Network Coding: An Introduction

Tracey Ho

Desmond S. Lun

Contents

Preface

The basic idea behind network coding is extraordinarily simple. As it is defined in this book, network coding amounts to no more than performing coding operations on the contents of packets—performing arbitrary mappings on the contents of packets rather than the restricted functions of replication and forwarding that are typically allowed in conventional, store-and-forward architectures. But, although simple, network coding has had little place in the history of networking. This is for good reason: in the traditional wireline technologies that have dominated networking history, network coding is not very practical or advantageous.

Today, we see the emergence, not only of new technologies, but also of new services, and, in designing new network protocols for these new technologies and services, we must be careful not to simply transplant old protocols because we are familiar with them. Instead, we must consider whether other, hitherto unused, ideas may lead to better solutions for the new situation.

Network coding shows much promise as such an idea. In particular, various theoretical and empirical studies suggest that significant gains can be obtained by using network coding in multi-hop wireless networks and for serving multicast sessions, which are certainly examples of fast-emerging technologies or services. These studies have, for example, encouraged Microsoft to adopt network coding as a core technology of its Avalanche project—a research project that aims to develop a peer-to-peer file distribution system—exploiting the advantages offered by network coding for multicast services. Thus, the time may finally be ripe for network coding.

Hence the motivation for this book: we feel that network coding may have a great deal to offer to the future design of packet networks, and we would like to help this potential be realized. We would like also to

encourage more research in this burgeoning field. Thus, we have aimed the book at two (not necessarily distinct) audiences: first, the practitioner, whose main interest is applications; and, second, the theoretician, whose main interest is developing further understanding of the properties of network coding. Of these two audiences, we have tended to favor the first, though the content of the book is nevertheless theoretical. We have aimed to expound the theory in such a way that it is accessible to those who would like to implement network coding, serving an important purpose that was, in our opinion, inadequately served. The theoretician, in contrast to the practitioner, is spoiled. Besides this book, a survey of important theoretical results in network coding is provided in Yeung et al.'s excellent review, *Network Coding Theory* [149, 150]. Because of our inclination toward applications, however, our presentation differs substantially from that of Yeung et al.

Our presentation draws substantially from our doctoral dissertations [60, 93], and a bias toward work with which we have personally been involved is shown. We endeavor, however, to ensure that most of the significant work in network coding is either covered in the text or mentioned in summary—a goal aided by the notes and further reading in the final section of each chapter. There will inevitably be some unintended omissions, for which we apologize in advance.

Broadly, we intend for the book to be accessible to any reader who has a background in electrical engineering or computer science. Some mathematical methods that we use, such as some algebraic methods and some optimization techniques, may be unfamiliar to readers in this category and, though we do not cover these methods, we provide references to suitable textbooks where relevant.

We have many to thank for their help and support during the development of this book. Foremost among them are Muriel Médard and Ralf Koetter, doctoral and postdoctoral advisers, respectively, to us both, who have been exemplary personal and professional mentors. We would also like to thank the wonderful group of collaborators that worked with us on network coding: Ebad Ahmed, Yu-Han Chang, Supratim Deb, Michelle Effros, Atilla Eryilmaz, Christina Fragouli, Mario Gerla, Keesook J. Han, Nick Harvey, Sidharth (Sid) Jaggi, David R. Karger, Dina Katabi, Sachin Katti, Jörg Kliewer, Michael Langberg, Hyunjoo Lee, Ben Leong, Petar Maymounkov, Payam Pakzad, Joon-Sang Park, Niranjan Ratnakar, Siddharth Ray, Jun Shi, Danail Traskov, Sriram Vishwanath, Harish Viswanathan, and Fang Zhao. In general, the entire network coding community has been a very delightful, friendly, and

intellectually-stimulating community in which to work, and we would like to thank all its members for making it so. We would also like to thank Tao Cui, Theodoros Dikaliotis and Elona Erez for their helpful suggestions and comments on drafts of this book. There are two further groups that we would like to thank—without them this book certainly could not have been produced. The first is the great group of professionals at Cambridge University Press who saw the book to publication: we thank, in particular, Phil Meyler, Anna Littlewood, and Daisy Barton. The second is our families, for their love and support during our graduate studies and the writing of this book.

1
Introduction

Network coding, as a field of study, is young. It was only in 2000 that the seminal paper by Ahlswede, Cai, Li, and Yeung [4], which is generally attributed with the "birth" of network coding, was published. As such, network coding, like many young fields, is characterized by some degree of confusion, of both excitement about its possibilities and skepticism about its potential. Clarifying this confusion is one of the principal aims of this book. Thus, we begin soberly, with a definition of network coding.

1.1 What is network coding?

Defining network coding is not straightforward. There are several definitions that can be and have been used.

In their seminal paper [4], Ahlswede, Cai, Li, and Yeung say that they "refer to coding at a node in a network as *network coding*", where, by coding, they mean an arbitrary, causal mapping from inputs to outputs. This is the most general definition of network coding. But it does not distinguish the study of network coding from network, or multiterminal, information theory—a much older field with a wealth of difficult open problems. Since we do not wish to devote this book to network information theory (good coverage of network information theory already exists, for example, in [28, Chapter 14]), we seek to go further with our definition.

A feature of Ahlswede et al.'s paper that distinguishes it from most network information theory papers is that, rather than looking at general networks where essentially every node has an arbitrary, probabilistic effect on every other node, they look specifically at networks consisting of nodes interconnected by error-free point-to-point links. Thus the network model of Ahlswede et al. is a special case of those ordinarily

studied in network information theory, albeit one that is very pertinent to present-day networks—essentially all wireline networks can be cast into their model once the physical layer has been abstracted into error-free conduits for carrying bits.

Another possible definition of network coding, then, is coding at a node in a network with *error-free* links. This distinguishes the function of network coding from that of channel coding for noisy links; we can similarly distinguish the function of network coding from that of source coding by considering the former in the context of independent incompressible source processes. This definition is frequently used and, under it, the study of network coding reduces to a special case of network information theory. This special case has in fact been studied well before 2000 (see, for example, [52, 133]), which detracts from some of the novelty of network coding, but we can still go further with our definition.

Much work in network coding has concentrated around a particular form of network coding: *random linear network coding*. Random linear network coding was introduced in [62] as a simple, randomized coding method that maintains "a vector of coefficients for each of the source processes," which is "updated by each coding node". In other words, random linear network coding requires messages being communicated through the network to be accompanied by some degree of extra information—in this case a vector of coefficients. In today's communications networks, there is a type of network that is widely-used, that easily accommodates such extra information, and that, moreover, consists of error-free links: packet networks. With packets, such extra information, or side information, can be placed in packet headers and, certainly, placing side information in packet headers is common practice today (e.g., sequence numbers are often placed in packet headers to keep track of order).

A third definition of network coding, then, is coding at a node in a *packet* network (where data is divided into packets and network coding is applied to the contents of packets), or more generally, coding above the physical layer. This is unlike network information theory, which is generally concerned with coding at the physical layer. We use this definition in this book. Restricting attention to packet networks does, in some cases, limit our scope unnecessarily, and some results with implications beyond packet networks may not be reported as such. Nevertheless, this definition is useful because it grounds our discussion in a concrete setting relevant to practice.

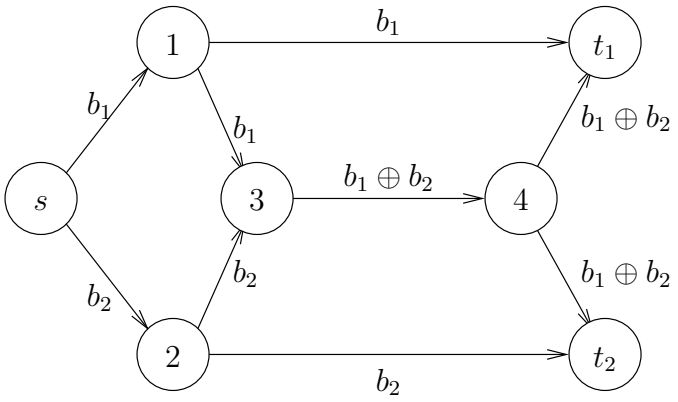


Fig. 1.1. The butterfly network. In this network, every arc represents a directed link that is capable of carrying a single packet reliably. There are two packets, b_1 and b_2, present at the source node s, and we wish to communicate the contents of these two packets to both of the sink nodes, t_1 and t_2.

1.2 What is network coding good for?

Equipped with a definition, we now proceed to discuss the utility of network coding. Network coding can improve throughput, robustness, complexity, and security. We discuss each of these performance factors in turn.

1.2.1 Throughput

The most well-known utility of network coding—and the easiest to illustrate—is increase of throughput. This throughput benefit is achieved by using packet transmissions more efficiently, i.e., by communicating more information with fewer packet transmissions. The most famous example of this benefit was given by Ahlswede et al. [4], who considered the problem of multicast in a wireline network. Their example, which is commonly referred to as the butterfly network (see Figure 1.1). features a multicast from a single source to two sinks, or destinations. Both sinks wish to know, in full, the message at the source node. In the capacitated network that they consider, the desired multicast connection can be established only if one of the intermediate nodes (i.e., a node that is neither source nor sink) breaks from the traditional routing paradigm of packet networks, where intermediate nodes are allowed only to make copies of received packets for output, and performs a coding operation—it takes

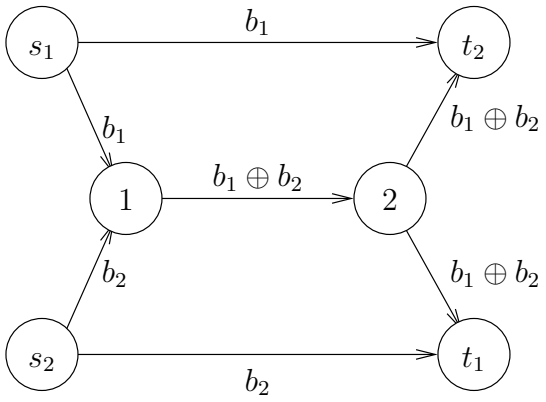


Fig. 1.2. The modified butterfly network. In this network, every arc represents a directed link that is capable of carrying a single packet reliably. There is one packet b_1 present at source node s_1 that we wish to communicate to sink node t_1 and one packet b_2 present at source node s_2 that we wish to communicate to sink node t_2.

two received packets, forms a new packet by taking the binary sum, or XOR, of the two packets, and outputs the resulting packet. Thus, if the contents of the two received packets are the vectors b_1 and b_2, each comprised of bits, then the packet that is output is $b_1 \oplus b_2$, formed from the bitwise XOR of b_1 and b_2. The sinks decode by performing further coding operations on the packets that they each receive. Sink t_1 recovers b_2 by taking the XOR of b_1 and $b_1 \oplus b_2$, and likewise sink t_2 recovers b_1 by taking the XOR of b_2 and $b_1 \oplus b_2$. Under routing, we could communicate, for example, b_1 and b_2 to t_1, but we would then only be able to communicate one of b_1 or b_2 to t_2.

The butterfly network, while contrived, illustrates an important point: that network coding can increase throughput for multicast in a wireline network. The nine packet transmissions that are used in the butterfly network communicate the contents of two packets. Without coding, these nine transmissions cannot used to communicate as much information, and they must be supplemented with additional transmissions (for example, an additional transmission from node 3 to node 4).

While network coding can increase throughput for multicast in a wireline network, its throughput benefits are not limited to multicast or to wireline networks. A simple modification of the butterfly network leads to an example that involves two unicast connections that, with coding, can be established and, without coding, cannot (see Figure 1.2). This

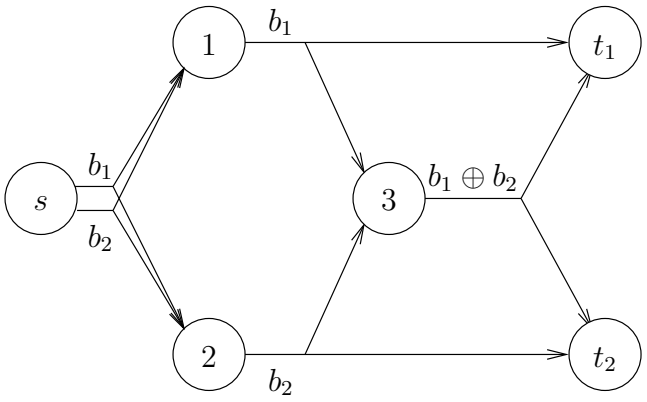


Fig. 1.3. The wireless butterfly network. In this network, every hyperarc represents a directed link that is capable of carrying a single packet reliably to one or more nodes. There are two packets, b_1 and b_2, present at the source node s, and we wish to communicate the contents of these two packets to both of the sink nodes, t_1 and t_2.

example involves *two* unicast connections. For unicast in the lossless wireline networks that have been considered so far, a minimum of two unicast connections is necessary for there to be a throughput gain from network coding. As we establish more concretely in Section 2.3, network coding yields no throughput advantage over routing for a *single* unicast connection in a lossless wireline network.

Network coding can also be extended to wireless networks and, in wireless networks, it becomes even easier to find examples where network coding yields a throughput advantage over routing. Indeed, the wireless counterparts of the butterfly network (Figure 1.3) and the modified butterfly network (Figure 1.4) involve fewer nodes—six and three nodes, respectively, as opposed to seven and six. As before, these examples show instances where the desired communication objective is not achievable using routing, but is achievable using coding. These wireless examples differ in that, rather than assuming that packet transmissions are from a single node to a single other node, they allow for packet transmissions to originate at a single node and end at more than one node. Thus, rather than representing transmissions with arcs, we use *hyperarcs*—generalizations of arcs that may have more than one end node.

The examples that we have discussed so far demonstrate that, even in the absence of losses and errors, network coding can yield a through-

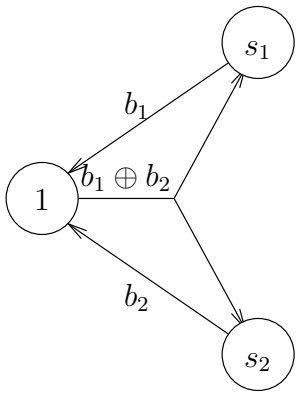


Fig. 1.4. The modified wireless butterfly network. In this network, every hyperarc represents a directed link that is capable of carrying a single packet reliably to one or more nodes. There is one packet b_1 present at source node s_1 that we wish to communicate to node s_2 and one packet b_2 present at source node s_2 that we wish to communicate to node s_1.

put advantage when it is applied either to one or more simultaneous multicast connections or two or more simultaneous unicast connections. This is true both when packets are transmitted only from a single node to a single other node (wireline networks) and when they are transmitted from a single node to one or more other nodes (wireless networks). These examples are, however, mere contrived, toy examples, and it is natural to wonder whether network coding can be generalized and, if so, to what end. Much of the remainder of this book will be devoted to generalizing the observations made thus far to network coding in more general settings.

1.2.2 Robustness

1.2.2.1 Robustness to packet losses

But before we proceed, we address an important issue in packet networks, particularly wireless packet networks, that we have thus far neglected: packet loss. Packet loss arises for various reasons in networks, which include buffer overflow, link outage, and collision. There are a number of ways to deal with such losses. Perhaps the most straightforward, which is the mechanism used by the transmission control protocol (TCP), is to set up a system of acknowledgments, where packets received by the sink are acknowledged by a message sent back to the source and, if

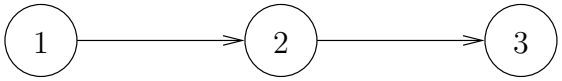


Fig. 1.5. Two-link tandem network. Nodes 1 and 2 are each capable of injecting a single packet per unit time on their respective outgoing links.

the source does not receive the acknowledgment for a particular packet, it retransmits the packet. An alternative method that is sometimes used is channel coding or, more specifically, *erasure coding*. An erasure code, applied by the source node, introduces a degree of redundancy to the packets so that the message can be recovered even if only a subset of the packets sent by the source are received by the sink.

Erasure coding is coding applied by the source node. What about coding applied by intermediate nodes? That is, what about network coding? Is network coding useful in combating against packet losses? It is; and the reason it is can be seen with a very simple example. Consider the simple, two-link tandem network shown in Figure 1.5. In this network, packets are lost on the link joining nodes 1 and 2 with probability ε_{12} and on the link joining nodes 2 and 3 with probability ε_{23}. An erasure code, applied at node 1, allows us to communicate information at a rate of $(1-\varepsilon_{12})(1-\varepsilon_{23})$ packets per unit time. Essentially we have, between nodes 1 and 3, an erasure channel with erasure probability $1-(1-\varepsilon_{12})(1-\varepsilon_{23})$, whose capacity, of $(1-\varepsilon_{12})(1-\varepsilon_{23})$, can be achieved (or approached) with a suitably-designed code. But the true capacity of the system is greater. If we apply an erasure code over the link joining nodes 1 and 2 and another over the link joining nodes 2 and 3, i.e., if we use two stages of erasure coding with full decoding and re-encoding at node 2, then we can communicate information between nodes 1 and 2 at a rate of $1-\varepsilon_{12}$ packets per unit time and between nodes 2 and 3 at a rate of $1-\varepsilon_{23}$ packets per unit time. Thus, we can communicate information between nodes 1 and 3 at a rate of $\min(1-\varepsilon_{12}, 1-\varepsilon_{23})$, which is in general greater than $(1-\varepsilon_{12})(1-\varepsilon_{23})$.

So why isn't this solution used in packet networks? A key reason is delay. Each stage of erasure coding, whether it uses a block code or a convolutional code, incurs some degree of delay because the decoder of each stage needs to receive some number of packets before decoding can begin. Thus, if erasure coding is applied over every link a connection, the total delay would be large. But applying extra stages of erasure coding is simply a special form of network coding—it is coding applied

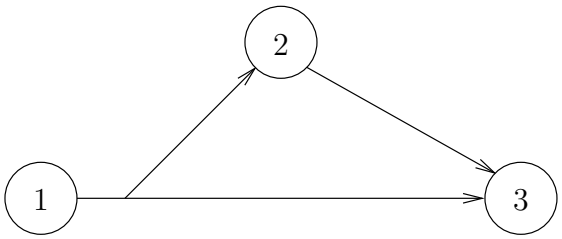


Fig. 1.6. The packet relay channel. Nodes 1 and 2 are each capable of injecting a single packet per unit time on their respective outgoing links.

at intermediate nodes. Thus, network coding can be used to provide robustness against packet losses, which can be translated into throughput gains. What we want from a network coding solution is not only increased throughput, however, we want a solution that goes beyond merely applying additional stages of erasure coding—we want a network coding scheme that applies additional coding at intermediate code *without decoding*. In Chapter 4, we discuss how random linear network coding satisfies the requirements of such a coding scheme.

Losses add an additional dimension to network coding problems and, when losses are present, even a single unicast connection suffices for gains to be observed. Losses are very pertinent to wireless networks, and considering losses makes network coding more relevant to wireless applications. Another characteristic of wireless networks that we have discussed is the presence of broadcast links—links that reach more than one end node—and we have yet to combine losses and broadcast links.

In Figure 1.6, we show a modification of the two-link tandem network that we call the packet relay channel. Here, the link coming out of node 1 doesn't only reach node 2, but also reaches node 3. Because of packet loss, however, whether a packet transmitted by node 1 is received by neither node 2 nor node 3, by node 2 only, by node 3 only, or by both nodes 2 and 3 is determined probabilistically. Let's say packets transmitted by node 1 are received by node 2 only with probability $p_{1(23)2}$, by node 3 only with probability $p_{1(23)3}$, and by both nodes 2 and 3 with probability $p_{1(23)(23)}$) (they are lost entirely with probability $1 - p_{1(23)2} - p_{1(23)3} - p_{1(23)(23)}$). As for packets transmitted by node 2, let's say packets transmitted by node 2 are received by node 3 with probability p_{233} (they are lost entirely with probability $1 - p_{233}$). Network coding, in particular random linear network coding, allows for the maximum achievable throughput in such a set-up,

known as the min-cut capacity, to be reached, which in this case is $\min(p_{1(23)2} + p_{1(23)3} + p_{1(23)(23)}, p_{1(23)3} + p_{1(23)(23)} + p_{233})$.

This is no mean feat: first, from the standpoint of network information theory, it is not even clear that there would exist a simple, capacity-achieving network code and, second, it represents a significant shift from the prevailing approach to wireless packet networks. The prevailing, routing approach advocates treating wireless packet networks as an extension of wireline packet networks. Thus, it advocates sending information along routes; in this case, either sending information from node 1 to node 2, then to node 3, or directly from node 1 to node 3, or, in more sophisticated schemes, using a combination of the two. With network coding, there are no paths as such—nodes contribute transmissions to a particular connection, but these nodes do not necessarily fall along a path. Hence a rethink of routing is necessary. This rethink results in subgraph selection, which we examine in Chapter 5.

1.2.2.2 Robustness to link failures

Besides robustness against random packet losses, network coding is also useful for protection from non-ergodic link failures. Live path protection, where a primary and a backup flow are transmitted for each connection, allows very fast recovery from link failures, since rerouting is not required. By allowing sharing of network resources among different flows, network coding can improve resource usage. For a single multicast session, there exists, for any set of failure patterns from which recovery is possible with arbitrary rerouting, a static network coding solution that allows recovery from any failure pattern in the set.

1.2.3 Complexity

In some cases, although optimal routing may be able to achieve similar performance to that of network coding, the optimal routing solution is difficult to obtain. For instance, minimum-cost subgraph selection for multicast routing involves Steiner trees, which is complex even in a centralized setting, while the corresponding problem with network coding is a linear optimization that admits low-complexity distributed solutions. This is discussed further in Section 5.1.1.

Network coding has also been shown to substantially improve performance in settings where practical limitations necessitate suboptimal solutions, e.g., gossip-based data dissemination [33] and 802.11 wireless ad hoc networking [76].

1.2.4 Security

From a security standpoint, network coding can offer both benefits and drawbacks. Consider again the butterfly network (Figure 1.1). Suppose an adversary manages to obtain only the packet $b_1 \oplus b_2$. With the packet $b_1 \oplus b_2$ alone, the adversary cannot obtain either b_1 or b_2; thus we have a possible mechanism for secure communication. In this instance, network coding offers a security benefit.

Alternatively, suppose that node 3 is a malicious node that does not send out $b_1 \oplus b_2$, but rather a packet masquerading as $b_1 \oplus b_2$. Because packets are coded rather than routed, such tampering of packets is more difficult to detect. In this instance, network coding results in a potential security drawback. We discuss the security implications of network coding in Chapter 6.

We have now given a number of toy examples illustrating some benefits of network coding. That these examples bear some relevance to packet networks should be evident; exactly how the principles they illustrate can be exploited in actual settings is perhaps not. We address more general cases using the model that we put forth in the following section.

1.3 Network model

Packet networks, especially wireless packet networks, are immensely complex and, as such, difficult to accurately model. Moreover, network coding is used in such a wide variety of contexts that it is not sensible to always use the same model. Nevertheless, there are common aspects to all the models that we employ, which we now discuss. The specific aspects of the various models we use are discussed as we encounter them.

As a starting point for our model, we assume that there are a number of *connections*, or *sessions*, that we wish to establish. These connections may be unicast (with a single source node and a single sink node) or multicast (with a single source node and more than one sink node). In a multicast connection, all the sink nodes wish to know the same message originating from the source node. These connections are accompanied with packets that we wish to communicate at rates that may or may not be known. Thus, our model ignores congestion control, i.e., our model does not consider having to regulate the rates of connections. We consider congestion control to be separate problem that is beyond the scope of this book.

We represent the topology of the network with a directed hypergraph

$\mathcal{H} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of hyperarcs. A hypergraph is a generalization of a graph, where, rather than arcs, we have hyperarcs. A hyperarc is a pair (i, J), where i, the start node, is an element of $\mathcal{N}$ and J, the set of end nodes, is a non-empty subset of $\mathcal{N}$. Each hyperarc (i, J) represents a broadcast link from node i to nodes in the non-empty set J. In the special case where J consists of a single element j, we have a point-to-point link. The hyperarc is now a simple arc and we sometimes write (i, j) instead of $(i, \{j\})$. If the network consists only of point-to-point links (as in wireline network), then $\mathcal{H}$ is a graph, denoted alternatively as $\mathcal{G}$ rather than $\mathcal{H}$. The link represented by hyperarc (i, J) may be lossless or lossy, i.e., it may or may not be subject to packet erasures.

To establish the desired connection or connections, packets are injected on hyperarcs. Let z_{iJ} be the average rate at which packets are injected on hyperarc (i, J). The vector z, consisting of z_{iJ}, $(i, J) \in \mathcal{A}$, defines the rate at which packets are injected on all hyperarcs in the network. In this abstraction, we have not explicitly accounted for any queues. We assume that queueing occurs at a level that is hidden from our abstraction and, provided that z lies within some constraint set Z, all queues in the network are stable. In wireline networks, links are usually independent, and the constraint set Z decomposes as the Cartesian product of $|\mathcal{A}|$ constraints. In wireless networks, links are generally dependent and the form of Z may be complicated (see, for example, [29, 74, 75, 82, 140, 144]). For the time being, we make no assumptions about Z except that it is a convex subset of the positive orthant and that it contains the origin.

The pair $(\mathcal{H}, Z)$ defines a capacitated graph that represents the network at our disposal, which may be a full, physical network or a subnetwork of a physical network. The vector z, then, can be thought of as a subset of this capacitated graph—it is the portion actually under use—and we call it the *coding subgraph* for the desired connection or connections. We assume that the coding subgraph defines not only the rates of packet injections on hyperarcs, but also the specific times at which these injections take place. Thus, the classical networking problems of routing and scheduling are special subproblems of the problem of selecting a coding subgraph.

The examples discussed in the previous section give instances of coding subgraphs—instances where packet injections have been chosen, and the task that remains is to use them as effectively as possible. Perhaps the simplest way of representing a coding subgraph in a lossless network

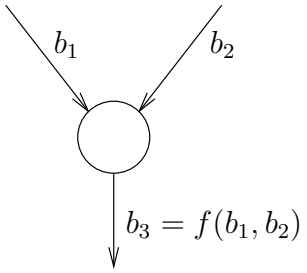


Fig. 1.7. Coding at a node in a static subgraph. Two packets, b_1 and b_2, are each carried by one of the incoming arcs. The outgoing arc carries a packet that is a function of b_1 and b_2.

is to represent each packet transmission over some time period as a separate hyperarc, as we have done in Figures 1.1–1.4. We may have parallel hyperarcs as we do in Figure 1.3 (where there are two hyperarcs $(s, \{1, 2\})$), representing multiple packets transmitted and received by the same nodes in a single time period. Coding at a node is shown in Figure 1.7. We call this representation of a subgraph a *static subgraph*. In a static subgraph, time is not represented explicitly, and it appears as though events occur instantaneously. Presumably in reality, there is some delay involved in transmitting packets along links, so the output of packets on a link are delayed from their input. Thus, static subgraphs hide some timing details that, though not difficult to resolve, must be kept in mind. Moreover, we must restrict our attention to acyclic graphs, because cyclic graphs lead to the instantaneous feedback of packets. Despite its limitations, static subgraphs suffice for much that we wish to discuss, and they will be used more or less exclusively in Chapter 2, where we deal with lossless networks.

For lossy networks, the issue of time becomes much more important. The network codes that are used for lossy networks are generally much longer than those for lossless networks, i.e., one coding block involves many more source packets. Looked at another way, the time period that must be considered for a network code in a lossy network is much longer than that in a lossless network. Hence it becomes imperative to examine the interplay of coding and time at a coding node. To do this, we extend static subgraphs to *time-expanded subgraphs*.

A time-expanded subgraph represents not only the injection and reception points of packets, but also the times at which these injections and receptions take place. We draw only successful receptions, hence,

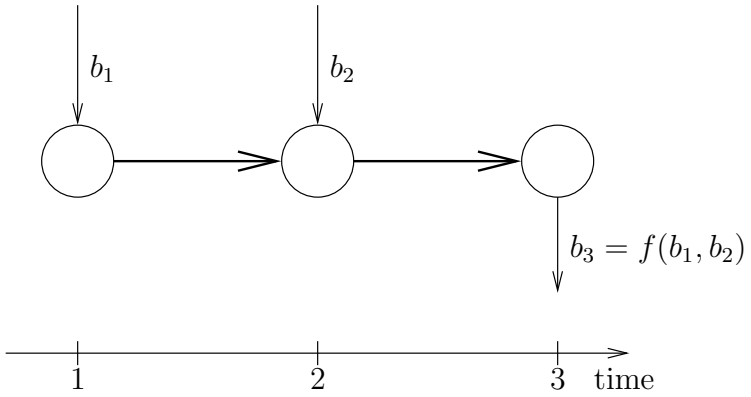


Fig. 1.8. Coding at a node in a time-expanded subgraph. Packet b_1 is received at time 1, and packet b_2 is received at time 2. The thick, horizontal arcs have infinite capacity, and represent data stored at a node. Thus, at time 3, packets b_1 and b_2 can be used to form packet b_3.

in a lossy network, a time-expanded subgraph in fact represents a particular element in the random ensemble of a coding subgraph. Suppose for example that, in Figure 1.7, packet b_1 is received at time 1, packet b_2 is received at time 2, and packet b_3 is injected at time 3. In a time-expanded subgraph, we represent these injections and receptions as shown in Figure 1.8. In this example, we have used integral values of time, but real values of time can just as easily be used. We now have multiple instances of the same node, with each instance representing the node at a different time. Joining these instances are infinite capacity links that go forward in time, representing the ability of nodes to store packets. We use time-expanded subgraphs in Chapter 4, where we deal with lossy networks.

1.4 Outline of book

We begin, in Chapter 2, with the setting for which network coding theory was originally developed: multicast in a lossless wireline network. The prime example illustrating the utility of network coding in this setting is the butterfly network (Figure 1.1), and, in Chapter 2, we extend the insight developed from this example to general topologies. We characterize the capacity of network coding for the multicast problem and discuss both deterministic and random code constructions that allow this capacity to be achieved. In the theoretical discussion, we deal with static

subgraphs, which, as we have noted, hide the details of time. Thus, it is not immediately apparent how the theory applies to real packet networks, where dynamical behavior is usually important. We discuss packet networks explicitly in Section 2.5 and show how random network coding can be applied naturally to dynamic settings. The strategy discussed here is revisited in Chapter 4. We conclude the chapter with two other extensions of the basic theory for lossless multicast: the case of networks with cycles (recall the issue of instantaneous feedback in static subgraphs) and that of correlated source processes.

