'), $p_u^j(z) \neq p_{u'}^{j'}(z)$.) This proves that the network code has rate 2, since any two global coding filters $(1, q^e(z))$ and $(1, q^{e'}(z))$ must be linearly independent. To decode, each receiver chooses two linearly independent incoming global coding filters and inverts the corresponding transformation.

We now present a conceptually more complicated design that results in filters with lower degrees, and in addition does not require any node to know how many nodes exist in the network.

Cantor diagonal assignment: Arrange binary polynomials in a two-dimensional lattice so that the $(i, j)^{th}$ element corresponds to the $((i + j - 2)(i + j - 2)/2 + j)^{th}$ binary polynomial according to the natural lexicographic ordering on binary polynomials. It can be seen that this corresponds to arranging the lexicographically ordered binary polynomials along successive diagonals in the positive quadrant of the two-dimensional integer lattice. The idea for this enumerative scheme is borrowed from Cantor's celebrated theorem [5] showing the bijection between integers and rational numbers.

If a node has only one linearly independent global coding filter on its incoming edges, it forwards that on all outgoing edges. If it has two, then for the j^{th} outgoing edge from the node it choose the global coding filter $(1, q_{u,j}(z))$, where $q_{u,j}(z)$ corresponds to the polynomial in the $(u, j)^{th}$ position on the lattice. This assignment scheme still results in global coding filters that are pairwise linearly independent for any two edges for which there exist edge-disjoint paths to the two edges. Hence the scheme gives codes that are decodable. Also, the highest degree of any polynomial in any global coding vector (and hence in any local encoder) can be seen to be $\mathcal{O}(\log(|\mathcal{V}|U))$. $\square$

The previous distributed design only works for multicast codes of rate 2. We now demonstrate distributed design algorithms for general multicast rates R; the cost of higher rate is either higher design complexity, or higher implementation complexity.

We first state definitions and a result which will be useful in our code design. The following result demonstrates an elegant connection between the combinatorial properties of a network and the existence of a network code.

Let $\mathcal{C}$ be a convolutional linear multicast code for the multicast network coding problem $\mathcal{P}(\mathcal{G})$. For any such code, we define the following three matrices. The *input matrix* $A(z)$ is the $C \times |\mathcal{E}|$ matrix comprising the source encoders; the $(i, j)^{th}$ element of $A(z)$ equals the filter $\beta^{e(-i),e(j)}(z)$, where $\beta^{e(-i),e(j)}(z)$ equals zero if $v_t(e(j)) \neq s$. The *line graph matrix* $F(z)$ is the $|\mathcal{E}| \times |\mathcal{E}|$ matrix comprising the internal encoders; the $(i, j)^{th}$ element of $F(z)$ equals the filter $\beta^{e(i),e(j)}(z)$, where $\beta^{e(i),e(j)}(z)$ equals zero if $v_h(e(i)) \neq v_t(e(j))$. The *output matrix* $B(z)$ is the $|\mathcal{E}| \times |\mathcal{T}|C$ matrix comprising the decoders; the $(i, j)^{th}$ element of $B(z)$ equals the decoder filter $\beta^{e(i),t}(z)$, where $\beta^{e(i),t}(z)$ equals zero if $v_h(e(i)) \neq t$ for some $t \in \mathcal{T}$.

Theorem 7 (Theorem 2 [27]) *Convolutional code $\mathcal{C}$ solves $\mathcal{P}(\mathcal{G})$ if and only if the* *corresponding Edmonds matrix is non-singular.*

$$E(z) = \begin{bmatrix} A(z) & 0 \\ I - F(z) & B(z) \end{bmatrix} \tag{5.6}$$

Theorem 8 shows a simple decentralized design of convolutional network multicast codes; the codes employ exponentially large encoding and decoding filters.

We perform a pre-processing step to trim the network so that there are no more than C edges between any two nodes.

Theorem 8 *Let $\beta^{u,i,j}(z)$ represent the transfer coefficient from the i^{th} incoming edge to the j^{th} outgoing edge at the node with identification u. Let*

$$\beta^{u,i,j}(z) = z^{2^{u((|\mathcal{V}|-1)C)^2+(|\mathcal{V}|-1)C)i+j}}.$$

Then the corresponding convolutional code $\mathcal{C}$ solves the network coding problem.

Proof: The function $u((|\mathcal{V}|-1)C)^2+(|\mathcal{V}|-1)C)i+j$ is distinct for any two distinct triples of (u,i,j). Therefore each $\beta^{u,i,j}(z)$ equals z exponentiated to a distinct power of 2. The determinant of the Edmonds matrix $E(z)$ equals the sum of powers of z, each of which consists of the product of a single $\beta^{u,i,j}(z)$ from each row of the Edmonds matrix. Therefore each term equals z exponentiated to a distinct integer, and therefore the sum cannot be zero. Hence the Edmonds matrix is non-singular. By Theorem 7, this implies that the code $\mathcal{C}$ solves the multicast network coding problem $\mathcal{P}(\mathcal{G})$. □

Note 1: In the above proof we assume that each node knows the value $|\mathcal{V}|$. As in Theorem 6, this can be replaced with a Cantor-diagonal assignment scheme, which obviates this requirement. Under such a scheme, we only need to maintain the requirement that each $\beta^{u,i,j}(z) = z^{2^{\nu(u,i,j)}}$, where $\nu(u,i,j)$ is an integer-valued function that takes distinct values for every distinct triple (u,i,j).

Note 2: Another scheme, which requires lower complexity filters (though still exponential in length), was proposed in [24].

The above theorem is interesting in that it shows a conceptually simple distributed design for a zero error multicast network code. However, it requires the use of prohibitively expensive encoding and decoding operations, and is hence of only academic interest.

We now show that there exist codes that require only polynomial complexity to implement. We have only an existence proof, no polynomial-time construction has been found to date. The proof is a simple extension of Theorem 16 in [37]. Once again, we pare $\mathcal{G}$ so that there are at most C edges between any pair of nodes.

Theorem 9 *For any $\mathcal{P}(\mathcal{G})$ there exists a distributed degree-$\mathcal{O}(|\mathcal{V}|^2C)$ $\mathbb{F}_2(z)$ convolutional network code that solves $\mathcal{P}(\mathcal{G})$.*

Proof: The proof follows directly from the proof of Theorem 16 in [37], which proves the existence of a single network code that achieves rate C over a sufficiently large field for any of a family of networks, each with multicast capacity C. In our case, nodes in general do not have knowledge of the entire network, but under the assumption that they have an upper bound on the size of the network, can individually construct an appropriate family of networks in which $\mathcal{G}$ is contained.

The family of networks we consider, $\mathcal{J}(C, |\mathcal{V}|)$, consists of all subgraphs with min-cut at least C, of the graph $\mathcal{G}(C, |\mathcal{V}|)$ that has $|\mathcal{V}|$ and C edges between each pair of nodes. By assumption, $\mathcal{G}$ is in $\mathcal{J}(C, |\mathcal{V}|)$. A (loose) upper bound on $|\mathcal{J}(C, |\mathcal{V}|)|$ is the size of the power-set of the number of edges in $\mathcal{J}(C, |\mathcal{V}|)$. Each of $|\mathcal{V}|$ nodes in $\mathcal{G}$ is connected to most $|\mathcal{V}| - 1$ other nodes, with at most C edges each. Therefore the number of $\mathcal{J}(C, |\mathcal{V}|)$ $|\mathcal{V}|^2C$. By Theorem 16 in [37], there exists a network code with filter-size at most $\mathcal{O}(|\mathcal{V}|^2 C)$ that achieves rate C for all networks in $\mathcal{J}(C, |\mathcal{V}|)$, and therefore in particular achieves rate C for $\mathcal{G}$. $\square$

Chapter 6 Relationships between Types of Linear Network Codes

To begin studying linear network codes, we must first answer the question "What is linearity?" [39]. In this chapter we study relationships between algebraic, block, and convolutional network codes. (We distinguish between FIR convolutional codes and IIR convolutional codes.)

For a given network coding problem $\mathcal{P}(\mathcal{G})$, a *reduction* is a transformation of network code $\mathcal{C}$ into a new network code $\mathcal{C}'$. We investigate reductions that enable us to transform one type of linear network code (such as algebraic, block, or convolutional) into another. Some of the proposed reductions apply only to a limited class of network coding problems (e.g., multicasting), while others apply to general network coding problems (multiple sources, multiple sinks). We distinguish between three types of reductions.

Results currently in the literature deal with *global reductions.* A global reduction for a class of networks implies that codes defined under one notion of linearity can be replaced at every node by codes defined under another notion of linearity, and the new codes have identical rate regions.

However, for practical codes on networks, distributed design and implementation is desirable. We therefore consider *local reductions.* A local reduction is a design algorithm for replacing codes of one type at every node of a network with codes of another type, independently of the global network structure; the result must be a code of the second type with a rate vector identical to that of the code of the first type.

Input-output nonequivalent local reductions are local reductions that do not necessarily preserve the input-out transfer function of every node. These reductions are of interest because they enable distributed design for a new type of code when distributed design is possible for the initial family of codes.

Input-output equivalent local reductions are local reductions that preserve the input-output transfer function of every node. This class of reductions is of interest since this enables different nodes in a network to employ different notions of linearity.

We note that the existence of local input-output equivalent reductions implies the existence of local input-output nonequivalent local reductions, which in turn implies the existence of global reductions. In the reverse direction, a counter-example for global reductions for a class of networks implies that local input-output nonequivalent reductions for that class of networks do not exist, which in turn implies that input-output equivalent reductions for that class of networks do not exist. In a similar vein, the multicast network coding problem is a subset of the general network coding problem, and a counter-example of a reduction for a multicast network coding problem implies that no such reduction would exist for general network coding problems.

A summary of our and other previously known results is provided in Figure 6.1. We summarize only the strongest known reductions or counter-examples.

Finally, we introduce the class of *filter bank network codes*, which subsume each of the three previously mentioned classes of linear network codes. We show that under the assumptions of causality, finite memory, and L-shift-invariance (invariance of operations at nodes under shifts by an integer L) there exists a local input-output equivalent reduction between any code linear over a finite field and a corresponding filter bank network code.

We begin by reminding ourselves of some basic finite field arithmetic. In this chapter we consider finite fields which are extensions over any prime p, rather than

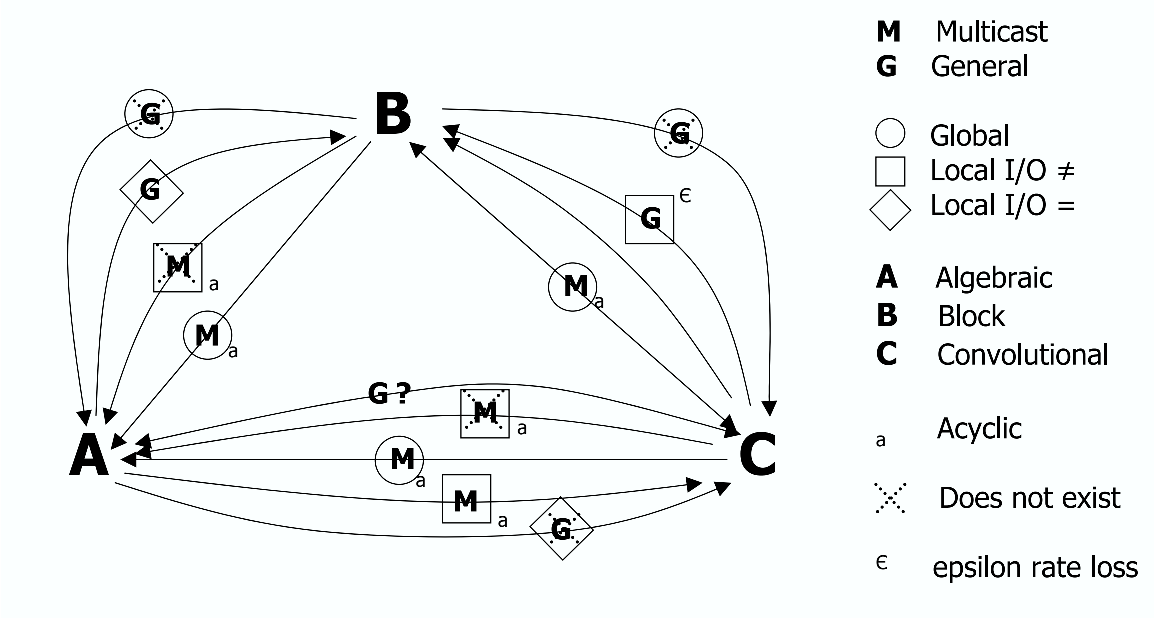

Figure 6.1: Diagrammatic representation of relationships between different notions of linearity

simply extensions of the binary field. Let $q = p^m$. Let $P_m(z)$ be the degree-m irreducible polynomial that is used to represent elements in the finite field $\mathbb{F}_q$. That is, all additions and multiplications over $\mathbb{F}_q$ are computed via corresponding additions or multiplications of p-ary polynomials modulo $P_m(z)$. This defines a one-to-one onto linear mapping between the set of all elements of $\mathbb{F}_q$ and the set of all p-ary polynomials of degree less than m. Each such p-ary polynomial can also be represented by the length-m bit vector comprising of the coefficients of the polynomial. Several of our proofs rely on the linear bijection $\mathcal{B}_{P_m} : \mathbb{F}_q \rightarrow (\mathbb{F}_p)^m$ that takes an element in the finite field $\mathbb{F}_q$ to its length-m bit-vector representation in $(\mathbb{F}_p)^m$. For example, $\mathcal{B}_{P_m}$ takes the multiplicative identity in $\mathbb{F}_a$ to the length-m bit vector with a 1 in the

mth position and zeroes everywhere else.

Reductions between Algebraic and block network codes

General networks

For general networks, there exists an input-output equivalent local reduction from algebraic network codes to block network codes. Even global reductions in the opposite direction are impossible.

Lemma 10 *For any algebraic network code $\mathcal{C}_{A,p,m}$ that solves network coding problem $\mathcal{R}_{\mathcal{G}}$ there exists an input-output equivalent local reduction to a block network code $\mathcal{C}_{B,p,m}$.*

Proof: Let $\vec{u_i}$, $i \in \{1, \ldots, m\}$ be unit vectors in $\mathbb{F}_q$. Given any $\beta_{i',i} \in \mathbb{F}_q$, we define the corresponding $[\beta_{i',i}]$ so that its ith row vector equals $\mathcal{B}_{P_m}(\beta_{i',i}\mathcal{B}_{P_m}^{-1}(\vec{u_i}))$. To see that this preserves the input-output relationship at every node, consider the following. For all $i \in \{1, \ldots, m\}$, let $u_i \in \mathbb{F}_q$ equal $(z^{i-1})mod(P_m(z))$. Then the span of $\{u_i(z)\}_{i=1}^m$, with scalars from $\mathbb{F}_p$, is exactly the set of p-ary polynomials of degree less than m. Hence any element $\alpha \in \mathbb{F}_{p^m}$, denoted as the polynomial $(\alpha(z))mod(P_m(z))$, may be written as $\sum_{i=1}^m (a_i u_i(z))mod(P_m(z))$, where $a_i \in \mathbb{F}_p$ for all $i \in \{1, \ldots, m\}$. Therefore, due to the linearity of $\mathcal{B}_{P_m}$, we are done. $\square$

Lemma 11 ([52],[44]) *There exist network coding problems $\mathcal{R}_{\mathcal{G}}$ that are solved by block network codes $\mathcal{C}_{B,p,m}$, such that there are no algebraic network codes $\mathcal{C}_{A,p',m'}$ that solve $\mathcal{R}_{\mathcal{G}}$ for any p' and m'.*

Multicast networks without cycles

While Lemma 11 proves that there does not exist a reduction from general block network codes to algebraic network codes, this failure does not necessarily apply for every network coding problem. We here examine the special case of multicast networks.