In Chapter 3, we extend our discussion to non-multicast problems in lossless wireline networks, i.e., we consider the situation where we wish to establish multiple connections, and we potentially code packets from separate connections together, as in the case of the modified butterfly network (Figure 1.2). We refer to this type of coding as *inter-session coding*. This is as opposed to the situation where each connection is kept separate (as in Chapter 2, where we only consider a single connection), which we call *intra-session coding*. We show that linear network coding is not in general sufficient to achieve capacity and that non-linear codes may be required. Given the difficulty of constructing codes without a linear structure, work on constructing inter-session codes has generally focused on suboptimal approaches that give non-trivial performance improvements. We discuss some of these approaches in Chapter 3.

In Chapter 4, we consider lossy networks. We show how random linear network coding can be applied in lossy networks, such as the packet relay channel (Figure 1.6), and that it yields a capacity-achieving strategy for single unicast or multicast connections in such networks. We also derive an error exponent that quantifies the rate at which the probability of error decays with coding delay.

Chapters 2–4 all assume that the coding subgraph is already defined. Chapter 5 considers the problem of choosing an appropriate coding subgraph. Two types of approaches are considered: flow-based approaches, where we assume that the communication objective is to establish connections at certain, given flow rates, and queue-length-based approaches, where the flow rates, though existent, are not known. We deal primarily with subgraph selection for intra-session coding, though we do also discuss approaches to subgraph selection for inter-session coding.

In Chapter 6, we consider the problem of security against adversarial errors. This problem is motivated by the application of network coding to overlay networks, where not all nodes can necessarily be trusted.

Thus, mechanisms are necessary to allow errors introduced by malicious nodes to be either corrected or detected.

1.5 Notes and further reading

Error-free networks have been considered for some time, and work on error-free networks includes that of Han [52] and Tsitsiklis [133]. Ahlswede et al. [4] were the first to consider the problem multicast of an error-free network. In their work, which had its precursor in earlier work relating to specific network topologies [146, 120, 152, 151], they showed that coding at intermediate nodes is in general necessary to achieve the capacity of a multicast connection in an error-free network and characterized that capacity. This result generated renewed interest in error-free networks, and it was quickly strengthened by Li et al. [88] and Koetter and Médard [85], who independently showed that linear codes (i.e., codes where nodes are restricted to performing operations that are linear over some base finite field) suffice to achieve the capacity of a multicast connection in an error-free network.

Ho et al. [62] introduced random linear network coding as a method for multicast in lossless packet networks and analyzed its properties. Random linear network coding for multicast in lossless packet networks was further studied in [27, 66]; it was also studied as a method for data dissemination in [34] and as a method for data storage in [1]; in [94, 98], its application to lossy packet networks was examined. Protocols employing random linear network coding in peer-to-peer networks and mobile ad-hoc networks (MANETs) are described in [50] and [112], respectively.

The butterfly network first appears in [4]; its modified form first appears in [84, 119]. The wireless butterfly network first appears in [97]; its modified form first appears in [139].

The basic static subgraph model that we use derives principally from [85]. The use of time-expanded subgraphs first appears in [136].

2
Lossless Multicast Network Coding

Multicast refers to the case where the same information is transmitted to multiple sink nodes. The first application of network coding to be discovered was that network coding allows the maximum possible multicast rate to be achieved in a noise-free, or lossless, network. Clearly, this rate can be no more than the capacity between the source and each sink individually. As we will see, network coding allows joint use of network resources by multiple sink nodes, so that any rate possible for all sinks individually is simultaneously achievable for all sinks together.

2.0.1 Notational conventions

We denote matrices with bold uppercase letters and vectors with bold lowercase letters. All vectors are row vectors unless indicated otherwise with a subscript T. We denote by $[\mathbf{x}, \mathbf{y}]$ the concatenation of two row vectors $\mathbf{x}$ and $\mathbf{y}$. For any vector (or matrix) whose entries (rows/columns) are indexed by the arcs of a network, we assume a consistent ordering of the vector entries (matrix rows/columns) corresponding to a topological ordering of the arcs.

2.1 Basic network model and multicast network coding problem formulation

Leaving aside for a start the complexities of real packet networks, we consider a very simple network model and problem formulation, widely used in the network coding literature, to gain some fundamental insights and develop some basic results. We will later apply these insights and results to more complex network models.

The basic problem we consider is a single-source multicast on an

acyclic network. The network is represented by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of arcs. There are r *source processes* $X_1, \ldots, X_r$ originating at a given source node $s \in \mathcal{N}$. Each source process X_i is a stream of independent random bits of rate one bit per unit time.

The arcs are directed and lossless, i.e. completely reliable. Each arc $l \in \mathcal{A}$ can transmit one bit per time unit from its start (origin) node $o(l)$ to its end (destination) node $d(l)$; there may be multiple arcs connecting the same pair of nodes. Arc l is called an *input arc* of $d(l)$ and an *output arc* of $o(l)$. We denote by $\mathcal{I}(v), \mathcal{O}(v)$ the set of input links and output links respectively of a node v. We refer to the random bitstream transmitted on an arc l as an *arc process* and denote it by Y_l. For each node v, the *input processes* of v are the arc processes Y_k of input arcs k of v and, if $v = s$, the source processes $X_1, \ldots, X_r$. The arc process on each of v's output arcs l is a function of one or more of v's input processes, which are called the input processes of l; we assume for now that all of v's input processes are input processes of l.†

All the source processes must be communicated to each of a given set $\mathcal{T} \subset \mathcal{N} \backslash s$ of sink nodes. We assume without loss of generality that the sink nodes do not have any output arcs‡. Each sink node $t \in \mathcal{T}$ forms *output processes* $Z_{t,1}, \ldots, Z_{t,r}$ as a function of its input processes. Specifying a graph $\mathcal{G}$, a source node $s \in \mathcal{N}$, source rate r, and a set of sink nodes $\mathcal{T} \subset \mathcal{N} \backslash s$ defines a multicast network coding problem. A *solution* to such a problem defines coding operations at network nodes and decoding operations at sink nodes such that each sink node reproduces the values of all source processes perfectly, i.e. $Z_{t,i} = X_i \; \forall \; t \in \mathcal{T}, i = 1, \ldots, r,$. A network coding problem for which a solution exists is said to be *solvable*.

2.2 Delay-free scalar linear network coding

We first consider the simplest type of network code, delay-free scalar linear network coding over a finite field. As we will see, this type of network code is sufficient to achieve optimal throughput for the acyclic multicast problem described above, but is not sufficient in general.

Each bitstream corresponding to a source process X_i or arc process

† We will drop this assumption in Section 2.5 when we extend the model to packet networks. In the case without network coding, each arc has only one input process.

‡ Given any network, we can construct an equivalent network coding problem satisfying this condition by adding, for each sink node t, a virtual sink node t' and r links from t to t'.

Y_l is divided into vectors of m bits. Each m-bit vector corresponds to an element of the finite field $\mathbb{F}_q$ of size $q = 2^m$. We can accordingly view each source or arc process as a vector of finite field symbols instead of bits.

Since we are considering an acyclic network, we do not have to explicitly consider arc transmission delays – we simply assume that the nth symbol for each arc l is transmitted only after $o(l)$ has received the nth symbol of each of its input processes. This is equivalent, as far as the analysis is concerned, to assuming that all transmissions happen instantaneously and simultaneously, hence the term *delay-free*. This assumption would lead to stability problems if there is a cycle of dependent arcs, i.e. a directed cycle of arcs each transmitting data that is a function of data on its predecessor in the cycle, possibly coded together with other inputs.

In scalar linear network coding, the nth symbol transmitted on an arc l is a scalar linear function, in $\mathbb{F}_q$, of the nth symbol of each input process of node $o(l)$, and this function is the same for all n. Thus, instead of working with processes that are streams of symbols, it suffices to consider just one symbol for each process. For notational convenience, in the remainder of this section, X_i, Y_l and $Z_{t,i}$ refer to a single symbol of the corresponding source, arc and output processes respectively.

Scalar linear network coding for an arc l can be represented by the equation

$$Y_l = \sum_{k \in \mathcal{I}(o(l))} f_{k,l} Y_k + \begin{cases} \sum_i a_{i,l} X_i & \text{if } o(l) = s \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where $a_{i,l}, f_{k,l}$ are scalar elements from $\mathbb{F}_q$, called *(local) coding coefficients*, specifying the coding operation. In the case of multicast with independent source processes, we will see that it suffices for each sink node t to form its output symbols as a scalar linear combination of its input symbols

$$Z_{t,i} = \sum_{k \in \mathcal{I}(t)} b_{t,i,k} Y_k \tag{2.2}$$

We denote by $\mathbf{a}$ and $\mathbf{f}$ the vectors of coding variables $(a_{i,l} : \ 1 \le i \le r, l \in \mathcal{A})$ and $(f_{k,l} : , l, k \in \mathcal{A})$ respectively, and by $\mathbf{b}$ the vector of decoding variables $(b_{t,i,k} : \ t \in \mathcal{T}, 1 \le i \le r, k \in \mathcal{A})$.

Since all coding operations in the network are scalar linear operations of the form (2.1), it follows inductively that for each arc l, Y_l is a scalar

linear function of the source symbols X_i. In equation form,

$$Y_l = \sum_{i=1}^{r} c_{i,l} X_i \tag{2.3}$$

where coefficients $c_{i,l} \in \mathbb{F}_q$ are functions of the coding variables $(\mathbf{a}, \mathbf{f})$. The vector

$$\mathbf{c}_l = [c_{1,l} \dots c_{r,l}] \in \mathbb{F}_q^r,$$

is called the *(global) coding vector* of arc l. It specifies the overall mapping from the source symbols to Y_l resulting from the aggregate effect of local coding operations at network nodes. For the source's output arcs l, $\mathbf{c}_l = [a_{1,l} \dots a_{r,l}]$. Since the network is acyclic, we can index the arcs *topologically*, i.e. such that at each node all incoming arcs have lower indexes than all outgoing arcs†, and inductively determine the coding vectors $\mathbf{c}_l$ in ascending order of l using (2.1).

Outputs $Z_{t,i}$ are likewise scalar linear operations of the source symbols X_i. Thus, the mapping from the vector of source symbols $\mathbf{x} = [X_1 \dots X_r]$ to the vector of output symbols $\mathbf{z}_t = [Z_{t,1} \dots Z_{t,r}]$ at each sink t is specified by a linear matrix equation

$$\mathbf{z}_t = \mathbf{x}\mathbf{M}_t.$$

$\mathbf{M}_t$ is a function of $(\mathbf{a}, \mathbf{f}, \mathbf{b})$ and can be calculated as the matrix product

$$\mathbf{M}_t = \mathbf{A}(\mathbf{I} - \mathbf{F})^{-1}\mathbf{B}_t^T$$

where

- $\mathbf{A} = (a_{i,l})$ is an $r \times |\mathcal{A}|$ matrix whose nonzero entries $a_{i,l}$ are the coefficients with which source symbols X_i are linearly combined to form symbols on the source's output arcs l (ref Equation (2.1)). Columns corresponding to all other arcs are all zero. Matrix $\mathbf{A}$ can be viewed as a transfer matrix from the source symbols to the source's output arcs.
- $\mathbf{F} = (f_{k,l})$ is a $|\mathcal{A}| \times |\mathcal{A}|$ matrix whose nonzero entries $f_{k,l}$ are the coefficients with which node $d(k)$ linearly combines arc symbols Y_k to form symbols on output arcs l (ref Equation (2.1)). $f_{k,l} = 0$ if $d(k) \neq o(l)$. For $n = 1, 2, \dots,$ the (k,l)th entry of $\mathbf{F}^n$ gives the mapping from Y_k to Y_l due to $(n+1)$-hop (or arc) paths. Since we are

† The topological order is an extension, generally non-unique, of the partial order defined by the graph.

considering an acyclic network, $\mathbf{F}$ is nilpotent, i.e. $\mathbf{F}^{\tilde{n}} = \mathbf{0}$ for some $\tilde{n}$, and the (k,l)th entry of

$$(\mathbf{I}-\mathbf{F})^{-1} = \mathbf{I} + \mathbf{F} + \mathbf{F}^2 + \ldots$$

gives the overall mapping from Y_k to Y_l due to all possible paths between arcs k and l. $(\mathbf{I}-\mathbf{F})^{-1}$ can thus be considered as a transfer matrix from each arc to every other arc.

- $\mathbf{B}_t = (b_{t,i,k})$ is an $r \times |\mathcal{A}|$ matrix. Its nonzero entries $b_{t,i,k}$ are the coefficients with which sink t linearly combines symbols Y_k on its input arcs k to form output symbols $Z_{t,i}$ (ref Equation (2.2)). Columns corresponding to all other arcs are all zero. $\mathbf{B}_t$ is the transfer matrix representing the decoding operation at sink t.

The value of $(\mathbf{a}, \mathbf{f}, \mathbf{b})$, or equivalently, the value of $(\mathbf{A}, \mathbf{F}, \mathbf{B}_t : \ t \in \mathcal{T})$, specifies a scalar linear network code. Finding a scalar linear solution is equivalent to finding a value for $(\mathbf{a}, \mathbf{f}, \mathbf{b})$ such that $\mathbf{A}(\mathbf{I}-\mathbf{F})^{-1}\mathbf{B}_t^T = \mathbf{I}\ \forall\ t \in \mathcal{T}$, i.e. the source symbols are reproduced exactly at each sink node.

Defining the matrix $\mathbf{C} := \mathbf{A}(\mathbf{I}-\mathbf{F})^{-1}$, we have that the lth column of $\mathbf{C}$ gives the transpose of the coding vector $\mathbf{c}_l$ for arc l.

Having developed a mathematical framework for scalar linear network coding, we proceed in the following to address some basic questions:

- Given a multicast network coding problem, how do we determine the maximum multicast rate for which the problem is solvable?
- What is the maximum throughput advantage for multicast network coding over routing?
- Given a solvable multicast network coding problem, how do we construct a solution?

2.3 Solvability and throughput

2.3.1 The unicast case

As a useful step towards characterizing solvability of a multicast problem, we consider the special case of unicast: communicating at rate r between a source node s and a single sink node t in our basic network model. This can be viewed as a degenerate multicast network coding problem with r source processes originating at s and a single sink t.

The famous max-flow/min-cut theorem for a point-to-point connection tells us that the following two conditions are equivalent:

(C1) There exists a flow of rate r between the s and t.

(C2) The value of the minimum cut between s and t† is at least r.

The network coding framework provides another equivalent condition which is of interest to us because it generalizes readily to the multicast case:

(C3) The determinant of the transfer matrix $\mathbf{M}_t$ is nonzero over the ring of polynomials $\mathbb{F}_2[\mathbf{a}, \mathbf{f}, \mathbf{b}]$.

Theorem 2.1 *Conditions (C1) and (C3) are equivalent.*

The proof of this theorem and the next uses the following lemma:

Lemma 2.1 *Let f be a nonzero polynomial in variables $x_1, x_2, \ldots, x_n$ over $\mathbb{F}_2$, and let d be the maximum degree of f with respect to any variable. Then there exist values for $x_1, x_2, \ldots, x_n$ in $\mathbb{F}_{2^m}^n$ such that $f(x_1, x_2, \ldots, x_n) \neq 0$, for any m such that $2^m > d$.*

Proof Consider f as a polynomial in $x_2, \ldots, x_n$ with coefficients from $\mathbb{F}_2[x_1]$. Since the coefficients of f are polynomials of degree at most d they are not divisible by $x_1^{2^m} - x_1$ (the roots of which are the elements of $\mathbb{F}_{2^m}$). Thus, there exists an element $\alpha \in \mathbb{F}_{2^m}$ such that f is nonzero when $x_1 = \alpha$. The proof is completed by induction on the variables. □

Proof of Theorem 2.1: If (C1) holds, we can use the Ford-Fulkerson algorithm to find r arc-disjoint paths from s to t. This corresponds to a solution where $\mathbf{M}_t = \mathbf{I}$, so (C3) holds. Conversely, if (C3) holds, by Lemma 2.1, there exists a value for $(\mathbf{a}, \mathbf{f}, \mathbf{b})$ over a sufficiently large finite field, such that $\det(\mathbf{M}_t) \neq 0$. Then $\tilde{\mathbf{B}}_t = (\mathbf{M}_t^T)^{-1}\mathbf{B}_t$ satisfies $\mathbf{C}\tilde{\mathbf{B}}_t^T = \mathbf{I}$, and $(\mathbf{A}, \mathbf{F}, \tilde{\mathbf{B}})$ is a solution to the network coding problem, implying (C1). □

2.3.2 The multicast case

The central theorem of multicast network coding states that if a communication rate r is possible between a source node and each sink node individually, then with network coding it is possible to multicast at rate

† A cut between s and t is a partition of $\mathcal{N}$ into two sets $Q, \mathcal{N}\backslash Q$, such that $s \in Q$ and $t \in \mathcal{N}\backslash Q$. Its value is the number of arcs whose start node is in Q and whose end node is in $\mathcal{N}\backslash Q$.

r to all sink nodes simultaneously. An intuitive proof is obtained by extending the preceding results for the unicast case.

Theorem 2.2 *Consider an acyclic delay-free multicast problem where r source processes originating at source node s are demanded by a set $\mathcal{T}$ of sink nodes. There exists a solution if and only if for each sink node $t \in \mathcal{T}$ there exists a flow of rate r between s and t.*

Proof We have the following sequence of equivalent conditions:

$\forall t \in \mathcal{T}$ there exists a flow of rate r between s and t
$\Leftrightarrow \forall t \in \mathcal{T}$ the transfer matrix determinant $\det\ \mathbf{M}_t$ is nonzero over the ring of polynomials $\mathbb{F}_2[\mathbf{a}, \mathbf{f}, \mathbf{b}]$
$\Leftrightarrow \prod_{t \in \mathcal{T}} \det\ \mathbf{M}_t$ is nonzero over the ring of polynomials $\mathbb{F}_2[\mathbf{a}, \mathbf{f}, \mathbf{b}]$
$\Leftrightarrow$ there exists a value for $(\mathbf{a}, \mathbf{f}, \mathbf{b})$ in a large enough finite field such that $\prod_{t \in \mathcal{T}} \det\ \mathbf{M}_t$ evaluates to a nonzero value. From this we can obtain a solution $(\mathbf{a}, \mathbf{f}, \mathbf{b}')$, since each sink t can multiply the corresponding vector of output values $\mathbf{z}_t$ by $\mathbf{M}_t^{-1}$ to recover the source values $\mathbf{x}$.

where the first step follows from applying Theorem 2.1 to each sink and the last step follows from Lemma 2.1. □

Corollary 2.1 *The maximum multicast rate is the minimum, over all sink nodes, of the minimum cut between the source node and each sink node.*

2.3.3 Multicasting from multiple source nodes

The analysis developed for the case of multicasting from one source node readily generalizes to the case of multicasting from multiple source nodes to the same set of sink nodes. Consider a multiple-source multicast problem on a graph $(\mathcal{N}, \mathcal{A})$ where each source process $X_i, i = 1, \ldots, r$, instead of originating at a common source node, originates at a (possibly different) source node $s_i \in \mathcal{N}$.

One approach is to allow $a_{i,l}$ to take a nonzero value only if X_i originates at $o(l)$, i.e. we replace Equation 2.1 with

$$Y_l = \sum_{i\,:\,o(l)=s_i} a_{i,l} X_i + \sum_{k \in \mathcal{I}(o(l))} f_{k,l} Y_k \tag{2.4}$$

An alternative is to convert the multiple-source multicast problem

into an equivalent single-source multicast problem, by adding to $\mathcal{N}$ a virtual source node s from which r source processes originate, and to $\mathcal{A}$ one virtual arc (s, s_i) for each source process X_i. We can then apply similar analysis as in the single source node case to obtain the following multiple source counterpart to Theorem 2.2.

Theorem 2.3 *Consider an acyclic delay-free multicast problem on a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with r source processes $X_i, i = 1, \ldots, r$ originating at source nodes $s_i \in \mathcal{N}$ respectively, demanded by a set $\mathcal{T}$ of sink nodes. There exists a solution if and only if for each sink node $t \in \mathcal{T}$ and each subset $\mathcal{S} \subset \{s_i : \ i = 1, \ldots, r\}$ of source nodes, the max flow/min cut between $\mathcal{S}$ and t is greater than or equal to $|\mathcal{S}|$.*

Proof Let $\mathcal{G}'$ be the graph obtained by adding to $\mathcal{N}$ a virtual source node s and to $\mathcal{A}$ one virtual arc (s, s_i) for each source process X_i. We apply Theorem 2.2 to the equivalent single-source multicast problem on $\mathcal{G}'$. For any cut Q such that $s \in Q, t \in \mathcal{N}\backslash Q$, let $\mathcal{S}(Q) = Q \cap \{s_i : \ 1, \ldots, r\}$ be the subset of actual source nodes in Q. The condition that the value of the cut Q in $\mathcal{G}'$ is at least r is equivalent to the condition that the value of the cut $\mathcal{S}(Q)$ in $\mathcal{G}$ is at least $|\mathcal{S}(Q)|$, since there are $r - |\mathcal{S}(Q)|$ virtual arcs crossing the cut Q from s to the actual source nodes not in Q. □

2.3.4 Maximum throughput advantage

The multicast throughput advantage of network coding over routing for a given network graph $(\mathcal{N}, \mathcal{A})$ with arc capacities $\mathbf{z} = (z_l : \ l \in \mathcal{A})$, source node $s \in \mathcal{N}$ and sink nodes $\mathcal{T} \subset \mathcal{N}\backslash s$ is defined as the ratio of the multicast capacity with network coding to the multicast capacity without network coding. The capacity with network coding is given by the max flow min cut condition, from Corollary 2.1. The capacity without network coding is equal to the fractional Steiner tree packing number, which is given by the following linear program:

$$\begin{aligned} &\max_{\mathbf{u}} \sum_{k \in \mathcal{K}} u_k \\ &\text{subject to} \sum_{k \in \mathcal{K}:\ l \in k} u_k \leq z_l \ \ \forall\, l \in \mathcal{A} \\ &u_k \geq 0 \ \ \forall\, k \in \mathcal{K} \end{aligned} \tag{2.5}$$

where $\mathcal{K}$ is the set of all possible Steiner trees in the network, and u_k is the flow rate on tree $k \in \mathcal{K}$.

It is shown in [3] that for a given directed network, the maximum multicast throughput advantage of network coding over routing is equal to the integrality gap of a linear programming relaxation for the minimum weight directed Steiner tree. Specifically, consider a network $(\mathcal{N}, \mathcal{A}, s, \mathcal{T})$ with arc weights $\mathbf{w} = (w_l : \ l \in \mathcal{A})$. The minimum weight Steiner tree problem can be formulated as an integer program

$$\begin{aligned} &\min_{\mathbf{a}} \sum_{l \in \mathcal{A}} w_l a_l \\ &\text{subject to} \sum_{l \in \Gamma_+(Q)} a_l \geq 1 \ \ \forall\, Q \in \mathcal{C} \\ &a_l \in \{0,1\} \ \ \forall\, l \in \mathcal{A} \end{aligned} \tag{2.6}$$

where $\mathcal{C} := \{Q \subset \mathcal{N} : \ s \in Q, \mathcal{T} \not\subset Q\}$ denotes the set of all cuts between the source and at least one sink, $\Gamma_+(Q) := \{(i,j) : \ i \in Q, j \notin Q\}$ denotes the set of forward arcs of a cut Q, and a_l is the indicator function specifying whether arc l is in the Steiner tree. This integer program has an integer relaxation obtained by replacing the integer constraint $a_l \in \{0,1\}$ with the linear constraint $0 \leq a_l \leq 1$.

Theorem 2.4 *For a given network* $(\mathcal{N}, \mathcal{A}, s, \mathcal{T})$,

$$\max_{\mathbf{z} \geq 0} \frac{M_c(\mathcal{N}, \mathcal{A}, s, \mathcal{T}, \mathbf{z})}{M_r(\mathcal{N}, \mathcal{A}, s, \mathcal{T}, \mathbf{z})} = \max_{\mathbf{w} \geq 0} \frac{W_{IP}(\mathcal{N}, \mathcal{A}, s, \mathcal{T}, \mathbf{w})}{W_{LP}(\mathcal{N}, \mathcal{A}, s, \mathcal{T}, \mathbf{w})}$$

where $M_c(\mathcal{N}, \mathcal{A}, s, \mathcal{T}, \mathbf{z})$ *and* $M_r(\mathcal{N}, \mathcal{A}, s, \mathcal{T}, \mathbf{z})$ *denote the multicast capacity with and without network coding respectively, under arc capacities* $\mathbf{z}$*, and* $W_{IP}(\mathcal{N}, \mathcal{A}, s, \mathcal{T}, \mathbf{w})$ *and* $W_{LP}(\mathcal{N}, \mathcal{A}, s, \mathcal{T}, \mathbf{w})$ *denote the optimum of the integer program (2.6) and its linear relaxation respectively.*

Determining the maximum value of the integrality gap, $\max_{\mathbf{w} \geq 0} \frac{W_{IP}(\mathcal{N}, \mathcal{A}, s, \mathcal{T}, \mathbf{w})}{W_{LP}(\mathcal{N}, \mathcal{A}, s, \mathcal{T}, \mathbf{w})}$, is a long-standing open problem in computer science. From a known lower bound on this integrality gap, we know that the multicast throughput advantage for coding can be $\Omega((\log n / \log \log n)^2)$ for a network with n sink nodes. For undirected networks, there is a similar correspondence between the maximum multicast throughput advantage for coding and the integrality gap of the bidirected cut relaxation for the undirected Steiner tree problem. Interested readers can refer to [3] for details.

2.4 Multicast network code construction

Next, we address the question: given a solvable multicast network coding problem, how do we construct a solution? Note that here we are considering a single multicast session (i.e. all the sink nodes demand the same information) on a given graph whose arcs have their entire capacity dedicated to supporting that multicast session. When there are multiple sessions sharing a network, one possible approach for intra-session network coding is to first allocate to each session a subset of the network, called a *subgraph*, and then apply the techniques described in this section to construct a code for each session on its allocated subgraph. The issue of subgraph selection is covered in Chapter 5.

2.4.1 Centralized polynomial-time construction

Consider a solvable multicast network coding problem on an acyclic network with r source processes and d sink nodes. The following centralized algorithm constructs, in polynomial time, a solution over a finite field $\mathbb{F}_q$, where $q \geq d$.

The algorithm's main components and ideas are as follows:

- The algorithm first finds r arc-disjoint paths $\mathcal{P}_{t,1}, \ldots, \mathcal{P}_{t,r}$ from the source s to each sink $t \in \mathcal{T}$. Let $\mathcal{A}' \subset \mathcal{A}$ be the set of arcs in the union of these paths. By Theorem 2.2, the subgraph consisting of arcs in $\mathcal{A}'$ suffices to support the desired multicast communication, so the coding coefficients for all other arcs can be set to zero.
- The algorithm sets the coding coefficients of arcs in $\mathcal{A}'$ in topological order, maintaining the following invariant: for each sink t, the coding vectors of the arcs in the set $\mathcal{S}_t$ form a basis for $\mathbb{F}_q^r$, where $\mathcal{S}_t$ comprises the arc from each of the paths $\mathcal{P}_{t,1}, \ldots, \mathcal{P}_{t,r}$ whose coding coefficients were set most recently. The invariant is initially established by adding a virtual source node s' and r virtual arcs from s' to s that have linearly independent coding vectors $[\mathbf{0}^{i-1}, 1, \mathbf{0}^{r-i}], i = 1, \ldots, r$, where $\mathbf{0}^j$ denotes the length-j all zeros row vector. The invariant ensures that at termination, each sink has r linearly independent inputs.
- To facilitate efficient selection of the coding coefficients, the algorithm maintains, for each sink t and arc $l \in \mathcal{S}_t$, a vector $\mathbf{d}_t(l)$ satisfying the condition

$$\mathbf{d}_t(l) \cdot \mathbf{c}_k = \delta_{l,k} \; \forall \, l, k \in \mathcal{S}_t$$

where

$$\delta_{l,k} := \begin{cases} 1 & l = k \\ 0 & l \neq k \end{cases} .$$

By this condition, $\mathbf{d}_t(l)$ is orthogonal to the subspace spanned by the coding vectors of the arcs in $\mathcal{S}_t$ excluding l. Note that a vector $\mathbf{v} \in \mathbb{F}_q^r$ is linearly independent of vectors $\{\mathbf{c}_k : \ k \neq l, k \in \mathcal{S}_t\}$ if and only if $\mathbf{v} \cdot \mathbf{d}_t(l) \neq 0$. This can be seen by expressing $\mathbf{v}$ in the basis corresponding to $\{\mathbf{c}_k : \ k \in \mathcal{S}_t\}$ as $\mathbf{v} = \sum_{k \in \mathcal{S}_t} b_k \mathbf{c}_k$, and noting that

$$\mathbf{v} \cdot \mathbf{d}_t(l) = \sum_{k \in \mathcal{S}_t} b_k \mathbf{c}_k \cdot \mathbf{d}_t(l) = b_l.$$

- For an arc l on a path $\mathcal{P}_{t,i}$, the arc immediately preceding l on $\mathcal{P}_{t,i}$ is denoted $p_t(l)$, and the set of sinks t for which an arc l is in some path $\mathcal{P}_{t,i}$ is denoted $\mathcal{T}(l)$. To satisfy the invariant, the coding coefficients of each arc l are chosen such that the resulting coding vector $\mathbf{c}_l$ is linearly independent of the coding vectors of all arcs in $\mathcal{S}_t \backslash \{p_t(l)\}$ for all $t \in \mathcal{T}(l)$, or equivalently, such that

$$\mathbf{c}_l \cdot \mathbf{d}_t(l) \neq 0 \ \forall \ t \in \mathcal{T}(l). \tag{2.7}$$

This can be done by repeatedly choosing random coding coefficients until the condition (2.7) is met. Alternatively, this can be done deterministically by applying Lemma 2.2 below to the set of vector pairs $\{(\mathbf{c}_{p_t(l)}, \mathbf{d}_t(p_t(l))) : \ t \in \mathcal{T}(l)\}$.