By Lemma 10, there is a local input-output equivalent reduction from algebraic network codes to block network codes for multicast networks. Further, by [37],[33], algebraic network codes are optimal for acyclic multicast problems, which implies that there exists a global reduction from algebraic network codes to block network codes. (The case for networks with cycles is different and will be discussed in the next subsection.)

Lemma 12 shows that local input-output non-equivalent reductions from algebraic network codes to block network codes are not possible for multicast networks.

Lemma 12 *For all n_1 let $\mathcal{C}_{B,2,2}(\mathcal{G}_n)$ (i.e., the $(\mathbb{F}_2)^2$-block network code for all networks of the form Figure 6(a)) be such that $[\beta_1] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ and $[\beta_2] = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$. Then for any finite field $\mathbb{F}_{p^m}$ there exists a network $\mathcal{G}_N$ such that there does not exist an algebraic network code $\mathcal{C}_{A,p,m}$ that is local input-output non-equivalent to $\mathcal{C}_{B,2,2}(N)$.*

Proof: Code $\mathcal{C}_{B,2,2}(\mathcal{G}_n)$ achieves a rate of 1 bit per coding instant. We wish to replace each $[\beta_1]$ matrix on the left branch of the networks in Figure 6 with an element, say β_1, from a suitable finite field $\mathbb{F}_q$. We also wish to replace each $[\beta_2]$ matrix on the right branch with another element, say β_2. Since the finite field $\mathbb{F}_{p^m}$ is a cyclic group of order p^m under multiplication, for any β_1, β_2 in $\mathbb{F}_{p^m}$, $\beta_1^{p^m} = \beta_2^{p^m} = 1$. Thus, if the network $\mathcal{G}_n$ in Figure 6 is such that $n = N - 1$, the messages from the two branches will destructively interfere at the output, and the sink will receive 0 regardless of the input . □

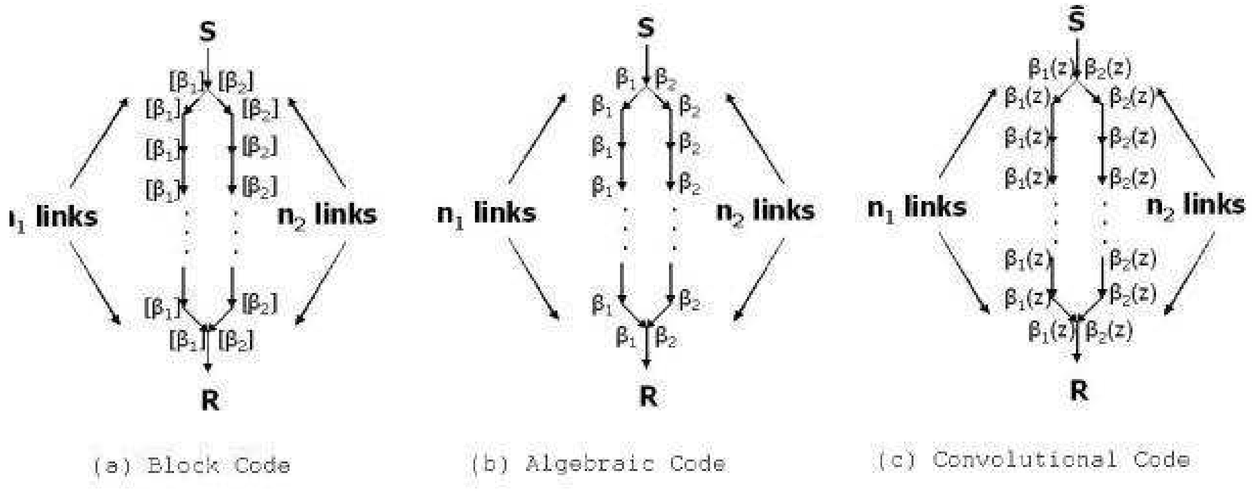

Figure 6.2: This figure shows a single-sender (S) single-receiver (R) network $\mathcal{G}_n$, such that both branches of the network have n edges. Sub-figures (a), (b), and (c), respectively, show particular block, algebraic, and convolutional network codes for $\mathcal{G}_n$.

Reductions between convolutional and algebraic network codes

General networks

For general networks, there does not exist any input-output equivalent local reduction in either direction between algebraic network codes and convolutional network codes. For general acyclic networks, we do not know of any other reductions in either direction.

In the following lemma, we distinguish between FIR and IIR convolutional codes.

Lemma 13 *1. For any algebraic network code $\mathcal{C}_{A,2,2}(\mathcal{G})$ that contains a single-input single-output node with $\beta_{i',i} = (z) mod (z^2 + z + 1)$, there does not exist a local input-output equivalent convolutional network code $\mathcal{C}_{C,p,m}(\mathcal{G})$ for any p, m.*

2. For any convolutional network code $\mathcal{C}_{C,2,1}(\mathcal{G})$ that contains a single-input single-output node with $\beta_{i',i}(z) = 1/(z+1)$, there does not exist a local input-output equivalent algebraic network code $\mathcal{C}_{A,p,m}(\mathcal{G})$ for any p, m.

3. For any convolutional network code $\mathcal{C}_{C,2,1}(\mathcal{G})$ that contains a single-input single-output node with $\beta_{i',i}(z) = z$, there does not exist a local input-output equivalent algebraic network code $\mathcal{C}_{A,p,m}(\mathcal{G})$ for any p, m.

Proof:

1. For the algebraic code, consider the input $X(n) = 1$ for all n. The output due to the incoming message equals $X(n)$ for odd n and $X(n) + X(n-1)$ for even n. No convolutional filter can mimic this behavior.

2. For the convolutional code, consider the input $X(n) = 1$ for $n = 0$ and 0 otherwise. The corresponding output has infinite support. This behavior cannot be mimicked by algebraic codes.

3. Consider the sequence of inputs $X_j(n) = \delta(j)$ for all n, where $\delta(j)$ is the Kronecker-δ function. On input $\delta(j)$, the output is $\delta(j+1)$. Let us assume that $\beta_{i',i}$, an element of some $\mathbb{F}_{p^m}$ is input-output equivalent to z. Because the blocklength of this $\beta_{i',i}$ equals m, the output corresponding to input $\delta(j)$ cannot equal $\delta(j+1)$.

$\square$

Multicast networks

Algebraic network codes that achieve the multicast capacity are not possible for some networks with cycles. Convolutional network codes asymptotically achieve capacity [1], [37] in these networks.

We show a local input-output nonequivalent reduction from algebraic convolutional network codes to convolutional network codes for multicast coding problems over acyclic graphs. This reduction, combined with the existence of algebraic codes, simplifies the arguments presented in [16], [19]. Further, by [37],[33], algebraic network codes are optimal for acyclic multicast problems. Thus there exists a global reduction between algebraic and convolutional network codes for multicast problems on acyclic networks.

For all n let $\mathcal{C}_{C,2,2}(\mathcal{G}_n)$ (Figure 6(c)) be such that $\beta_1(z) = 1$ and $\beta_2(z) = z$. Lemma 14 shows that local reductions from algebraic network codes to convolutional network codes are not possible for multicast networks.

Lemma 14 *For any integers p, m, there exists a multicast problem $\mathcal{P}(\mathcal{G})$ such that there is no algebraic network code $\mathcal{C}_{A,p,m}(\mathcal{G})$ that is local input-output non-equivalent to $\mathcal{C}_{C,2,2}(\mathcal{G})$.*

Proof: We note $\mathcal{C}_{C,2,2}(\mathcal{G}_n)$ achieves the capacity of one bit per coding cycle. The remainder of the proof is identical to that of Lemma 12. □

Lemma 15 *For any algebraic network code $\mathcal{C}_{A,p,m}$ that solves a multicast network coding problem $\mathcal{R}_{\mathcal{G}}$ there exists an input-output nonequivalent local reduction to a convolutional network code $\mathcal{C}_{C,p,m}$.*

Proof: Given any $\beta_{i',i} = (B(z))mod(P_m(z))$ in $\mathcal{C}_{A,p,m}$ such that $B(z)$ is of degree less than m, we define the corresponding $\beta_{i',i}(z)$ in $\mathcal{C}_{C,p,m}$ as $B(z)$. We denote this mapping by $\mathcal{M} : \mathbb{F}_{2^m} \to \mathbb{F}_2(z)$.

Let $E_{c'} = \{e_{c'_i}\}$ and $E_c = \{e_{c_i}\}$ be any two subsets of E such that each forms a cut of $\mathcal{G}$, and $E_{c'}$ is the set of all edges that are predecessors to E_c. Let $\vec{v}(E_{c'}, j)$ be the $|E_{c'}|$-length vector over $\mathbb{F}_{p^m}$ such that the ith entry of $\vec{v}(E_{c'}, j)$ equals $\alpha(e_{c'_i}(j))$. Let $\vec{v}(E_c, j)$ be the $|E_c|$-length vector over $\mathbb{F}_{p^m}$ such that the ith entry of $\vec{v}(E_c, j)$ equals $\alpha(e_{c'_i}(j))$. Similarly, let $\vec{v}(z)(E_{c'})$ be the $|E_{c'}|$-length vector over $\mathbb{F}_2(z)$ such that ith entry of $\vec{v(z)}(E_{c'})$ equals $\alpha(e_{c'_i}(z))$, and let $\vec{v}(E_c)$ be the $|E_c|$-length vector over $\mathbb{F}_2(z)$ such that ith entry of $\vec{v(z)}(E_c)$ equals $\alpha(e_{c'_i}(z))$. Then for all $E_{c'}$, E_c, and j, any network code $\mathcal{C}_{A,p,m}$ is defined by a linear map $L_{E_{c'} \to E_c} : (\mathbb{F}_{p^m})^{|E_{c'}|} \longrightarrow (\mathbb{F}_{p^m})^{|E_c|}$ from $\vec{v}(E_{c'}, j)$ to $\vec{v}(E_c, j)$. Given any such $L_{E_{c'} \to E_c}$, we define $L(z)_{E_{c'} \to E_c} : (\mathbb{F}_2(z))^{|E_{c'}|} \longrightarrow (\mathbb{F}_2(z))^{|E_c|}$ as $L(z)_{E_{c'} \to E_c}(\beta(z)) = \mathcal{M}(L_{E_{c'} \to E_c}(\mathcal{M}^{-1}(\beta(z))))$. Then $L(z)_{E_{c'} \to E_c}$ describes the linear transformation between $\vec{v}(z)(E_{c'})$ and $\vec{v}(z)(E_c)$ implemented by the convolutional network code $\mathcal{C}_{C,p,m}$. The rank of $L(z)_{E_{c'} \to E_c}$ is at least that of $L_{E_{c'} \to E_c}$ (since the $(.)mod(P_m(z))$ operation is linear), and therefore if $\mathcal{C}_{A,p,m}$ solved a particular network multicast problem, then so does $\mathcal{C}_{C,p,m}$. $\square$.

Reductions between convolutional and block network codes

General networks

For general networks, there does not exist a global reduction from block network codes to convolutional network codes [52],[44]. For some networks with cycles, block network codes that achieve capacity are not possible due to feedback, but convolutional network codes achieve capacity [1], [37]. If we weaken our requirements of reductions to allow asymptotically negligible rate-loss, we can demonstrate the existence of an input-output nonequivalent local reduction from convolutional network codes to block network codes.

Let $\mathcal{R}_{\mathcal{G}}$ be the network coding problem referred to in [44].

Lemma 16 ([52],[44]) *1. There do not exist convolutional network codes $\mathcal{C}_{C,p',m'}$ that solve $\mathcal{R}_{\mathcal{G}}$.*

2. There exists a block network code $\mathcal{C}_{B,p,m}$ that solves $\mathcal{R}_{\mathcal{G}}$.

Given a network coding problem $R_{\mathcal{G}} = \{r_{st}\}$, we define the corresponding *ϵ-loss network coding problem $R^{\epsilon}_{\mathcal{G}} = \{r^{\epsilon}_{st}\}$* so that $r^{\epsilon}_{st} = (1-\epsilon)r_{st}$. An ϵ-loss network coding problem $R^{\epsilon}_{\mathcal{G}}$ is said to be *solvable* if there exists a code $\mathcal{C}$ such that for all $s \in \mathcal{S}$ and $t \in \mathcal{T}$ such that $r^{\epsilon}_{st} > 0$, sink t decodes $X_i(s)$, $i \in \{1, \ldots, \lceil m(1-\epsilon)\rceil\}$ without error.

We also define the *decoding delay of a convolutional network code d* as the number of coding instants from the start of encoding operations before all sinks can start decoding their messages.

Lemma 17 *Given $\epsilon > 0$, network $\mathcal{G}$ with decoding delay d, and a convolutional network code that solves $R_{\mathcal{G}}$, there exists a block network code that solves the $R^{\epsilon}_{\mathcal{G}}$ network coding problem with blocklength $m = d/\epsilon$.*

Note: The factor d/m decreases with m, and is asymptotically equal to 0.

Proof: Each source s transmits the sequence $\{X_0(s), X_1(s), \ldots, X_{m-d}(s), 0, \ldots, 0\}$ in the first m coding instants. Each internal node follows the time domain transfer function of the convolutional code for the first m instants. Each sink is able to decode the first $m-d$ symbols of the desired source symbols, by the definition of decoding delay. This is equivalent to m-length block network codes. The transfer matrices at each node corresponds to the truncated time-domain transfer functions of the convolutional network codes. $\square$.

Multicast networks

For multicast network problems, input-output equivalent local reductions cannot exist in either direction, as the following arguments show.

Lemma 18 *1. For any block network code $\mathcal{C}_{A,2,2}(\mathcal{G})$ that contains a single-input single-output node with $[\beta_{i',i}] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, there does not exist a local input-output equivalent convolutional network code $\mathcal{C}_{C,p,m}(\mathcal{G})$ for any p, m.*

2. For any convolutional network code $\mathcal{C}_{C,2,1}(\mathcal{G})$ that contains a single-input single-output node with $\beta_{i',i}(z) = 1/(z+1)$, there does not exist a local input-output equivalent block network code $\mathcal{C}_{B,p,m}(\mathcal{G})$ for any p, m.

3. For any convolutional network code $\mathcal{C}_{C,2,1}(\mathcal{G})$ that contains a single-input single-output node with $\beta_{i',i}(z) = z$, there does not exist a local input-output equivalent algebraic network code $\mathcal{C}_{B,p,m}(\mathcal{G})$ for any p, m.

Proof:

1. For the block code, consider the input $M(e_{i'}(n)) = 1$ for all n. The output on edge i due to the incoming message equals $M(e_{i'}(n))$ for odd n, and 0 for even n. No convolutional filter can mimic this behavior.

   The proofs of 2 and 3 are identical to that in Lemma 13

□

General Formulation for Linear Network Codes

In this section we give a general formulation for linear network codes under reasonable additional restrictions. Let us consider linear systems that are causal, have finite memory, and are L-shift invariant, i.e., operations at every node are periodic with period L.