Lemma 2.2 *Let $n \leq q$. For a set of pairs $\{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{F}_q^r \times \mathbb{F}_q^r : \ 1 \leq i \leq n\}$ such that $\mathbf{x} \cdot \mathbf{y}_i \neq 0 \ \forall \ i$, we can, in $O(n^2 r)$ time, find a vector $\mathbf{u}_n$ that is a linear combination of $\mathbf{x}_1, \ldots, \mathbf{x}_n$, such that $\mathbf{u}_n \cdot \mathbf{y}_i \neq 0 \ \forall \ i$.*

Proof This is done by the following inductive procedure, which constructs vectors $\mathbf{u}_1, \ldots, \mathbf{u}_n$ such that $\mathbf{u}_i \cdot \mathbf{y}_l \neq 0 \ \forall \ 1 \leq l \leq i \leq n$. Set $\mathbf{u}_1 := \mathbf{x}_1$. Let $\mathcal{H}$ be a set of n distinct elements of $\mathbb{F}_q$. For $i = 1, \ldots, n-1$, if $\mathbf{u}_i \cdot \mathbf{y}_{i+1} \neq 0$, set $\mathbf{u}_{i+1} := \mathbf{u}_i$; otherwise set $\mathbf{u}_{i+1} := \alpha \mathbf{u}_i + \mathbf{x}_{i+1}$ where α is any element in

$$\mathcal{H} \backslash \left\{ -(\mathbf{x}_{i+1} \cdot \mathbf{y}_l)/(\mathbf{u}_i \cdot \mathbf{y}_l) : \ l \leq i \right\},$$

which is nonempty since $|\mathcal{H}| > i$. This ensures that

$$\mathbf{u}_{i+1} \cdot \mathbf{y}_l = \alpha \mathbf{u}_i \cdot \mathbf{y}_l + \mathbf{x}_{i+1} \cdot \mathbf{y}_l \neq 0 \ \forall \ l \leq i.$$

Each dot product involves length-r vectors and is found in $O(r)$ time,

each $\mathbf{u}_i$ is found in $O(nr)$ time, and $\mathbf{u}_1, \ldots, \mathbf{u}_n$ is found in $O(n^2 r)$ time. □

The full network code construction algorithm is given in Algorithm 1. It is straightforward to verify that $\mathbf{c}_k \cdot \mathbf{d}_t(l) = \delta_{k,l}\ \forall\ k, l \in \mathcal{S}'_t$ at the end

Algorithm 1: Centralized polynomial-time algorithm for multicast linear network code construction

Input: $\mathcal{N}, \mathcal{A}, s, \mathcal{T}, r$
$\mathcal{N} := \mathcal{N} \cup \{s'\}$
$\mathcal{A} := \mathcal{A} \cup \{l_1, \ldots, l_r\}$ where $o(l_i) = s', d(l_i) = s$ for $i = 1, \ldots, r$
Find r arc-disjoint paths $\mathcal{P}_{t,1}, \ldots, \mathcal{P}_{t,r}$ from s' to each sink $t \in \mathcal{T}$;
Choose field size $q = 2^m \geq |\mathcal{T}|$
foreach $i = 1, \ldots, r$ **do** $\mathbf{c}_{l_i} := [\mathbf{0}^{i-1}, 1, \mathbf{0}^{r-i}]$
foreach $t \in \mathcal{T}$ **do**
  $\mathcal{S}_t := \{l_1, \ldots, l_r\}$
  foreach $l \in \mathcal{S}_t$ **do** $\mathbf{d}_t(l) := \mathbf{c}_l$
foreach $k \in \mathcal{A} \backslash \{l_1, \ldots, l_r\}$ *in topological order* **do**
  choose, by repeated random trials or by the procedure of Lemma 2.2, $\mathbf{c}_k = \sum_{k' \in \mathcal{P}(k)} f_{k',k} \mathbf{c}_{k'}$ such that $\mathbf{c}_k$ is linearly independent of $\{\mathbf{c}_{k'} :\ k' \in \mathcal{S}_t, k' \neq p_t(k)\}$ for each $t \in \mathcal{T}(k)$
  foreach $t \in \mathcal{T}(k)$ **do**
    $\mathcal{S}'_t := \{k\} \cup \mathcal{S}_t \backslash p_t(k)$
    $\mathbf{d}'_t(k) := (\mathbf{c}_k \cdot \mathbf{d}_t(p_t(k)))^{-1} \mathbf{d}_t(p_t(k))$
    foreach $k' \in \mathcal{S}_t \backslash p_t(k)$ **do**
A $\mathbf{d}'_t(k') := \mathbf{d}_t(k') - (\mathbf{c}_k \cdot \mathbf{d}_t(k')) \mathbf{d}'_t(k)$
    $(\mathcal{S}_t, \mathbf{d}_t) := (\mathcal{S}'_t, \mathbf{d}'_t)$
return f

of step A.

Theorem 2.5 *For a solvable multicast network coding problem on an acyclic network with r source processes and d sink nodes, Algorithm 1 deterministically constructs a solution in $O(|\mathcal{A}|\, dr(r+d))$ time.*

Proof The full proof is given in [72]. □

Corollary 2.2 *A finite field of size $q \geq d$ is sufficient for a multicast network coding problem with d sink nodes on an acyclic network.*

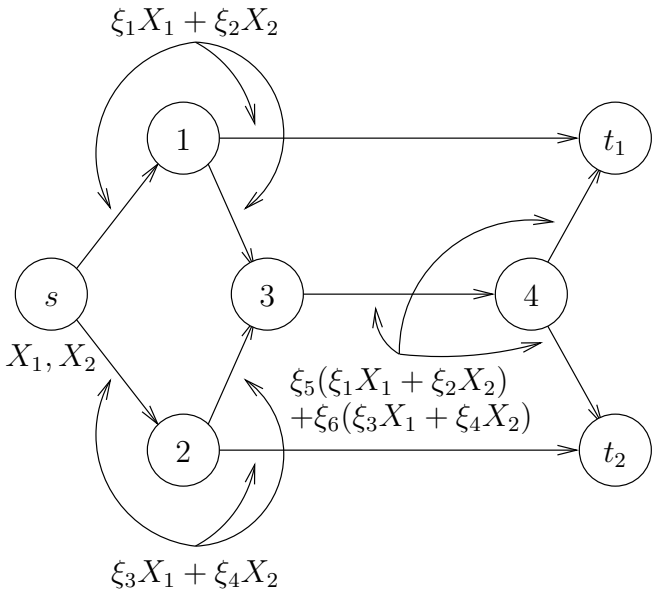


Fig. 2.1. An example of random linear network coding. X_1 and X_2 are the source processes being multicast to the receivers, and the coefficients ξ_i are randomly chosen elements of a finite field. The label on each arc represents the process being transmitted on the arc. Reprinted with permission from [56].

For the case of two source processes, a tighter bound of $q \geq \sqrt{2d - 7/4} + 1/2$ is shown in [46] using a coloring approach.

2.4.2 Random linear network coding

A simple approach that finds a solution with high probability is to choose coding coefficients $(\mathbf{a}, \mathbf{f})$ independently at random from a sufficiently large finite field. The value of $(\mathbf{a}, \mathbf{f})$ determines the value of the coding vector Y_l for each network arc l (which equals the lth column of $\mathbf{C} = \mathbf{A}(\mathbf{I} - \mathbf{F})^{-1}$).

It is not always necessary to do random coding on every arc. For instance, in our lossless network model, a node with a single input can employ simple forwarding, as in Figure 2.1. Or if we have found r disjoint paths from the source to each sink as in the algorithm of the previous section, we can restrict coding to occur only on arcs where two or more paths to different sinks merge. We will bound the probability

that random network coding on η arcs yields a solution to a feasible multicast problem.

Recall from the proof of Theorem 2.2 that for a solvable multicast problem, the product of transfer matrix determinants $\prod_{t\in\mathcal{T}} \det(\mathbf{A}(\mathbf{I}-\mathbf{F})^{-1}\mathbf{B}_t^T)$ is nonzero over the ring of polynomials $\mathbb{F}_2[\mathbf{a},\mathbf{f},\mathbf{b}]$. Since the only nonzero rows of $\mathbf{B}_t^T$ are those corresponding to input arcs of sink t, $\mathbf{A}(\mathbf{I}-\mathbf{F})^{-1}\mathbf{B}_t^T$ is nonsingular only if t has a set $\mathcal{I}_t \subset \mathcal{I}(t)$ of r input arcs with linearly independent coding vectors, or equivalently, the submatrix $\mathbf{C}_{\mathcal{I}_t}$ formed by the r columns of $\mathbf{C}$ corresponding to $\mathcal{I}_t$ is nonsingular. Then each sink t can decode by setting the corresponding submatrix of $\mathbf{B}_t$ (whose columns correspond to arcs in $\mathcal{I}_t$) to $\mathbf{C}_{\mathcal{I}_t}^{-1}$, which gives $\mathbf{M}_t = \mathbf{I}$.

To obtain a lower bound on the probability that random coding yields a solution, we assume that for each sink t the set $\mathcal{I}_t$ is fixed in advance and other inputs, if any, are not used for decoding. A solution then corresponds to a value for $(\mathbf{a},\mathbf{f})$ such that

$$\psi(\mathbf{a},\mathbf{f}) = \prod_{t\in\mathcal{T}} \det\ \mathbf{C}_{\mathcal{I}_t} \tag{2.8}$$

is nonzero. The Schwartz-Zippel theorem (e.g., [105]) states that for any nonzero polynomial in $\mathbb{F}_2[x_1,\ldots,x_n]$, choosing the values of variables $x_1,\ldots,x_n$ independently and uniformly at random from $\mathbb{F}_{2^m}$ results in a nonzero value for the polynomial with probability at least $1-d/2^m$, where d is the total degree of the polynomial. To apply this theorem to the polynomial $\psi(\mathbf{a},\mathbf{f})$, we need a bound on its total degree; we can obtain a tighter bound by additionally bounding the degree of each variable.

These degree bounds can be obtained from the next lemma, which expresses the determinant of the transfer matrix $\mathbf{M}_t = \mathbf{A}(\mathbf{I}-\mathbf{F})^{-1}\mathbf{B}_t^T$ in a more transparent form, in terms of the related matrix

$$\mathbf{N}_t = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{I}-\mathbf{F} & \mathbf{B}_t^T \end{bmatrix}. \tag{2.9}$$

Lemma 2.3 *For an acyclic delay-free network, the determinant of the transfer matrix* $\mathbf{M}_t = \mathbf{A}(\mathbf{I}-\mathbf{F})^{-1}\mathbf{B}_t^T$ *for receiver* t *is equal to*

$$\det \mathbf{M}_t = (-1)^{r(|\mathcal{A}|+1)} \det \mathbf{N}_t.$$

Proof Note that

$$\left[\begin{array}{cc} \mathbf{I} & -\mathbf{A}(\mathbf{I}-\mathbf{F})^{-1} \\ \mathbf{0} & \mathbf{I} \end{array}\right]\left[\begin{array}{cc} \mathbf{A} & \mathbf{0} \\ \mathbf{I}-\mathbf{F} & \mathbf{B}_t^T \end{array}\right]=\left[\begin{array}{cc} \mathbf{0} & -\mathbf{A}(\mathbf{I}-\mathbf{F})^{-1}\mathbf{B}_t^T \\ \mathbf{I}-\mathbf{F} & \mathbf{B}_t^T \end{array}\right]$$

Since $\left[\begin{array}{cc} \mathbf{I} & -\mathbf{A}(\mathbf{I}-\mathbf{F})^{-1} \\ \mathbf{0} & \mathbf{I} \end{array}\right]$ has determinant 1,

$$\begin{aligned}
\det\left(\left[\begin{array}{cc} \mathbf{A} & \mathbf{0} \\ \mathbf{I}-\mathbf{F} & \mathbf{B}_t^T \end{array}\right]\right) &= \det\left(\left[\begin{array}{cc} \mathbf{0} & -\mathbf{A}(\mathbf{I}-\mathbf{F})^{-1}\mathbf{B}_t^T \\ \mathbf{I}-\mathbf{F} & \mathbf{B}_t^T \end{array}\right]\right) \\
&= (-1)^{r|\mathcal{A}|}\det\left(\left[\begin{array}{cc} -\mathbf{A}(\mathbf{I}-\mathbf{F})^{-1}\mathbf{B}_t^T & \mathbf{0} \\ \mathbf{B}_t^T & \mathbf{I}-\mathbf{F} \end{array}\right]\right) \\
&= (-1)^{r|\mathcal{A}|}\det(-\mathbf{A}(\mathbf{I}-\mathbf{F})^{-1}\mathbf{B}_t^T)\det(\mathbf{I}-\mathbf{F}) \\
&= (-1)^{r(|\mathcal{A}|+1)}\det(\mathbf{A}(\mathbf{I}-\mathbf{F})^{-1}\mathbf{B}_t^T)\det(\mathbf{I}-\mathbf{F})
\end{aligned}$$

The result follows from observing that $\det(\mathbf{I}-\mathbf{F})=1$ since $\mathbf{F}$ is upper-triangular with zeros along the main diagonal. □

This lemma can be viewed as a generalization of a classical result linking (uncoded) network flow and bipartite matching. The problem of checking the feasibility of an $s-t$ flow of size r on graph $\mathcal{G}=(\mathcal{N},\mathcal{A})$ can be reduced to a bipartite matching problem by constructing the following bipartite graph: one node set of the bipartite graph has r nodes $u_1,\ldots,u_r$, and a node $v_{l,1}$ corresponding to each arc $l\in\mathcal{A}$; the other node set of the bipartite graph has r nodes $w_1,\ldots,w_r$, and a node $v_{l,2}$ corresponding to each arc $l\in\mathcal{A}$. The bipartite graph has

- an arc joining each node u_i to each node $v_{l,1}$ such that $o(l)=s$ (corresponding to an output link of source s)
- an arc joining node $v_{l,1}$ to the corresponding node $v_{l,2}$ for all $l\in\mathcal{A}$,
- an arc joining node $v_{l,2}$ to $v_{j,1}$ for each pair $(l,j)\in\mathcal{A}\times\mathcal{A}$ such that $d(l)=o(j)$ (corresponding to incident links), and
- an arc joining each node w_i to each node $v_{l,2}$ such that $d(l)=t$ (corresponding to input links of sink t.

The $s-t$ flow is feasible if and only if the bipartite graph has a perfect matching. The matrix $\mathbf{N}_t$ defined in Equation (equation:N) can be viewed as a network coding generalization of the Edmonds matrix (see e.g., [105]) used for checking if the bipartite graph has a perfect matching.

Since each coding coefficient appears in only one entry of $\mathbf{N}_t$, we can

easily obtain degree bounds for det $\mathbf{N}_t$ using the complete expansion of the determinant, as shown in the following lemma.

Lemma 2.4 *Consider a random network code in which η arcs l have associated coding coefficients $a_{i,l}$ and/or $f_{k,l}$ that are randomly chosen. The determinant of $\mathbf{N}_t$ has maximum degree η in the random variables $\{a_{i,l}, f_{k,l}\}$, and is linear in each of these variables.*

Proof Note that the variables $a_{i,l}, f_{k,l}$ corresponding to an arc l each appear once, in column l of $\mathbf{N}_t$. Thus, only the η columns corresponding to arcs with associated random coefficients contain variable terms. The determinant of $\mathbf{N}_t$ can be written as the sum of products of $r + |\mathcal{A}|$ entries, each from a different column (and row). Each such product is linear in each variable $a_{i,l}, f_{k,l}$, and has degree at most η in these variables. □

Noting that det $\mathbf{C}_{\mathcal{I}_t}$ equals det $\mathbf{M}_t$ for some $\mathbf{b}_t$†, and using Lemmas 2.3 and 2.4, we have that $\psi(\mathbf{a}, \mathbf{f})$ (defined in (2.8)) has total degree at most $d\eta$ in the randomly chosen coding coefficients and each coding coefficient has degree at most d, where η is the number of arcs with randomly chosen coding coefficients and d is the number of sink nodes.

Theorem 2.6 *Consider a multicast problem with d sink nodes, and a network code in which some or all of the coding coefficients $(\mathbf{a}, \mathbf{f})$ are chosen uniformly at random from a finite field $\mathbb{F}_q$ where $q > d$, and the remaining coding coefficients, if any, are fixed. If there exists a solution to the multicast problem with these fixed coding coefficients, then the probability that the random network code yields a solution is at least $(1 - d/q)^\eta$, where η is the number of arcs l with associated random coding coefficients $a_{i,l}, f_{k,l}$.*

Proof See Appendix 2.A. □

The bound of Theorem 2.6 is a worst-case bound applying across all networks with d sink nodes and η links with associated random coding coefficients. For many networks, the actual probability of obtaining a solution is much higher. Tighter bounds can be obtained by considering additional aspects of network structure. For example, having more redundant capacity in the network increases the probability that a random linear code will be valid.

† when $\mathbf{B}_t$ is the identity mapping from arcs in $\mathcal{I}_t$ to outputs at t

In the rest of this chapter, we will extend our basic network model and lossless multicast problem in a few ways: from static source and arc processes to time-varying packet networks, from acyclic networks to networks with cycles, and from independent to correlated source processes.

2.5 Packet networks

The algebraic description of scalar linear network coding in Section 2.2, developed for the idealized static network model of Section 2.1, is readily adapted to the case of transmission of a finite batch (generation) of packets over a time-varying network where each packet can potentially undergo different routing and coding operations.

Let the source message be composed of a batch of r exogenous source packets. A packet transmitted by a node v is formed as a linear combination of one or more constituent packets, which may be source packets originating at v or packets received previously by v. For a multicast problem, the objective is to reproduce, at each sink, the r source packets.

For scalar linear network coding in a field $\mathbb{F}_q$, the bits in each packet are grouped into vectors of length m which are viewed as symbols from $\mathbb{F}_q$, $q = 2^m$. We thus consider each packet as a vector of symbols from $\mathbb{F}_q$; we refer to such a vector as a *packet vector*.

We can think of source packets and transmitted packets as analogous to source processes and arc processes respectively in the static network model. The kth symbol of a transmitted packet is a scalar linear function of the kth symbol of each of its constituent packets, and this function is the same for all k. This is analogous to the formation of an arc process Y_l as a linear combination of one or more of the input processes of node $o(l)$ in the static model.

For a given sequence $\mathcal{S}$ of packet transmissions, we can consider a corresponding static network $\mathcal{G}$ with the same node set and with arcs corresponding to transmissions in $\mathcal{S}$, where for each packet p transmitted from node v to w in $\mathcal{S}$, $\mathcal{G}$ has one unit-capacity arc $\tilde{p}$ from v to w. The causality condition that each packet p transmitted by a node v is a linear combination of only those packets received by v earlier in the sequence translates into a corresponding restriction in $\mathcal{G}$ on the subset of v's inputs that can be inputs of $\tilde{p}$. This departs from our previous assumption in Section 2.2 that each of node v's inputs is an input of each of v's outgoing arcs. The restriction that arc k is not an input of arc l is equivalent to setting the coding coefficient $f_{k,l}$ to zero. Such restrictions

are conveniently specified using a *line graph*: the line graph $\mathcal{G}'$ of $\mathcal{G}$ has one node w_l for every arc l of $\mathcal{G}$, and contains the arc (w_k, w_l) if w_k is an input of w_l.

2.5.1 Distributed random linear coding for packet networks

2.5.1.1 Coding vector approach

The random linear network coding approach of Section 2.4.2 can form the basis of a practical, distributed multicast technique for time-varying packet networks. Applying this approach to the packet network model, each packet transmitted by a node v is an independent random linear combination of previously received packets and source packets generated at v. The coefficients of these linear combinations are chosen with the uniform distribution from the finite field $\mathbb{F}_q$, and the same linear operation is applied to each symbol in a packet.

In a distributed setting, network nodes independently choose random coding coefficients, which determine the network code and the corresponding decoding functions at the sinks. Fortunately, a sink node does not need to know all these coefficients in order to know what decoding function to use. It suffices for the sink to know the overall linear transformation from the source packets to the packets it has received. As in the static model, the overall linear transformation from the source packets to a packet p is called the (global) coding vector of p.

There is a convenient way to convey this information to the sinks, which is analogous to using a pilot tone or finding an impulse response. For a batch of source packets with indexes $i = 1, 2, \ldots, r$, we add to the header of the ith source packet its coding vector, which is the length r unit vector $[0 \ldots 0 \ 1 \ 0 \ldots 0]$ with a single nonzero entry in the ith position. For each packet formed subsequently by a coding operation, the same coding operation is applied to each symbol of the coding vector as to the data symbols of the packet. Thus, each packet's header contains the coding vector of that packet.

A sink node can decode the whole batch when it has received r linearly independent packets. Their coding vectors form the rows of the transfer matrix from the source packets to the received packets. The transformation corresponding to the inverse of this matrix can be applied to the received packets to recover the original source packets. Decoding can alternatively be done incrementally using Gaussian elimination.

Note that each coding vector is $r \log q$ bits long, where q is the coding

field size. The proportional overhead of including the coding vector in each packet decreases with the amount of data in each packet, so for large packets this overhead is relatively small. For small packets, this overhead can be reduced by decreasing the field size q or batch size r (by dividing a large batch of source packets into smaller batches, and only allowing coding among packets of the same batch). Decreasing q and r also reduces decoding complexity. However, the smaller the field size the higher the probability that more transmissions are required, since there is higher probability of randomly picking linearly dependent transmissions. Also, reducing batch size reduces our ability to code across bursty variations in source rates or arc capacities, resulting in reduced throughput if packets near the batch boundaries have to be transmitted without coding. An illustration is given in Figure 2.2. In this example, a source node s is multicasting to sink nodes y and z. All the arcs have average capacity 1, except for the four labeled arcs which have average capacity 2. In the optimal solution, arc (w, x) should transmit coded information for both receivers at every opportunity. However, variability in the instantaneous capacities of arcs (u, w), (u, y), (t, w) and (t, z) can cause the number of sink y packets in a batch arriving at node w to differ from the number of sink z packets of that batch arriving at w, resulting in some throughput loss.

Because of such trade-offs, appropriate values for q and r will depend on the type of network and application. The effect of these parameter choices, and performance in general, are also dependent on the choice of subgraph (transmission opportunities) for each batch.† The effects of such parameters are investigated by Chou et al. [26] under a particular distributed policy for determining when a node switches to sending packets of the next batch.

2.5.1.2 Vector space approach

A more general approach for distributed random linear coding encodes a batch of information in the choice of the vector space spanned by the source packet vectors.

Specifically, let $\mathbf{x}_1, \ldots, \mathbf{x}_r$ denote the source packet vectors, which are length-n row vectors of symbols from $\mathbb{F}_q$. We denote by $\mathbf{X}$ the $r \times n$ matrix whose ith row is $\mathbf{x}_i$. Consider a sink node t, and let $\mathbf{Y}_t$ be the matrix whose rows are given by t's received packet vectors. $\mathbf{X}$ and $\mathbf{Y}_t$

† Subgraph selection is the topic of Chapter 5.

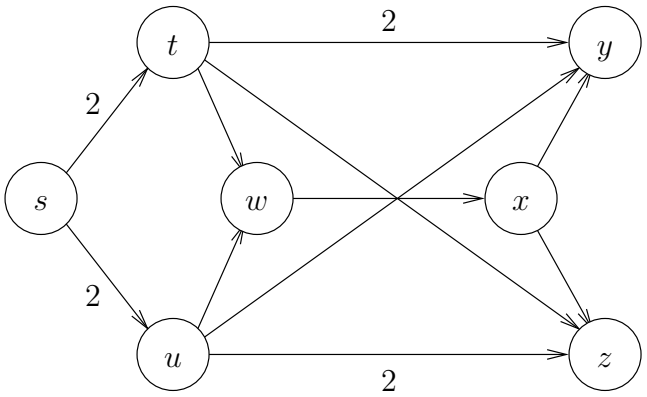


Fig. 2.2. An example illustrating throughput loss caused by restricting coding to occur only among packets of a batch. Reprinted with permission from [58].

are linearly related by a matrix equation

$$\mathbf{Y} = \mathbf{G}_t \mathbf{X} ‡$$

In a random network code, $\mathbf{G}_t$ is determined by the random coding coefficients of network nodes. The vector space approach is based on the observation that for any value of $\mathbf{G}_t$, the row space of $\mathbf{Y}$ is a subspace of the row space of $\mathbf{X}$. If the sink receives r linearly independent packets, it recovers the row space of $\mathbf{X}$.

Let $\mathcal{P}(\mathbb{F}_q^n)$ denote the set of all subspaces of $\mathbb{F}_q^n$, i.e. the projective geometry of $\mathbb{F}_q^n$. In this approach, a code corresponds to a nonempty subset of $\mathcal{P}(\mathbb{F}_q^n)$, and each codeword is a subspace of $\mathbb{F}_q^n$. A codeword is transmitted as a batch of packets; the packet vectors of the source packets in the batch form a generating set for the corresponding subspace or its orthogonal complement. It is natural to consider codes whose codewords all have the same dimension r.† Note that the coding vector approach of the previous section is a special case where the code consists of all subspaces with generator matrices of the form $[\mathbf{U}|\mathbf{I}]$, where $\mathbf{U} \in \mathbb{F}_q^{r\times(n-r)}$ and $\mathbf{I}$ is the $r \times r$ identity matrix (corresponding to the coding vectors). Since only a subset of all r-dimensional subspaces of $\mathbb{F}_q^n$ correspond to codewords in the coding vector approach, the number of

‡ $\mathbf{G}_t$ is analogous to the transpose of matrix $\mathbf{C}_{\mathcal{I}(t)}$ defined in Section 2.4.2 for the static network model.

† Such codes can be described as particular vertices of a Grassmann graph/q-Johnson scheme. Details are given in [83].

codewords and hence the code rate is lower than in the vector space approach, though the difference becomes asymptotically negligible as the packet length n grows relative to the batch size r.

An important motivation for the vector space approach is its application to correction of errors and erasures in networks. We discuss this briefly in Section 6.2.2.2.

2.6 Networks with cycles and convolutional network coding

In our basic network model, which is acyclic, a simple delay-free network coding approach (ref Section 2.2) can be used. Many networks of interest contain cycles, but can be operated in a delay-free fashion by imposing restrictions on the network coding subgraph to prevent cyclic dependencies among arcs. For instance, we can restrict network coding to occur over an acyclic subgraph of the network line graph (defined in Section 2.5). Another type of restriction is temporal, as in the finite batch packet model of the previous section: if we index the transmitted packets according to their creation time, each transmitted packet has a higher index than the constituent packets that formed it, so there are no cyclic dependencies among packets. This can be viewed conceptually as expanding the network in time. In general, a cyclic graph with v nodes and rate r can be converted to a time-expanded acyclic graph with κv nodes and rate at least $(\kappa - v)r$; communication on this expanded graph can be emulated in κ time steps on the original cyclic graph.

For some network problems, such as those in Figure 2.3, the optimal rate cannot be achieved over any acyclic subgraph of the network line graph. In this example, to multicast both sources simultaneously to both sinks, information must be continuously injected into the cycle (the square in the middle of the network) from both sources. Converting the network to a time-expanded acyclic graph gives a time-varying solution that asymptotically achieves the optimal rate, but at the expense of increasing delay and decoding complexity. An alternative for such as networks is to take an approach akin to convolutional coding, where delays are explicitly considered, and information from different time steps is linearly combined. This approach, termed *convolutional network coding*, enables the optimal rate to be achieved with a time-invariant solution.

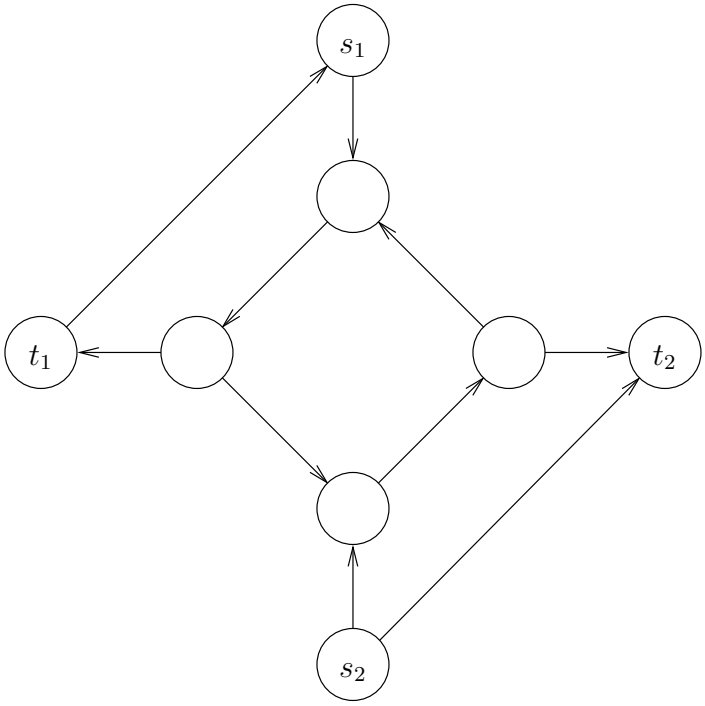


Fig. 2.3. An example of a multicast problem in which the optimal rate cannot be achieved over any acyclic subgraph of the network line graph. Each arc has a constant rate of one packet per unit time. Reprinted with permission from [64].

2.6.1 Algebraic representation of convolutional network coding

Convolutional network codes can be cast in a mathematical framework similar to that of Section 2.2 for delay-free scalar network codes, by representing the random processes algebraically in terms of a delay operator variable D which represents a unit time delay or shift: if

$$\begin{aligned} X_i(D) &= \sum_{\tau=0}^{\infty} X_i(\tau) D^\tau \\ Y_l(D) &= \sum_{\tau=0}^{\infty} Y_\tau(l) D^\tau, \qquad Y_l(0) = 0 \\ Z_{t,i}(D) &= \sum_{\tau=0}^{\infty} Z_{t,i}(\tau) D^\tau, \qquad Z_{t,i}(0) = 0. \end{aligned}$$

The results for delay-free scalar linear network coding carry over to this model by replacing the finite field $\mathbb{F}_q$ with the field $\mathbb{F}_q(D)$ of rational functions in the delay variable D. Analogously to the delay-free case, the transfer matrix from source processes to sink output processes can be calculated as the matrix product

$$\mathbf{M}_t = \mathbf{A}(D)(\mathbf{I} - \mathbf{F}(D))^{-1}\mathbf{B}_t(D)^T$$

where $\mathbf{A}(D) = (a_{i,l}(D)), \mathbf{F}(D) = (f_{k,l}(D)), \mathbf{B}_t(D) = (b_{t,i,k}(D))$ are matrices whose entries are elements of $\mathbb{F}_q(D)$.