This restricts the set of permissible encoding operations at intermediate nodes to be of the form

$$y(iL+j) = \sum_{k=1}^{m} c_{jk} y(iL+j-k) + \sum_{k=0}^{m} d_{jk} x(iL+j-k),\ j \in \{0, \ldots, L-1\}.$$

The above formulation of network codes leads naturally to a state-space formulation for describing encoding operations at a node, and therefore for the entire network. These sets of operations can be implemented by filter-banks.

Chapter 7 Complexity

Discussions of computational complexity in previous chapters focus primarily on the time-complexity of design. For networks that are not changing rapidly, implementation complexity may be more important. In this chapter we investigate delay and numbers of basic arithmetic operations required as measures of implementation complexity.

Other measures of complexity examined by other authors include bounds on the number of nodes that need to perform non-trivial network coding operations. The work in [42] finds upper and lower bounds on this number and shows that finding the minimal number of nodes that need to perform coding is an intractable problem. An alternative approach assigns costs to various network resources (such as links or nodes) and designs minimal cost network codes according to these metrics [47].

7.1 Coding Delay/blocklength

We give a unified definition of coding delay based on the *effective blocklength* N of the code. For an $\mathbb{F}_{2^m}$-algebraic network code, an $(\mathbb{F}_2)^m$-block network code, and a degree-m $\mathbb{F}_2(z)$-convolutional network code (whether FIR or IIR), we define N as m. In this section we do not consider *hybrid* network codes, which are mixtures of the three previously defined types of networks codes. However, the results we state can in most cases be generalized to such codes. We also restrict our attention to multicast networks; for the general network coding problems even the rate region is not known in general.

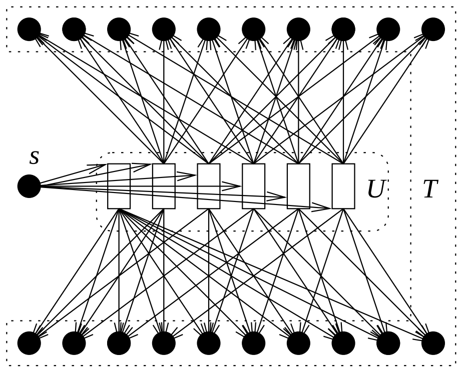

Figure 7.1: An example of a 3-layer graph

7.1.1 Algebraic Network Codes

The upper bound on the effective blocklength of network codes described in Corollary 19 follows from our deterministic code design algorithm from Chapter 5.

Corollary 19 *For any directed acyclic graph $\mathcal{G}$ and multicast network coding problem $\mathcal{P}(\mathcal{G})$, there exists a block network code $\mathcal{C}$ with effective blocklength $N = \lceil \log(|\mathcal{T}|) \rceil$.*

The smallest effective blocklength N required for general multicast network coding problems is a quantity of interest. We exhibit a family of networks (presented in [30]) that require N of at least $\lceil (\log(|\mathcal{T}|))/2 \rceil + \mathcal{O}(1)$. The result is upper and lower bounds that match up to a multiplicative constant. Similar results were independently obtained in [44] and [17]. In addition, in [44] the same lower bound is extended to general block codes (including non-linear codes) and it is also shown that computing the smallest alphabet size required to solve a particular network coding is NP-hard.

We first define the following family of three-layer graphs: $\mathcal{G}_{a,C} = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V} = \{s\} \cup U \cup \mathcal{T}$ where $U = \{1, \ldots, a\}$, $\mathcal{T} = \{t_W \mid W \subseteq U, |W| = C\}$, and edges $\mathcal{E} = \{(s, u) \mid u \in U\} \cup \{(u, t_W) \mid t_W \in \mathcal{T}, u \in W\}$. The source s constitutes the first layer, the a nodes in U constitute the second layer, and the $\binom{a}{C}$ nodes described by $\mathcal{T}$ constitute the third. Each node in $\mathcal{T}$ is connected by unit capacity links to a distinct C-element subset W of U. Figs. 7.1 shows $\mathcal{G}_{6,3}$ as an example.

In passing, we note that this family of graphs is extremely useful in proving performance and complexity bounds of various kinds for network coding problems. This family of graphs is used in [33] to prove that the ratio between the achievable multicast rates with and without network coding can be arbitrarily large and in [6] to show that if one only cares about average throughput rather than multicasting, then the ratio is bounded. This is also the family of graphs used in [44] to prove lower bounds on the effective blocklength of any block codes. We prove this lower bound for block linear codes in Theorem 20 below. We show in the next section (Theorem 22) that for sliding window codes, this bound does not hold, but a weaker version does.

Theorem 20 *Any algebraic network code that solves $\mathcal{G}_{a,2}$ and achieves the min-cut capacity requires an effective blocklength $N = \lceil \log(a-1) \rceil = \lceil (\log(|\mathcal{T}|))/2 \rceil + \mathcal{O}(1)$.*

Proof: The symbol that any node $u \in U$ receives is of the form $\beta_1 X(1) + \beta_2 X(2)$, which corresponds to a global coding vector of the form (β_1, β_2). If any two nodes u, u' in U receive symbols with global coding vectors that are linearly dependent, then the sink t that is connected to u and u' will be unable to decode successfully. Therefore all of the global coding vectors corresponding to different nodes u must be pairwise linearly independent. For any $\mathbb{F}_q$, there are at most $q+1$ vectors of length 2 such that any two vectors are pairwise linearly independent. More precisely, any vector in $(\mathbb{F}_q)^2$ is a multiple of one of the following $q+1$ vectors $\{(1,0), (1,1), (1,2), \ldots, (1, q-1), (0,1)\}$; these vectors are pairwise linearly independent. $\square$

A very similar argument can be used to prove that a lower bound on the effective blocklength N for block linear codes is also $\lceil (\log(|\mathcal{T}|))/2 \rceil + \mathcal{O}(1)$. This is not the case for convolutional network codes.

7.1.2 Convolutional Network Codes

Convolutional network codes can require as little as half the effective blocklength of block or algebraic network codes, or even non-linear block codes, as is shown by the following theorem.

We first state a lemma by Morrison [55].

Lemma 21 *([55, Section 3]) The number of pairs of coprime polynomials of degree at most m over a finite field of size q equals $q^{2m+2} - q^{2m+1} + q - 1$.*

The above lemma allows us to construct convolutional codes for $\mathcal{G}_{a,2}$ with half the effective blocklength required by block codes.

Theorem 22 *([31, Corollary 11]) There exists a convolutional network code that solves $\mathcal{G}_{a,2}$ and achieves the min-cut capacity and requires an effective blocklength $N = \lceil (\log(|\mathcal{T}|))/4 \rceil + \mathcal{O}(1)$.*

Proof: The number of length-2 vectors of degree-m polynomials over $\mathbb{F}_2$, such that any two are linearly independent, is at least $2^{2m+1} + 1$. This is because any two vectors $(\beta_1(z), \beta_2(z))$ and $(\beta_1'(z), \beta_2'(z))$ satisfying $\beta_i(z) \neq 0$ and $\beta_i'(z) \neq 0$ for all i in $\{1, 2\}$ are linearly independent if and only if $\beta_1(z)/\beta_2(z) \neq \beta_1'(z)/\beta_2'(z)$. The number of distinct values that $\beta_1(z)/\beta_2(z)$ can take is exactly twice the number of pairs of coprime polynomials. Calculating the value in Lemma 21 for $q = 2$ and adding two (for the vectors $(1, 0)$ and $(0, 1)$ gives the required result. $\square$

The above theorem implicitly assumes that FIR convolutional codes are used. We conjecture that if IIR convolutional codes are used, then a reduction in N by another factor of 2 is possible.

7.2 Per-bit Computational Complexity

In this section we bound the number of encoder arithmetic operations required per bit transmitted from s to any t. In the case of the algebraic linear network codes already designed, each $Y(e)$ is generated by taking linear combinations of up to $|\mathcal{T}|$ elements over a field $\mathbb{F}_q$ with at least $|\mathcal{T}|$ elements. Thus the message on any incoming edge to some node v undergoes $\log(|\mathcal{T}|)$ bit operations to produce a bit on an outgoing edge. To compare, for replicate-and-forward strategies, only a single bit operation is required per bit on outgoing edges. For networks with large values of C or $|\mathcal{T}|$, we may wish to reduce the encoding computational complexity.

For appropriate choice of code parameters (i.e., $n = \mathcal{O}(\log(|\mathcal{E}||\mathcal{T}|))$ and $p = (\log(n))/n$), the randomized code design algorithm presented in Theorem 4 results in codes that at any node require $\mathcal{O}(\log\log(|\mathcal{E}||\mathcal{T}|))$ bit operations per bit transmitted, reducing the bit coding complexity exponentially at the cost of a negligibly small loss of achievable rate. We here show that for a different random choice of block network codes that we call *permute-and-add codes*, where each $[\beta^{e',e}]$ is chosen to be a $n \times n$ permutation matrix, even lower complexity is possible. This work also appears in [29].

Permutation matrices have the property that each row in any $[\beta^{e',e}]$ has exactly 1 non-zero element. We design the permute-and-add network code $\mathcal{C}_\pi$ as follows. Let $\sigma_{U,n,R}$ be the uniform distribution on the set of all $n \times nR$ binary matrices. For all $e \in \mathcal{E}$ such that $v_t(e) = s$, we choose each source encoder f^e i.i.d. from $\sigma_{U,n,R}$. (Hence our choice of source encoders actually results in dense matrices rather than sparse ones. We conjecture that appropriately choosing sparse matrices would also result in network codes with the desired properties, but our current proof techniques do not suffice to prove this). Let $\sigma_{\pi,n}$ be the uniform distribution on the set of all $n \times n$ permutation matrices. We choose each internal encoder $f^{e',e}$ i.i.d. from $\sigma_{\pi,n}$.

The transfer function from s to each t is then computed inductively (as in [25]) as the linear map represented by the matrix $[\beta^t] = [[\beta^e]^T_{e \in \Gamma_I(t)}]^T$ whose row vectors are all the row vectors of all the global coding matrices for all edges e incoming to sink t. Let r^t be the rank of $[\beta^t]$. Suppose $r^t = nR$. In this case the matrix $[\beta^t]$ has full column rank, and $[\beta^t]^{-1}$, any pseudo-inverse of $[\beta^t]$, can be used as the decoder at sink t.

The following lemma about random permutation matrices is useful in proving our result.

Lemma 23 *Let $[L]$ be an arbitrary $n \times n$ matrix and $[\beta]$ be chosen uniformly at random from the set of all $n \times n$ permutation matrices. For any $\epsilon > 0$, the probability that the rank of $[\beta] + [L]$ is less than $n(1-\epsilon)$ is at most $2^{-n\epsilon + \log(n)}$.*

Proof: For a fixed length-n vector V, the probability that V is in the null-space of $[L] + [\beta]$ is

$$\begin{aligned}
&\mathbf{Pr}[[\beta]][([L] + [\beta])V = 0] \\
&= \mathbf{Pr}[[\beta]][V' = [\beta]V] \\
&\leq \frac{1}{\binom{n}{w_H(V)}},
\end{aligned}$$

where V' represents $[L]V$, and $w_H(V)$ denotes the Hamming weight of the vector V. Here $[\beta]$ acts as a random permutation on the locations of the non-zero elements of V. Thus if V and V' have different weights, then $\mathbf{Pr}[[\beta]][V' = [\beta]V] = 0$. If V and V' have the same weight, then $V' \neq [\beta]V$ unless the random permutation maps the set of non-zero locations of V to the non-zero locations of V'. Now let V be a random binary vector uniformly distributed on the set of all length-n binary vectors. Then the probability over V and $[\beta]$ that V is in the null-space of $[L]+[\beta]$ is bounded as

$$\begin{aligned}
&\mathbf{Pr}[[\beta],V]][([L]+[\beta])V=0]\\
&\leq \frac{1}{2^n}\sum_{V}\frac{1}{\binom{n}{w_H(V)}}\\
&= \frac{1}{2^n}\sum_{w_H(V)=0}^{n}\binom{n}{w_H(V)}\frac{1}{\binom{n}{w_H(V)}}\\
&= \frac{n+1}{2^n},
\end{aligned}$$

As both the permutation matrix $[\beta]$ and the vector V are drawn uniformly from their corresponding domains (of size $n!$ and 2^n respectively), the number of $(V,[\beta])$ pairs that give $V'=[\beta]V$ is bounded from above by

$$n!2^n\frac{n+1}{2^n}=(n+1)!$$

If we want to bound from above the probability, over random permutations, that the number of vectors V in the null-space of the transformation is greater than or equal to $2^{n\epsilon}$ we can assume that in the worst case each of $\frac{(n+1)!}{2^{n\epsilon}}$ permutations results in a null-space of $[L]$ of size $2^{n\epsilon}$. Thus the probability that the transformation has a null-space of dimension at least $n\epsilon$ is at most $\frac{n+1}{2^{n\epsilon}}=2^{-n\epsilon}$, where $\epsilon'=\epsilon-\frac{\log(n+1)}{n}$. $\square$

Let ϵ_n, a parameter in the design of $\mathcal{C}_\pi$, be any function of n such that $\lim_{n\to\infty}\epsilon_n=0$. Define $\epsilon'_n=\epsilon_n-\frac{\log(n+1)}{n}$ and require $\lim_{n\to\infty}n\epsilon'_n=\infty$. We now state and prove our main result on permute-and-add network codes.

Theorem 24 *For any $\epsilon_n>0$, with probability greater than $1-2^{-n\epsilon'_n+\log(|\mathcal{E}|(||\mathcal{T}||+1))}$, network code $\mathcal{C}_\pi$ achieves rate $R=C-(|\mathcal{E}|+1)\epsilon_n$.*

Proof: We first present a high-level outline of the proof. We need to show that with high probability the transfer function is invertible at each sink. To prove this, using the directed acyclic nature of $\mathcal{G}$ we first define a partial order on specific cutsets of $\mathcal{G}$.

We then show that with high probability the rank of the linear transformation between any two successive cut-sets is nearly full, i.e., almost all the information carried by edges in one cut-set is retrievable from edges in the successive cut-set. We use the union bound to bound by a function exponentially decaying in n the probability that even a single linear map between successive cut-sets results in a rank-loss which is not asymptotically negligible. We then note the composition of linear maps of almost full-rank results in a linear map of almost full rank. Therefore the transfer-function to each receiver is, with high probability, almost full-rank. Lastly, we show that a random encoding function will not intersect the null-space of the transfer function to any receiver. We next formalize these concepts. First, we initialize a step-counter a to 1. Counter a keeps track of the stage of our inductive proof-checking algorithm. For each step a and sink t, let $D^{t,a}$ be an ordered set of C edges defining a frontier edge set for t, and $[B^{t,a}]$ be an $nR \times nR$ frontier edge-set matrix composed of the global coding matrices $[\beta^a]$ for the edges in $D^{t,a}$.

Our proof-checking procedure calculates, for each $t \in \mathcal{T}$, and each count a, the probability of the event $E^{t,a}$ that the linear transformation between $[B^{t,1}]$ and $[B^{t,a}]$ is of rank at least $n(R - a\epsilon_n)$

A lower bound for $\mathbf{Pr}[][E^{t,a}]$ can be obtained as follows. By the inductive hypothesis, with probability $1 - 2^{-n\epsilon'_n + \log(|a|||\mathcal{T}||)}$ the linear transformation between $[B^{t,1}]$ and $[B^{t,a}]$ is of rank at least $n(R - a\epsilon_n)$. Let $[\hat{B}^{t,a}]$ be any matrix consisting of a subset of rows of $[B^{t,a}]$ which forms a basis for $[B^{t,a}]$. Matrix $[\hat{B}^{t,a}]$ is partitioned into $[\hat{B}^{t,a}_{e(a)}]$, which corresponds to the $n' \leq n$ rows of $[\hat{B}^{t,a}]$ that carry coding vectors of edge $e(a)$ and $[\hat{B}^{t,a}_{D\setminus e(a)}]$, which corresponds to the rows that carry coding vectors for the rest of $D^{t,a}$. Let $[\bar{B}^{t,a}]$ be any basis for the null-space of the linear transformation between $[B^{t,1}]$ and $[B^{t,a}]$, which by the rank-nullity theorem and the inductive hypothesis is of rank at most $a\epsilon_n$ with high probability. For the inductive step we consider the linear transform (in the basis composed of rows of $[B^{t,a}]$ and $[\bar{B}^{t,a}]$) between the matrix

$[[B^{t,a}]^T[\bar{B}^{t,a}]^T]^T$ and $[B^{t,a+1}]$. We denote this linear transform by the matrix $[L^{t,a}]$. (The reason we must include $[\bar{B}^{t,a}]$ in the transformation is because the global coding matrix corresponding to $e(a+1)$ may be linearly dependent on information carried on edges $e \notin D^{t,a}$.)

First, note that $[L^{t,a}]$ can be represented by a $nR \times nR$ matrix. Our goal will be to bound from below the rank of $[L^{t,a}]$. While the vectors of $[B^{t,a+1}]$ may be linearly dependent on vectors from $[\bar{B}^{t,a}]$, such dependencies can only increase the rank of $[L^{t,a}]$. We therefore assume that no such linear dependencies exist. In other words we restrict ourselves to $[\hat{L}^{t,a}]$, the square sub-matrix of $[L^{t,a}]$ acting on $[\hat{B}^{t,a}]$. Since $D^{t,a}$ and $D^{t,a+1}$ differ in only one edge ($e(a)$ is replaced with $e(a+1)$), rearranging rows and columns gives an alternative expression for $[\hat{L}^{t,a}]$ given by

$$[\hat{L}^{t,a}] = \begin{bmatrix} I & 0 \\ \left[\hat{L}_1^{t,a}\right] & \left[\hat{L}_2^{t,a}\right] + \left[\hat{\beta}^{e(a),e(a+1)}\right] \end{bmatrix}.$$

The top blocks of $[\hat{L}^{t,a}]$ (the identity and the zero matrices) represent the linear transformation of the vectors of $D^{t,a} \setminus e(a)$ (which, by the definition of the inductive step, are unchanged). The bottom blocks represent the linear dependencies between $[B^{t,a}]$ and the information carried on $e(a+1)$. Here $[\hat{\beta}^{e(a),e(a+1)}]$ refers to the part of $[\beta^{e(a),e(a+1)}]$ corresponding to which basis vectors exist in $[\hat{B}^{t,a}_{e(a)}]$. It is therefore a minor of the random permutation matrix $[\beta^{e(a),e(a+1)}]$ such that it is itself a random permutation matrix. The matrix $[\hat{L}_1^{t,a}]$ corresponds to the linear combinations of global coding matrices from edges other than $e(a)$ that contribute to the global coding matrix on $e(a+1)$.

We need to show that with high probability (at least $1-2^{-n\epsilon'_n}$) the nullity of this matrix $[\hat{L}^{t,a}]$ is small (at most $n\epsilon_n$). But difference in the rank of the transformation between $[B^{t,1}]$ and $[B^{t,a}]$ and that of the transformation between $[B^{t,1}]$ and $[B^{t,a+1}]$ is at most the nullity of $[L^{t,a}]$, which in turn is at most the nullity of $[\hat{L}^{t,a}]$. Therefore, by taking the union bound of the probability of a rank-loss greater than $n\epsilon_n$ over all $|\mathcal{T}|$ receivers and $|\mathcal{E}|$ edges, we are done proving the inductive hypothesis.

We now analyze the rank of $[\hat{L}^{t,a}]$. Since all but the last at most n rows of $[\hat{L}^{t,a}]$ correspond to an identity matrix appended with an all-zero matrix, Gaussian elimination results in a matrix of the form

$$\begin{bmatrix} I & 0 \\ 0 & [\hat{L}_2^{t,a}] + [\hat{\beta}^{e(a),e(a+1)}] \end{bmatrix}.$$

Therefore the rank of $[\hat{L}^{t,a}]$ depends crucially on the rank of $[\hat{L}_2^{t,a}] + [\hat{\beta}^{e(a),e(a+1)}]$. We denote the dimension of $[\hat{L}_2^{t,a}] + [\hat{\beta}^{e(a),e(a+1)}]$ by $\hat{n}$. By Lemma 23, the rank of $[\hat{L}_2^{t,a}] + [\hat{\beta}^{e(a),e(a+1)}]$ is $\hat{n}$ with probability at least $1 - 2^{-n\epsilon'_n}$, where $\epsilon'_n = \epsilon_n - \frac{\log(n+1)}{n}$.

The above shows that the overall linear transform to any receiver is, with high probability, close to full rank. To complete the proof we need only show that with high probability the span of the vectors generated by the source encoder does not intersect with the null-space of $[L^{t,|F^{\mathcal{T}}|}]$ for any $t \in \mathcal{T}$. The probability of such an intersection is equals the probability that a vector space of dimension $n(C-(|\mathcal{E}|+1)\epsilon_n)$ (the space spanned by the source encoder's vectors) chosen uniformly and at random from a space of dimension n intersects a fixed vector space of dimension $n|\mathcal{E}|\epsilon_n$ (the null-space of $L^{t,|F^{\mathcal{T}}|}$). It can be computed that the probability that this event does not occur is given by $\Pi_{i=n\epsilon_n}^{n(C-|\mathcal{E}|\epsilon_n)}(1-2^{-i})$, which can be bounded from below by $1 - n(C-(|\mathcal{E}|+1)\epsilon_n)2^{-n\epsilon_n}$. Using the union bound over all $t \in \mathcal{T}$ gives the required error probability. □

Note: The above provides a proof that random distributed design of permute-and-add network codes works with high probability. If a small amount of feedback is permitted from each t to s, then each t can tell s the matrix $[\beta^t]$, and the expected number of code-design attempts required to guarantee that the permute-and-add code works is at most 2.

Chapter 8 Networks with Adversaries

8.1 Introduction

Consider the following point-to-point adversarial channel coding problem. The network $\mathcal{G}$ consists of $|\mathcal{E}|$ parallel, directed, binary-input, binary-output edges $\mathcal{E} = \{e(1), e(2), \ldots, e(|\mathcal{E}|)\}$ between the source s with encoder Xavier and the sink t with decoder Yvonne. Encoder Xavier wishes to describe the source's information across the network. Xavier also has access to a fair coin, which he can use to generate as many bits as he wants. Xavier and Yvonne share no private key or common randomness. Xavier wishes to transmit all of the information generated by s to Yvonne, who wishes to decode the received message with asymptotically negligible error probability. Xavier and Yvonne agree on low-complexity encoding and decoding schemes in advance. The encoding and decoding schemes are also known to the computationally unbounded adversary Zorba. The adversary Zorba knows the message generated by s but not the outcomes of Xavier's coin flips. Zorba can also see and control the transmissions on $\mathcal{Z} \subseteq \mathcal{E}$, where $\mathcal{Z}$ has size M; Zorba cannot observe or change transmissions on $\mathcal{E} \setminus \mathcal{Z}$. Zorba wishes to minimize the rate R at which Yvonne can reconstruct the information from s with asymptotically negligible error probability. We first consider the case where Zorba's interference patterns on the links he controls can be based only on the knowledge he already possesses (code design, source message, and causal knowledge of symbols transmitted on links he controls). We then show, using more complex arguments, that the rate region is identical even if Zorba has non-causal knowledge of the information transmitted on links he controls.

Previous work [28] exhibits a low-complexity algorithm for each sink to detect an adversarial attack with high probability as long as there is at least one packet in the network whose contents the adversary cannot infer.

We obtain an intriguing two-part rate region for the corresponding error-correction problem. We construct low-complexity block codes, that asymptotically achieve the capacity $C_{Adv}(M, |\mathcal{E}|) = (|\mathcal{E}| - M)1(M/|\mathcal{E}| < 0.5)$ of this channel model. (The indicator function $1(\cdot)$ is one when its argument is true and zero otherwise.) Viewing the ratio $M/|\mathcal{E}|$ as the noise parameter of this adversarial channel, the capacity of the channel for the regime $M/|\mathcal{E}| < 0.5$ equals $|\mathcal{E}|(1 - M/|\mathcal{E}|)$. That is, it equals the capacity of $|\mathcal{E}|$ parallel binary erasure channels (BECs) with erasure probability $M/|\mathcal{E}|$. This result is striking since the location of all erasures is explicitly known to the decoder of an erasure channel whereas $\mathcal{Z}$ is unknown to Yvonne. Indeed, our code construction relies on BEC channel codes. The construction also employs parity information, which enables Yvonne to estimate, with high reliability, the set $\mathcal{J}^z \subseteq \mathcal{Z}$ of links that Zorba corrupts. Yvonne decodes the messages on $\mathcal{E} \setminus \mathcal{J}^z$. Conversely, we show that no matter which code Xavier uses, if he transmits at a rate higher than $C_{Adv}(M, |\mathcal{E}|)$, then there exists a strategy by which Zorba can force Yvonne's probability of decoding error to be bounded away from 0.

Section 8.3 presents our results for the case where the network consists of parallel edges. These set the stage for the more interesting multicast model of Section 8.4. Non-trivial coding must be performed at internal nodes in order to achieve the multicast capacity [1]. This makes error-correction harder than in the parallel link case, since in principle the information injected into the network by an adversary controlling even a single link can contaminate all of the information reaching any sink.

Section 8.5 treats generalizations. These include allowing small amounts of feedback, which increases the rate region to $(|\mathcal{E}| - M)$, and knowledge at the sinks of the adversary's location, which enlarges the rate region to $(|\mathcal{E}| - M)$. In contrast, knowledge by the source of the adversary's location leaves the region unchanged.

We also show a separation between channel and network coding for this problem. That is, if the links in the network in addition to possible adversarial interference also suffer corruption by random noise, then overlaying network coding to combat the adversary's actions on top of link-by-link channel coding achieves the optimal performance. We also provide an algorithm for detecting which edges need to be removed from the network so as to eliminate the contamination from the information being injected by adversaries. We then consider the case where the adversary does not know the message at the source, showing that the maximal rate at which secret information can be embedded in an information-theoretically secure manner into the message being transmitted equals $(1-2p)1(M/|\mathcal{E}| < 0.5)$. Finally, Section 8.6 details algorithms for a scenario where Zorba possesses non-causal information on the links that he controls, treating both the unicast and multicast cases.

8.2 Related Work

In the class of *noisy* channels, where communication is limited by the presence of random noise. Shannon's seminal paper [59] considers the problem of reliable communication over a memoryless noisy channel. Two standard noisy channel models are the Binary Symmetric Channel (transmitted bits are flipped with probability p) with capacity $C_{BSC} = 1 - H(p)$, and the Binary Erasure Channel (transmitted bits are erased with probability p) with capacity $C_{BEC} = 1 - p$. The *block-interference* model presented in [51] considers a type of channel with memory. There are N parallel binary-input binary-output edges between the source and the sink. For each coding interval of length n, a fraction p of these N edges are BSCs whose cross-over probability equals 1/2, and the remaining $(1-p)N$ are noiseless. If the sink has state information about the channels telling him which are noiseless and which are not, or if n is large, the authors show that the capacity of this channel approaches

$NC_{BEC}(p)$. If state information is not known to the sink and n is small (≈ 1), the capacity is close to $NC_{BSC}(p)$.

In the class of *adversarial channels* communication is limited by the presence of a malicious adversary. For instance, [22] presents results on a single binary-input, binary-output channel on which an adversary can observe the full blocklength-n channel input and change at most a fraction p of these bits. The sink is required to reconstruct the input with asymptotically negligible error probability. While the capacity of such channels is not known, the best known non-trivial upper bound is everywhere less than $C_{BSC}(p)$ [50]. When the source and sink share a length-$O(\log(n))$ private key for use in a blocklength-n transmission, the channel capacity is $1 - H(p)$ [41]. Similar results follow when the sender and receiver share randomness instead of a private key [10]. The work of [12] and [54] gives explicit constructions of codes with rates approaching $C_{BSC}(p)$ when the computational capabilities of the adversaries are limited. An excellent survey of results for channels with uncertainty can be found in [43]. Results on Verifiable Secret Sharing with an honest dealer ([9],[56]) can be used to prove some results on the secret capacities of the adversarial channel model we consider.

8.3 Unicast Model

We start with results for the parallel-edge unicast model.

The codes we use are not linear; however, they have design, encoding, and decoding complexity that is polynomial in all the network problem parameters.

We block both source bits and Xavier's random coin flips into m-dimensional vectors that we treat as elements of the finite field $\mathbb{F}_q$, where $q = 2^m$. The length-nR source input vector is $X = (X(1), X(2), \ldots, X(n))^T$, where each $X(i)$ vector comprises R elements of $\mathbb{F}_q$. Thus $X(1), X(2), \ldots, X(n)$ represents the R source bits from the first mn units of time. The m-vector of random coin outcomes is $\rho \in \mathbb{F}_q$.

A *code against adversarial attack* $\mathcal{C}$ is defined by its encoder $\{f^e\}_{e\in\mathcal{E}}$ and decoder h. For each $e \in \mathcal{E}$, $f^e : (\mathbb{F}_q)^{nR} \times \mathbb{F}_q \to (\mathbb{F}_q)^n$ maps a source vector X and random symbol ρ to the length-n vector $Y^e = (Y^e(i))_{i=1}^n = f^e(X, \rho)$ transmitted across edge e. We use $Y = f(X, \rho) = (f^e(X, \rho))_{e\in\mathcal{E}}$ to denote the full channel input and $\hat{Y} = (\hat{Y}^e)_{e\in\mathcal{E}}$ to describe the full channel output. In particular, we use the length-$|\mathcal{E}|$ vectors $Y(i)$ and $\hat{Y}(i)$ to denote the channel input and output at time i. A decoder $h : (\mathbb{F}_q)^{n|\mathcal{E}|} \to (\mathbb{F}_q)^{nR}$ maps the collection $\hat{Y}$ of received channel outputs to a reconstruction $\hat{X} = h(\hat{Y})$ of source message X.

Xavier and Yvonne together choose a code $\mathcal{C} = ((f^e)_{e\in\mathcal{E}}, h)$. This code choice is fixed and known to Zorba, who also has full knowledge of the source message X to be transmitted. Zorba uses this information to choose the jamming function g used to corrupt the channel input Y to give channel output $\hat{Y}$. In designing his jamming function, Zorba first chooses a set $\mathcal{Z}$ of edges to control. The size of $\mathcal{Z}$ cannot exceed M, the jamming dimension. For each $e \in \mathcal{E} \setminus \mathcal{Z}$, $\hat{Y}^e = Y^e$. For each $e \in \mathcal{Z}$, Zorba uses jamming functions $g^{e,i} : (\mathbb{F}_q)^{nR} \times (\mathbb{F}_q)^{iM} \to (\mathbb{F}_q)$ to produce $\hat{Y}^e = g^e(X, (Y^e)_{e\in\mathcal{Z}}) = (g^{e,i}(X, (Y^e(j))_{e\in\mathcal{Z}, j\in\{1,2,\ldots,i\}}))_{i\in\{1,2,\ldots,n\}}$; thus the corrupted information on any edge $e \in \mathcal{Z}$ can rely on both the source message X and causally on the channel inputs Y^e on all edges $e \in \mathcal{Z}$. For notational simplicity we henceforth write $\hat{Y} = g(X, Y)$ to denote the full collection of channel outputs.