A simple type of convolutional network code uses a network model where each arc has fixed unit delay; arcs with longer delay can be modeled as arcs in series. At time $\tau + 1$, each non-sink node v receives symbols $Y_k(\tau)$ on its input arcs k and/or source symbols $X_i(\tau)$ if $v = s_i$, and linearly combines them to form symbols $Y_l(\tau+1)$ on its output arcs l. The corresponding coding operation at an arc l at time τ is similar to Equation 2.4 but with time delay:

$$\begin{aligned} Y_l(\tau+1) &= \sum_{\{i\,:\,s_i = o(l)\}} a_{i,l} X_i(\tau) \\ &\quad + \sum_{\{k\,:\,d(k) = o(l)\}} f_{k,l} Y_k(\tau) \end{aligned}$$

which can be expressed in terms of D as

$$\begin{aligned} Y_l(D) &= \sum_{\{i\,:\,s_i = o(l)\}} D a_{i,l} X_i(D) \\ &\quad + \sum_{\{k\,:\,d(k) = o(l)\}} D f_{k,l} Y_k(D) \end{aligned}$$

In this case, the coding coefficients at non-sink nodes are given by $a_{i,l}(D) = Da_{i,l}, f_{k,l}(D) = Df_{k,l}$. By considering $D = 0$, we can see that the matrix $\mathbf{I} - \mathbf{F}(D)$ is invertible. In a synchronous setting, this does not require memory at non-sink nodes (though in practical settings where arc delays are variable, some buffering is needed since the $(\tau+1)$st symbol of each output arc is transmitted only after reception of the τth symbol of each input). The sink nodes, on the other hand, require memory: the decoding coefficients $b_{t,i,k}(D)$ are, in general, rational functions of D, which corresponds to the use of past received and decoded symbols

for decoding. The corresponding equations are

$$\begin{aligned} Z_{t,i}(\tau+1) &= \sum_{u=0}^{\mu} b'_{t,i}(u) Z_{t,i}(\tau-u) \\ &\quad + \sum_{\{k\,:\,d(k)=t\}} \sum_{u=0}^{\mu} b''_{t,i,k}(u) Y_k(\tau-u) \end{aligned}$$

and

$$Z_{t,i}(D) = \sum_{\{k\,:\,d(k)=t\}} b_{t,i,k}(D) Y_k(D),$$

where

$$b_{t,i,k}(D) = \frac{\sum_{u=0}^{\mu} D^{u+1} b''_{t,i,k}(u)}{1 - \sum_{u=0}^{\mu} D^{u+1} b'_{t,i}(u)}. \tag{2.10}$$

The amount of memory required, μ, depends on the structure of the network. A rational function is *realizable* if it is defined when $D = 0$†, and a matrix of rational entries is realizable if all its entries are realizable. By similar arguments as in the acyclic delay-free case, we can extend Theorem 2.2 to the case with cycles.

Theorem 2.7 *Consider a multicast problem where r source processes originating at source node s are demanded by a set $\mathcal{T}$ of sink nodes. There exists a solution if and only if for each sink node $t \in \mathcal{T}$ there exists a flow of rate r between s and t.*

Proof The proof is similar to that of Theorem 2.2, but with a change of field. Consider the simple unit arc delay model and network code. We have the following equivalent conditions:

$\forall t \in \mathcal{T}$ there exists a flow of rate r between s and t
$\Leftrightarrow \forall t \in \mathcal{T}$ the transfer matrix determinant $\det\ \mathbf{M}_t$ is a nonzero ratio of polynomials from the ring $\mathbb{F}_2(D)[\mathbf{a}, \mathbf{f}, \mathbf{b}', \mathbf{b}'']$
$\Leftrightarrow \prod_{t\in\mathcal{T}} \det\ \mathbf{M}_t$ is a nonzero ratio of polynomials from the ring $\mathbb{F}_2(D)[\mathbf{a}, \mathbf{f}, \mathbf{b}', \mathbf{b}'']$
$\Leftrightarrow$ there exists a value for $(\mathbf{a}, \mathbf{f}, \mathbf{b}', \mathbf{b}'')$ over $\mathbb{F}_{2^m}$, for sufficiently large m, such that $\prod_{t\in\mathcal{T}} \det\ \mathbf{M}_t$ is nonzero in $\mathbb{F}_{2^m}(D)$
$\Leftrightarrow$ there exist realizable matrices $\mathbf{B}_t(D)$ such that $\mathbf{M}_t = D^u \mathbf{I}\ \forall t \in \mathcal{T}$ for some sufficiently large decoding delay u.

† i.e. For a realizable rational function, the denominator polynomial in lowest terms has a nonzero constant coefficient

□

More generally, it is not necessary to consider delays on every arc. To ensure stability and causality of information flow, we only need every directed cycle in the network to contain at least one delay element. Furthermore, the delays can be associated with nodes instead of links: an alternative model for convolutional network coding considers delay-free links and associates all delays with coding coefficients at network nodes. With this model, we can work in the binary field $\mathbb{F}_2$; having delay or memory at nodes corresponds to coding coefficients that are polynomials in $\mathbb{F}_2[D]$. For acyclic networks, such codes can be constructed in polynomial time using an approach analogous to that in Section 2.4.1, where the invariant becomes: for each sink t, the coding vectors of the arcs in the set $\mathcal{S}_t$ span $\mathbb{F}_2[D]^r$. For each arc, the coding coefficients can be chosen from a set of $d+1$ values, where d is the number of sinks. The block network codes we have considered in previous sections achieve capacity for acyclic networks, but in some cases convolutional network codes can have lower delay and memory requirements. One reason is that for block codes each coding coefficient is from the same field, whereas for convolutional network codes the amount of delay/memory can be different across coding coefficients. The case of cyclic networks is more complicated since there is no well-defined topological order in which to set the coding coefficients. An algorithm described in [**?**] updates the coding coefficients associated with each sink's subgraph in turn (each sink's subgraph consists of r arc-disjoint paths and the associated coding coefficients can be updated in topological order).

2.7 Correlated source processes

In our basic network model, the source processes are independent. In this section we consider correlated, or jointly distributed, source processes. Such correlation can be exploited to improve transmission efficiency.

The problem of lossless multicasting from correlated sources is a generalization of the classical distributed source coding problem of Slepian and Wolf, where correlated sources are separately encoded but jointly decoded. The classical Slepian-Wolf problem corresponds to the special case of a network consisting of one direct arc from each of two source nodes to a common receiver. In the network version of the problem, the sources are multicast over a network of intermediate nodes that can perform network coding. It turns out that a form of random linear network

coding with nonlinear decoding is asymptotically rate-optimal. This can be viewed as a network coding generalization of a classical random linear coding approach for the Slepian-Wolf problem, and as an instance of joint source-network coding.

2.7.1 Joint source-network coding

For simplicity, we consider two sources† X_1, X_2 with respective rates r_1 and r_2 bits per unit time. The source bits at X_i are grouped into vectors of r_i bits which we refer to as symbols. The two sources' consecutive pairs of output symbols are drawn i.i.d. from the same joint distribution Q.

We employ a vector linear network code that operates on blocks of bits corresponding to n symbols from each source. Specifically, linear coding is done in $\mathbb{F}_2$ over blocks consisting of nr_i bits from each source X_i.‡ Let c_k be the capacity of arc k. For each block, each node v transmits, on each of its output arcs k, nc_k bits formed as random linear combinations of input bits (source bits originating at v and bits received on input arcs). This is illustrated in Figure 2.4. $\mathbf{x}_1 \in \mathbb{F}_2^{nr_1}$ and $\mathbf{x}_2 \in \mathbb{F}_2^{nr_2}$ are vectors of source bits being multicast to the receivers, and the matrices Υ_i are matrices of random bits. Suppose the capacity of each arc is c. Matrices Υ_1 and Υ_3 are $nr_1 \times nc$, Υ_2 and Υ_4 are $nr_2 \times nc$, and Υ_5 and Υ_6 are $nc \times nc$.

To decode, sink maps its block of received bits to a block of decoded values that has minimum entropy or maximum Q-probability among all possible source values consistent with the received values.

We give a lower bound on the probability of decoding error at a sink for Let m_1 and m_2 be the minimum cut capacities between the receiver and each of the sources respectively, and let m_3 be the minimum cut capacity between the receiver and both sources. We denote by L the maximum source-receiver path length. The type $P_{\mathbf{x}}$ of a vector $\mathbf{x} \in \mathbb{F}_2^{\tilde{n}}$ is the distribution on $\mathbb{F}_2$ defined by the relative frequencies of the elements of $\mathbb{F}_2$ in $\mathbf{x}$, and joint types $P_{\mathbf{xy}}$ are analogously defined.

Theorem 2.8 *The error probability of the random linear network code*

† We use the term "source" in place of "source process" for brevity.
‡ The approach can be extended to coding over larger finite fields.

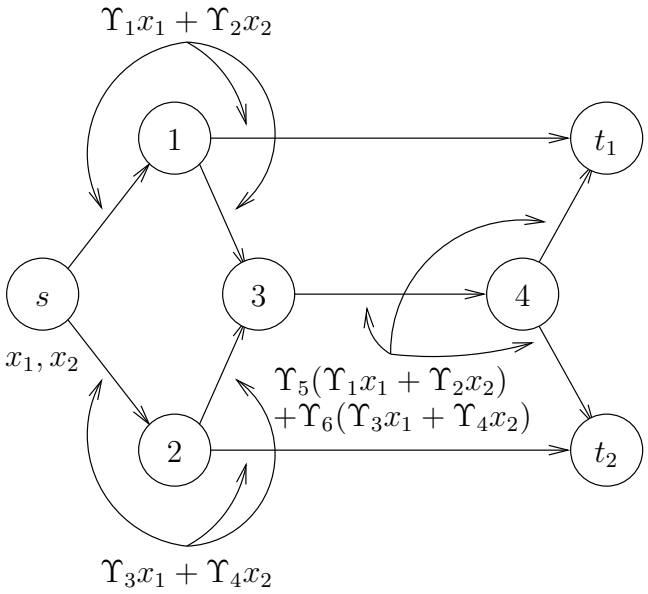


Fig. 2.4. An example illustrating vector linear coding. The label on each arc represents the process being transmitted on the arc. Reprinted with permission from [56].

is at most $\sum_{i=1}^{3} p_e^i$, *where*

$$
\begin{aligned}
p_e^1 \;\le\; \exp\Bigg\{ & -n \min_{X,Y} \Bigg(D(P_{XY}||Q) \\
& + \Big| m_1(1 - \frac{1}{n}\log L) - H(X|Y) \Big|^+ \Bigg) \\
& + 2^{2r_1 + r_2} \log(n+1) \Bigg\} \\
p_e^2 \;\le\; \exp\Bigg\{ & -n \min_{X,Y} \Bigg(D(P_{XY}||Q) \\
& + \Big| m_2(1 - \frac{1}{n}\log L) - H(Y|X) \Big|^+ \Bigg) \\
& + 2^{r_1 + 2r_2} \log(n+1) \Bigg\} \\
p_e^3 \;\le\; \exp\Bigg\{ & -n \min_{X,Y} \Bigg(D(P_{XY}||Q) \\
& + \Big| m_3(1 - \frac{1}{n}\log L) - H(XY) \Big|^+ \Bigg) \\
& + 2^{2r_1 + 2r_2} \log(n+1) \Bigg\}
\end{aligned}
$$

and X, Y are dummy random variables with joint distribution P_{XY}.

Proof See Appendix 2.A. □

The error exponents

$$\begin{aligned} e^1 &= \min_{X,Y} \Bigg(D(P_{XY}||Q) \\ &\quad + \left| m_1(1 - \frac{1}{n}\log L) - H(X|Y) \right|^+ \Bigg) \\ e^2 &= \min_{X,Y} \Bigg(D(P_{XY}||Q) \\ &\quad + \left| m_2(1 - \frac{1}{n}\log L) - H(Y|X) \right|^+ \Bigg) \\ e^3 &= \min_{X,Y} \Bigg(D(P_{XY}||Q) \\ &\quad + \left| m_3(1 - \frac{1}{n}\log L) - H(XY) \right|^+ \Bigg), \end{aligned}$$

for general networks reduce to those for the Slepian-Wolf network where $L = 1, m_1 = R_1, m_2 = R_2, m_3 = R_1 + R_2$:

$$\begin{aligned} e^1 &= \min_{X,Y} \left(D(P_{XY}||Q) + |R_1 - H(X|Y)|^+ \right) \\ e^2 &= \min_{X,Y} \left(D(P_{XY}||Q) + |R_2 - H(Y|X)|^+ \right) \\ e^3 &= \min_{X,Y} \left(D(P_{XY}||Q) + |R_1 + R_2 - H(XY)|^+ \right). \end{aligned}$$

2.7.2 Separation of source coding and network coding

A separate source coding and network coding scheme first performs source coding to describe each source as a compressed collection of bits, and then uses network coding to losslessly transmit to each sink a subset of the resulting bits. Each sink first decodes the network code to recover the transmitted bits, and then decodes the source code to recover the original sources. Such an approach allows use of existing low complexity source codes. However, separate source and network coding is in general not optimal. This is shown by an example in Figure 2.5 which is formally presented in [114]; we give here an informal description providing

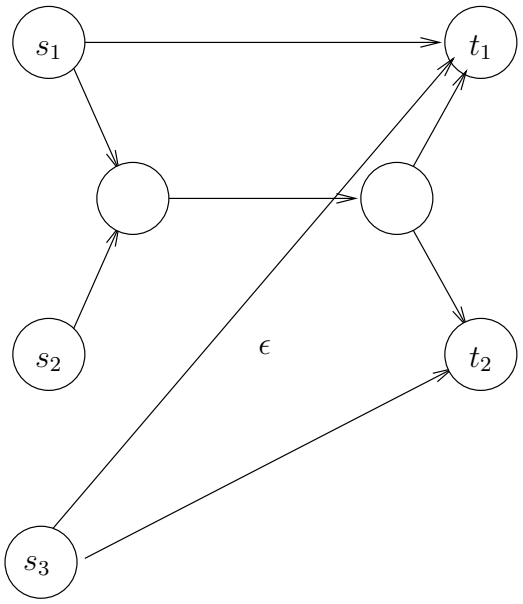


Fig. 2.5. An example in which separate source coding and network coding is suboptimal. Each arc has capacity $1 + \epsilon$, except the arc from s_3 to t_1 which has capacity ϵ, where $0 < \epsilon < 1$. Reprinted with permission from [114].

intuition for the result. Suppose source s_1 is independent of sources s_2 and s_3, while the latter two are highly correlated, and all three sources have entropy 1. In the limit as ϵ goes to 0 and sources s_2 and s_3 are invertible functions of each other, we have essentially the equivalent of the modified butterfly network in Figure 2.6, where sources s_2 and s_3 in Figure 2.5 together play the role of source s_2 in Figure 2.6. The problem is thus solvable with joint source-network coding. It is not however solvable with separate source and network coding– sink t_2 needs to do joint source and network decoding based on the correlation between sources s_2 and s_3.

2.8 Notes and further reading

The field of network coding has its origins in the work of Yeung et al. [151], Ahlswede et al. [4] and Li et al. [88]. The famous butterfly network and the max flow min cut bound for network coded multicast were given in Ahlswede et al. [4]. Li et al. [88] showed that linear coding with finite symbol size is sufficient for multicast connections. Koetter

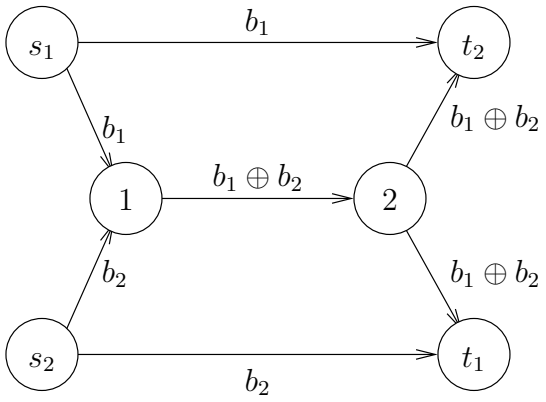


Fig. 2.6. The modified butterfly network. In this network, every arc has capacity 1.

and Médard [85] developed the algebraic framework for linear network coding used in this chapter.

The connection between multicast throughput advantage and integrality gap of the minimum weight Steiner tree problem was given by Agarwal and Charikar [3], and extended to the case of average throughput by Chekuri et al. [23].

Concurrent independent work by Sanders et al. [123] and Jaggi et al. [68] developed the centralized polynomial-time algorithm for acyclic networks presented in this chapter. The coding vector approach for distributed random linear network coding was introduced in Ho et al. [62]. The network coding generalization of the Edmonds matrix was given in Ho et al. [55, 57] and used in Harvey et al. [54], which presented a deterministic construction for multicast codes based on matrix completion. A practical batch network coding protocol based on random network coding was presented in Chou et al. [27]. The vector space approach for distributed random network coding was proposed by Koetter and Kschischang [83]. A gossip protocol using random linear network coding was presented in Deb and Médard [34]. Fragouli and Soljanin [47] developed a code construction technique based on information flow decomposition.

A number of works have considered the characteristics of network codes needed for achieving capacity on different types of multicast network problems. Lower bounds on coding field size were presented by Rasala Lehman and Lehman [116] and Feder et al. [43]. Upper bounds were given by Jaggi et al. [73] (acyclic networks), Ho et al. [66] (general

networks), Feder et al. [43] (graph-specific) and Fragouli et al. [47] (two sources).

Convolutional network coding was first discussed in Ahlswede et al. [4]. The algebraic convolutional network coding approach presented in this chapter is from Koetter and Médard [85]. Various aspects of convolutional network coding, including constructive coding and decoding techniques, are addressed in Erez and Feder [40, 41], Fragouli and Soljanin [45], and Li and Yeung [87]. The linear network coding approach for correlated sources is an extension by Ho et al. [65] of the linear coding approach in Csiszár [30] for the Slepian-Wolf problem. Separation of source coding and network coding is addressed in [114].

2.A Appendix: Random network coding

Lemma 2.5 *Let P be a nonzero polynomial in $\mathbb{F}[\xi_1, \xi_2, \dots]$ of degree less than or equal to $d\eta$, in which the largest exponent of any variable ξ_i is at most d. Values for $\xi_1, \xi_2, \dots$ are chosen independently and uniformly at random from $\mathbb{F}_q \subseteq \mathbb{F}$. The probability that P equals zero is at most $1 - (1 - d/q)^\eta$ for $d < q$.*

Proof For any variable ξ_1 in P, let d_1 be the largest exponent of ξ_1 in P. Express P in the form $P = \xi_1^{d_1} P_1 + R_1$, where P_1 is a polynomial of degree at most $d\eta - d_1$ that does not contain variable ξ_1, and R_1 is a polynomial in which the largest exponent of ξ_1 is less than d_1. By the Principle of Deferred Decisions (e.g., [105]), the probability $\Pr[P = 0]$ is unaffected if we set the value of ξ_1 last after all the other coefficients have been set. If, for some choice of the other coefficients, $P_1 \neq 0$, then P becomes a polynomial in $\mathbb{F}[\xi_1]$ of degree d_1. By the Schwartz-Zippel Theoremhim, this probability $\Pr[P = 0 | P_1 \neq 0]$ is upper bounded by d_1/q. So

$$\begin{aligned} \Pr[P = 0] &\leq \Pr[P_1 \neq 0]\frac{d_1}{q} + \Pr[P_1 = 0] \\ &= \Pr[P_1 = 0]\left(1 - \frac{d_1}{q}\right) + \frac{d_1}{q}. \end{aligned} \tag{2.11}$$

Next we consider $\Pr[P_1 = 0]$, choosing any variable ξ_2 in P_1 and letting d_2 be the largest exponent of ξ_2 in P_1. We express P_1 in the form $P_1 = \xi_2^{d_2} P_2 + R_2$, where P_2 is a polynomial of degree at most $d\eta - d_1 - d_2$ that does not contain variables ξ_1 or ξ_2, and R_2 is a polynomial in which

the largest exponent of ξ_2 is less than d_2. Proceeding similarly, we assign variables ξ_i and define d_i and P_i for $i = 3, 4, \ldots$ until we reach $i = k$ where P_k is a constant and $\Pr[P_k = 0] = 0$. Note that $1 \le d_i \le d < q \,\forall\, i$ and $\sum_{i=1}^{k} d_i \le d\eta$, so $k \le d\eta$. Applying Schwartz-Zippel as before, we have for $k' = 1, 2, \ldots, k$

$$Pr[P_{k'} = 0] \le Pr[P_{k'+1} = 0] \left(1 - \frac{d_{k'+1}}{q}\right) + \frac{d_{k'+1}}{q}. \tag{2.12}$$

Combining all the inequalities recursively, we can show by induction that

$$\begin{aligned} \Pr[P = 0] \quad \le \quad & \frac{\sum_{i=1}^{k} d_i}{q} - \frac{\sum_{i \ne l} d_i d_l}{q^2} \\ & + \cdots + (-1)^{k-1} \frac{\prod_{i=1}^{k} d_i}{q^k}. \end{aligned}$$

Now consider the integer optimization problem

$$\begin{aligned} \text{Maximize} \quad & f = \frac{\sum_{i=1}^{d\eta} d_i}{q} - \frac{\sum_{i \ne l} d_i d_l}{q^2} + \\ & \cdots + (-1)^{d\eta-1} \frac{\prod_{i=1}^{d\eta} d_i}{q^{d\eta}} \\ \text{subject to} \quad & 0 \le d_i \le d < q \,\forall\, i \in [1, d\eta], \\ & \sum_{i=1}^{d\eta} d_i \le d\eta, \quad \text{and} \quad d_i \text{ integer} \end{aligned} \tag{2.13}$$

whose maximum is an upper bound on $\Pr[P = 0]$.

We first consider the problem obtained by relaxing the integer condition on the variables d_i. Let $\mathbf{d}^* = \{d_1^*, \ldots, d_{d\eta}^*\}$ be an optimal solution.

For any set S_h of h distinct integers from $[1, d\eta]$, let $f_{S_h} = 1 - \frac{\sum_{i \in S_h} d_i}{q} + \frac{\sum_{i,l \in S_h, i \ne l} d_i d_l}{q^2} - \cdots + (-1)^h \frac{\prod_{i \in S_h} d_i}{q^h}$. We can show by induction on h that $0 < f_{S_h} < 1$ for any set S_h of h distinct integers in $[1, d\eta]$. If $\sum_{i=1}^{d\eta} d_i^* < d\eta$, then there is some $d_i^* < d$, and there exists a feasible solution $\mathbf{d} = \{d_1, \ldots, d_{d\eta}\}$ such that $d_i = d_i^* + \epsilon$, $\epsilon > 0$, and $d_h = d_h^*$ for $h \ne i$, which satisfies

$$\begin{aligned} & f(\mathbf{d}) - f(\mathbf{d}^*) \\ & \quad = \frac{\epsilon}{q} \left(1 - \frac{\sum_{h \ne i} d_h^*}{q} + \cdots + (-1)^{d\eta-1} \frac{\prod_{h \ne i} d_h^*}{q^{d\eta-1}}\right). \end{aligned}$$

This is positive, contradicting the optimality of $\mathbf{d}^*$, so $\sum_{i=1}^{d\eta} d_i^* = d\eta$.

Next suppose $0 < d_i^* < d$ for some d_i^*. Then there exists some d_l^* such

that $0 < d_l^* < d$, since if $d_l^* = 0$ or d for all other l, then $\sum_{i=1}^{d\eta} d_i^* \neq d\eta$. Assume without loss of generality that $0 < d_i^* \leq d_l^* < d$. Then there exists a feasible vector $\mathbf{d} = \{d_1, \ldots, d_{d\eta}\}$ such that $d_i = d_i^* - \epsilon$, $d_l = d_l^* + \epsilon$, $\epsilon > 0$, and $d_h = d_h^* \; \forall \; h \neq i, l$, which satisfies

$$f(\mathbf{d}) - f(\mathbf{d}^*) = -\left(\frac{(d_i^* - d_l^*)\epsilon - \epsilon^2}{q^2}\right) \left(1 - \frac{\sum_{h\neq i,l} d_h^*}{q} - \cdots + (-1)^{d\eta-2}\frac{\prod_{h\neq i,land} d_h^*}{q^{d\eta-2}}\right).$$

This is again positive, contradicting the optimality of $\mathbf{d}^*$.
Thus, $\sum_{i=1}^{d\eta} d_i^* = d\eta$, and $d_i^* = 0$ or d. So exactly η of the variables d_i^* are equal to d. Since the optimal solution is an integer solution, it is also optimal for the integer program (2.13). The corresponding optimal $f = \eta\frac{d}{q} - \binom{\eta}{2}\frac{d^2}{q^2} + \cdots + (-1)^{\eta-1}\frac{d^\eta}{q^\eta} = 1 - \left(1 - \frac{d}{q}\right)^\eta$. □

Proof of Theorem 2.8: We consider transmission, by random linear network coding, of one block of source bits, represented by vector $[\ \mathbf{x}_1, \quad \mathbf{x}_2\ ] \in \mathbb{F}_2^{n(r_1+r_2)}$. The transfer matrix $\mathbf{C}_{\mathcal{I}(t)}$ specifies the mapping from the vector of source bits $[\ \mathbf{x}_1, \quad \mathbf{x}_2\ ]$ to the vector $\mathbf{z}$ of bits on the set $\mathcal{I}(t)$ of terminal arcs incident to the receiver.

The decoder maps a vector $\mathbf{z}$ of received bits onto a vector $[\ \tilde{\mathbf{x}_1}, \quad \tilde{\mathbf{x}_2}\ ] \in \mathbb{F}_2^{n(r_1+r_2)}$ minimizing $\alpha(P_{\mathbf{x}_1\mathbf{x}_2})$ subject to $[\ \mathbf{x}_1, \quad \mathbf{x}_2\ ]\mathbf{C}_{\mathcal{I}(t)} = \mathbf{z}$. For a minimum entropy decoder, $\alpha(P_{\mathbf{x}_1\mathbf{x}_2}) \equiv H(P_{\mathbf{x}_1\mathbf{x}_2})$, while for a maximum Q-probability decoder, $\alpha(P_{\mathbf{x}_1\mathbf{x}_2}) \equiv -\log Q^n(\mathbf{x}_1\mathbf{x}_2)$. We consider three types of errors: in the first type, the decoder has the correct value for $\mathbf{x}_2$ but outputs the wrong value for $\mathbf{x}_2$; in the second, the decoder has the correct value for $\mathbf{x}_1$ but outputs the wrong value for $\mathbf{x}_2$; in the third, the decoder outputs wrong values for both $\mathbf{x}_1$ and $\mathbf{x}_2$. The error probability is upper bounded by the sum of the probabilities of the three types of errors, $\sum_{i=1}^{3} p_e^i$.