The error probability is defined as $P_e^{(n)} = \mathbf{Pr}[h(g(X, (Y^e)_{e\in\mathcal{Z}}))] \neq X)$. Rate R is achievable for the channel g for jamming dimension M if for any $\epsilon > 0$ and n sufficiently large there exists a blocklength-n code $\mathcal{C}$ with $P_e^{(n)} < \epsilon$ for every jamming function g in the family of jamming functions described above. The capacity $C_{Adv}(M, |\mathcal{E}|)$ equals the maximal achievable rate over all g.

We now state and prove our main result for unicast channels.

Theorem 25

$$C_{Adv}(M, |\mathcal{E}|) \quad = \quad (|\mathcal{E}| - M)1(M/|\mathcal{E}| < 0.5)$$

Further, for any n and any $m = \omega(\log(n|\mathcal{E}|))$ there exist blocklength-n codes with $R = (1 - (|\mathcal{E}| + 1)/n)C_{Adv}(M, |\mathcal{E}|)$, $P_e^{(n)} < n|\mathcal{E}|2^{-m}$, and complexity of design and encoding and decoding implementation equal to $\mathcal{O}((nm|\mathcal{E}|)^2)$.

Proof: **Upper Bounds:** The bound $R \leq |\mathcal{E}| - M$ is immediate since Zorba can set $\hat{Y}^e = 0^n$ for all $e \in \mathcal{Z}$, thereby giving rate zero on all edges controlled by Zorba.

If $M \geq |\mathcal{E}|/2$, $R = 0$ since Zorba can use the following strategy to make decoding with $P_e^{(n)} < 1/2$ impossible. Zorba selects an arbitrary jamming subset $\mathcal{J}^z$ of size $|\mathcal{E}| - M$ of $\mathcal{Z}$. Then, for arbitrary $X' \neq X$ and ρ', Zorba sets $\hat{Y}^e = f^e(X', \rho')$ for each $e \in \mathcal{J}^z$ and $\hat{Y}^e = 0^n$ for $e \in \mathcal{Z} \setminus \mathcal{J}^z$. Yvonne does not know $\mathcal{Z}$ and is therefore unable to decide which of X and X' to decode to, leading to an error probability of at least 1/2.

Lower Bound: We first sketch the achievability argument and then give a precise code construction. Assume $M/|\mathcal{E}| < 1/2$ and $R = |\mathcal{E}| - M$. In the first $n - |\mathcal{E}| - 1$ symbols on each $e \in \mathcal{E}$, Xavier transmits X using an erasure code. Xavier uses the remaining $|\mathcal{E}| + 1$ symbols to send a marker containing ρ and $D = (D^e)_{e \in \mathcal{E}}$. The vector D is a hash of the vectors $(Y(i))_{i=1}^{n-|\mathcal{E}|-1}$ with ρ. Yvonne decodes by looking for consistency among the received channel outputs. Since Zorba controls fewer than half of the edges, Yvonne can determine (ρ, D) by majority rule. She then recomputes the hash using ρ and the received transmissions. Since Zorba does not know ρ *a priori*, any changes he makes on $(Y^e)_{e \in \mathcal{Z}}$ will, with high probability, be inconsistent with the hash values. This enables Yvonne to determine which edges have been corrupted. She then uses $\hat{Y}^e$ from $e \notin \mathcal{J}^z$ to reconstruct $\hat{X}$, via the erasure code.

We now describe our coding scheme in detail. For any n and $m = \omega(\log(n|\mathcal{E}|))$, fix $R = \lfloor (1 - (|\mathcal{E}| + 1)/n)(|\mathcal{E}| - M) \rfloor$ and design the functions f^e using the following procedure.

Let $\mathcal{L}$ be any $(n-|\mathcal{E}|-1)|\mathcal{E}| \times nR$ Vandermonde matrix over $\mathbb{F}_q$ (such a matrix exists since $q = 2^m$ and $m = \omega(\log(n|\mathcal{E}|))$ [21]). For the ith edge $e(i) \in \mathcal{E}$, the matrix $\mathcal{L}^{e(i)}$, known *a priori* to Xavier, Yvonne, and Zorba, is defined to be the $(n - |\mathcal{E}| - 1) \times nR$ matrix consisting of row $[(n - |\mathcal{E}| - 1)(i - 1) + 1]$ through $[(n - |\mathcal{E}| - 1)i]$ of $\mathcal{L}$. For all $e \in \mathcal{E}$ we define T^e, U, and D as

$$\begin{aligned} T^e &= (\mathcal{L}^e X)^T, \\ U &= (1, \rho, \ldots, \rho^{n-|\mathcal{E}|-1}) \text{ and} \\ D &= U[T^{e(1)}, T^{e(2)}, \ldots, T^{e(|\mathcal{E}|)}] \end{aligned}$$

and set $Y^e = (T^e, D, \rho)$. Thus for each $e \in \mathcal{E}$, the first $n - |\mathcal{E}| - 1$ symbols in Y^e are the erasure-coded message symbols, the next $|\mathcal{E}|$ symbols are the hash function output, and the last symbol is the hash-function's key ρ.

Yvonne's decoding scheme h is as follows. Let $\hat{Y}^e = \left(\hat{T}^e, \hat{D}, \hat{\rho}\right)$ denote the channel output on $e \in \mathcal{E}$. As described above, Yvonne first determines the correct value of the marker (D, ρ) by choosing the value that appears on the majority of the links. She then checks, for the ith edge $e(i) \in \mathcal{E}$, whether or not the ith symbol of D equals the ith symbol in $U(\hat{T}^{e(1)}, \hat{T}^{e(2)}, \ldots, \hat{T}^{e(|\mathcal{E}|)})$. She calls the set of edges for which this is true the *decoding set of edges* $\mathcal{E}^D$.

In the second stage of decoding Yvonne constructs $\mathcal{L}^D$, an $|\mathcal{E}^D|(n - |\mathcal{E}| - 1) \times nR$ matrix created by concatenating the matrices in $\{\mathcal{L}^e\}_{e \in \mathcal{E}^D}$. Since $\mathcal{L}$ is a Vandermonde matrix, so is $\mathcal{L}^D$. Yvonne obtains $\hat{X}$ by inverting the matrix equation $\hat{Y}^D = \mathcal{L}^D X$, where $\hat{Y}^D$ is the dimension $|\mathcal{E}^D|(n - |\mathcal{E}| - 1)$ vector obtained by the ordered concatenation of $\hat{Y}^e$, $e \in \mathcal{E}^D$. There is a decoding error only if $\mathcal{E}^J \cap \mathcal{E}^D \neq \phi$, where $\mathcal{E}^J \subseteq \mathcal{Z}$ is the *jamming set of edges*, i.e., the set of edges for which $\hat{T}^e \neq T^e$. We now bound the probability of this event.

It suffices to prove that the probability that the ith symbol of D does not equal $U\hat{T}^{e(i)}$ for any $e(i) \in \mathcal{E}^J$ is at least $1 - n|\mathcal{E}|2^{-m}$. By definition of $\mathcal{E}^J$, $T^e \neq \hat{T}^e$. Thus $U(T^e)^T = U(\hat{T}^e)^T$, i.e., $U(T^e - \hat{T}^e)^T = 0$ only if ρ is a root of the degree $n - |\mathcal{E}| - 1$ polynomial $U(T^e - \hat{T}^e)^T$. Zorba does not know the value of ρ, and the polynomial contains at most $n - |\mathcal{E}| - 1$ roots in field of size $q = 2^m$. Therefore $e \notin \mathcal{E}^D$ are inconsistent with probability at least $1 - (n - |\mathcal{E}| - 1)/2^m$. Since there are fewer than $|\mathcal{E}|/2$ edges in $\mathcal{E}^J$, the total probability that $\mathcal{E}^J \cap \mathcal{E}^D \neq \phi$ is at most $(n - |\mathcal{E}| - 1))|\mathcal{E}|/2^{m+1} < n|\mathcal{E}|2^{-m}$.

Lastly, it can be verified that the complexity of the encoder f^e at each edge e is determined by the complexity of computing the vectors T^e over a field of size q, and that the complexity of decoder h is determined by the complexity of inverting a Vandermonde matrix of dimension nR over the same finite field [21]. □

Note 1*:* Any Maximum Distance Separable code [49] may be used in place of $\mathcal{L}$. We choose Vandermonde matrices due to their low design and implementation complexity.

8.4 Multicast Model

We now examine the problem of multicasting information on more complex networks with a hidden adversary. The codes we use are linear at the internal nodes, but are not linear at the source or sink nodes. However, the design, encoding, and decoding complexities are polynomial in all the network problem parameters. We assume that $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a directed acyclic network with unit-capacity directed edges. The encoder Xavier at the source node s uses the network $\mathcal{G}$ to transmit the source's information X as defined in Section 8.3 to a set of decoders, $\{\text{Yvonne}_1, \ldots, \text{Yvonne}_{|\mathcal{T}|}\}$, located respectively at the sink nodes $\mathcal{T} = \{t_1, t_2, \ldots, t_{|\mathcal{T}|}\}$. Xavier uses $M|\mathcal{T}|$ random

m-vectors, denoted $\rho = (\rho^{i,k})_{i\in\{1,\ldots,M\},k\in\{1,\ldots,|\mathcal{T}|\}}$.

A *network code against adversarial attack* $\mathcal{C}$ is defined by its source encoder, internal encoders, and decoders at receiver nodes.

The source encoder comprises a collection of functions $\{f^e\}_{e\in\Gamma_O(s)}$. For each $e \in \Gamma_O(s)$, $f^e : (\mathbb{F}_q)^{nR} \times (\mathbb{F}_q)^{M|\mathcal{T}|} \to (\mathbb{F}_q)^n$ maps X and a set of random symbols $\{\rho^{i,k}\}$ to the length-n vector Y^e transmitted across edge e. We denote by $Y^e(i)$ the ith symbol input to edge $e \in \mathcal{E}$, and denote by $\hat{Y}^e(i)$ the ith symbol output on edge e.

The internal encoder for any edge $e \notin \Gamma_O(s)$ is a function $f^e : (\mathbb{F}_q)^{n|\Gamma_I(v_t(e))|} \to (\mathbb{F}_q)^n$ which maps messages $Y^{e'}$ on all links e' incoming to $v_t(e)$ to the vector Y^e transmitted across edge e.

For each $k \in \{1,\ldots,|\mathcal{T}|\}$, decoder $h^k : (\mathbb{F}_q)^{n|\Gamma_I(t_k)|} \to (\mathbb{F}_q)^{nR}$ maps the collection $\hat{Y}^k = (\hat{Y}^e(i))_{e\in\Gamma_I(t_k),i\in\{1,\ldots,n\}}$ of received channel outputs to a reconstruction $\hat{X}^k$ of source X. Let $\hat{Y}^k(i) = (\hat{Y}^e(i))_{e\in\Gamma_I(t_k)}$ denote the set of ith symbols from all channel outputs and $\hat{Y}^e_k = (\hat{Y}^e(i))_{i\in\{1,\ldots,n\}}$ denote the symbols on link e.

Xavier and the Yvonnes together choose a code $\mathcal{C} = ((f^e)_{e\in\mathcal{E}}, (h^k)_{k\in\{1,\ldots|\mathcal{T}|\}})$, and inform each e of f^e. This code choice is fixed and known to Zorba, who also has full knowledge of the source message X to be transmitted. Zorba uses this information to choose the set $\mathcal{Z} \subseteq \mathcal{E}$ of edges to control. The size of $\mathcal{Z}$ cannot exceed the jamming dimension M. We note that adversarial control of a vertex $v \in \mathcal{V}$ is equivalent to adversarial control of all edges in $\Gamma_O(v)$, and therefore we need only treat the case where Zorba controls edges. (For the case that Zorba is constrained in the number of nodes he controls, the *vertex connectivity* of $\mathcal{G}$ is the important parameter, in terms of which results similar to those presented here can be derived.)

For each $e \in \mathcal{E}$, we use $\hat{Y}^e$ to describe the channel output of link e received by node $v_h(e)$. For each $e \notin \mathcal{Z}$ $\hat{Y}^e = Y^e$. For each $e \in \mathcal{Z}$ and $i \in \{1,2,\ldots,n\}$, Zorba uses causal jamming functions $g^{e,i} : (\mathbb{F}_q)^{nR} \times (\mathbb{F}_q)^i \to (\mathbb{F}_q)$ to produce a corrupted output $\hat{Y}^e = g^e(X, (Y^e)_{e\in\mathcal{Z}}) = (g^{e,i}(X, (Y^e(j))_{e\in\mathcal{Z},j\in\{1,2,\ldots,i\}}))_{i\in\{1,2,\ldots,n\}}$. As in the unicast case, we defer the discussion of the corresponding theorem for the case with non-causal jamming functions to Section 8.6. The error probability is defined as $P_e^{(n)} = \mathbf{Pr}[\exists k \text{ such that } h^k(g(X, (Y^e)_{e\in\mathcal{Z}})) \neq X]$. Rate R is achievable for jamming dimension M if for any $\epsilon > 0$ and n sufficiently large there exists a blocklength-n code $\mathcal{C}$ with $P_e^{(n)} < \epsilon$ for every jamming function g in the family of jamming functions described above. The capacity $C_{Adv,Mul}(M, |\mathcal{E}|)$ of the given adversarial channel model equals the maximal achievable rate.

We take the following approach.

For each $v \in \mathcal{V}$ the encoding functions $\{f^e\}_{e\in\Gamma_O(v)}$ perform approximately n rounds of a robust algebraic network code ([38], [33]). The input to this algebraic network code in the ith round is $X(i)$. After these rounds of transmitting the source information, s transmits to each receiver in succession $M(R+1)$ symbols of marker information using C edge-disjoint paths.

We model the effect of the jamming functions as follows. Let $\mathcal{G}^Z$ be the graph obtained by attaching a new unit-rate source node $s^{e,Z}$ to the midpoint of e for each $e \in \mathcal{Z}$. The message $X^e(i)$ generated over the ith time interval by $s^{e,Z}$ may be an arbitrary function of X and $(Y^{e'}(i'))_{e'\in\mathcal{Z}, i'\leq i}$. For each $e \in \mathcal{Z}$, the link output $\hat{Y}^e(i)$ over coding interval i equals $Y^e(i) + X^e(i)$. Denote by $X^Z(i)$ the length-M vector $(X^e(i))_{e\in\mathcal{Z}}$.

Since the set $\mathcal{Z}$ is fixed and $\mathcal{C}$ is linear, for each $k \in \{1, 2, \ldots, |\mathcal{T}|\}$

$$\hat{Y}^k(i) = T^k X(i) + T^{Z,k} X^Z(i) \tag{8.1}$$

for some fixed linear transforms T^k and $T^{Z,k}$. We define the *interference at t_k* as $\delta^k(i) = T^{Z,k} X^Z(i)$. The linear span of $\{X(i)\}_{i\in\{1,\ldots,n-M|\mathcal{T}|(C_m+1)\}}$ is a vector-space (denoted $\mathbf{V_X}$) of dimension at most R. Denote by $T^k\mathbf{V_X}$ the linear span of $\{T^k X(i)\}_{i\in\{1,\ldots,n-M|\mathcal{T}|(C_m+1)\}}$. The linear span of $\{X^Z(i)\}_{i\in\{1,\ldots,n-M|\mathcal{T}|(C_m+1)\}}$ is a vector-space (denoted by $\mathbf{V_Z}$) of dimension at most M. Denote by $T^{Z,k}\mathbf{V_Z}$ the linear span of $\{\delta^k(i)\}_{i\in\{1,\ldots,n-M|\mathcal{T}|(C_m+1)\}}$. By Theorem 4.3, with high probability over code design, $X(i)$ is retrievable from $T^kX(i)$, and $\mathbf{V_X}\cap\mathbf{V_Z}$ equals only the zero vector. This implies that if Yvonne$_k$ knows $T^{Z,k}\mathbf{V_Z}$, then Yvonne$_k$ can recover $X(i)$ for all i.

In contrast to the unicast case, Yvonne$_k$ does not here first infer the set of jamming edges $\mathcal{E}^J$. She cannot do this in general, since, for example, if e' is the only edge that satisfies $v_t(e')=v_h(e)$ and $\mathcal{E}^J=\{e,e'\}$, then Yvonne$_k$ cannot determine whether or not $e\in\mathcal{E}^J$. The best she can do is to cancel out the interference effect. Theorem 29 in Section 8.5 shows a scheme for Yvonne$_k$ to detect a set of edges such that cutting them isolates $\mathcal{E}^J$ from the network without changing the set of achievable rates. The drawback of that scheme is that, as currently implemented, its complexity is exponential in M. To ascertain $T^{Z,k}\mathbf{V_Z}$, we use a scheme similar to the one developed in Section 8.3.

Theorem 26

$$C_{Adv,Mul}(M,C_m) \quad=\quad (C_m-M)1(M/C_m<0.5)$$

Further, for any n and any $m=\omega(\log(n|\mathcal{E}||\mathcal{T}|))$, there exist blocklength-n codes with $R=(1-M|\mathcal{T}|(C_m+1)/n)C_{Adv,Mul}(M,C_m)$, $P_e^{(n)}<n|\mathcal{E}|2^{-m(C_m-R)}+|\mathcal{T}|(n/q)^M$, complexity of design and encoding $\mathcal{O}(nm)$, and of decoding $\mathcal{O}((nmC_m)^3)$.

Proof: **Upper bounds:** The bound $R\le C_m-M$ follows since Zorba can choose $\mathcal{Z}$ to be in a cut-set, and set $\hat{Y}^e=0^n$ for all $e\in\mathcal{Z}$.

If $M>C_m/2$ edges, $R=0$ by the following argument. Zorba chooses $\mathcal{Z}$ to be a subset of some min-cut $\mathcal{E}(s,t,S)$, and an arbitrary set of jamming edges $\mathcal{J}^Z\subseteq\mathcal{Z}$ of size C_m-M. For any $X'\neq X$ and any ρ', Zorba mimics the network code $\mathcal{C}$, and for each $e\in\mathcal{J}^Z$ sets $\hat{Y}^e$ to what the message would have been on $\mathcal{J}^Z$ if s had input

(X', ρ'), and $\hat{Y}^e = 0^n \; \forall e \in \mathcal{Z} \setminus \mathcal{J}^Z$. As in Theorem 25, Yvonne$_k$ is unable to decide which of X and X' to decode to.

Lower bound: We present a coding strategy using ideas from the proof of Theorem 25.

Let $R = (1 - M|\mathcal{T}|(C_m + 1)/n)(C_m - M)$ and $m = \Theta(\log(n|\mathcal{T}||\mathcal{E}|))$. We show the existence of codes that achieve R with $P_e^{(n)} < 2^{-\Omega(m(C_m - R))}$. There are two encoding steps.

First, Xavier uses a robust network code of the form in Theorem 4.3 $(n - M|\mathcal{T}|(C_m + 1))$ times to multicast information to each t_k. The input to $\mathcal{C}$ during the ith use is $X(i)$.

In the second step, C_m edge-disjoint paths $\{P^i(s, t_k)\}_{i \in \{1,\ldots,C_m\}}$ are used to transmit identical copies of the marker information. This marker information consists of M blocks, each of length $R + 1$, for each receiver. Since there are $|\mathcal{T}|$ receivers, this process requires at most $(R + 1)M|\mathcal{T}|$ channel uses over $\mathbb{F}_q$. The marker information sent to t_k is $\left(D^{j,k}, \rho^{j,k}\right)_{j=1}^{M}$. That is, each of the M blocks of length $R + 1$ in the marker to t_k contains the random symbol $\rho^{j,k}$ and the length-R hash-vector $D^{j,k}$. Each hash-vector $D^{j,k}$ is a distinct linear combination of $\{X(i)\}_{i=1}^{n-(R+1)M|\mathcal{T}|}$, defined as

$$D^{j,k} = \sum_{i=1}^{n-(R+1)M|\mathcal{T}|} (\rho^{j,k})^{i-1} X(i).$$

For any receiver Yvonne$_k$, Zorba controls edges in less than half of the edge-disjoint paths $\{P^i(s, t_k)\}_{i=1}^{C_m}$, hence the marker information each sink receives on more than half the paths is identical. At each t_k Yvonne$_k$ retrieves $\left(D^{j,k}, \rho^{j,k}\right)_{j=1}^{M}$ by a majority decision.

Yvonne$_k$ decodes as follows. For all $j \in \{1, \ldots, M\}$ she computes the vectors

$T^kD^{j,k}$ and the vectors $\sum_{i=1}^{n-(R+1)M|\mathcal{T}|}(\rho^{j,k})^{i-1}\hat{Y}^k(i)$. Using (8.1) we have

$$\begin{aligned}
&\sum_{i=1}^{n-(R+1)M|\mathcal{T}|}(\rho^{j,k})^{i-1}\hat{Y}^k(i)\\
&= T^kD^{j,k}+T^{Z,k}\left(\sum_{i=1}^{n-(R+1)M|\mathcal{T}|}(\rho^{j,k})^{i-1}X^Z(i)\right)\\
&= T^kD^{j,k}+\sum_{i=1}^{n-(R+1)M|\mathcal{T}|}(\rho^{j,k})^{i-1}\delta^k(i).
\end{aligned}$$

Hence Yvonne can retrieve M length-R vectors in $T^{Z,k}\mathbf{V_Z}$, namely $\sum_{i=1}^{n-(R+1)M|\mathcal{T}|}(\rho^{j,k})^{i-1}\delta^Z(i)$, denoted respectively by $A^{j,k}$. We now prove that, with high probability, $\{A^{j,k}\}_{j\in\{1,\ldots,M\}}$ forms a basis for $T^{Z,k}\mathbf{V_Z}$ for each $k\in\{1,\ldots,|\mathcal{T}|\}$.

We denote by $[\Delta]$ the matrix that has $\delta^k(i)$s as row vectors. Let $U^k(i)=((\rho^{i,k})^{j-1})_{j=1}^{n-(R+1)M|\mathcal{T}|}$. We denote by $[U]$ the matrix that has $U^k(i)$ as row vectors. Since Zorba controls at most M links, rank($[\Delta]$) is at most M. We choose $[\Delta']$ to be any set of rank($[\Delta]$) linearly independent columns of $[\Delta]$. Suppose that $\{A^{j,k}\}_{j=1}^{C_m}$ does not form a basis for $T^{Z,k}\mathbf{V_Z}$. This means that for some linear combination $c^k=(c_{1,k},c_{2,k},\ldots,c_{\mathrm{rank}([\Delta]),k})$ the length-M column vector $[U][\Delta']c^k$ equals the zero vector, though the column vector $[\Delta']c^k$ is non-zero (since by definition $[\Delta']$ has full column rank). Thus the adversary would have to choose the matrix $[\Delta]$ so that the M polynomials that are the elements of the column vector $[U][\Delta']c^k$ are all zero. By an argument similar to the one for Theorem 25, the probability that this happens is $(n/q)^M$. Taking the union bound over all receivers, the error probability equals $|\mathcal{T}|(n/q)^M$. Taking into account the error probability in the design of robust network codes gives the required result. □

Note 1*:* The codes described in Theorem 26 operate under the assumption that network conditions remain static during each block of transmissions. If nodes conditions are dynamic then communication at the time-averaged rate of $(C_{mavg}-M_{avg})$

using the same code is still possible as long as some verifiably correct marker information can still be transmitted through the network. A more detailed treatment for such dynamic scenarios is considered in [32].

Note 2*:* The codes described in Theorem 26 may have significant decoding delay associated with the time required for the decoder to obtain a sufficient number of transmissions to be able to decode. A more desirable characteristic would be for the network codes to have scalable performance; the more communication they receive, the better their probability of decoding correctly. As long as over half of the marker information each receiver obtains at any point in the decoding process is correct, our decoding scheme still works.

8.5 Variations on the Theme

We now analyze various related models.

Model 1**:** Suppose that in addition to the conditions described in Chapter 8, Xavier knows $\mathcal{Z}$ but the Yvonnes do not. We denote the resulting capacity by $C_{\mathcal{Z}\to X}(M, C_m)$. Alternatively, if all of the Yvonnes know $\mathcal{Z}$ but Xavier does not, we denote the capacity by $C_{\mathcal{Z}\to Y}(M, C_m)$. Theorem 27 shows that knowledge of $\mathcal{Z}$ at the receivers is more useful than knowledge of $\mathcal{Z}$ at the source.

Theorem 27

$$\begin{aligned} C_{\mathcal{Z}\to X}(M, C_m) &= (C_m - M)1(M/C_m < 0.5) \\ C_{\mathcal{Z}\to Y}(M, C_m) &= (C_m - M) \end{aligned}$$

Sketch of Proof: Both $C_{\mathcal{Z}\to X}(M, C_m)$ and $C_{\mathcal{Z}\to Y}(M, C_m)$ must be at least as large as $C_{Adv,Mul}(M, C_m)$ since Xavier and Yvonne$_k$ can ignore $\mathcal{Z}$ and follow the strategy of Theorem 26. If Yvonne$_k$ does not know $\mathcal{Z}$, then regardless of Xavier's knowledge of $\mathcal{Z}$, Zorba can still follow the strategy of pretending to be Xavier, as in the upper bound of Theorem 25, and therefore $C_{\mathcal{Z}\to X}(M, C_m) = C_{Adv,Mul}(M, C_m)$. However, if Xavier uses f^e as in Theorem 26 and Yvonne$_k$ knows $\mathcal{Z}$, she can, with high probability, infer $T^{k,Z}$ and cancel the effect of $X^Z(i)$. Hence $C_{\mathcal{Z}\to Y}(M, C_m) = C_m - M$ for all values of M. □

Model 2**:** Suppose that each $e \in \mathcal{E}$ is noisy with channel capacity $C_{Noise} < 1$. We denote the overall capacity of this channel by $C_{Adv,Noise}(M, C_m)$.

Theorem 28

$$C_{Adv,Noise}(M, C_m) = C_{Noise}C_{Adv,Mul}(M, C_m).$$

Sketch of Proof: Xavier first uses a channel code to make each e noiseless and then uses the code of Theorem 25. No higher rate is achievable since Zorba can use the same strategy as in the upper bound in Theorem 25. □

Model 3**:** Suppose that Yvonne$_k$ wishes to find a set of links L so that removing L neutralizes the effect of Zorba without diminishing the multicast capacity. That is, define the graph $\mathcal{G}^L = (\mathcal{V}, \mathcal{E} - L)$. Define the network code $\mathcal{C}^L$ as the code with the linear coefficients $\beta_{e,e'}$ unchanged if $e \notin L$, and 0 otherwise. For every set of links L such that $\hat{Y}^k(i)$ is a linear function only of $X(i)$ for each k, let $T^{L,k}$ be a matrix such that $\hat{Y}^k(i) = T^{L,k}X(i)$.

Theorem 29 *For any rate satisfying $R \leq C_{Adv}(M, C_m)$, there exists a set of edges $L \subseteq \mathcal{E}$ that can be determined by Yvonne$_k$, such that $\hat{Y}^k(i) = T^{L,k}X(i)$.*

Sketch of Proof: We use the codes from Theorem 26. Each Yvonne$_k$ first determines $T^{L,k}\mathbf{V_Z}$, and then sequentially considers all size-M subsets of $\mathcal{E}$ to see if any of them induces the transform $T^{L,k}$. She chooses the first such set and calls it L. Due to random code design, with high probability, such a choice suffices. □

Model 4**:** Suppose that we allow a small amount of secret and noiseless feedback

($\mathcal{O}(\log(n))$ bits) from each Yvonne$_k$ to Xavier. We denote the capacity of this channel by $C_{FB}(M, C_m)$.

Theorem 30

$$C_{FB}(M, C_m) = C_m - M.$$

Sketch of Proof: We use essentially the codes described in Theorem 26. Each Yvonne$_k$ transmits a secret key (not known to Zorba) to Xavier. Instead of transmitting just the marker, as in Theorem 26, Xavier signs the marker with the secret key using an information-theoretic authentication scheme (e.g., [20]). This enables each Yvonne$_k$ to receive an uncorrupted marker even if only a single path from Xavier is uncorrupted. $\square$

Model 5: Finally, suppose Zorba is unaware of X, and $X = (X_{sec}, X_{pub})$ contains a message X_{sec} which we wish to keep information-theoretically secret ([60]) from Zorba.

Theorem 31

$$C_{Adv,S}(M, C_m) = (C_m - 2M)1(M/C_m < 0.5).$$

Sketch of Proof: Every set U of $C_m - M$ links in every min-cut must contain enough information to be able to decode X correctly. Therefore for the worst case set of links U, the maximum amount of information that can be transmitted through U and still be statistically independent of Zorba's observations on $\mathcal{Z}$ has rate at most $C_m - 2M$, which proves our upper bound. We use the codes from Theorem 26 to prove achievability. Let the linear transform from X to the set of messages $Y^Z = \{Y^e\}_{e \in \mathcal{Z}}$ observed by Zorba be denoted by T^Z. Since $\mathcal{Z}$ is of bounded size M, for a network code $\mathcal{C}$ operating (asymptotically in n) at rate $C_m - M$, the null-space of T^Z must have a dimension of at least $n(C_m - 2M)$. This implies that, with high probability over network code design, a randomly chosen network code $\mathcal{C}$ has the property that for every set of edges $\mathcal{Z}$ satisfying $|\mathcal{Z}| < M$ and for each pair (Y^Z, X_{sec}), there exists a corresponding X_{pub} such that $Y^Z = T(X_{sec}, X_{pub})^T$. Thus for any observed message Y^Z each X_{sec} is equiprobable, and therefore X_{sec} is information-theoretically secret from Zorba. □

8.6 Non-causal Adversary

We now discuss the case when Zorba has non-causal knowledge of the information transmitted on links he controls, i.e., the gs are non-causal functions. As in the case of causal gs, we do this in two stages. We first describe an algorithm for the unicast case that runs in time polynomial in the network code parameters. We then describe an algorithm for the multicast case that runs in time exponential in M.

The algorithms used in the causal case do not work here because Zorba now has access to some marker information prior to choosing his jamming function g. This means that if the marker information is identical on all links, he can change information on the links he controls to match the marker information.

In the unicast case (Section 8.6.2), we get around this by sending different secret keys and corresponding hash functions on different links (Theorem 32). To decode, Yvonne checks for mutual consistency among the received markers and decodes using only messages from links that are all consistent. Since Zorba does not know the secret keys on the links he does not control, with high probability, Yvonne can detect any corrupted messages Zorba introduces. We show how this can be done with low complexity. In the more complex multicast case (Section 8.6.2), we use a similar strategy of checking for mutual consistency among marker information sent on edge-disjoint paths to each receiver. The definitions of encoders, decoders, and jamming functions are identical here to those employed in the proofs in Sections 8.3 and 8.4, except that in the unicast case the encoder uses $|\mathcal{E}|$ random symbols $\{\rho^e\}_{e\in\mathcal{E}}$ instead of just one. The proofs for the upper bounds for these cases are identical to the ones already described in Sections 8.3 and 8.4, and so we omit them.