(Joint) types of sequences are considered as (joint) distributions P_X ($P_{X,Y}$, etc.) of dummy variables X, Y, etc. The set of different types of sequences in $\mathbb{F}_2^k$ is denoted by $\mathcal{P}(\mathbb{F}_2^k)$. Defining the sets of types

$$\mathcal{P}_n^i = \begin{cases} \{P_{X\tilde{X}Y\tilde{Y}} \in \mathcal{P}(\mathbb{F}_2^{nr_1} \times \mathbb{F}_2^{nr_1} \times \mathbb{F}_2^{nr_2} \times \mathbb{F}_2^{nr_2}) \mid \\ \quad \tilde{X} \neq X, \tilde{Y} = Y\} & i = 1 \\ \{P_{X\tilde{X}Y\tilde{Y}} \in \mathcal{P}(\mathbb{F}_2^{nr_1} \times \mathbb{F}_2^{nr_1} \times \mathbb{F}_2^{nr_2} \times \mathbb{F}_2^{nr_2}) \mid \\ \quad \tilde{X} = X, \tilde{Y} \neq Y\} & i = 2 \\ \{P_{X\tilde{X}Y\tilde{Y}} \in \mathcal{P}(\mathbb{F}_2^{nr_1} \times \mathbb{F}_2^{nr_1} \times \mathbb{F}_2^{nr_2} \times \mathbb{F}_2^{nr_2}) \mid \\ \quad \tilde{X} \neq X, \tilde{Y} \neq Y\} & i = 3 \end{cases}$$

sequences

$$\begin{aligned}\mathcal{T}_{XY} &= \{[\ \mathbf{x}_1,\quad \mathbf{x}_2\ ] \in \mathbb{F}_2^{n(r_1+r_2)} \mid \\ &\qquad P_{\mathbf{x}_1\mathbf{x}_2} = P_{XY}\} \\ \mathcal{T}_{\tilde{X}\tilde{Y}|XY}(\mathbf{x}_1\mathbf{x}_2) &= \{[\ \tilde{\mathbf{x}}_1,\quad \tilde{\mathbf{x}}_2\ ] \in \mathbb{F}_2^{n(r_1+r_2)} \mid \\ &\qquad P_{\tilde{\mathbf{x}}_1\tilde{\mathbf{x}}_2\mathbf{x}_1\mathbf{x}_2} = P_{\tilde{X}\tilde{Y}XY}\}\end{aligned}$$

we have

$$\begin{aligned}p_e^1 \leq & \sum_{\substack{P_{X\tilde{X}Y\tilde{Y}} \in \mathcal{P}_n^1 : \\ \alpha(P_{\tilde{X}\tilde{Y}}) \leq \alpha(P_{XY})}} \sum_{\substack{(\mathbf{x}_1,\,\mathbf{x}_2) \in \\ \mathcal{T}_{XY}}} Q^n(\mathbf{x}_1\mathbf{x}_2) \Pr\Bigg(\exists(\tilde{\mathbf{x}}_1,\tilde{\mathbf{x}}_2) \in \\ & \qquad \mathcal{T}_{\tilde{X}\tilde{Y}|XY}(\mathbf{x}_1\mathbf{x}_2) \text{ s.t.} [\ \mathbf{x}_1 - \tilde{\mathbf{x}}_1,\quad \mathbf{0}\ ]\mathbf{C}_{\mathcal{I}(t)} = \mathbf{0}\Bigg) \\ \leq & \sum_{\substack{P_{X\tilde{X}Y\tilde{Y}} \in \mathcal{P}_n^1 : \\ \alpha(P_{\tilde{X}\tilde{Y}}) \leq \alpha(P_{XY})}} \sum_{(\mathbf{x}_1,\,\mathbf{x}_2) \in \mathcal{T}_{XY}} Q^n(\mathbf{x}_1\mathbf{x}_2) \\ & \min\left\{ \sum_{\substack{(\tilde{\mathbf{x}}_1,\,\tilde{\mathbf{x}}_2) \in \\ \mathcal{T}_{\tilde{X}\tilde{Y}|XY}(\mathbf{x}_1\mathbf{x}_2)}} \Pr\left([\ \mathbf{x}_1 - \tilde{\mathbf{x}}_1,\quad \mathbf{0}\ ]\mathbf{C}_{\mathcal{I}(t)} = \mathbf{0}\right)\ ,\ 1 \right\}\end{aligned}$$

Similarly,

$$\begin{aligned}p_e^2 &\leq \sum_{\substack{P_{X\tilde{X}Y\tilde{Y}} \in \mathcal{P}_n^2 : \\ \alpha(P_{X\tilde{Y}}) \leq \alpha(P_{XY})}} \sum_{(\mathbf{x}_1,\,\mathbf{x}_2) \in \mathcal{T}_{XY}} Q^n(\mathbf{x}_1\mathbf{x}_2) \\ &\qquad \min\left\{ \sum_{\substack{(\tilde{\mathbf{x}}_1,\,\tilde{\mathbf{x}}_2) \in \\ \mathcal{T}_{\tilde{X}\tilde{Y}|XY}(\mathbf{x}_1\mathbf{x}_2)}} \Pr\left([\ \mathbf{0},\quad \mathbf{x}_2 - \tilde{\mathbf{x}}_2\ ]\mathbf{C}_{\mathcal{I}(t)} = \mathbf{0}\right)\ ,\ 1 \right\} \\ p_e^3 &\leq \sum_{\substack{P_{X\tilde{X}Y\tilde{Y}} \in \mathcal{P}_n^3 : \\ \alpha(P_{\tilde{X}\tilde{Y}}) \leq \alpha(P_{XY})}} \sum_{(\mathbf{x}_1,\,\mathbf{x}_2) \in \mathcal{T}_{XY}} Q^n(\mathbf{x}_1\mathbf{x}_2) \\ &\qquad \min\left\{ \sum_{\substack{(\tilde{\mathbf{x}}_1,\,\tilde{\mathbf{x}}_2) \in \\ \mathcal{T}_{\tilde{X}\tilde{Y}|XY}(\mathbf{x}_1\mathbf{x}_2)}} \Pr\left([\ \mathbf{x}_1 - \tilde{\mathbf{x}}_1,\quad \mathbf{x}_2 - \tilde{\mathbf{x}}_2\ ]\mathbf{C}_{\mathcal{I}(t)} = \mathbf{0}\right)\ ,\ 1 \right\}\end{aligned}$$

where the probabilities are taken over realizations of the network transfer matrix $\mathbf{C}_{\mathcal{I}(t)}$ corresponding to the random network code. The probabil-

ities

$$\begin{aligned} P_1 &= \Pr\left([\ \mathbf{x}_1 - \tilde{\mathbf{x}_1},\quad \mathbf{0}\ ]\mathbf{C}_{\mathcal{I}(t)} = \mathbf{0}\right) \\ P_2 &= \Pr\left([\ \mathbf{0},\quad \mathbf{x}_2 - \tilde{\mathbf{x}_2}\ ]\mathbf{C}_{\mathcal{I}(t)} = \mathbf{0}\right) \\ P_3 &= \Pr\left([\ \mathbf{x}_1 - \tilde{\mathbf{x}_1},\quad \mathbf{x}_2 - \tilde{\mathbf{x}_2}\ ]\mathbf{C}_{\mathcal{I}(t)} = \mathbf{0}\right) \end{aligned}$$

for nonzero $\mathbf{x}_1 - \tilde{\mathbf{x}_1}, \mathbf{x}_2 - \tilde{\mathbf{x}_2}$ can be calculated for a given network, or bounded in terms of n and parameters of the network as we will show later.

We can apply some simple cardinality bounds

$$\begin{aligned} |\mathcal{P}_n^1| &< (n+1)^{2^{2r_1+r_2}} \\ |\mathcal{P}_n^2| &< (n+1)^{2^{r_1+2r_2}} \\ |\mathcal{P}_n^3| &< (n+1)^{2^{2r_1+2r_2}} \\ |\mathcal{T}_{XY}| &\leq \exp\{nH(XY)\} \\ |\mathcal{T}_{\tilde{X}\tilde{Y}|XY}(\mathbf{x}_1\mathbf{x}_2)| &\leq \exp\{nH(\tilde{X}\tilde{Y}|XY)\} \end{aligned}$$

and the identity

$$\begin{aligned} Q^n(\mathbf{x}_1\mathbf{x}_2) &= \exp\{-n(D(P_{XY}||Q) + H(XY))\}, \\ &\quad (\mathbf{x}_1, \mathbf{x}_2) \in \mathcal{T}_{XY} \end{aligned} \tag{2.14}$$

to obtain

$$\begin{aligned} p_e^1 \leq \exp\Bigg\{ &- n \min_{\substack{P_{X\tilde{X}Y\tilde{Y}} \in \mathcal{P}_n^1: \\ \alpha(P_{\tilde{X}\tilde{Y}}) \leq \alpha(P_{XY})}} \Big(D(P_{XY}||Q) + \\ &\left|-\frac{1}{n}\log P_1 - H(\tilde{X}|XY)\right|^+\Big) + 2^{2r_1+r_2}\log(n+1)\Bigg\} \\ p_e^2 \leq \exp\Bigg\{ &- n \min_{\substack{P_{X\tilde{X}Y\tilde{Y}} \\ \in \mathcal{P}_n^2: \\ \alpha(P_{X\tilde{Y}}) \leq \\ \alpha(P_{XY})}} \Big(D(P_{XY}||Q) + \\ &\left|-\frac{1}{n}\log P_2 - H(\tilde{Y}|XY)\right|^+\Big) + 2^{r_1+2r_2}\log(n+1)\Bigg\} \\ p_e^3 \leq \exp\Bigg\{ &- n \min_{\substack{P_{X\tilde{X}Y\tilde{Y}} \\ \in \mathcal{P}_n^3: \\ \alpha(P_{\tilde{X}\tilde{Y}}) \leq \\ \alpha(P_{XY})}} \Big(D(P_{XY}||Q) + \\ &\left|-\frac{1}{n}\log P_3 - H(\tilde{X}\tilde{Y}|XY)\right|^+\Big) + 2^{2r_1+2r_2}\log(n+1)\Bigg\}, \end{aligned}$$

where the exponents and logs are taken with respect to base 2.

For the minimum entropy decoder, we have

$$\alpha(P_{\tilde{X}\tilde{Y}}) \le \alpha(P_{XY}) \Rightarrow$$
$$\begin{cases} H(\tilde{X}|XY) \le H(\tilde{X}|Y) \le H(X|Y) & \text{for } Y = \tilde{Y} \\ H(\tilde{Y}|XY) \le H(\tilde{Y}|X) \le H(Y|X) & \text{for } X = \tilde{X} \\ H(\tilde{X}\tilde{Y}|XY) \le H(\tilde{X}\tilde{Y}) \le H(XY) \end{cases}$$

which gives

$$p_e^1 \le \exp\left\{ -n \min_{XY} \left(D(P_{XY}||Q) + \left| -\frac{1}{n}\log P_1 - H(X|Y) \right|^+ \right) + 2^{2r_1+r_2}\log(n+1) \right\} \tag{2.15}$$

$$p_e^2 \le \exp\left\{ -n \min_{XY} \left(D(P_{XY}||Q) + \left| -\frac{1}{n}\log P_2 - H(Y|X) \right|^+ \right) + 2^{r_1+2r_2}\log(n+1) \right\} \tag{2.16}$$

$$p_e^3 \le \exp\left\{ -n \min_{XY} \left(D(P_{XY}||Q) + \left| -\frac{1}{n}\log P_3 - H(XY) \right|^+ \right) + 2^{2r_1+2r_2}\log(n+1) \right\}. \tag{2.17}$$

We next show that these bounds also hold for the maximum Q-probability decoder, for which, from (2.14), we have

$$\begin{aligned} \alpha(P_{\tilde{X}\tilde{Y}}) &\le \alpha(P_{XY}) \\ \Rightarrow D(P_{\tilde{X}\tilde{Y}}||Q) + H(\tilde{X}\tilde{Y}) &\le D(P_{XY}||Q) + H(XY). \end{aligned} \tag{2.18}$$

For $i = 1$, $\tilde{Y} = Y$, and (2.18) gives

$$D(P_{\tilde{X}Y}||Q) + H(\tilde{X}|Y) \le D(P_{XY}||Q) + H(X|Y). \tag{2.19}$$

We show that

$$\begin{aligned}
&\min_{\substack{P_{X\tilde{X}Y\tilde{Y}} \in \mathcal{P}_n^1 : \\ \alpha(P_{\tilde{X}\tilde{Y}}) \leq \\ \alpha(P_{XY})}} \left(D(P_{XY}||Q) + \left| -\frac{1}{n}\log P_1 - H(\tilde{X}|XY) \right|^+ \right) \\
&\geq \min_{\substack{P_{X\tilde{X}Y\tilde{Y}} \in \mathcal{P}_n^1 : \\ \alpha(P_{\tilde{X}\tilde{Y}}) \leq \\ \alpha(P_{XY})}} \left(D(P_{XY}||Q) + \left| -\frac{1}{n}\log P_1 - H(\tilde{X}|Y) \right|^+ \right) \\
&\geq \min_{XY} \left(D(P_{XY}||Q) + \left| -\frac{1}{n}\log P_1 - H(X|Y) \right|^+ \right)
\end{aligned}$$

by considering two possible cases for any $X, \tilde{X}, Y$ satisfying (2.19):

Case 1: $-\frac{1}{n}\log P_1 - H(X|Y) < 0$. Then

$$\begin{aligned}
&D(P_{XY}||Q) + \left| -\frac{1}{n}\log P_1 - H(\tilde{X}|Y) \right|^+ \\
&\geq D(P_{XY}||Q) + \left| -\frac{1}{n}\log P_1 - H(X|Y) \right|^+ \\
&\geq \min_{XY} \left(D(P_{XY}||Q) + \left| -\frac{1}{n}\log P_1 - H(X|Y) \right|^+ \right)
\end{aligned}$$

Case 2: $-\frac{1}{n}\log P_1 - H(X|Y) \geq 0$. Then

$$\begin{aligned}
&D(P_{XY}||Q) + \left| -\frac{1}{n}\log P_1 - H(\tilde{X}|Y) \right|^+ \\
&\geq \quad D(P_{XY}||Q) + \left(-\frac{1}{n}\log P_1 - H(\tilde{X}|Y) \right) \\
&\geq \quad D(P_{\tilde{X}Y}||Q) + \left(-\frac{1}{n}\log P_1 - H(X|Y) \right) \text{ by (2.19)} \\
&= \quad D(P_{\tilde{X}Y}||Q) + \left| -\frac{1}{n}\log P_1 - H(X|Y) \right|^+
\end{aligned}$$

which gives

$$\begin{aligned}
&D(P_{XY}||Q) + \left|-\frac{1}{n}\log P_1 - H(\tilde{X}|Y)\right|^+ \\
&\geq \frac{1}{2}\left[D(P_{XY}||Q) + \left|-\frac{1}{n}\log P_1 - H(\tilde{X}|Y)\right|^+\right. \\
&\quad \left. + D(P_{\tilde{X}Y}||Q) + \left|-\frac{1}{n}\log P_1 - H(X|Y)\right|^+\right] \\
&\geq \frac{1}{2}\left[D(P_{XY}||Q) + \left|-\frac{1}{n}\log P_1 - H(X|Y)\right|^+\right. \\
&\quad \left. + D(P_{\tilde{X}Y}||Q) + \left|-\frac{1}{n}\log P_1 - H(\tilde{X}|Y)\right|^+\right] \\
&\geq \min_{XY}\left(D(P_{XY}||Q) + \left|-\frac{1}{n}\log P_1 - H(X|Y)\right|^+\right).
\end{aligned}$$

A similar proof holds for $i = 2$.

For $i = 3$, we show that

$$\begin{aligned}
&\min_{\substack{P_{X\tilde{X}Y\tilde{Y}} \in \mathcal{P}_n^3 : \\ \alpha(P_{\tilde{X}\tilde{Y}}) \leq \\ \alpha(P_{XY})}} \left(D(P_{XY}||Q) + \left|-\frac{1}{n}\log P_3 - H(\tilde{X}\tilde{Y}|XY)\right|^+\right) \\
&\geq \min_{\substack{P_{X\tilde{X}Y\tilde{Y}} \in \mathcal{P}_n^3 : \\ \alpha(P_{\tilde{X}\tilde{Y}}) \leq \\ \alpha(P_{XY})}} \left(D(P_{XY}||Q) + \left|-\frac{1}{n}\log P_3 - H(\tilde{X}\tilde{Y})\right|^+\right) \\
&\geq \min_{XY}\left(D(P_{XY}||Q) + \left|-\frac{1}{n}\log P_3 - H(XY)\right|^+\right)
\end{aligned}$$

by considering two possible cases for any $X, \tilde{X}, Y, \tilde{Y}$ satisfying (2.18):

Case 1: $-\frac{1}{n}\log P_3 - H(XY) < 0$. Then

$$\begin{aligned}
&D(P_{XY}||Q) + \left|-\frac{1}{n}\log P_3 - H(\tilde{X}\tilde{Y})\right|^+ \\
&\geq D(P_{XY}||Q) + \left|-\frac{1}{n}\log P_3 - H(XY)\right|^+ \\
&\geq \min_{XY}\left(D(P_{XY}||Q) + \left|-\frac{1}{n}\log P_3 - H(XY)\right|^+\right)
\end{aligned}$$

Case 2: $-\frac{1}{n}\log P_3 - H(XY) \geq 0$. Then

$$\begin{aligned}
& D(P_{XY}||Q) + \left|-\frac{1}{n}\log P_3 - H(\tilde{X}\tilde{Y})\right|^+ \\
&\geq \quad D(P_{XY}||Q) + \left(-\frac{1}{n}\log P_3 - H(\tilde{X}\tilde{Y})\right) \\
&\geq \quad D(P_{\tilde{X}\tilde{Y}}||Q) + \left(-\frac{1}{n}\log P_3 - H(XY)\right) \text{ by (2.18)} \\
&= \quad D(P_{\tilde{X}\tilde{Y}}||Q) + \left|-\frac{1}{n}\log P_3 - H(XY)\right|^+
\end{aligned}$$

which gives

$$\begin{aligned}
& D(P_{XY}||Q) + \left|-\frac{1}{n}\log P_3 - H(\tilde{X}\tilde{Y})\right|^+ \\
&\geq \quad \frac{1}{2}\left[D(P_{XY}||Q) + \left|-\frac{1}{n}\log P_3 - H(\tilde{X}\tilde{Y})\right|^+\right. \\
&\qquad \left. + D(P_{\tilde{X}\tilde{Y}}||Q) + \left|-\frac{1}{n}\log P_3 - H(XY)\right|^+\right] \\
&\geq \quad \min_{XY}\left(D(P_{XY}||Q) + \left|-\frac{1}{n}\log P_3 - H(XY)\right|^+\right).
\end{aligned}$$

We bound the probabilities P_i in terms of n and the network parameters $m_i, i = 1, 2$, the minimum cut capacity between the receiver and source X_i, m_3, the minimum cut capacity between the receiver and both sources, and L, the maximum source-receiver path length.

Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be subgraphs of graph $\mathcal{G}$ consisting of all arcs downstream of sources 1 and 2 respectively, where an arc k is considered downstream of a source X_i if $s_i = o(k)$ or if there is a directed path from the source to $o(k)$. Let $\mathcal{G}_3$ be equal to $\mathcal{G}$.

Note that in a random linear network code, any arc k which has at least one nonzero input transmits the zero process with probability $\frac{1}{2^{nc_k}}$, where c_k is the capacity of k. Since the code is linear, this probability is the same as the probability that a pair of distinct values for the inputs of k are mapped to the same output value on k.

For a given pair of distinct source values, let E_k be the event that the corresponding inputs to arc k are distinct, but the corresponding values on k are the same. Let $E(\tilde{\mathcal{G}})$ be the event that E_k occurs for some arc k on every source-receiver path in graph $\tilde{\mathcal{G}}$. P_i is then equal to the probability of event $E(\mathcal{G}_i)$.

Let $\mathcal{G}'_i, i = 1, 2, 3$ be the graph consisting of m_i node-disjoint paths, each consisting of L arcs each of unit capacity. We show by induction on m_i that P_i is upper bounded by the probability of event $E(\mathcal{G}'_i)$.

We let $\tilde{\mathcal{G}}$ be the graphs $\mathcal{G}_i, \mathcal{G}'_i, i = 1, 2, 3$ in turn, and consider any particular source-receiver path $\mathcal{P}_{\tilde{\mathcal{G}}}$ in $\tilde{\mathcal{G}}$. We distinguish two cases:

Case 1: E_k does not occur for any of the arcs k on the path $\mathcal{P}_{\tilde{\mathcal{G}}}$. In this case the event $E(\tilde{\mathcal{G}})$ occurs with probability 0.

Case 2: There exists some arc $\hat{k}$ on the path $\mathcal{P}_{\tilde{\mathcal{G}}}$ for which E_k occurs.

Thus, we have $\Pr(E(\tilde{\mathcal{G}})) = \Pr(\text{case 2}) \Pr(E(\tilde{\mathcal{G}})|\text{case 2})$. Since $\mathcal{P}_{\mathcal{G}'_i}$ has at least as many arcs as $\mathcal{P}_{\mathcal{G}_i}$, $\Pr(\text{case 2 for } \mathcal{G}'_i) \geq \Pr(\text{case 2 for } \mathcal{G}_i)$. Therefore, if we can show that $\Pr(E(\mathcal{G}'_i)|\text{case 2}) \geq \Pr(E(\mathcal{G}_i)|\text{case 2})$, the induction hypothesis $\Pr(E(\mathcal{G}'_i)) \geq \Pr(E(\mathcal{G}_i))$ follows.

For $m_i = 1$, the hypothesis is true since $\Pr(E(\mathcal{G}'_i)|\text{case 2}) = 1$. For $m_i > 1$, in case 2, removing the arc $\hat{k}$ leaves, for $\mathcal{G}'_i$, the effective equivalent of a graph consisting of $m_i - 1$ node-disjoint length-L paths, and, for $\mathcal{G}_i$, a graph of minimum cut at least $m_i - 1$. The result follows from applying the induction hypothesis to the resulting graphs.

Thus, $\Pr(E(\mathcal{G}'_i))$ gives an upper bound on probability P_i:

$$\begin{aligned} P_i &\leq \left(1 - (1 - \frac{1}{2^n})^L\right)^{m_i} \\ &\leq \left(\frac{L}{2^n}\right)^{m_i}. \end{aligned}$$

Substituting this into the error bounds (2.15)-(2.17) gives the desired result. □

3
Inter-Session Network Coding

So far, we have considered network coding for a single communication session, i.e. unicast communication to one sink node or multicast of common information to multiple sink nodes. This type of coding is called *intra-session network coding*, since we only code together information symbols that will be decoded by the same set of sink nodes. For intra-session network coding, it suffices for each node to form its outputs as random linear combinations of its inputs. Each sink node can decode once it has received enough independent linear combinations of the source processes.

When there are multiple sessions sharing the network, a simple practical approach is to allocate disjoint subsets of the network capacity to each session. If each session's allocated subgraph satisfies the max-flow/min-cut condition for each sink (Theorems 2.2 and 2.7), we can obtain a solution with intra-session network coding among information symbols of each session separately. Sections 5.1.1 and 5.2.1 discuss such an approach.

In general, however, achieving optimal rates may require *inter-session* network coding, i.e. coding among information symbols of different sessions. Inter-session network coding is more complicated than intra-session network coding. Coding must be done strategically in order to ensure that each sink can decode its desired source processes – nodes cannot simply combine all their inputs randomly, since the sink nodes may not have sufficient incoming capacity to decode all the randomly combined source processes. Unlike intra-session network coding, decoding may have to be done at non-sink nodes. We will see an example further on, in Section 3.5.1.

At present, for general multi-session network problems, it is not yet known how to determine feasibility or construct optimal network codes.

In this chapter, we first discuss some theoretical approaches and results. We then describe constructions and implementations of suboptimal but practical inter-session network codes.

3.1 Scalar and vector linear network coding

Scalar linear network coding was described in Section 2.2 for a single multicast session. In the general case with multiple sessions, each sink $t \in \mathcal{T}$ can demand an arbitrary subset

$$\mathcal{D}_t \subset \{1, 2, \ldots, r\} \tag{3.1}$$

of the information sources. Scalar linear network coding for the general case is defined similarly, the only difference being that each sink needs only decode its demanded subset of information sources. We can generalize the scalar linear solvability criterion as follows. In the single session case, the criterion is that the transfer matrix determinant $\det \mathbf{M}_t$ for each sink node t, as a function of coding coefficients $(\mathbf{a}, \mathbf{f}, \mathbf{b})$, is nonzero – this corresponds to each sink node being able to decode all the source processes. In the inter-session case, the criterion is that there exists a value for coefficients $(\mathbf{a}, \mathbf{f}, \mathbf{b})$ such that

(i) the submatrix of $\mathbf{M}_t$ consisting of rows whose indexes are in $\mathcal{D}_t$ is nonsingular
(ii) the remaining rows of $\mathbf{M}_t$ are all zero.

This corresponds to each sink node being able to extract its demanded source processes while removing the effect of other interfering (unwanted) source processes.

The problem of determining whether a general network problem has a scalar linear solution has been shown (by reduction from 3-CNF) to be NP-hard. It can be reduced to the problem of determining whether a related algebraic variety is empty, as follows. Let $m_1(\mathbf{a}, \mathbf{f}, \mathbf{b}), \ldots, m_K(\mathbf{a}, \mathbf{f}, \mathbf{b})$ denote all the entries of $\mathbf{M}_t, t \in \mathcal{T}$ that must evaluate to zero according to condition (ii). Let $d_1(\mathbf{a}, \mathbf{f}, \mathbf{b}), \ldots, d_L(\mathbf{a}, \mathbf{f}, \mathbf{b})$ denote the determinants of the submatrices that must be nonzero according to condition (i). Let ξ be a new variable, and let I be the ideal generated by $m_1(\mathbf{a}, \mathbf{f}, \mathbf{b}), \ldots, m_K(\mathbf{a}, \mathbf{f}, \mathbf{b}), 1 - \xi \prod_{i=1}^{L} d_i(\mathbf{a}, \mathbf{f}, \mathbf{b})$. Then the decoding conditions are equivalent to the condition that the variety associated with the ideal I is nonempty. This can be decided by computing a Gröbner basis for I, which is not polynomial in complexity but for which standard mathematical software exists.

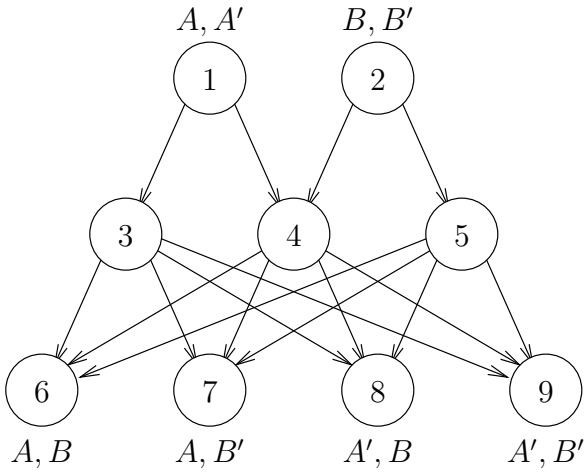
Unlike the single multicast session case, scalar linear network coding is not optimal in the general case. Scalar coding is time-invariant. The approach outlined above for determining scalar linear solvability does not cover time-sharing among scalar solutions. Figure 3.1 gives an example network problem which can be solved by time-sharing among different routing strategies, but not by scalar linear network coding.

The class of vector linear network codes includes both scalar linear network coding and time-sharing as special cases. In vector linear network coding, the bitstream corresponding to each source and arc process is divided into vectors of finite field symbols; the vector associated with an arc is a linear function, specified by a matrix, of the vectors associated with its inputs. Vector linear network coding was used in Section 2.7 for multicasting from correlated sources.

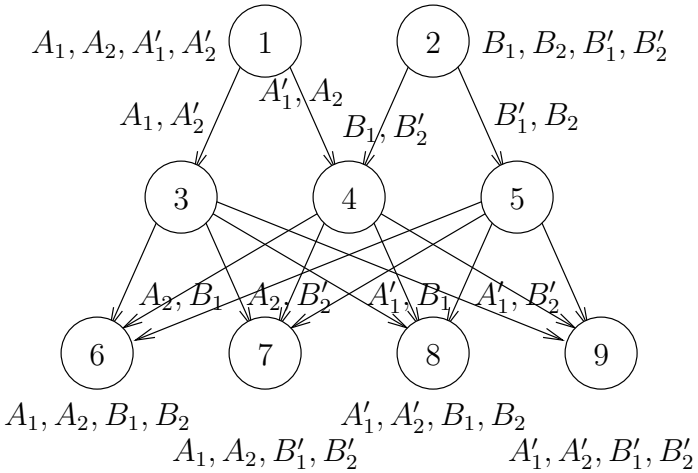
3.2 Fractional coding problem formulation

In the basic model and problem formulation of Section 2.1, a solution is defined with respect to fixed source rates and arc capacities (all assumed to be equal); for a given network problem and class of coding/routing strategies, a solution either exists or does not. A more flexible approach, useful for comparing various different classes of strategies in the multiple session case, is to define a network problem in terms of source/sink locations and demands, and to ask what rates are possible relative to the arc capacities.

The most general characterization of the rates achievable with some class of strategies is a rate region which gives the trade-offs between the rates of different sources. A simpler characterization, which suffices for the purpose of comparing different classes of network codes, assumes that the source rates are equal to each other and that the arc rates are equal to each other, and asks what is the maximum ratio between the rate of a source and the rate of an arc. Specifically, we consider sending a vector of k symbols from each source with a vector of n symbols transmitted on each arc. The symbols are from some alphabet (in the codes we have considered so far, the alphabet is a finite field, but we can consider more general alphabets such as rings). Such a network problem is defined by a graph $(\mathcal{N}, \mathcal{A})$, source nodes $s_i \in \mathcal{N}, i = 1, \ldots, r$, a set $\mathcal{T} \subset \mathcal{N}$ of sink nodes, and the sets $\mathcal{D}_t \subset \{1, \ldots, r\}$ of source processes demanded by each sink $t \in \mathcal{T}$. For brevity, we will in the following refer to such a network problem simply as a *network*. A (k, n) *fractional solution* defines coding operations at network nodes and decoding operations at



(a) An example network problem which can be solved by time-sharing among different routing strategies, but not by scalar linear network coding.



(b) A time-sharing routing solution.

Fig. 3.1.

sink nodes such that each sink perfectly reproduces the values of its demanded source processes. A *solution*, as in Section 2.1, corresponds to the case where $k = n$. A *scalar solution* is a special case where $k = n = 1$. The *coding capacity of a network with respect to an alphabet*

$\mathcal{A}$ and a class $\mathcal{C}$ of network codes is

$$\sup\{k/n : \exists \text{ a } (k,n) \text{ fractional coding solution in } \mathcal{C} \text{ over } \mathcal{A}\}$$

3.3 Insufficiency of linear network coding

Linear network coding is not sufficient in general for the case of multiple sessions. This was shown by an example network $\mathcal{P}$ which has a nonlinear coding capacity of 1, while there is no linear solution. The network $\mathcal{P}$ and its nonlinear solution are shown in Figure 3.2. The class of linear codes for which $\mathcal{P}$ has no solution is more general than the class of vector linear codes over finite fields described in Section 3.1. It includes linear network codes where the source and arc processes are associated with elements of any finite R-module G with more than one element, for any ring R. (A ring is a generalization of a field, the difference being that elements of a ring do not need to have multiplicative inverses. An R-module is a generalization of a vector space using a ring R in place of a field.)

The construction of $\mathcal{P}$ is based on connections with matroid theory. By identifying matroid elements with source and arc processes in a network, a (non-unique) *matroidal network* can be constructed from a given matroid, such that dependencies and independencies of the matroid are reflected in the network. Circuits (minimal dependent sets) of the matroid are reflected in the dependence of output arc processes of a node (or decoded output processes of a sink node) on the input processes of the node. Bases (maximal independent sets) of the matroid correspond to the set of all source processes, or the input processes of sink nodes that demand all the source processes. Properties of the matroid thus carry over to its associated matroidal network(s).

The network $\mathcal{P}$ is based on two matroidal networks $\mathcal{P}_1$ and $\mathcal{P}_2$ (shown in Figures 3.4 and 3.6) associated with the well-known Fano and non-Fano matroids respectively (shown in Figures 3.3 and 3.5. In the case of vector linear coding, $\mathcal{P}_1$ has no vector linear solution of any dimension over a finite field with odd characteristic, while $\mathcal{P}_2$ has no vector linear solution of any dimension over a finite field with characteristic 2.† This

† In the more general case of R-linear coding over G, vector linear solvability corresponds to scalar linear solvability over a larger module. It can be shown that if $\mathcal{P}$ has a scalar linear solution, then it has a scalar linear solution for the case where R acts faithfully on G and is a ring with an identity element I. In this case, $\mathcal{P}_1$ has no scalar R-linear solution over G if $I+I \neq 0$, while $\mathcal{P}_2$ has no scalar R-linear solution over G if $I+I=0$.