8.6.1 Unicast

We first consider the unicast problem ($\mathcal{G} = (\{s,t\},\mathcal{E})$, where information must be transmitted from s to t over the parallel edges $e \in \mathcal{E}$).

Theorem 32

$$C_{Adv}(M,|\mathcal{E}|) \;\; = \;\; (|\mathcal{E}|-M)1(M/|\mathcal{E}| < 0.5)$$

Further, for any n and any $m = \omega(\log(n|\mathcal{E}|))$ there exist blocklength-n codes with $R = (1-(|\mathcal{E}|+1)/n)C_{Adv}(M,|\mathcal{E}|)$, $P_e^{(n)} < n|\mathcal{E}|^2 2^{-m}$, and complexity of design and encoding and decoding implementation equal to $\mathcal{O}((nm|\mathcal{E}|)^2)$.

Lower Bound: We describe the design of codes for the regime $M/|\mathcal{E}| < 1/2$ achieving $R = |\mathcal{E}| - M$. In our strategy for achieving this bound, for each $e \in \mathcal{E}$, Y^e contains a *different* marker with a secret random symbol ρ^e and $|\mathcal{E}|$ hash symbols $D^{e,e'}$.

We now describe our coding scheme in detail. As in Theorem 25, we first choose n, which fixes $R = \lfloor (1-(|\mathcal{E}|+1)/n)(|\mathcal{E}|-M)\rfloor$. Also, $m = \Theta(\log(n|\mathcal{E}|))$. Let $\mathcal{L}$ be any $(n-|\mathcal{E}|-1)|\mathcal{E}| \times nR$ Vandermonde matrix over $\mathbb{F}_q$. For each $e(i) \in \mathcal{E}$, the matrix $\mathcal{L}^{e(i)}$, known *a priori* to Xavier, Yvonne, and Zorba, is defined as the $(n-|\mathcal{E}|-1)\times nR$ matrix consisting of the $(n-|\mathcal{E}|-1)(i-1)+1^{th}$ through the $(n-|\mathcal{E}|-1)i^{th}$ rows of

$\mathcal{L}$. For all $e, e' \in \mathcal{E}$ we define T^e, u^e, and $D^{e,e'}$ as

$$\begin{aligned} T^e &= (\mathcal{L}^e x)^T, \\ u^e &= (1, \rho^e, \ldots, (\rho^e)^{n-|\mathcal{E}|-1}), \text{ and} \\ D^{e,e'} &= u^e (T^{e'})^T. \end{aligned}$$

Thus for each $e \in \mathcal{E}$, the first $n - |\mathcal{E}| - 1$ are the channel-coded message symbols, the next symbol is the hash-function's key, and the remaining $|\mathcal{E}|$ symbols are the hash function output dot-products.

Yvonne's decoding scheme h is as follows. Let $\hat{Y}^e = \left(\hat{T}^e, \hat{\rho}^e, (\hat{D}^{e,e'})_{e' \in \mathcal{E}}\right)$ denote the channel output. (The first $n - |\mathcal{E}| - 1$ symbols are $\hat{T}^e$, the next symbol is $\hat{\rho}^e$, and the remaining symbols are $\hat{D}^{e,e'}$.) Let $\mathcal{E}^J \subseteq \mathcal{Z}$ be the set of edges in $\mathcal{E}$ for which $\hat{T}^e \neq T^e$. In the first stage of decoding, Yvonne constructs a *large* set (of size at least $|\mathcal{E}| - M$) of "good" edges that, with high probability, does not contain any edges in $\mathcal{E}^J$. The information on these good edges will then be used to reconstruct the value of the transmitted information X. To do this Yvonne builds a *consistency graph* $\mathcal{G}_{con}(\mathcal{E})$ with vertex set of $\mathcal{G}_{con}(\mathcal{E}) = \mathcal{E}$ and edge-set of $\mathcal{G}_{con}(\mathcal{E}) = \mathcal{E} \times \mathcal{E}$. Two vertices $e, e' \in \mathcal{G}_{con}(\mathcal{E})$ are connected by the edge (e, e') if e and e' are *consistent*, where edges e and e' are consistent if $\hat{Y}^e$ and $\hat{Y}^{e'}$ satisfy

$$\begin{aligned} \hat{D}^{e,e'} &= \hat{u}^e (\hat{T}^{e'})^T \text{ and} \\ \hat{D}^{e',e} &= \hat{u}^{e'} (\hat{T}^e)^T. \end{aligned}$$

Here $\hat{u}^e = (1, \hat{\rho}^e, (\hat{\rho}^e)^2, \ldots, (\hat{\rho}^e)^{n-|\mathcal{E}|-1})$ is Yvonne's estimate for u^e given the received symbol $\hat{\rho}^e$. The *degree of a vertex* $e \in \mathcal{G}_{con}(\mathcal{E})$ is the number of vertices $e' \in \mathcal{G}_{con}(\mathcal{E})$ to which e is connected; self-loops are allowed, and a self-loop contributes 1 to the degree of a vertex. We define $\mathcal{E}^D = \{e \in \mathcal{G}_{con}(\mathcal{E}) | \text{degree}(e) \geq |\mathcal{E}| - M\}$. Note that for any $e, e' \in \mathcal{E} \setminus \mathcal{Z}$, $\hat{Y}^e = Y^e$ and $\hat{Y}^{e'} = Y^{e'}$, and hence $(e, e') \in \mathcal{G}_{con}(\mathcal{E})$. Therefore $\mathcal{E} \setminus \mathcal{Z}$ is contained in $\mathcal{E}^D$.

In the second stage of decoding Yvonne constructs $\mathcal{L}^D$, the $|\mathcal{E}^D|(n-|\mathcal{E}|-1) \times nR$ matrix created by concatenating the matrices in $\{\mathcal{L}^e\}_{e\in\mathcal{E}^D}$. Yvonne obtains $\hat{X}$ by inverting the matrix equation $\hat{Y}^D = \mathcal{L}^D x$, where $\hat{Y}^D$ is the $|\mathcal{E}^D|(n-|\mathcal{E}|-1)$ length vector obtained by the ordered concatenation of $y(e)$, $e \in \mathcal{E}^D$. There is a decoding error only if $\mathcal{E}^J \cap \mathcal{E}^D \neq \phi$, which we now show happens with low probability.

Consider any $e \in \mathcal{E} \setminus \mathcal{Z}$, $e' \in \mathcal{E}^J$. It suffices to prove that with probability at least $1 - 2^{-\Omega(m)}$, no such pair (e, e') will be consistent. By definition of $\mathcal{E}^J$ and $\mathcal{E} \setminus \mathcal{Z}$, $T^{e'} \neq \hat{T}^{e'}$, $\hat{u}^e = u^e$, and $\hat{D}^{e,e'} = D^{e,e'}$. Thus for e and e' to be consistent $u^e(T^{e'})^T = u^e(\hat{T}^{e'})^T$, i.e., $u^e(T^{e'} - \hat{T}^{e'})^T = 0$, which happens only if ρ^e is a root of the degree $n - |\mathcal{E}| - 1$ polynomial $u^e(T^{e'} - \hat{T}^{e'})^T$. Zorba does not know the value of ρ^e, and the polynomial contains at most $n-|\mathcal{E}|-1$ roots in the field of size 2^m. Therefore e and e' are consistent with probability at most $1-(n-|\mathcal{E}|-1)/2^m$. Since there are at most $|\mathcal{E}|^2/4$ such $(e, e') \in (\mathcal{E} \setminus \mathcal{Z}) \times \mathcal{E}^J$, the total probability that $\mathcal{E}^J \cap \mathcal{E}^D \neq \phi$ is at most $(n - |\mathcal{E}| - 1))|\mathcal{E}|^2/2^{m+2} = n|\mathcal{E}|^2 2^{-\Omega(m)}$.

Lastly, it can be verified that the complexity of encoder f^e at each edge e is determined by the complexity of computing the vectors T^e over a field of size $\mathcal{O}(\Delta_n)$, and that the complexity of decoder h is determined by the complexity of inverting a Vandermonde matrix of dimension nR over the same finite field ([21]). □

8.6.2 Multicast

We now consider the multicast problem ($\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a general acyclic graph, and information has to be multicast from s to $\{t_k\}_{k\in\{1,2,\ldots,|\mathcal{T}|\}}$).

Lower bound: We now present a coding strategy that achieves rate $(C_m - M)1(M/C_m < 0.5)$ using ideas from the proof of Theorem 25. We set $R = (1-(|\mathcal{E}|+1)/n)(C_m - M)$ and $m = \Theta(\log(n|\mathcal{E}|))$ and show the existence of codes that achieve

R with asymptotically negligible error probability.

There are three encoding steps. First, as in Theorem 25, let $\mathcal{L}$ be any $(n-(C_m-1)|\mathcal{T}|)C_m \times nR$ Vandermonde matrix over $\mathbb{F}_q$ known *a priori* to Xavier, the Yvonnes, and Zorba; $\{\mathcal{L}^i\}$ for $i \in \{1,\ldots,C_m\}$ be $(n-(C_m-1)|\mathcal{T}|)\times nR$ sub-matrices of $\mathcal{L}$; and T^i the matrix product $(\mathcal{L}^i X)^T$. Now, we generate $|\mathcal{T}|$ sets of consistency information. For each sink t_k, let $\{P^i(s,t_k)\}_{i=1}^{C_m}$ be C_m predetermined edge-disjoint paths from s to t_k. For each $i \in \{1,2,\ldots,C_m\}$, $k \in \{1,2,\ldots,|\mathcal{T}|\}$ we associate the random symbol $\rho^{i,k}$ with the path $P^i(s,t_k)$. For each $t_k \in t$, we denote by $D^{i,i',k}$ the dot-product $u^{i,k}(T^{i'})^T$ where $u^{i,k}$ is defined as $(1,\rho^{i,k},(\rho^{i,k})^2\ldots,(\rho^{i,k})^{n-C_m-1})$.

In the second step, the paths $\{P^i(s,t_k)\}_{i\in\{1,2,\ldots,C_m\}}$ are used to transmit the *consistency information* $\left(\rho^{i,k},(D^{i,i',k})_{i'=1}^{C_m}\right)$ to t_k. This step requires (C_m+1) symbols per t_k, for a total of $(C_m+1)|\mathcal{T}|$ channel uses. For any k Zorba may control e in (less than half of) $\{P^i(s,t_k)\}_{i=1}^{C_m}$, and hence each Yvonne$_k$ at t_k receives (partially corrupted) consistency information denoted by $\left(\hat{\rho}^{i,k},(\hat{D}^{i,i',k})_{i'=1}^{C_m}\right)_{i=1}^{C_m}$.

In the third step, Xavier uses a robust linear network code of rate $C_m - M - \log(|\mathcal{E}|n|\mathcal{T}|)/n$ as described in Theorem 26. Since the rate at which adversarial source nodes s^Z generate information is at most M, a rate of $C_m - M$ would be achievable between s and any receiver t_k if $\mathcal{Z}$ were known to each decoder.

Decoding by Yvonne$_k$ proceeds as follows. For each possible $\mathcal{E}' \subset \mathcal{E}$ of size at most M (there are at most $|\mathcal{E}|^M$ of these), Yvonne$_k$ decodes to $\hat{X}^{\mathcal{E}'}$ by guessing $\mathcal{G}^Z = \mathcal{G}^{\mathcal{E}'}$. More specifically, the symbols $(\hat{Y}^e)_{e\in\Gamma_I(t_k)}$ corresponding to the third step of the encoding process are decoded under the assumption that $\mathcal{Z} = \mathcal{E}'$. If $\mathcal{E}'$ indeed equals $\mathcal{E}^Z$, Theorem 4.3 proves that, with high probability over the choice of the robust network code, $\hat{X}^{\mathcal{E}'} = X$. If $\mathcal{E}' \neq \mathcal{Z}$ then $\hat{X}^{\mathcal{E}'}$ may or may not equal X. We now provide a consistency check similar to the one in Theorem 32, which, with high probability, distinguishes between the cases $\hat{X}^{\mathcal{E}'} = X$ and $\hat{X}^{\mathcal{E}'} \neq X$. This will conclude the proof of our theorem.

After decoding to $\hat{X}^{\mathcal{E}'}$ Yvonne$_k$ computes $\{\hat{T}^{i,k} = (\mathcal{L}^i\hat{X}^{\mathcal{E}'})^T\}_{i=1}^{C_m}$ (where $\mathcal{L}^i$ are the matrices used by Xavier in the first step of encoding). The information $\left(\hat{T}^{i,k}, \hat{\rho}^{i,k}, (\hat{D}^{i,i',k})_{i'=1}^{C_m}\right)_{i=1}^{C_m}$ is used to construct a consistency graph $\mathcal{G}_{con}(\mathcal{E})^k$ with C_m vertices $v(i), i \in \{1, 2, \ldots, C_m\}$. An edge is drawn between $v(i)$ and $v(i')$ if and only if i and i' are consistent, i.e., $\hat{D}^{i,i',k} = \hat{u}^{i,k}(\hat{T}^{i',k})^T$ and $\hat{D}^{i',i,k} = \hat{u}^{i',k}(\hat{T}^{i,k})^T$. Here $\hat{u}^{i,k} = (1, \hat{\rho}^{i,k}, (\hat{\rho}^{i,k})^2, \ldots, (\hat{\rho}^{i,k})^{n-C_m-1})$ is computed by Yvonne$_k$ given $\hat{\rho}^{i,k}$. We now show that if Yvonne$_k$ declares successful decoding when $\mathcal{G}_{con}(\mathcal{E})^k$ has at least $C_m - M$ vertices each of degree at least $C_m - M$ then, with high probability, $\hat{X}^{\mathcal{E}'} = X$.

We first note that if $\hat{X}^{\mathcal{E}'} = X$ for all k, $\hat{T}^{i,k} = T^{i,k}$ for all $i \in \{1, 2, \ldots, C_m\}$. Also, the consistency information satisfies $\left(\hat{\rho}^{i,k}, (\hat{D}^{i,i',k})_{i'=1}^{C_m}\right) = \left(\rho^{i,k}, (D^{i,i',k})_{i'=1}^{C_m}\right)$ for at least $C_m - M$ indices i in $\mathcal{G}_{con}(\mathcal{E})^k$, which are therefore pairwise consistent.

Now assume $\hat{X}^{\mathcal{E}'} \neq X$. We define the following four subsets of vertices of $\mathcal{G}_{con}(\mathcal{E})^k$ whose union equals the entire vertex-set of $\mathcal{G}_{con}(\mathcal{E})^k$.

$$A = \left\{v(i) | \hat{T}^{i,k} = T^{i,k} \text{ and } P^i(s, t_k) \cap \mathcal{Z} = \phi\right\}.$$

$$B = \left\{v(i) | \hat{T}^{i,k} = T^{i,k} \text{ and } P^i(s, t_k) \cap \mathcal{Z} \neq \phi\right\}.$$

$$C = \left\{v(i) | \hat{T}^{i,k} \neq T^{i,k} \text{ and } P^i(s, t_k) \cap \mathcal{Z} = \phi\right\}.$$

$$D = \left\{v(i) | \hat{T}^{i,k} \neq T^{i,k} \text{ and } P^i(s, t_k) \cap \mathcal{Z} \neq \phi\right\}.$$

We now use three observations to prove our assertion. First we note that $|A \cup B| < C_m - M$. If not, let $\mathcal{L}^{A\cup B}$ be the matrix whose rows are exactly the rows of the matrices $\{\mathcal{L}^i\}_{v(i)\in A\cup B}$. By construction $\mathcal{L}^{A\cup B}$ has full rank and therefore one can obtain X by using the inverse transformation. This contradicts the assumption that $X^{\mathcal{E}'} \neq X$. Secondly, we note that $|B \cup D| \leq M$, as for all $v(i) \in B \cup D$, the information on path $P^i(s, t_k)$ was viewed by Zorba, which by assumption can occur on at most M indices i. Finally, we notice that, with high probability, over random $\rho^{i,k}$ there are no edges in $\mathcal{G}_{con}(\mathcal{E})^k$ between vertices $v(i)$ for which Zorba does not control $P^i(s, t_k)$ and vertices

$v(i')$ for which $\hat{T}^{i',k} \neq T^{i',k}$. This is because for such (i,i'), as analyzed in the non-causal unicast case, $\hat{D}^{i,i',k} = \hat{u}^{i,k}(\hat{T}^{i',k})^T$ with probability at most $(n - C_m - 1))/2^m$.

We now analyze the degree of vertices $\mathcal{G}_{con}(\mathcal{E})^k$. By our third observation $v(i)$ in C cannot be connected to any vertex $v(i') \in A$, any vertex $v(i') \in C$ (including self-loops), and any vertex $v(i') \in D$. Thus the degree of vertices in C is at most $|B|$, which is at most M by our second observation, which in turn is strictly less than $C_m - M$.

For $v(i) \in D$, by our third observation $v(i) \in D$ cannot be connected to any $v(i') \in A$ or any $v(i') \in C$. Thus the degree of vertices in D is at most $|B \cup D|$, which is at most $M < C_m - M$ as above.

Finally, by our first observation $|A \cup B| \leq C_m - M - 1$, and thus there are at most $C_m - M - 1$ vertices in $\mathcal{G}_{con}(\mathcal{E})^k$ with degree at least $C_m - M$. $\square$

Chapter 9 Summary and Future Work

9.1 Summary

This thesis explores information-theoretic and algorithmic aspects of the exciting new field of network coding. For various network models this work theoretically characterizes which rates are achievable and which are not; it also provides algorithms that attain the achievable performance.

The idea of all nodes in a network sharing the task of processing information is a simple one, but as the results in this work and others show, the resulting behavior of networks can be significantly altered and improved. The notion is also counter-intuitive; at first sight it seems that mixing uncorrelated information should make communication harder rather than easier.

The key underlying idea tying together the improvements achieved through network coding is the increase in diversity of information flowing through the network.

The greater freedom in choice of messages which can be transmitted by nodes in the network makes code design a less constrained problem, which enables the polynomial-time designs presented in Chapter 5. Classical information-theoretic proofs for many problems consider random design of codes which are shown to achieve capacity with high probability. Such an approach is also followed in this work, but due to the larger class of operations each node is allowed to perform, we can restrict our attention to codes (such as linear codes) that are efficient to implement. The underlying linear structure of the codes enables low complexity deterministic code design. Since design of rate-achieving codes requires only the linear-algebraic invariant of rank being preserved across a number of cutsets linear in the size of the network, code design itself also reduces to a "small" number of linear operations.

What is perhaps the most surprising aspect of code design is that even very decentralized design is possible for capacity-achieving codes; once again, this is possible because nodes in the network have greater leeway in generating new combinations of information on outgoing links, which reduces the possibility of redundancy of information flowing in different parts of the network. It turns out that if nodes are allowed to choose their encoding operations randomly from a sufficiently rich class, with high probability the resulting code ensures that information flowing through the network is maximally linearly independent. One such class, the set of permutation matrices, leads to permute-and-add codes, which have the desirable property of having essentially the same complexity of encoding as classical copy-and-forward codes.

Diversity comes in many forms for network codes; we examine in Chapter 6 three different types of linearity – algebraic, block, and convolutional – and the corresponding types of network codes. The different types of linear network code are well-suited for different types of network problems. We show equivalences between these types of codes, which enable unified code design techniques even in networks running different types of network codes.

We examine the complexity of network code implementation in Chapter 7. Since messages transmitted from nodes can be composed of different combinations of incoming messages, it is conceivable that the freedom afforded by code design diversity comes at the price of a prohibitive complexity or number of arithmetic operations required to implement network codes. This turns out not to be the case – the class of linear operations is exponentially large in the block-length, and so guaranteeing an exponentially small probability of error in maintaining a number of linear-independence invariants that is linear in the size of the network requires a complexity and block-length that is only logarithmic in the size of the network.

Lastly, in Chapter 8, we show that in the case of fault-tolerant networks, the path and message diversity afforded by network coding enables network code design that is resilient even against an attack by a malicious adversary who controls significant parts of the network. In this paradigm, the network coding diversity acts as the redundancy in a *network error-correcting code.* We can maintain robustness to adversarial attack in a causal system by sending key information late; since the adversary does not know what the signature will be, with high probability any changes he makes in the information will be detected. For packet-based systems, however, the assumption of causality is less valid. In this case, the signatures that are used to detect the adversary are those that are transmitted on links not controlled by him. However, this problem is inherently more complex, since the decoders do not know a priori which links these are; this has to be part of the decoding. Decoding proceeds by first guessing which parts of the network are adversarially controlled, verifying whether this guess successfully explains both the information and the key received, and if so, declaring successful decoding. With high probability over network code design, only a correct guess will result in each decoder declaring success in decoding. We tightly bound the rate region for several such problems, and show efficient design and implementation schemes for such attack-resilient network codes.

9.2 Future Work

The trend in network code design has been toward design of progressively simpler codes for ever harder problems. The work in this thesis has several natural extensions in such directions.

The work on low-complexity encoders raises the natural question of whether we can design codes with low-complexity decoders. Another possibility is the design of low-complexity codes that only need to perform one of a small set of encoding operations, which would make it possible to use off-the-shelf components for network code design. Polynomial-time deterministic construction of such low-complexity network codes is also an interesting combinatorial problem.

Wireless networks are a natural medium for the use of network codes, since broadcasting is an inherent property. However, wireless networks are very vulnerable to passive or active attacks by malicious nodes (for instance hidden nodes may eavesdrop, jam transmissions, inject fake packets). An interesting design question is whether we can use the message diversity of a network code to protect against such attacks in the face of inherent wireless communication challenges such as varying network topologies, varying noise levels, interference between nodes, and packet erasures.

Network tomography, i.e., estimation of network topology by probing the edges of the network, is another fertile ground for network coding ideas to help in the design of algorithms with new properties. Since network coding produces a linear transformation between pairs of nodes, a structured choice of code coefficients can result in codes for which knowing the linear transformation is equivalent to knowing the topology. Thus purely linear-algebraic techniques might suffice for the purpose of network identification.

For structured sources of information that have multiple levels of resolution, such as image, audio or video files, a practical approach to rate-distortion via network coding for sinks with variable bandwidths available to them might be worth investigating. In such an approach, interior nodes of a network, on receiving representations of the data in a particular wavelet basis, could perform linear transformations to produce multi-resolution representations of the data in other wavelet bases, to increase the diversity of information reaching each sink.

In each of the above promising directions, the core idea of information diversity via network coding provides a fresh perspective on approaches for attacking classically studied problems.

Bibliography

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

[2] A. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows.* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[3] J. Blömer, R. Karp, and E. Welzl. The rank of sparse random matrices over finite fields. *Random Struct. Alg.*, 10(4):407–419, 1997.

[4] E. Soljanin C. Fragouli. Decentralized network coding. *Information Theory Workshop*, 2004.

[5] G. Cantor. Eigenschaft des inbegriffes aller reelen algebraischen zahlen. *Crelles Journal*, 77:258–262, 1874.

[6] C. Chekuri, C. Fragouli, and E. Soljanin. On throughput benefits and alphabet size in network coding. Submitted to the *IEEE Transactions on Information Theory*, March 2005.

[7] P. A. Chou, Y. Wu, and K. Jain. Practical network coding. In *Proceedings of the 41st Allerton Conference on Communication, Control and Computing*, 2003.

[8] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:251–280, 1990.

[9] R. Cramer, I. Damgård, and S. Fehr. On the cost of reconstructing a secret, or vss with optimal reconstruction phase. In *Proceedings of the 21st Annual*

International Cryptology Conference on Advances in Cryptology, pages 503–523, 2001.

[10] I. Csiszár and P. Narayan. Common randomness and secret key generation with a helper. *IEEE Transactions on Information Theory*, 46(2):344–366, March 2000.

[11] S. Deb and M. Médard. Algebraic gossip: A network coding approach to optimal multiple rumour mongering. Submitted to the *IEEE Transactions Information Theory*, April 2004.

[12] Y. Z. Ding, P. Gopalan, and R. J. Lipton. Error correction against computationally bounded adverseries. In *DIMACS Workshop on Codes and Complexity*, December 2001.

[13] R. Dougherty, C. Freiling, and K. Zeger. Insufficiency of linear coding in network information flow. Submitted to the *IEEE Transactions on Information Theory*, 2004.

[14] J. Edmonds. Minimum partition of a matroid into independent sets. *J. Res. Nat. Bur. Standards Sect.*, 869:67–72, 1965.

[15] P. Elias, A. Feinstein, and C. E. Shannon. A note on maximum flow through a network. *IRE Trans. Inform. Theory*, IT-2:117–119, 1956.

[16] E. Erez and M. Feder. Convolutional network codes. In *IEEE International Symposium on Information Theory*, 2004.

[17] M. Feder, D. Ron, and A. Tavory. Bounds on linear codes for network multicast. In *Electronic Colloquium on Computational Complexity (ECCC) 10(033)*, 2003.

[18] J. Feldman, T. Malkin, C. Stein, and R. A. Servedio. On the capacity of secure network coding. In *Proceedings of 42nd Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, 2004.

[19] C. Fragouli and E. Soljanin. A connection between network coding and convolutional codes. In *2004 IEEE International Conference on Communications*, pages 661–666, 2004.

[20] P. Gemmell and M. Naor. Codes for interactive authentication. *In Proceedings of CRYPTO 93': Lecture Notes in Computer Science*, 773:355–367, 1993.

[21] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, 1996.

[22] R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29:147–160, 1950.

[23] T. Ho. *Networking from a network coding perspective.* Dissertation, Massachusetts Institute of Technology, 2004.

[24] T. Ho. A note on distributed zero-error network code design. personal communication, 2004.

[25] T. Ho., R. Koetter, M. Médard, D. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *IEEE International Symposium on Information Theory (ISIT)*, page 442, Yokohama, July 2003.

[26] T. Ho, B. Leong, R. Koetter, and M. Médard. On the dynamic multicast problem for networks. In *Network Coding Workshop*, Trento, Italy, 2005.

[27] T. Ho, M. Médard, J. Shi, M. Effros, and D. Karger. On randomized network coding. In *Proceedings of 41st Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, 2003.

[28] T. C. Ho, B. Leong, R. Koetter, M. Médard, M. Effros, and D. R. Karger. Byzantine modification detection in multicast networks using randomized network coding. In *International Symposium on Information Theory*, 2004.

[29] S. Jaggi, Y. Cassuto, and M. Effros. Low complexity encoding for network codes. In *International Symposium on Information Theory*, 2006.

[30] S. Jaggi, P. A. Chou, and K. Jain. Low complexity algebraic multicast network codes. In *IEEE International Symposium on Information Theory (ISIT)*, page 368, Yokohama, July 2003.

[31] S. Jaggi, M. Effros, T. Ho, and M. Médard. On linear network coding. In *Proceedings of 42nd Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, 2004.

[32] S. Jaggi, T. Ho, M. Langberg, M. Médard, D. Katabi. Low-complexity distributed design of network error-correcting codes. To be submitted to INFOCOM 2006.

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

[34] K. Jain. Security based on network topology against the wiretapping attack. *IEEE Wireless Communications*, pages 68–71, Feb 2004.

[35] K. Jain, M. Mahdian, and M. R. Salavatipour. Packing Steiner trees. In *14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2003.

[36] L. R. Ford Jr. and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[37] R. Koetter and M. Médard. Beyond routing: An algebraic approach to network coding. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOMM)*, volume 1, pages 122–130, 2002.

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

[39] Ralf Koetter. What is linearity? Personal communication.

[40] S. Lang. *Algebra.* Addison-Wesley, 2nd Edition, 1984.

[41] M. Langberg. Private codes or succint random codes that are (almost) perfect. In *45th Annual Symposium on the Foundations of Computer Science*, 2004.

[42] M. Langberg, A. Sprintson, and S. Bruck. The encoding complexity of network coding. In *International Symposium on Information Theory*, Sept. 2005.

[43] A. Lapidoth and P. Narayan. Reliable communication under channel uncertainty. *IEEE Transactions on Information Theory*, 44(6):2148–2177, October 1998.

[44] A. Rasala Lehman and E. Lehman. Complexity classification of network information flow problems. In *Proceedings of the 41st Allerton Conference on Communication, Control and Computing*, 2003.

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

[46] Z. Li and B. Li. Network coding in undirected networks. In *Proceedings of CISS*, 2004.

[47] D. Lun, M. Médard, T. Ho, and R. Koetter. Network coding with a cost criterion. In *Proceedings of International Symposium in Information Theory and its Applications*, October 2004.

[48] Webpage maintained by R. Koetter. Network coding homepage. *http://tesla.csl.uiuc.edu/ koetter/NWC*, 2003–present.

[49] R. J. McEliece. *The Theory of Information and Coding*, volume 3 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, Reading, Mass., 1977.

[50] R. J. McEliece, E. R. Rodemich, L. Rumsey, and L. R. Welch. New upper bounds on the rate of a code via the delsarte-mcwilliams inequalities. *IEEE Transactions on Information Theory*, 23:157–166, 1977.

[51] R. J. McEliece and W. E. Stark. Channels with block interference. *IEEE Transactions on Information Theory*, 30(1):44–53, January 1984.

[52] M. Médard, M. Effros, T. Ho, and D. Karger. On coding for non-multicast networks. In *Proceedings of the 41st Allerton Conference on Communication, Control and Computing*, 2003.

[53] K. Menger. Zur allgemeinen kurventheorie. *Fund. Math.*, 10:95–115, 1927.

[54] S. Micali, C. Peikert, M. Sudan, and D. Wilson. Optimal error correction against computationally bounded noise. In preparation, 2004.

[55] K. E. Morrison. Random polynomials over finite fields. In *Combinatorics of Algebraic Structures*, http://www.calpoly.edu/ kmorriso/Research/RPFF.pdf, 1999.

[56] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing: Edmonton, Alberta, Canada*, pages 73–85, 1989.

[57] P. Sanders, S. Egner, and L. Tolhuizen. Polynomial time algorithms for network information flow. In *15th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 286–294, 2003.

[58] P. Sanders, S. Egner, and L. M. G. M. Tolhuizen. Algorithms for network information flow. Unpublished, private communication, August 2002.

[59] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 623–656, 1948.

[60] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[61] D. Slepian and J. K. Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, 19:471–480, July 1973.

[62] Y. Wu, P. A. Chou, and S.-Y.Kung. Minimum-energy multicast in mobile ad hoc network using network coding. Submitted to the *IEEE Transactions on Communications*, March 2004.