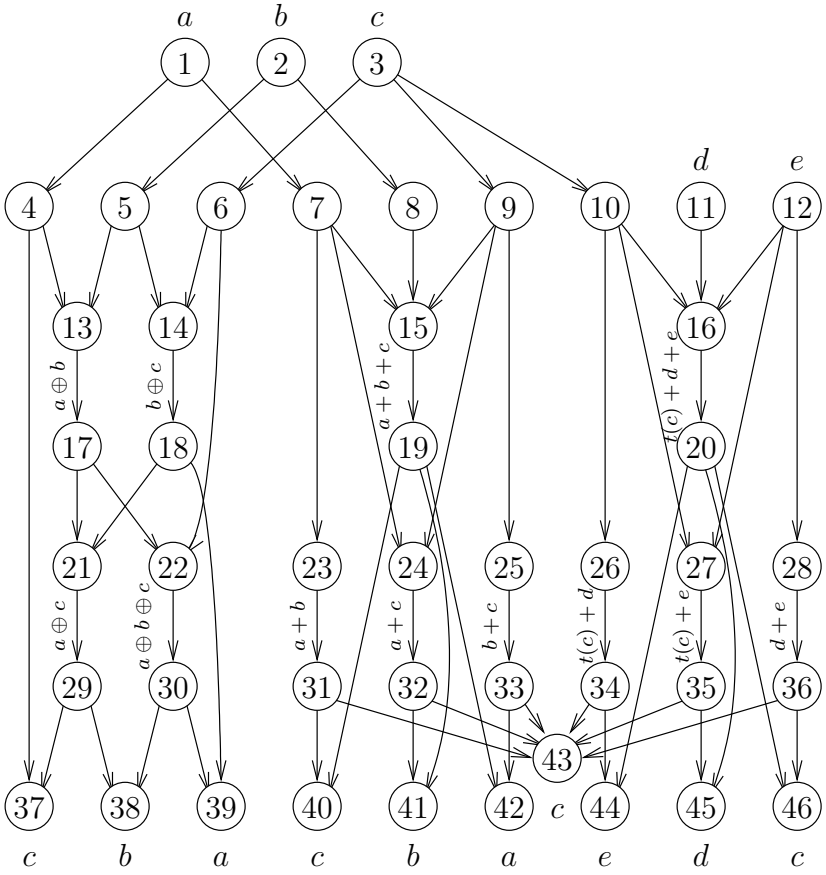


Fig. 3.2. An example network problem whose nonlinear coding capacity is greater than its linear coding capacity. The arc labels give a nonlinear solution over an alphabet of size 4. Symbols $+$ and $-$ denote addition and subtraction in the ring $\mathbb{Z}_4$ of integers modulo 4, $\oplus$ denotes bitwise XOR, and t denotes the operation of reversing the order of the bits in a 2-bit binary string.

incompatibility is exploited to construct the example network which does not have a linear solution, but has a nonlinear solution over an alphabet of size 4, shown in Figure 3.2.

For the case of vector linear codes over finite fields, it can be shown, using information theoretic arguments described in the next section, that the linear coding capacity of this example is 10/11. The proof is given in [36]. Thus, vector linear network coding over finite fields is not even asymptotically optimal.

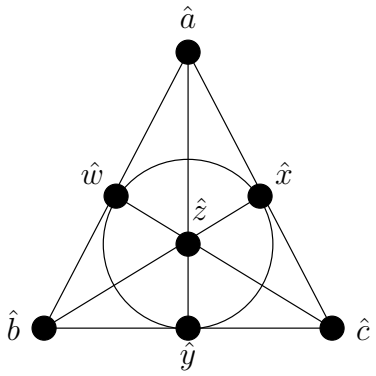


Fig. 3.3. A geometric representation of the Fano matroid. The labeled dots correspond to the elements of the underlying set, any three elements of which are dependent if and only if in the diagram they are collinear or on the circle.

3.4 Information theoretic approaches

Determining the coding capacity of an arbitrary network is an open problem, but progress in characterizing/bounding capacity in various cases has been made using information theoretic entropy arguments. The information theoretic approach represents the source and arc processes as random variables (or sequences of random variables). The entropies of the source random variables can be identified with (or lower bounded by) the source rates, while the entropies of the arc random variables can be upper bounded by the arc capacities. Other constraint relations involving the joint/conditional entropies of various subsets of these random variables can be derived from the coding dependencies and decoding requirements.

In the following we denote by $\mathcal{S} = \{1, \ldots, r\}$ the set of information sources, and by $\mathcal{I}(v), \mathcal{O}(v) \in \mathcal{A}, \mathcal{S}(v) \subset \mathcal{S}$ the set of input arcs, output arcs and information sources respectively of a node v. As before, $\mathcal{D}_t \subset \mathcal{S}$ denotes the set of information sources demanded by a sink t.

In the case of an acyclic network, suppose there exists a (k, n) fractional network coding solution over an alphabet $\mathcal{B}$. Let the ith source process, $1 \leq i \leq r$, be represented by a random vector X_i consisting of k independent random variables distributed uniformly over $\mathcal{B}$. Denote by Y_j the corresponding random vector transmitted on each arc $j \in \mathcal{A}$ under the fractional network code. We use the abbreviated notation $Y_{\mathcal{A}'} = \{Y_j : j \in \mathcal{A}'\}$ for a set of arcs $\mathcal{A}' \subset \mathcal{A}$, and $X_{\mathcal{S}'} = \{X_i : i \in \mathcal{S}'\}$ for a set of sources $\mathcal{S}' \subset \mathcal{S}$. Then we have the following entropy condi-

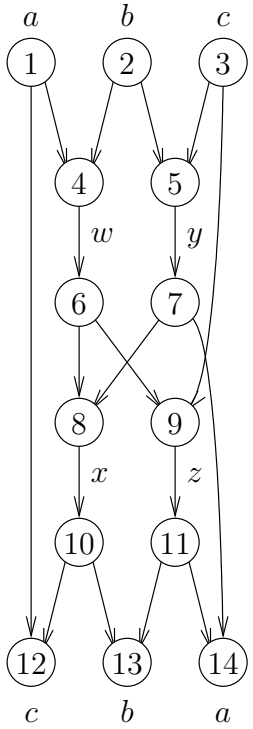


Fig. 3.4. A network associated with the Fano matroid. Source processes a, b, c originate at nodes 1,2,3 respectively, and are demanded by the sink nodes 14, 13, 12 respectively. The source and edge labels indicate the correspondence with the elements of the Fano matroid as shown in Figure 3.3. The network has no vector linear solution of any dimension over a finite field with odd characteristic.

tions:

$$H(X_i) = k \tag{3.2}$$

$$H(X_{\mathcal{S}}) = rk \tag{3.3}$$

$$H(Y_j) \leq n \ \forall\, j \in \mathcal{A} \tag{3.4}$$

$$H(Y_{\mathcal{O}(v)} \mid X_{\mathcal{S}(v)}, Y_{\mathcal{I}(v)}) = 0 \ \forall\, v \in \mathcal{N} \tag{3.5}$$

$$H(X_{\mathcal{D}_t} \mid Y_{\mathcal{I}(t)}) = 0 \ \forall\, t \in \mathcal{T} \tag{3.6}$$

Equations (3.2)-(3.3) state that the source vectors X_i each have entropy k (in units scaled by the log of the alphabet size) and are independent. Inequality (3.4) upper bounds the entropy of each arc vector Y_j by the arc capacity n. Equation (3.5) expresses the condition that each

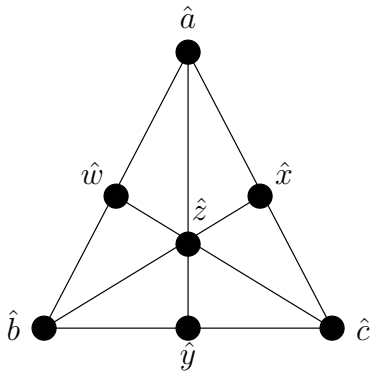


Fig. 3.5. A geometric representation of the non-Fano matroid. The labeled dots correspond to the elements of the underlying set, any three elements of which are dependent if and only if in the diagram they are collinear.

node's outputs are a deterministic function of its inputs. Equation (3.6) expresses the requirement that each sink can reproduce its demanded sources as a deterministic function of its inputs.

Given equations (3.2)-(3.6) for a particular network problem, *information inequalities* can be applied with the aim of simplifying the equations and obtaining a bound on the ratio k/n. Information inequalities are inequalities involving information measures (entropy, conditional entropy, mutual information and conditional mutual information) of subsets of a set of random variables, that hold under any joint distribution for the random variables. Intuitively, information inequalities are constraints which must be satisfied by the values of these information measures in order for the values to be consistent with some joint distribution. For any set $\mathcal{N}$ of discrete random variables, and any subsets $\mathcal{U}, \mathcal{U}', \mathcal{U}''$ of $\mathcal{N}$, we have the following *basic inequalities*:

$$\begin{aligned} H(\mathcal{U}) &\geq 0 \\ H(\mathcal{U}|\mathcal{U}') &\geq 0 \\ I(\mathcal{U};\mathcal{U}') &\geq 0 \\ I(\mathcal{U};\mathcal{U}'|\mathcal{U}'') &\geq 0 \end{aligned}$$

The basic inequalities and all inequalities implied by the basic inequalities are called *Shannon-type information inequalities*. For four or more random variables, there exist inequalities not implied by the basic inequalities, called *non-Shannon-type information inequalities*. An exam-

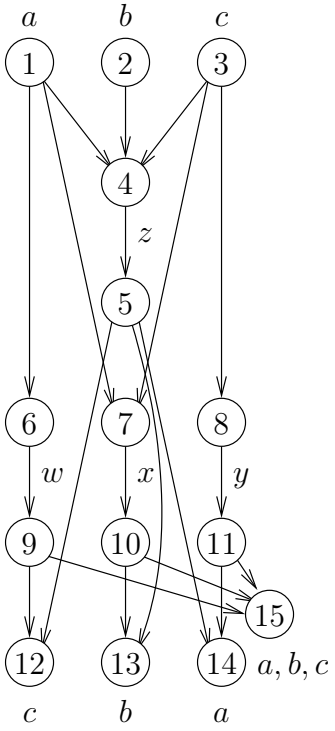


Fig. 3.6. A network associated with the non-Fano matroid. Source processes a, b, c originate at nodes 1,2,3 respectively, and are demanded by the sets of sink nodes $\{14, 15\}, \{13, 15\}, \{12, 15\}$ respectively. The source and edge labels indicate the correspondence with the elements of the non-Fano matroid as shown in Figure 3.5. The network has no vector linear solution of any dimension over a finite field with characteristic 2.

ple involving four random variables X_1, X_2, X_3, X_4 is the inequality

$$2I(X_3; X_4) \leq I(X_1; X_2)+I(X_1; X_3, X_4)+3I(X_3; X_4|X_1)+I(X_3; X_4|X_2), \tag{3.7}$$

Shannon-type inequalities are insufficient in general for computing coding capacity; this was shown in [37] by a network problem constructed based on the Vámos matroid (e.g., [110]), for which the use of the non-Shannon-type inequality (3.7) yields a tighter bound than any bound derived only with Shannon-type inequalities.

The information theoretic approach yields an implicit characterization of the rate region for deterministic network coding on a general acyclic network with arc capacities $c_k, k \in \mathcal{A}$. We first introduce some defini-

tions. Let $\mathcal{N}$ be a set of random variables denoted $\{X_i : i \in \mathcal{S}\} \cup \{Y_j : j \in \mathcal{A}\}$. Let $\mathcal{H}_\mathcal{N}$ be the $(2^{|\mathcal{N}|}-1)$-dimensional Euclidian space whose coordinates correspond to the $2^{|\mathcal{N}|}-1$ nonempty subsets of $\mathcal{N}$. A vector $\mathbf{g} \in \mathcal{H}_\mathcal{N}$ is *entropic* if there exists some joint distribution for the random variables in $\mathcal{N}$ such that each component of $\mathbf{g}$ equals the entropy of the corresponding subset of random variables. Define the region

$$\Gamma^*_\mathcal{N} = \{\mathbf{g} \in \mathcal{H}_\mathcal{N} : \mathbf{g} \text{ is entropic}\}$$

This region essentially summarizes the effect of all possible information inequalities involving variables in $\mathcal{N}$ (we do not yet have a complete characterization of this region or, equivalently, of all possible information inequalities).

We define the following regions in $\mathcal{H}_\mathcal{N}$:

$$\begin{aligned}
\mathcal{C}_1 &= \left\{\mathbf{h} \in \mathcal{H}_\mathcal{N} : H(X_\mathcal{S}) = \sum_{i\in\mathcal{S}} H(X_i)\right\} \\
\mathcal{C}_2 &= \left\{\mathbf{h} \in \mathcal{H}_\mathcal{N} : H(Y_{\mathcal{O}(v)} \mid X_{\mathcal{S}(v)}, Y_{\mathcal{I}(v)}) = 0 \ \forall\, v \in \mathcal{N}\right\} \\
\mathcal{C}_3 &= \{\mathbf{h} \in \mathcal{H}_\mathcal{N} : H(Y_j) < c_j \ \forall j \in \mathcal{A}\} \\
\mathcal{C}_4 &= \left\{\mathbf{h} \in \mathcal{H}_\mathcal{N} : H(X_{\mathcal{D}_t} | Y_{\mathcal{I}(t)}) = 0 \ \forall\, t \in \mathcal{T}\right\}
\end{aligned}$$

Theorem 3.1 *The capacity region for an arbitrary acyclic network with multiple sessions is given by*

$$\mathcal{R} = \Lambda\left(proj_{X_\mathcal{S}}\left(\overline{conv(\Gamma^*_\mathcal{N} \cap \mathcal{C}_{12})} \cap \mathcal{C}_3 \cap \mathcal{C}_4\right)\right),$$

where for any region $\mathcal{C} \subset \mathcal{H}_\mathcal{N}$, $proj_{X_\mathcal{S}}(\mathcal{C}) = \{\mathbf{h}_{X_\mathcal{S}} : \mathbf{h} \in \mathcal{C}\}$ is the projection of $\mathcal{C}$ on the coordinates h_{X_i} a corresponding to the source entropies, $conv(\cdot)$ denotes the convex hull operator, the overbar denotes the closure, and $\mathcal{C}_{12} = \mathcal{C}_1 \cap \mathcal{C}_2$.

Proof The proof is given in [145]. □

For networks with cycles, we need to also consider delay and causality constraints. This can be done by considering, for each arc j, a sequence $Y_j^{(1)}, \ldots, Y_j^{(T)}$ of random variables corresponding to time steps $\tau = 1, \ldots, T$. Then we have, in place of Equation (3.5),

$$H(Y^{(1)}_{\mathcal{O}(v)}, \ldots, Y^{(\tau)}_{\mathcal{O}(v)} \mid X_{\mathcal{S}(v)}, Y^{(1)}_{\mathcal{I}(v)}, \ldots, Y^{(\tau-1)}_{\mathcal{O}(v)}) = 0 \ \forall\, v \in \mathcal{N}.$$

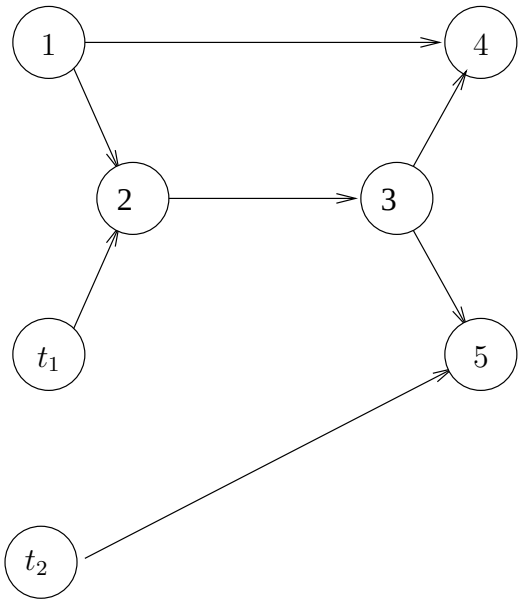


Fig. 3.7. Gadget for converting any general multiple session network coding problem into a multiple unicast problem. Reprinted with permission from [37].

3.4.1 Multiple unicast networks

A special case of the multiple session network coding problem is the multiple unicast case, i.e. multiple communication sessions each consisting of one source node and one sink node. For directed wired networks, any general multiple session network coding problem can be converted into a multiple unicast problem without changing the solvability or linear solvability of the problem. Thus, it suffices to study the multiple unicast case as far as capacity is concerned. The conversion procedure uses the gadget shown in Figure 3.7. Suppose t_1 and t_2 are sink nodes in an arbitrary directed network that both demand source X. We then add five additional nodes as shown, where node 1 is an additional source demanded by node 5, and node 4 demands source X. In the resulting network, t_1 and t_2 are not sink nodes but must decode X in order to satisfy the demands of the new sinks 4 and 5.

For an alternative undirected wired network model where each arc's capacity can be arbitrarily partitioned between the two directions of information flow, it was conjectured in [90] that network coding does not increase throughput for multiple unicast sessions. This is not however

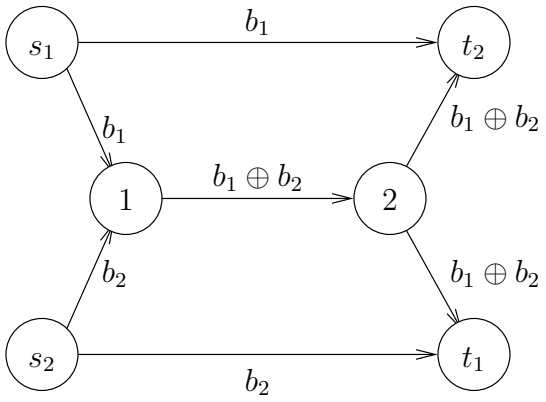


Fig. 3.8. The two-unicast butterfly network. Each arc represents a directed arc that is capable of carrying a single packet reliably. There is one packet b_1 present at source node s_1 that we wish to communicate to sink node t_1 and one packet b_2 present at source node s_2 that we wish to communicate to sink node t_2. $b_1 \oplus b_2$ is the packet formed from the bitwise XOR of the packets b_1 and b_2.

the case for undirected wireless networks, as we will see in the following section.

3.5 Constructive approaches

The complexity of inter-session network coding motivates consideration of suboptimal but useful classes of network codes that are feasible to construct and implement in practice. A number of constructive approaches are based on generalizations of simple example networks where bitwise XOR coding is useful.

3.5.1 Pairwise XOR coding in wireline networks

Figure 3.8, which we have seen before in Chapter 1, gives a simple example of a wireline network where bitwise XOR coding across two unicast sessions doubles the common throughput that can be simultaneously achieved for each session. This example has a "poison-antidote" interpretation: the coded $b_1 \oplus b_2$ packet is called a *poison* packet, since by itself it is not useful to either sink; each sink needs an *antidote* packet from the other session's source in order to recover its own session's packet.

To generalize this example, each arc can be replaced by a path seg-

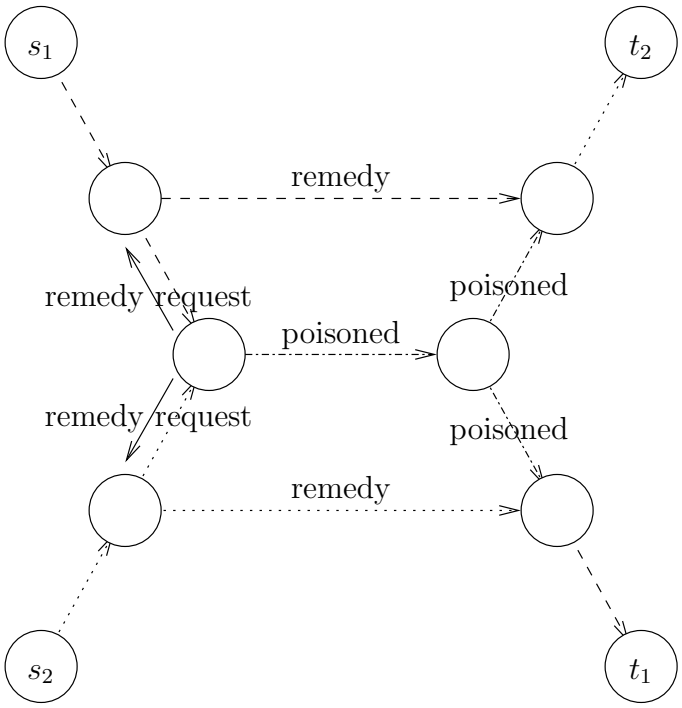


Fig. 3.9. The canonical poison-antidote scenario. This is an example of inter-session XOR coding between two unicast sessions where decoding must take place at intermediate non-sink nodes.

ment. The antidote segment can be between intermediate nodes rather than directly from source to sink node, as shown in Figure 3.9. We can also consider a stream of packets rather than a single packet from each source.

For more than two unicast sessions, a tractable approach is to restrict consideration to XOR coding across pairs of unicasts. The throughput-optimization problem then becomes the problem of optimally packing, in the network, poison-antidote butterfly structures of the form shown in Figure 3.9. This can be formulated as a linear optimization problem, which we discuss in Section 5.1.3.

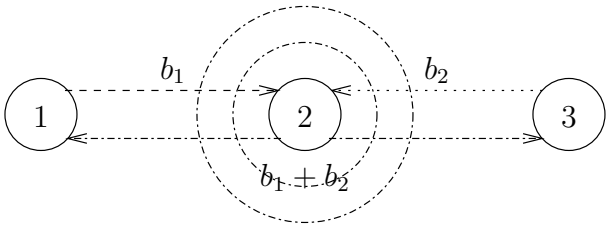


Fig. 3.10. Information exchange via a single relay node.

3.5.2 XOR coding in wireless networks

3.5.2.1 Canonical scenarios

The broadcast nature of the wireless medium gives rise to more situations where network coding is beneficial. The simplest example, where two sources exchange packets via a common relay node, is given in Figure 3.10 (it is the same as Figure 1.4 of Chapter 1, but drawn without the hyperarcs explicitly shown).

For the case where the two sources exchange a stream of packets, this example can be generalized by replacing the single relay node with a multiple-relay path, as illustrated in Figure 3.11. Each coded broadcast transmission by a relay node can transfer useful information in both directions, replacing two uncoded point-to-point transmissions. This canonical scenario is called the *information exchange scenario*.

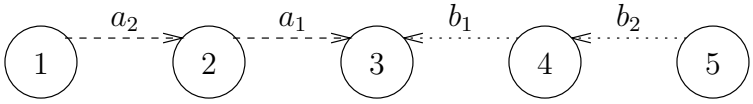
The information exchange scenario involves wireless broadcast of coded packets by relay nodes. The *path intersection scenario*, illustrated in Figure 3.12, involves wireless broadcast of coded packets as well as uncoded packets which are used for decoding the coded packets. The information exchange and path intersection scenarios are called wireless one-hop XOR coding scenarios, since each coded packet travels one hop before being decoded.

The poison-antidote scenario described earlier for wireline networks can also be adapted to wireless networks. The information exchange scenario and the path intersection scenario with two unicast sessions can be viewed as special cases where the coded packets travel exactly one hop, and the antidote packets travel zero and one hop respectively.

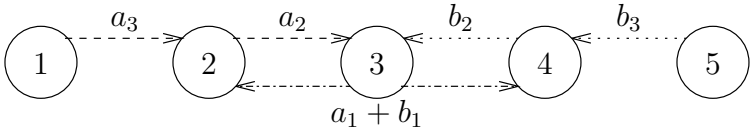
3.5.2.2 Opportunistic coding

The *Completely Opportunistic Coding* (COPE) protocol, proposed by Katti et al., is based on the canonical wireless one-hop XOR coding sce-

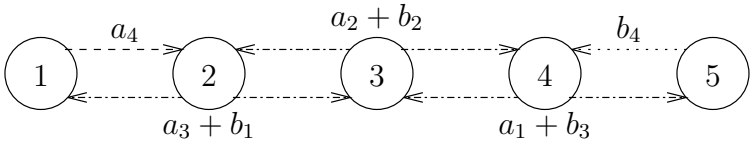
(a) The initial packets are sent uncoded from each source ...



(b) ... until a packet from each source reaches an intermediate relay node.



(c) This node can take the bitwise XOR of the two packets to form a coded packet. It then broadcasts the coded packet to its two neighbor nodes with a single wireless transmission. Each neighbor node has already seen (and stored) one of the two original packets, and so is able to decode the other. Thus, the coded broadcast is equivalent to forwarding the two original packets individually.



(d) Each neighbor node now has a packet from each source, and can similarly XOR them to form a coded packet, which is then broadcast.

Fig. 3.11. The multiple-relay information exchange scenario. After [138].

narios described in the previous section, and operates over any underlying routing protocol (e.g., shortest path or geographic routing) and medium access protocol (e.g., 802.11 MAC).

There is an *opportunistic listening (overhearing)* component: the broadcast nature of the wireless medium allows nodes to overhear transmissions intended for other nodes; all overheard packets are stored for some period of time. Nodes periodically send *reception reports*, which can be annotations to their own transmissions, informing their neighbors of which packets they have overheard.

In the *opportunistic coding* component, at each transmission opportunity, a node makes a decision on which of its queued packets to code and transmit, based on the packets' next-hop nodes and which of these packets have been overheard by the next-hop nodes. A node with packets to

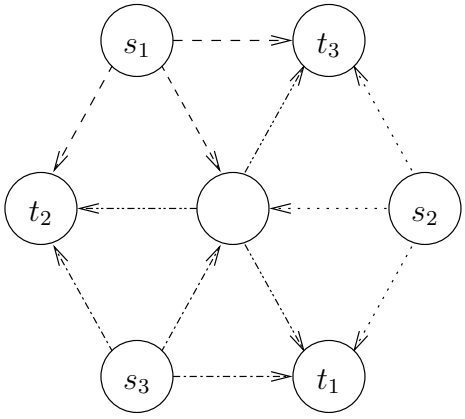


Fig. 3.12. The path intersection scenario with three unicast sessions. Each source communicates with its sink via a common relay node, and each source's transmission is received by the relay node as well as the other two sessions' sink nodes. The relay node broadcasts the bitwise XOR of the three sources' packets. Each sink can decode as it has the other two sources' packets. Reprinted with permission from [39].

forward to different next hop neighbors looks for the largest subset $\mathcal{S}$ of packets such that

- each packet $u \in \mathcal{S}$ has a different next hop node v_u
- each of these next hop nodes v_u already has all the packets in $\mathcal{S}$ except u.

The packets in $\mathcal{S}$ are XORed together to form a coded packet which is broadcast to nodes $v_u, u \in \mathcal{S}$, each of which has enough information to decode its intended packet u. This policy maximizes the number of packets that are communicated by its transmission, assuming that the nodes that receive the packet will attempt to decode immediately upon reception. For example, consider the situation illustrated in Figure 3.13. In this situation, the coding decision made by node 1 is to send out the packet $b_1 \oplus b_2 \oplus b_3$, because this allows three packets, b_1, b_2, and b_3, to be communicated to their next hops, while allowing the nodes that receive the packet to decode upon reception and recover the packets that they each desire. Node 1 does not, for example, send out the packet $b_1 \oplus b_2 \oplus b_4$, because only node 2 is able to immediately decode and recover the packet that it desires, b_4. The ability to make the coding decision requires each node to know the contents of the queues of the neighboring nodes.

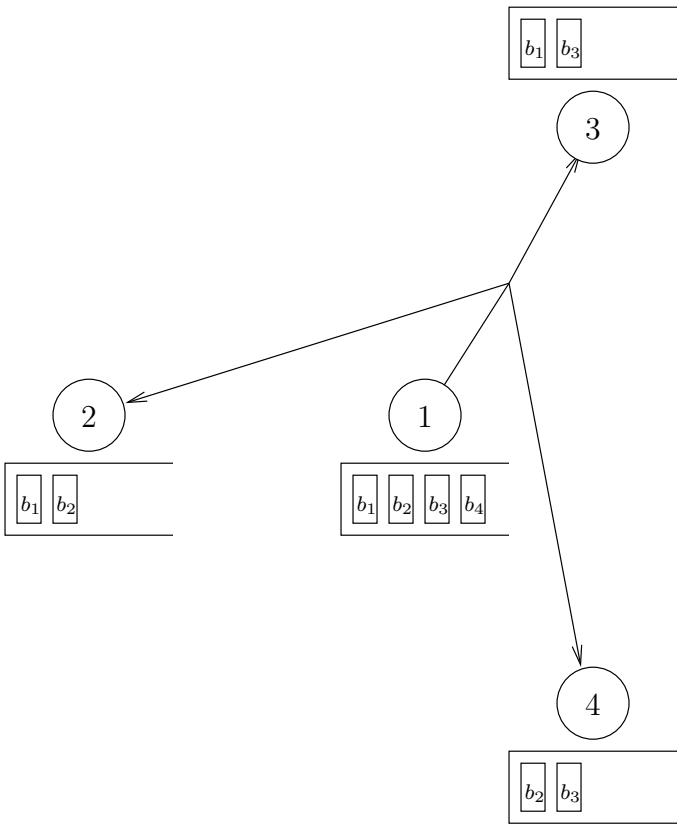


Fig. 3.13. An example of the Katti et al. queue-length-based approach. Suppose the next hop for packet b_1 is node 4, the next hop for packet b_2 is node 3, and the next hop for packets b_3 and b_4 is node 2. Node 1 has a transmission opportunity on its lossless broadcast arc reaching nodes 2, 3, and 4. The decision it makes is to send out the packet $b_1 \oplus b_2 \oplus b_3$, because this allows three packets, b_1, b_2, and b_3, to be communicated to their next hops. After [78].

Experiments have shown that opportunistic coding can significantly improve throughput in ad hoc wireless networks using 802.11 and geographic routing, particularly under congested network conditions. We discuss COPE further in Section 5.2.2, in relation to subgraph selection.

3.5.2.3 Coding-influenced routing

There can be greater scope for network coding gains when routing is influenced by potential coding opportunities. For instance, consider the

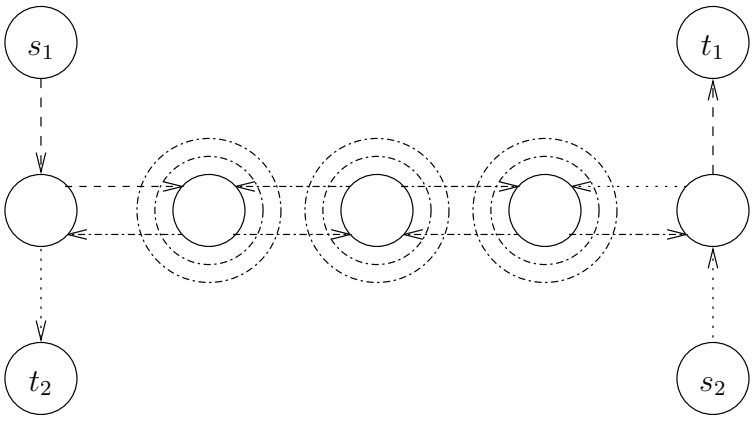


Fig. 3.14. The reverse carpooling scenario.

information exchange scenario, which involves two unicast flows such that the source of each unicast is co-located with the sink of the other. This can be generalized to the case of two unicast sessions whose source and sink nodes are not co-located, by selecting routes for the two sessions which overlap in opposite directions. In analogy to carpooling, each session's route may involve a detour which is compensated for by the sharing of transmissions on the common portion of the route, hence the name *reverse carpooling*. An illustration is given in Figure 3.14.

The problem of finding the best solution within a class of network codes is a subgraph selection problem. Tractable optimization problems can be obtained by limiting consideration to strategies involving reverse carpooling, poison-antidote and/or path intersection scenarios with a limit on the number of constituent sessions of a coded packet. This is discussed in Sections 5.1.3 and 5.2.2.

3.6 Notes and further reading

The algebraic characterization and construction of scalar linear non-multicast network codes presented in this chapter is from Koetter and Médard [85]. Rasala Lehman and Lehman [116] determined the complexity classes of different scalar linear network coding problems. Example networks requiring vector linear rather than scalar linear coding solutions were given in Rasala Lehman and Lehman [116], Médard et al. [104] and Riis [119]. Dougherty et al. showed in [38] that linear

coding is insufficient in general for non-multicast networks, and in [37] that Shannon-type inequalities are insufficient in general for analyzing network coding capacity, using connections between matroid theory and network coding described in [37]. Group network codes have been considered by Chan [21].

The entropy function-based characterization of capacity for acyclic networks presented in this chapter is due to Yan et al. [145]. Entropy-based approaches and other techniques for bounding communication rates in non-multicast problems have been given in various works including [104, 38, 147, 86, 2, 53].

The conversion processThe study of network coding on the undirected wired network model was initiated by Li and Li [89].

On the code construction side, the poison-antidote approach for multiple unicast network coding was introduced by Ratnakar et al. [117, 118]. The information exchange scenario was introduced by Wu et al. [139]. The (COPE) protocol was developed by Katti et al. [77, 78]. We discuss work on subgraph selection for inter-session network coding in Chapter 5.

4
Network Coding in Lossy Networks

In this chapter, we discuss the use of network coding, particularly random linear network coding, in lossy networks with packet erasures. The main result that we establish is that random linear network coding achieves the capacity of a single connection (unicast or multicast) in a given coding subgraph, i.e., by efficiently providing robustness, random linear network coding allows a connection to be established at the maximum throughput that is possible in a given coding subgraph.

Throughout this chapter, we assume that a coding subgraph is given; we address the problem of subgraph selection in Chapter 5, where we also discuss whether separating coding and subgraph selection is optimal. The lossy coding subgraphs we consider here are applicable to various types of network, including multi-hop wireless networks and peer-to-peer networks. In the latter case, losses are not caused so much by unreliable links, but rather by unreliable nodes that intermittently join and leave the network.

We model coding subgraphs using the time-expanded subgraphs described in Section 1.3. Recall that a time-expanded subgraph describes the times and locations at which packet injections and receptions occur. Since a coding subgraph specifies only to the times at which packet injections occur, a time-expanded subgraph is in fact an element in the random ensemble of a coding subgraph. A time-expanded subgraph is shown in Figure 4.1.

For simplicity, we assume that links are delay-free in the sense that the arrival time of a received packet corresponds to the time that it was injected to the link. This assumption does not alter the results that we derive in this chapter, and a link with delay can be converted into delay-free links by introducing an artificial node implementing a trivial coding function. On the left of Figure 4.2, we show a fragment of a

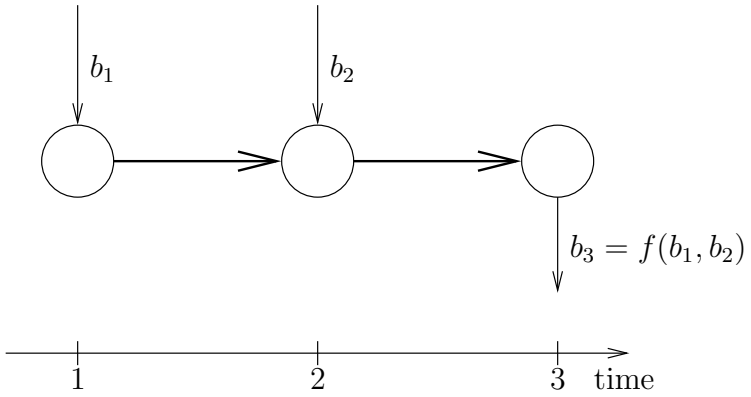


Fig. 4.1. Coding at a node in a time-expanded subgraph. Packet b_1 is received at time 1, and packet b_2 is received at time 2. The thick, horizontal arcs have infinite capacity, and represent data stored at a node. Thus, at time 3, packets b_1 and b_2 can be used to form packet b_3.

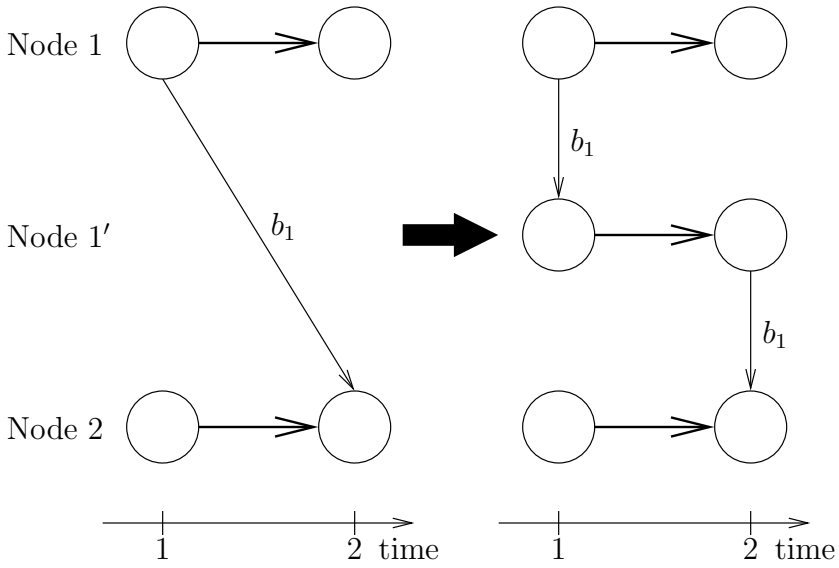


Fig. 4.2. Conversion of a link with delay into delay-free links.

time-expanded subgraph that depicts a transmission with delay. The packet b_1 is injected by node 1 at time 1, and this packet is not received by node 2 until time 2. On the right, we depict the same transmission, but we introduce an artificial node $1'$. Node $1'$ does nothing but store b_1

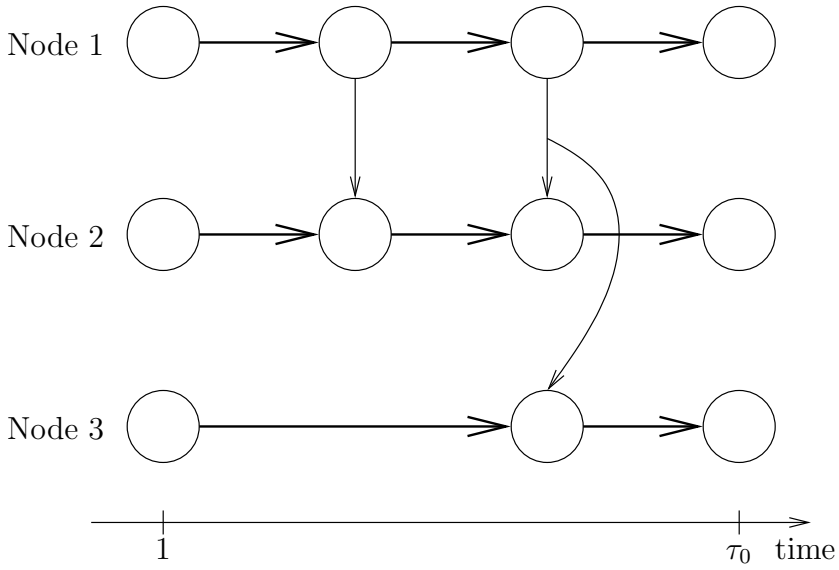


Fig. 4.3. A time-expanded subgraph where, between time 0 and time τ, a packet is successfully transmitted from node 1 to node 2 and another is successfully transmitted from node 1 to both nodes 2 and 3.

then transmit it out again at time 2. In this way, we can suppose that there is a delay-free transmission from node 1 to node $1'$ at time 1 and one from node $1'$ to node 2 at time 2.

Let A_{iJ} be the counting process describing the arrival of packets that are injected on hyperarc (i, J), and let A_{iJK} be the counting process describing the arrival of packets that are injected on hyperarc (i, J) and received by exactly the set of nodes $K \subset J$; i.e., for $\tau \geq 0$, $A_{iJ}(\tau)$ is the total number of packets that are injected on hyperarc (i, J) between time 0 and time τ, and $A_{iJK}(\tau)$ is the total number of packets that are injected on hyperarc (i, J) and received by all nodes in K (and no nodes in $\mathcal{N} \setminus K$) between time 0 and time τ. For example, suppose that three packets are injected on hyperarc $(1, \{2, 3\})$ between time 0 and time τ_0 and that, of these three packets, one is received by node 2 only, one is lost entirely, and one is received by both nodes 2 and 3; then we have $A_{1(23)}(\tau_0) = 3$, $A_{1(23)\emptyset}(\tau_0) = 1$, $A_{1(23)2}\tau_0) = 1$, $A_{1(23)3}(\tau_0) = 0$, and $A_{1(23)(23)}(\tau_0) = 1$. A possible time-expanded subgraph corresponding to these events is shown in Figure 4.3.

We assume that A_{iJ} has an average rate z_{iJ} and that A_{iJK} has an

average rate z_{iJK}; more precisely, we assume that

$$\lim_{\tau\to\infty} \frac{A_{iJ}(\tau)}{\tau} = z_{iJ}$$

and that

$$\lim_{\tau\to\infty} \frac{A_{iJK}(\tau)}{\tau} = z_{iJK}$$

almost surely. Hence we have $z_{iJ} = \sum_{K\subset J} z_{iJK}$ and, if the link is lossless, we have $z_{iJK} = 0$ for all $K \subsetneq J$. The vector z, consisting of z_{iJ}, $(i, J) \in \mathcal{A}$, is the coding subgraph that we are given.

In Section 4.1, we specify precisely what we mean by random linear network coding in a lossy network, then, in Section 4.2, we establish the main result of this chapter: we show that random linear network coding achieves the capacity of a single connection in a given coding subgraph. This result is concerned only with throughput and does not consider delay: in Section 4.3, we strengthen the result in the special case of Poisson traffic with i.i.d. losses by giving error exponents. These error exponents allow us to quantify the rate of decay of the probability of error with coding delay and to determine the parameters of importance in this decay.

4.1 Random linear network coding

The specific coding scheme we consider is as follows. We suppose that, at the source node, we have K message packets $w_1, w_2, \ldots, w_K$, which are vectors of length λ over some finite field $\mathbb{F}_q$. (If the packet length is b bits, then we take $\lambda = \lceil b/\log_2 q \rceil$.) The message packets are initially present in the memory of the source node.

The coding operation performed by each node is simple to describe and is the same for every node: received packets are stored into the node's memory, and packets are formed for injection with random linear combinations of its memory contents whenever a packet injection occurs on an outgoing link. The coefficients of the combination are drawn uniformly from $\mathbb{F}_q$.

Since all coding is linear, we can write any packet u in the network as a linear combination of $w_1, w_2, \ldots, w_K$, namely, $u = \sum_{k=1}^{K} \gamma_k w_k$. We call γ the *global encoding vector* of u, and we assume that it is sent along with u, as side information in its header. The overhead this incurs (namely, $K \log_2 q$ bits) is negligible if packets are sufficiently large.

Nodes are assumed to have unlimited memory. The scheme can be

Initialization:

- The source node stores the message packets $w_1, w_2, \ldots, w_K$ in its memory.

Operation:

- When a packet is received by a node,
  - the node stores the packet in its memory.
- When a packet injection occurs on an outgoing link of a node,
  - the node forms the packet from a random linear combination of the packets in its memory. Suppose the node has L packets $u_1, u_2, \ldots, u_L$ in its memory. Then the packet formed is

$$u_0 := \sum_{l=1}^{L} \alpha_l u_l,$$

    where α_l is chosen according to a uniform distribution over the elements of $\mathbb{F}_q$. The packet's global encoding vector γ, which satisfies $u_0 = \sum_{k=1}^{K} \gamma_k w_k$, is placed in its header.

Decoding:

- Each sink node performs Gaussian elimination on the set of global encoding vectors from the packets in its memory. If it is able to find an inverse, it applies the inverse to the packets to obtain $w_1, w_2, \ldots, w_K$; otherwise, a decoding error occurs.

Fig. 4.4. Summary of the random linear network coding scheme used in this chapter (cf. Section 2.5.1.1).

modified so that received packets are stored into memory only if their global encoding vectors are linearly-independent of those already stored. This modification keeps our results unchanged while ensuring that nodes never need to store more than K packets.

A sink node collects packets and, if it has K packets with linearly-independent global encoding vectors, it is able to recover the message packets. Decoding can be done by Gaussian elimination. The scheme can be run either for a predetermined duration or, in the case of rateless operation, until successful decoding at the sink nodes. We summarize the scheme in Figure 4.4.

The scheme is carried out for a single block of K message packets at the source. If the source has more packets to send, then the scheme is repeated with all nodes flushed of their memory contents.

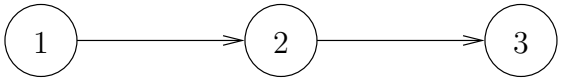


Fig. 4.5. A network consisting of two point-to-point links in tandem. Reprinted with permission from [93].

4.2 Coding theorems

In this section, we specify achievable rate intervals for random linear network coding in various scenarios. The fact that the intervals we specify are the largest possible (i.e., that random linear network coding is capacity-achieving) can be seen by simply noting that the rate of a connection must be limited by the rate at which distinct packets are being received over any cut between the source and the sink. A formal converse can be obtained using the cut-set bound for multi-terminal networks (see [28, Section 14.10]).

4.2.1 Unicast connections

4.2.1.1 Two-link tandem network

We develop our general result for unicast connections by extending from some special cases. We begin with the simplest non-trivial case: that of two point-to-point links in tandem (see Figure 4.5).

Suppose we wish to establish a connection of rate arbitrarily close to R packets per unit time from node 1 to node 3. Suppose further that random linear network coding is run for a total time Δ, from time 0 until time Δ, and that, in this time, a total of N packets is received by node 2. We call these packets $v_1, v_2, \ldots, v_N$.

Any packet u received by a node is a linear combination of $v_1, v_2, \ldots, v_N$, so we can write

$$u = \sum_{n=1}^{N} \beta_n v_n.$$

Now, since v_n is formed by a random linear combination of the message packets $w_1, w_2, \ldots, w_K$, we have

$$v_n = \sum_{k=1}^{K} \alpha_{nk} w_k$$

for $n = 1, 2, \ldots, N$. Hence

$$u = \sum_{k=1}^{K} \left(\sum_{n=1}^{N} \beta_n \alpha_{nk} \right) w_k,$$

and it follows that the kth component of the global encoding vector of u is given by

$$\gamma_k = \sum_{n=1}^{N} \beta_n \alpha_{nk}.$$

We call the vector β associated with u the *auxiliary encoding vector* of u, and we see that any node that receives $\lfloor K(1+\varepsilon) \rfloor$ or more packets with linearly-independent auxiliary encoding vectors has $\lfloor K(1+\varepsilon) \rfloor$ packets whose global encoding vectors collectively form a random $\lfloor K(1+\varepsilon) \rfloor \times K$ matrix over $\mathbb{F}_q$, with all entries chosen uniformly. If this matrix has rank K, then node 3 is able to recover the message packets. The probability that a random $\lfloor K(1+\varepsilon) \rfloor \times K$ matrix has rank K is, by a simple counting argument, $\prod_{k=1+\lfloor K(1+\varepsilon) \rfloor - K}^{\lfloor K(1+\varepsilon) \rfloor} (1 - 1/q^k)$, which can be made arbitrarily close to 1 by taking K arbitrarily large. Therefore, to determine whether node 3 can recover the message packets, we essentially need only to determine whether it receives $\lfloor K(1+\varepsilon) \rfloor$ or more packets with linearly-independent auxiliary encoding vectors.

Our proof is based on tracking the propagation of what we call *innovative* packets. Such packets are innovative in the sense that they carry new, as yet unknown, information about $v_1, v_2, \ldots, v_N$ to a node. It turns out that the propagation of innovative packets through a network follows the propagation of jobs through a queueing network, for which fluid flow models give good approximations. We first give a heuristic argument in terms of this fluid analogy before proceeding to a formal argument.

Since the packets being received by node 2 are the packets $v_1, v_2, \ldots, v_N$ themselves, it is clear that every packet being received by node 2 is innovative. Thus, innovative packets arrive at node 2 at a rate of z_{122}, and this can be approximated by fluid flowing in at rate z_{122}. These innovative packets are stored in node 2's memory, so the fluid that flows in is stored in a reservoir.

Packets, now, are being received by node 3 at a rate of z_{233}, but whether these packets are innovative depends on the contents of node 2's memory. If node 2 has more information about $v_1, v_2, \ldots, v_N$ than node 3 does, then it is highly likely that new information will be described to

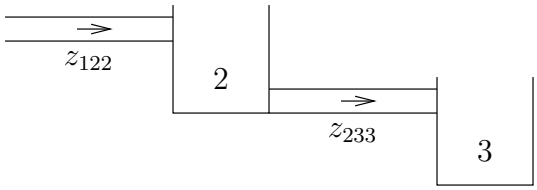


Fig. 4.6. Fluid flow system corresponding to two-link tandem network. Reprinted with permission from [93].

node 3 in the next packet that it receives. Otherwise, if node 2 and node 3 have the same degree of information about $v_1, v_2, \ldots, v_N$, then packets received by node 3 cannot possibly be innovative. Thus, the situation is as though fluid flows into node 3's reservoir at a rate of z_{233}, but the level of node 3's reservoir is restricted from ever exceeding that of node 2's reservoir. The level of node 3's reservoir, which is ultimately what we are concerned with, can equivalently be determined by fluid flowing out of node 2's reservoir at rate z_{233}.

We therefore see that the two-link tandem network in Figure 4.5 maps to the fluid flow system shown in Figure 4.6. It is clear that, in this system, fluid flows into node 3's reservoir at rate $\min(z_{122}, z_{233})$. This rate determines the rate at which packets with new information about $v_1, v_2, \ldots, v_N$—and, therefore, linearly-independent auxiliary encoding vectors—arrive at node 3. Hence the time required for node 3 to receive $\lfloor K(1+\varepsilon) \rfloor$ packets with linearly-independent auxiliary encoding vectors is, for large K, approximately $K(1+\varepsilon)/\min(z_{122}, z_{233})$, which implies that a connection of rate arbitrarily close to R packets per unit time can be established provided that

$$R \leq \min(z_{122}, z_{233}). \tag{4.1}$$

The right-hand side of (4.1) is indeed the capacity of the two-link tandem network, and we therefore have the desired result for this case.

We now proceed to establish the result formally. All packets received by node 2, namely $v_1, v_2, \ldots, v_N$, are considered innovative. We associate with node 2 the set of vectors U, which varies with time and is initially empty, i.e., $U(0) := \emptyset$. If packet u is received by node 2 at time τ, then its auxiliary encoding vector β is added to U at time τ, i.e., $U(\tau^+) := \{\beta\} \cup U(\tau)$.

We associate with node 3 the set of vectors W, which again varies with time and is initially empty. Suppose packet u, with auxiliary encoding

vector β, is received by node 3 at time τ. Let μ be a positive integer, which we call the *innovation order*. Then we say u is innovative if $\beta \notin \text{span}(W(\tau))$ and $|U(\tau)| > |W(\tau)| + \mu - 1$. If u is innovative, then β is added to W at time τ.†

The definition of innovative is designed to satisfy two properties: First, we require that $W(\Delta)$, the set of vectors in W when the scheme terminates, is linearly independent. Second, we require that, when a packet is received by node 3 and $|U(\tau)| > |W(\tau)|+\mu-1$, it is innovative with high probability. The innovation order μ is an arbitrary factor that ensures that the latter property is satisfied.

Suppose $|U(\tau)| > |W(\tau)| + \mu - 1$. Since u is a random linear combination of vectors in $U(\tau)$, it follows that u is innovative with some nontrivial probability. More precisely, because β is uniformly-distributed over $q^{|U(\tau)|}$ possibilities, of which at least $q^{|U(\tau)|} - q^{|W(\tau)|}$ are not in $\text{span}(W(\tau))$, it follows that

$$\begin{aligned}\Pr(\beta \notin \text{span}(W(\tau))) &\geq \frac{q^{|U(\tau)|} - q^{|W(\tau)|}}{q^{|U(\tau)|}} \\ &= 1 - q^{|W(\tau)|-|U(\tau)|} \\ &\geq 1 - q^{-\mu}.\end{aligned}$$

Hence u is innovative with probability at least $1 - q^{-\mu}$. Since we can always discard innovative packets, we assume that the event occurs with probability exactly $1 - q^{-\mu}$. If instead $|U(\tau)| \leq |W(\tau)| + \mu - 1$, then we see that u cannot be innovative, and this remains true at least until another arrival occurs at node 2. Therefore, for an innovation order of μ, the propagation of innovative packets through node 2 is described by the propagation of jobs through a single-server queueing station with queue size $(|U(\tau)| - |W(\tau)| - \mu + 1)^+$, where, for a real number x, $(x)^+ := \max(x, 0)$. We similarly define $(x)^- := \max(-x, 0)$.

The queueing station is serviced with probability $1-q^{-\mu}$ whenever the queue is non-empty and a received packet arrives on arc $(2, 3)$. We can equivalently consider "candidate" packets that arrive with probability $1 - q^{-\mu}$ whenever a received packet arrives on arc $(2, 3)$ and say that

† This definition of innovative differs from merely being informative, which is the sense in which innovative is used in [27]. Indeed, a packet can be informative, in the sense that in gives a node some new, as yet unknown, information about $v_1, v_2, \ldots, v_N$ (or about $w_1, w_2, \ldots, w_K$), and not satisfy this definition of innovative. We have defined innovative so that innovative packets are informative (with respect to other innovative packets at the node), but not necessarily conversely. This allows us to bound, or dominate, the behavior of random linear network coding, though we cannot describe it exactly.

the queueing station is serviced whenever the queue is non-empty and a candidate packet arrives on arc $(2, 3)$. We consider all packets received on arc $(1, 2)$ to be candidate packets.

The system we wish to analyze, therefore, is the following simple queueing system: Jobs arrive at node 2 according to the arrival of received packets on arc $(1, 2)$ and, with the exception of the first $\mu-1$ jobs, enter node 2's queue. The jobs in node 2's queue are serviced by the arrival of candidate packets on arc $(2, 3)$ and exit after being serviced. The number of jobs exiting is a lower bound on the number of packets with linearly-independent auxiliary encoding vectors received by node 3.

We analyze the queueing system of interest using the fluid approximation for discrete-flow networks (see, e.g., [24, 25]). We do not explicitly account for the fact that the first $\mu - 1$ jobs arriving at node 2 do not enter its queue because this fact has no effect on job throughput. Let B_1, B, and C be the counting processes for the arrival of received packets on arc $(1, 2)$, of innovative packets on arc $(2, 3)$, and of candidate packets on arc $(2, 3)$, respectively. Let $Q(\tau)$ be the number of jobs queued for service at node 2 at time τ. Hence $Q = B_1 - B$. Let $X := B_1 - C$ and $Y := C - B$. Then

$$Q = X + Y. \tag{4.2}$$

Moreover, we have

$$Q(\tau)dY(\tau) = 0, \tag{4.3}$$

$$dY(\tau) \geq 0, \tag{4.4}$$

and

$$Q(\tau) \geq 0 \tag{4.5}$$

for all $\tau \geq 0$, and

$$Y(0) = 0. \tag{4.6}$$

We observe now that equations (4.2)–(4.6) give us the conditions for a Skorohod problem (see, e.g., [25, Section 7.2]) and, by the oblique reflection mapping theorem, there is a well-defined, Lipschitz-continuous mapping Φ such that $Q = \Phi(X)$.

Let

$$\bar{C}^{(K)}(\tau) := \frac{C(K\tau)}{K},$$
$$\bar{X}^{(K)}(\tau) := \frac{X(K\tau)}{K},$$

and

$$\bar{Q}^{(K)}(\tau) := \frac{Q(K\tau)}{K}.$$

Recall that A_{233} is the counting process for the arrival of received packets on arc $(2,3)$. Therefore, $C(\tau)$ is the sum of $A_{233}(\tau)$ Bernoulli-distributed random variables with parameter $1-q^{-\mu}$. Hence

$$\begin{aligned}\bar{C}(\tau) &:= \lim_{K\to\infty} \bar{C}^{(K)}(\tau) \\ &= \lim_{K\to\infty} (1-q^{-\mu})\frac{A_{233}(K\tau)}{K} \qquad \text{a.s.} \\ &= (1-q^{-\mu})z_{233}\tau \qquad \text{a.s.},\end{aligned}$$

where the last equality follows by the assumptions of the model. Therefore

$$\bar{X}(\tau) := \lim_{K\to\infty} \bar{X}^{(K)}(\tau) = (z_{122} - (1-q^{-\mu})z_{233})\tau \qquad \text{a.s.}$$

By the Lipschitz-continuity of Φ, then, it follows that $\bar{Q} := \lim_{K\to\infty} \bar{Q}^{(K)} = \Phi(\bar{X})$, i.e., $\bar{Q}$ is, almost surely, the unique $\bar{Q}$ that satisfies, for some $\bar{Y}$,

$$\bar{Q}(\tau) = (z_{122} - (1-q^{-\mu})z_{233})\tau + \bar{Y}, \tag{4.7}$$
$$\bar{Q}(\tau)d\bar{Y}(\tau) = 0, \tag{4.8}$$
$$d\bar{Y}(\tau) \geq 0, \tag{4.9}$$

and

$$\bar{Q}(\tau) \geq 0 \tag{4.10}$$

for all $\tau \geq 0$, and

$$\bar{Y}(0) = 0. \tag{4.11}$$

A pair $(\bar{Q}, \bar{Y})$ that satisfies (4.7)–(4.11) is

$$\bar{Q}(\tau) = (z_{122} - (1-q^{-\mu})z_{233})^{+}\tau \tag{4.12}$$

and

$$\bar{Y}(\tau) = (z_{122} - (1-q^{-\mu})z_{233})^{-}\tau.$$

Hence $\bar{Q}$ is given by equation (4.12).

Fig. 4.7. A network consisting of L point-to-point links in tandem. Reprinted with permission from [93].

Recall that node 3 can recover the message packets with high probability if it receives $\lfloor K(1+\varepsilon)\rfloor$ packets with linearly-independent auxiliary encoding vectors and that the number of jobs exiting the queueing system is a lower bound on the number of packets with linearly-independent auxiliary encoding vectors received by node 3. Therefore, node 3 can recover the message packets with high probability if $\lfloor K(1+\varepsilon)\rfloor$ or more jobs exit the queueing system. Let ν be the number of jobs that have exited the queueing system by time Δ. Then

$$\nu = B_1(\Delta) - Q(\Delta).$$

Take $K = \lceil (1-q^{-\mu})\Delta R_c R/(1+\varepsilon)\rceil$, where $0 < R_c < 1$. Then

$$\begin{aligned}\lim_{K\to\infty} \frac{\nu}{\lfloor K(1+\varepsilon)\rfloor} &= \lim_{K\to\infty} \frac{B_1(\Delta) - Q(\Delta)}{K(1+\varepsilon)} \\ &= \frac{z_{122} - (z_{122} - (1-q^{-\mu})z_{233})^+}{(1-q^{-\mu})R_c R} \\ &= \frac{\min(z_{122}, (1-q^{-\mu})z_{233})}{(1-q^{-\mu})R_c R} \\ &\geq \frac{1}{R_c}\frac{\min(z_{122}, z_{233})}{R} > 1\end{aligned}$$

provided that

$$R \leq \min(z_{122}, z_{233}). \tag{4.13}$$

Hence, for all R satisfying (4.13), $\nu \geq \lfloor K(1+\varepsilon)\rfloor$ with probability arbitrarily close to 1 for K sufficiently large. The rate achieved is

$$\frac{K}{\Delta} \geq \frac{(1-q^{-\mu})R_c}{1+\varepsilon} R,$$

which can be made arbitrarily close to R by varying μ, R_c, and ε.

4.2.1.2 L-link tandem network

We extend our result to another special case before considering general unicast connections: we consider the case of a tandem network consisting of L point-to-point links and $L+1$ nodes (see Figure 4.7).

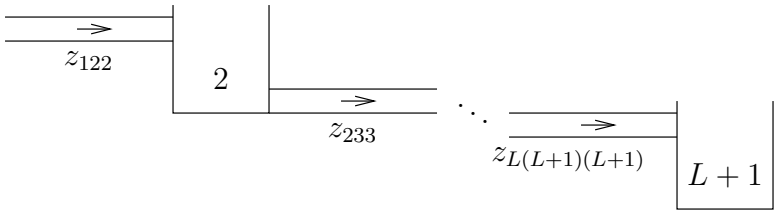


Fig. 4.8. Fluid flow system corresponding to L-link tandem network. Reprinted with permission from [93].

This case is a straightforward extension of that of the two-link tandem network. It maps to the fluid flow system shown in Figure 4.8. In this system, it is clear that fluid flows into node $(L+1)$'s reservoir at rate $\min_{1\leq i\leq L}\{z_{i(i+1)(i+1)}\}$. Hence a connection of rate arbitrarily close to R packets per unit time from node 1 to node $L+1$ can be established provided that

$$R \leq \min_{1\leq i\leq L}\{z_{i(i+1)(i+1)}\}. \tag{4.14}$$

Since the right-hand side of (4.14) is indeed the capacity of the L-link tandem network, we therefore have the desired result for this case.

The formal argument requires care. For $i = 2, 3, \ldots, L+1$, we associate with node i the set of vectors V_i, which varies with time and is initially empty. We define $U := V_2$ and $W := V_{L+1}$. As in the case of the two-link tandem, all packets received by node 2 are considered innovative and, if packet u is received by node 2 at time τ, then its auxiliary encoding vector β is added to U at time τ. For $i = 3, 4, \ldots, L+1$, if packet u, with auxiliary encoding vector β, is received by node i at time τ, then we say u is innovative if $\beta \notin \text{span}(V_i(\tau))$ and $|V_{i-1}(\tau)| > |V_i(\tau)| + \mu - 1$. If u is innovative, then β is added to V_i at time τ.

This definition of innovative is a straightforward extension of that in Section 4.2.1.1. The first property remains the same: we continue to require that $W(\Delta)$ is a set of linearly-independent vectors. We extend the second property so that, when a packet is received by node i for any $i = 3, 4, \ldots, L+1$ and $|V_{i-1}(\tau)| > |V_i(\tau)| + \mu - 1$, it is innovative with high probability.

Take some $i \in \{3, 4, \ldots, L+1\}$. Suppose that packet u, with auxiliary encoding vector β, is received by node i at time τ and that $|V_{i-1}(\tau)| > |V_i(\tau)| + \mu - 1$. Thus, the auxiliary encoding vector β is a random linear combination of vectors in some set V_0 that contains $V_{i-1}(\tau)$. Hence,

because β is uniformly-distributed over $q^{|V_0|}$ possibilities, of which at least $q^{|V_0|} - q^{|V_i(\tau)|}$ are not in $\text{span}(V_i(\tau))$, it follows that

$$\begin{aligned}\Pr(\beta \notin \text{span}(V_i(\tau))) &\geq \frac{q^{|V_0|} - q^{|V_i(\tau)|}}{q^{|V_0|}} \\ &= 1 - q^{|V_i(\tau)|-|V_0|} \\ &\geq 1 - q^{|V_i(\tau)|-|V_{i-1}(\tau)|} \\ &\geq 1 - q^{-\mu}.\end{aligned}$$

Therefore u is innovative with probability at least $1 - q^{-\mu}$. Following the argument in Section 4.2.1.1, we see, for all $i = 2, 3, \ldots, L$, that the propagation of innovative packets through node i is described by the propagation of jobs through a single-server queueing station with queue size $(|V_i(\tau)|-|V_{i+1}(\tau)|-\mu+1)^+$ and that the queueing station is serviced with probability $1-q^{-\mu}$ whenever the queue is non-empty and a received packet arrives on arc $(i, i+1)$. We again consider candidate packets that arrive with probability $1 - q^{-\mu}$ whenever a received packet arrives on arc $(i, i+1)$ and say that the queueing station is serviced whenever the queue is non-empty and a candidate packet arrives on arc $(i, i+1)$.

The system we wish to analyze in this case is therefore the following simple queueing network: Jobs arrive at node 2 according to the arrival of received packets on arc $(1, 2)$ and, with the exception of the first $\mu-1$ jobs, enter node 2's queue. For $i = 2, 3, \ldots, L-1$, the jobs in node i's queue are serviced by the arrival of candidate packets on arc $(i, i+1)$ and, with the exception of the first $\mu-1$ jobs, enter node $(i+1)$'s queue after being serviced. The jobs in node L's queue are serviced by the arrival of candidate packets on arc $(L, L+1)$ and exit after being serviced. The number of jobs exiting is a lower bound on the number of packets with linearly-independent auxiliary encoding vectors received by node $L+1$.

We again analyze the queueing network of interest using the fluid approximation for discrete-flow networks, and we again do not explicitly account for the fact that the first $\mu - 1$ jobs arriving at a queueing node do not enter its queue. Let B_1 be the counting process for the arrival of received packets on arc $(1, 2)$. For $i = 2, 3, \ldots, L$, let B_i, and C_i be the counting processes for the arrival of innovative packets and candidate packets on arc $(i, i+1)$, respectively. Let $Q_i(\tau)$ be the number of jobs queued for service at node i at time τ. Hence, for $i = 2, 3, \ldots, L$, $Q_i = B_{i-1} - B_i$. Let $X_i := C_{i-1} - C_i$ and $Y_i := C_i - B_i$, where $C_1 := B_1$. Then, we obtain a Skorohod problem with the following conditions: For

all $i = 2, 3, \ldots, L$,

$$Q_i = X_i - Y_{i-1} + Y_i.$$

For all $\tau \geq 0$ and $i = 2, 3, \ldots, L$,

$$Q_i(\tau) dY_i(\tau) = 0,$$
$$dY_i(\tau) \geq 0,$$

and

$$Q_i(\tau) \geq 0.$$

For all $i = 2, 3, \ldots, L$,

$$Y_i(0) = 0.$$

Let

$$\bar{Q}_i^{(K)}(\tau) := \frac{Q_i(K\tau)}{K}$$

and $\bar{Q}_i := \lim_{K\to\infty} \bar{Q}_i^{(K)}$ for $i = 2, 3, \ldots, L$. Then the vector $\bar{Q}$ is, almost surely, the unique $\bar{Q}$ that satisfies, for some $\bar{Y}$,

$$\bar{Q}_i(\tau) = \begin{cases} (z_{122} - (1 - q^{-\mu})z_{233})\tau + \bar{Y}_2(\tau) & \text{if } i = 2, \\ (1 - q^{-\mu})(z_{(i-1)ii} - z_{i(i+1)(i+1)})\tau & \\ \qquad + \bar{Y}_i(\tau) - \bar{Y}_{i-1}(\tau) & \text{otherwise,} \end{cases} \tag{4.15}$$

$$\bar{Q}_i(\tau) d\bar{Y}_i(\tau) = 0, \tag{4.16}$$

$$d\bar{Y}_i(\tau) \geq 0, \tag{4.17}$$

and

$$\bar{Q}_i(\tau) \geq 0 \tag{4.18}$$

for all $\tau \geq 0$ and $i = 2, 3, \ldots, L$, and

$$\bar{Y}_i(0) = 0 \tag{4.19}$$

for all $i = 2, 3, \ldots, L$.

A pair $(\bar{Q}, \bar{Y})$ that satisfies (4.15)–(4.19) is

$$\bar{Q}_i(\tau) = (\min(z_{122}, \min_{2\leq j<i}\{(1 - q^{-\mu})z_{j(j+1)(j+1)}\}) - (1 - q^{-\mu})z_{i(i+1)(i+1)})^+\tau \tag{4.20}$$

and

$$\bar{Y}_i(\tau) = (\min(z_{122}, \min_{2\leq j<i}\{(1 - q^{-\mu})z_{j(j+1)(j+1)}\}) - (1 - q^{-\mu})z_{i(i+1)(i+1)})^-\tau.$$

Hence $\bar{Q}$ is given by equation (4.20).

The number of jobs that have exited the queueing network by time Δ is given by

$$\nu = B_1(\Delta) - \sum_{i=2}^{L} Q_i(\Delta).$$

Take $K = \lceil (1 - q^{-\mu})\Delta R_c R/(1+\varepsilon) \rceil$, where $0 < R_c < 1$. Then

$$\begin{aligned} \lim_{K\to\infty} \frac{\nu}{\lfloor K(1+\varepsilon) \rfloor} &= \lim_{K\to\infty} \frac{B_1(\Delta) - \sum_{i=2}^{L} Q(\Delta)}{K(1+\varepsilon)} \\ &= \frac{\min(z_{122}, \min_{2\le i\le L}\{(1-q^{-\mu}) z_{i(i+1)(i+1)}\})}{(1-q^{-\mu})R_c R} \\ &\ge \frac{1}{R_c} \frac{\min_{1\le i\le L}\{z_{i(i+1)(i+1)}\}}{R} > 1 \end{aligned} \tag{4.21}$$

provided that

$$R \le \min_{1\le i\le L} \{z_{i(i+1)(i+1)}\}. \tag{4.22}$$

Hence, for all R satisfying (4.22), $\nu \ge \lfloor K(1+\varepsilon) \rfloor$ with probability arbitrarily close to 1 for K sufficiently large. The rate can again be made arbitrarily close to R by varying μ, R_c, and ε.

4.2.1.3 General unicast connection

We now extend our result to general unicast connections. The strategy here is simple: A general unicast connection can be formulated as a flow, which can be decomposed into a finite number of paths. Each of these paths is a tandem network, which is the case that we have just considered.

Suppose that we wish to establish a connection of rate arbitrarily close to R packets per unit time from source node s to sink node t. Suppose further that

$$R \le \min_{Q\in\mathcal{Q}(s,t)} \left\{ \sum_{(i,J)\in\Gamma_+(Q)} \sum_{K\not\subset Q} z_{iJK} \right\},$$

where $\mathcal{Q}(s,t)$ is the set of all cuts between s and t, and $\Gamma_+(Q)$ denotes the set of forward hyperarcs of the cut Q, i.e.,

$$\Gamma_+(Q) := \{(i,J) \in \mathcal{A} \,|\, i \in Q, J \setminus Q \ne \emptyset\}.$$

Therefore, by the max-flow/min-cut theorem (see, e.g., [5, Sections 6.5–6.7], [11, Section 3.1]), there exists a flow vector x satisfying

$$\sum_{\{J|(i,J)\in\mathcal{A}\}}\sum_{j\in J} x_{iJj} - \sum_{\{j|(j,I)\in\mathcal{A},i\in I\}} x_{jIi} = \begin{cases} R & \text{if } i=s,\\ -R & \text{if } i=t,\\ 0 & \text{otherwise,}\end{cases}$$

for all $i\in\mathcal{N}$,

$$\sum_{j\in K} x_{iJj} \le \sum_{\{L\subset J|L\cap K\neq\emptyset\}} z_{iJL} \tag{4.23}$$

for all $(i,J)\in\mathcal{A}$ and $K\subset J$, and $x_{iJj}\ge 0$ for all $(i,J)\in\mathcal{A}$ and $j\in J$.

Using the conformal realization theorem (see, e.g., [11, Section 1.1]), we decompose x into a finite set of paths $\{p_1,p_2,\ldots,p_M\}$, each carrying positive flow R_m for $m=1,2,\ldots,M$, such that $\sum_{m=1}^{M} R_m = R$. We treat each path p_m as a tandem network and use it to deliver innovative packets at rate arbitrarily close to R_m, resulting in an overall rate for innovative packets arriving at node t that is arbitrarily close to R. Some care must be take in the interpretation of the flow and its path decomposition because the same packet may be received by more than one node.

Consider a single path p_m. We write $p_m=\{i_1,i_2,\ldots,i_{L_m},i_{L_m+1}\}$, where $i_1=s$ and $i_{L_m+1}=t$. For $l=2,3,\ldots,L_m+1$, we associate with node i_l the set of vectors $V_l^{(p_m)}$, which varies with time and is initially empty. We define $U^{(p_m)}:=V_2^{(p_m)}$ and $W^{(p_m)}:=V_{L_m+1}^{(p_m)}$.

We note that the constraint (4.23) can also be written as

$$x_{iJj} \le \sum_{\{L\subset J|j\in L\}} \alpha_{iJL}^{(j)} z_{iJL}$$

for all $(i,J)\in\mathcal{A}$ and $j\in J$, where $\sum_{j\in L}\alpha_{iJL}^{(j)}=1$ for all $(i,J)\in\mathcal{A}$ and $L\subset J$, and $\alpha_{iJL}^{(j)}\ge 0$ for all $(i,J)\in\mathcal{A}$, $L\subset J$, and $j\in L$. Suppose packet u, with auxiliary encoding vector β, is placed on hyperarc (i_1,J) and received by $K\subset J$, where $K\ni i_2$, at time τ. We associate with u the independent random variable P_u, which takes the value m with probability $R_m\alpha_{i_1JK}^{(i_2)}/\sum_{\{L\subset J|i_2\in L\}}\alpha_{i_1JL}^{(i_2)}z_{iJL}$. If $P_u=m$, then we say u is innovative on path p_m, and β is added to $U^{(p_m)}$ at time τ.

Take $l=2,3,\ldots,L_m$. Now suppose packet u, with auxiliary encoding vector β, is placed on hyperarc (i_l,J) and received by $K\subset J$, where $K\ni i_{l+1}$, at time τ. We associate with u the independent random variable P_u, which takes the value m with probability

$R_m \alpha_{i_l JK}^{(i_{l+1})} / \sum_{\{L \subset J | i_{l+1} \in L\}} \alpha_{i_l JL}^{(i_{l+1})} z_{iJL}$. We say u is innovative on path p_m if $P_u = m$, $\beta \notin \text{span}(\cup_{n=1}^{m-1} W^{(p_n)}(\Delta) \cup V_{l+1}^{(p_m)}(\tau) \cup \cup_{n=m+1}^{M} U^{(p_n)}(\Delta))$, and $|V_l^{(p_m)}(\tau)| > |V_{l+1}^{(p_m)}(\tau)| + \mu - 1$.

This definition of innovative is somewhat more complicated than that in Sections 4.2.1.1 and 4.2.1.2 because we now have M paths that we wish to analyze separately. We have again designed the definition to satisfy two properties: First, we require that $\cup_{m=1}^{M} W^{(p_m)}(\Delta)$ is linearly-independent. This is easily verified: Vectors are added to $W^{(p_1)}(\tau)$ only if they are linearly independent of existing ones; vectors are added to $W^{(p_2)}(\tau)$ only if they are linearly independent of existing ones and ones in $W^{(p_1)}(\Delta)$; and so on. Second, we require that, when a packet is received by node i_l, $P_u = m$, and $|V_{l-1}^{(p_m)}(\tau)| > |V_l^{(p_m)}(\tau)| + \mu - 1$, it is innovative on path p_m with high probability.

Take $l \in \{3, 4, \ldots, L_m + 1\}$. Suppose that packet u, with auxiliary encoding vector β, is received by node i_l at time τ, that $P_u = m$, and that $|V_{l-1}^{(p_m)}(\tau)| > |V_l^{(p_m)}(\tau)| + \mu - 1$. Thus, the auxiliary encoding vector β is a random linear combination of vectors in some set V_0 that contains $V_{l-1}^{(p_m)}(\tau)$. Hence β is uniformly-distributed over $q^{|V_0|}$ possibilities, of which at least $q^{|V_0|} - q^d$ are not in $\text{span}(V_l^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m})$, where $d := \dim(\text{span}(V_0) \cap \text{span}(V_l^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m}))$. We have

$$\begin{aligned}
d &= \dim(\text{span}(V_0)) + \dim(\text{span}(V_l^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m})) \\
&\quad - \dim(\text{span}(V_0 \cup V_l^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m})) \\
&\le \dim(\text{span}(V_0 \setminus V_{l-1}^{(p_m)}(\tau))) + \dim(\text{span}(V_{l-1}^{(p_m)}(\tau))) \\
&\quad + \dim(\text{span}(V_l^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m})) \\
&\quad - \dim(\text{span}(V_0 \cup V_l^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m})) \\
&\le \dim(\text{span}(V_0 \setminus V_{l-1}^{(p_m)}(\tau))) + \dim(\text{span}(V_{l-1}^{(p_m)}(\tau))) \\
&\quad + \dim(\text{span}(V_l^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m})) \\
&\quad - \dim(\text{span}(V_{l-1}^{(p_m)}(\tau) \cup V_l^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m})).
\end{aligned}$$

Since $V_{l-1}^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m}$ and $V_l^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m}$ both form linearly-independent sets,

$$\begin{aligned}
&\dim(\text{span}(V_{l-1}^{(p_m)}(\tau))) + \dim(\text{span}(V_l^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m})) \\
&\quad = \dim(\text{span}(V_{l-1}^{(p_m)}(\tau))) + \dim(\text{span}(V_l^{(p_m)}(\tau))) + \dim(\text{span}(\tilde{V}_{\setminus m})) \\
&\quad = \dim(\text{span}(V_l^{(p_m)}(\tau))) + \dim(\text{span}(V_{l-1}^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m})).
\end{aligned}$$

Hence it follows that

$$\begin{aligned} d &\leq \dim(\text{span}(V_0 \setminus V_{l-1}^{(p_m)}(\tau))) + \dim(\text{span}(V_l^{(p_m)}(\tau))) \\ &\quad + \dim(\text{span}(V_{l-1}^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m})) \\ &\quad - \dim(\text{span}(V_{l-1}^{(p_m)}(\tau) \cup V_l^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m})) \\ &\leq \dim(\text{span}(V_0 \setminus V_{l-1}^{(p_m)}(\tau))) + \dim(\text{span}(V_l^{(p_m)}(\tau))) \\ &\leq |V_0 \setminus V_{l-1}^{(p_m)}(\tau)| + |V_l^{(p_m)}(\tau)| \\ &= |V_0| - |V_{l-1}^{(p_m)}(\tau)| + |V_l^{(p_m)}(\tau)|, \end{aligned}$$

which yields

$$d - |V_0| \leq |V_l^{(p_m)}(\tau)| - |V_{l-1}^{(p_m)}(\tau)| \leq -\rho.$$

Therefore, it follows that

$$\Pr(\beta \notin \text{span}(V_l^{(p_m)}(\tau) \cup \tilde{V}_{\setminus m})) \geq \frac{q^{|V_0|} - q^d}{q^{|V_0|}} = 1 - q^{d-|V_0|} \geq 1 - q^{-\mu}.$$

We see then that, if we consider only those packets such that $P_u = m$, the conditions that govern the propagation of innovative packets are exactly those of an L_m-link tandem network, which we dealt with in Section 4.2.1.2. By recalling the distribution of P_u, it follows that the propagation of innovative packets along path p_m behaves like an L_m-link tandem network with average arrival rate R_m on every link. Since we have assumed nothing special about m, this statement applies for all $m = 1, 2, \ldots, M$.

Take $K = \lceil (1 - q^{-\mu}) \Delta R_c R / (1 + \varepsilon) \rceil$, where $0 < R_c < 1$. Then, by equation (4.21),

$$\lim_{K \to \infty} \frac{|W^{(p_m)}(\Delta)|}{\lfloor K(1+\varepsilon) \rfloor} > \frac{R_m}{R}.$$

Hence

$$\lim_{K \to \infty} \frac{|\cup_{m=1}^{M} W^{(p_m)}(\Delta)|}{\lfloor K(1+\varepsilon) \rfloor} = \sum_{m=1}^{M} \frac{|W^{(p_m)}(\Delta)|}{\lfloor K(1+\varepsilon) \rfloor} > \sum_{m=1}^{M} \frac{R_m}{R} = 1.$$

As before, the rate can be made arbitrarily close to R by varying μ, R_c, and ε.

4.2.2 Multicast connections

The result for multicast connections is, in fact, a straightforward extension of that for unicast connections. In this case, rather than a single

sink t, we have a set of sinks T. As in the framework of static broadcasting (see [127, 128]), we allow sink nodes to operate at different rates. We suppose that sink $t \in T$ wishes to achieve rate arbitrarily close to R_t, i.e., to recover the K message packets, sink t wishes to wait for a time Δ_t that is only marginally greater than K/R_t. We further suppose that

$$R_t \leq \min_{Q \in \mathcal{Q}(s,t)} \left\{ \sum_{(i,J) \in \Gamma_+(Q)} \sum_{K \not\subset Q} z_{iJK} \right\}$$

for all $t \in T$. Therefore, by the max-flow/min-cut theorem, there exists, for each $t \in T$, a flow vector $x^{(t)}$ satisfying

$$\sum_{\{j|(i,J) \in \mathcal{A}\}} \sum_{j \in J} x^{(t)}_{iJj} - \sum_{\{j|(j,I) \in \mathcal{A}, i \in I\}} x^{(t)}_{jIi} = \begin{cases} R & \text{if } i = s, \\ -R & \text{if } i = t, \\ 0 & \text{otherwise,} \end{cases}$$

for all $i \in \mathcal{N}$,

$$\sum_{j \in K} x^{(t)}_{iJj} \leq \sum_{\{L \subset J | L \cap K \neq \emptyset\}} z_{iJL}$$

for all $(i, J) \in \mathcal{A}$ and $K \subset J$, and $x^{(t)}_{iJj} \geq 0$ for all $(i, J) \in \mathcal{A}$ and $j \in J$.

For each flow vector $x^{(t)}$, we go through the same argument as that for a unicast connection, and we find that the probability of error at every sink node can be made arbitrarily small by taking K sufficiently large.

We summarize our results with the following theorem statement.

Theorem 4.1 *Consider the coding subgraph z. The random linear network coding scheme described in Section 4.1 is capacity-achieving for single connections in z, i.e., for K sufficiently large, it can achieve, with arbitrarily small error probability, a connection from source node s to sink nodes in the set T at rate arbitrarily close to R_t packets per unit time for each $t \in T$ if*

$$R_t \leq \min_{Q \in \mathcal{Q}(s,t)} \left\{ \sum_{(i,J) \in \Gamma_+(Q)} \sum_{K \not\subset Q} z_{iJK} \right\}$$

for all $t \in T$.

Remark. The capacity region is determined solely by the average rates $\{z_{iJK}\}$ at which packets are received. Thus, the packet injection and

loss processes, which give rise to the packet reception processes, can in fact take any distribution, exhibiting arbitrary correlations, as long as these average rates exist.

4.3 Error exponents for Poisson traffic with i.i.d. losses

We now look at the rate of decay of the probability of error p_e in the coding delay Δ. In contrast to traditional error exponents where coding delay is measured in symbols, we measure coding delay in time units—time $\tau = \Delta$ is the time at which the sink nodes attempt to decode the message packets. The two methods of measuring delay are essentially equivalent when packets arrive in regular, deterministic intervals.

We specialize to the case of Poisson traffic with i.i.d. losses. Thus, the process A_{iJK} is a Poisson process with rate z_{iJK}. Consider the unicast case for now, and suppose we wish to establish a connection of rate R. Let C be the supremum of all asymptotically-achievable rates.

We begin by deriving an upper bound on the probability of error. To this end, we take a flow vector x from s to t of size C and, following the development in Section 4.2, develop a queueing network from it that describes the propagation of innovative packets for a given innovation order μ. This queueing network now becomes a Jackson network. Moreover, as a consequence of Burke's theorem (see, e.g., [79, Section 2.1]) and the fact that the queueing network is acyclic, the arrival and departure processes at all stations are Poisson in steady-state.

Let $\Psi_t(m)$ be the arrival time of the mth innovative packet at t, and let $C' := (1 - q^{-\mu})C$. When the queueing network is in steady-state, the arrival of innovative packets at t is described by a Poisson process of rate C'. Hence we have

$$\lim_{m\to\infty} \frac{1}{m} \log \mathbb{E}[\exp(\theta\Psi_t(m))] = \log \frac{C'}{C' - \theta} \tag{4.24}$$

for $\theta < C'$ [14, 113]. If an error occurs, then fewer than $\lceil R\Delta \rceil$ innovative packets are received by t by time $\tau = \Delta$, which is equivalent to saying that $\Psi_t(\lceil R\Delta \rceil) > \Delta$. Therefore,

$$p_e \leq \Pr(\Psi_t(\lceil R\Delta \rceil) > \Delta),$$

and, using the Chernoff bound, we obtain

$$p_e \leq \min_{0\leq\theta<C'} \exp\left(-\theta\Delta + \log \mathbb{E}[\exp(\theta\Psi_t(\lceil R\Delta \rceil))]\right).$$

Let ε be a positive real number. Then using equation (4.24) we obtain,

for Δ sufficiently large,

$$\begin{aligned} p_e &\leq \min_{0 \leq \theta < C'} \exp\left(-\theta\Delta + R\Delta\left\{\log\frac{C'}{C'-\theta} + \varepsilon\right\}\right) \\ &= \exp(-\Delta(C' - R - R\log(C'/R)) + R\Delta\varepsilon). \end{aligned}$$

Hence, we conclude that

$$\lim_{\Delta\to\infty} \frac{-\log p_e}{\Delta} \geq C' - R - R\log(C'/R). \tag{4.25}$$

For the lower bound, we examine a cut whose flow capacity is C. We take one such cut and denote it by Q^*. It is clear that, if fewer than $\lceil R\Delta \rceil$ distinct packets are received across Q^* in time $\tau = \Delta$, then an error occurs. The arrival of distinct packets across Q^* is described by a Poisson process of rate C. Thus we have

$$\begin{aligned} p_e &\geq \exp(-C\Delta) \sum_{l=0}^{\lceil R\Delta\rceil - 1} \frac{(C\Delta)^l}{l!} \\ &\geq \exp(-C\Delta)\frac{(C\Delta)^{\lceil R\Delta\rceil - 1}}{\Gamma(\lceil R\Delta\rceil)}, \end{aligned}$$

and, using Stirling's formula, we obtain

$$\lim_{\Delta\to\infty} \frac{-\log p_e}{\Delta} \leq C - R - R\log(C/R). \tag{4.26}$$

Since (4.25) holds for all positive integers μ, we conclude from (4.25) and (4.26) that

$$\lim_{\Delta\to\infty} \frac{-\log p_e}{\Delta} = C - R - R\log(C/R). \tag{4.27}$$

Equation (4.27) defines the asymptotic rate of decay of the probability of error in the coding delay Δ. This asymptotic rate of decay is determined entirely by R and C. Thus, for a packet network with Poisson traffic and i.i.d. losses employing random linear network coding as described in Section 4.1, the flow capacity C of the minimum cut of the network is essentially the sole figure of merit of importance in determining the effectiveness of random linear network coding for large, but finite, coding delay. Hence, in deciding how to inject packets to support the desired connection, a sensible approach is to reduce our attention to this figure of merit, which is indeed the approach that we take in Chapter 5.

Extending the result from unicast connections to multicast connections is straightforward—we simply obtain (4.27) for each sink.

4.4 Notes and further reading

Network coding for lossy networks has been looked at in [51, 94, 80, 98, 32, 136]. In [51, 32], a capacity result is established; in [80], a capacity result is established for the case where no side-information is placed in packet headers, and a code construction based on maximum distance separable (MDS) codes is proposed; in [94, 98, 136], the use of random linear network coding in lossy networks is examined. The exposition in this chapter is derived from [93, 94, 98].

Random linear network coding originates from [62, 27, 66], which deal with lossless networks. In [111, 99, 103], some variations to random linear network coding, as described in Section 4.1, are proposed. In [99], a variation that reduces memory usage at intermediate nodes is examined, while, in [111, 103], variations that reduce encoding and decoding complexity are examined. One of the schemes proposed in [103] achieves linear encoding and decoding complexity.

5
Subgraph Selection

In the previous two chapters, we assumed that a coding subgraph specifying the times and locations of packet injections was given. We dealt only with half the problem of establishing connections in coded packet networks—the coding half. This chapter deals with the other half: subgraph selection.

Subgraph selection, which is the problem of determining the coding subgraph to use, is the coded networks analog of the joint problems of routing and scheduling in conventional, routed networks. Subgraph selection and coding are very different problems, and the techniques used in the this chapter differ significantly from those in the previous two chapters. In particular, while the previous two chapters generally used techniques from information theory and coding theory, this chapter generally uses techniques from networking theory.

Subgraph selection is essentially a problem of network resource allocation: We have a limited resource (packet injections) that we wish to allocate to coded packets in such as way as to achieve certain communication objectives. We propose a number of solutions to the problem, and we divide these solutions into two categories: flow-based approaches (Section 5.1) and queue-length-based approaches (Section 5.2). In flow-based approaches, we assume that the communication objective is to establish a set of (unicast or multicast) connections at certain, given flow rates while, in queue-length-based approaches, we suppose that the flow rates, though existent, are not necessarily known, and we select coding subgraphs using the state of packet queues.

As in the previous two chapters, we deal primarily with *intra-session coding*—coding confined to a single session or connection. This allows us to make use the various results that we have established for network coding in a single session. Unfortunately, intra-session coding is sub-

optimal. Recall the modified butterfly network (Figure 1.2) and modified wireless butterfly network (Figure 1.4). In both these examples, a gain was achieved by using *inter-session coding*—coding across two or more independent sessions. There are far fewer results concerning inter-session coding than those concerning intra-session coding. One thing that is known, though, is linear codes do not suffice in general to achieve the inter-session coding capacity of a coding subgraph [38]. Since no non-linear network codes that seem practicable have yet been found, we are therefore forced to find suboptimal approaches to linear inter-session coding, or to simply use intra-session coding. We discuss subgraph selection techniques for both intra-session coding and suboptimal inter-session coding in this chapter.

We place particular emphasis on subgraph selection techniques that can computed in a distributed manner, with each node making computations based only on local knowledge and knowledge acquired from information exchanges. Such distributed algorithms are inspired by existing, successful distributed algorithms in networking, such as the distributed Bellman-Ford algorithm (see, e.g., [13, Section 5.2]), which is used to find routes in routed packet networks. In general, distributed subgraph selection techniques currently exist only in cases where arcs essentially behave independently and the capacities of separate arcs are not coupled.

5.1 Flow-based approaches

We discuss flow-based approaches under the assumption that there is a cost that we wish to minimize. This cost, which is a function of the coding subgraph z, reflects some notion of network efficiency (we could have, for example, an energy cost, a congestion cost, or even a monetary cost), and it allows us to favor particular subgraphs in the class of subgraphs that are capable of establishing the desired connections. A cost-minimization objective is certainly not the only possible (throughput maximization, for example, is another possibility), but it is very general, and much of the following discussion applies to other objectives also. Let f be the cost function. We assume, for tractability and simplicity, that f is convex.

We first discuss intra-session coding. For intra-session coding, we formulate the problem and discuss methods for its solution then, in Section 5.1.1.6, we consider applying these methods for communication

over wireless networks, and we compare their performance to existing methods. In Section 5.1.3, we discuss inter-session coding.

5.1.1 Intra-session coding

5.1.1.1 Problem formulation

We specify a multicast connection with a triplet $(s, T, \{R_t\}_{t\in T})$, where s is the source of the connection, T is the set of sinks, and $\{R_t\}_{t\in T}$ is the set of rates to the sinks (see Section 4.2.2). Suppose we wish to establish C multicast connections, $(s_1, T_1, \{R_{t,1}\}), \ldots, (s_C, T_C, \{R_{t,C}\})$. Using Theorem 4.1 and the max-flow/min-cut theorem, we see that subgraph selection in a lossy network with random linear network coding in each session can be phrased as the following mathematical programming problem:

$$\begin{aligned}
&\text{minimize } f(z)\\
&\text{subject to } z \in Z,\\
&\sum_{c=1}^{C} y_{iJK}^{(c)} \le z_{iJK}, \qquad \forall\ (i,J) \in \mathcal{A},\ K \subset J,\\
&\sum_{j\in K} x_{iJj}^{(t,c)} \le \sum_{\{L\subset J | L\cap K\neq\emptyset\}} y_{iJL}^{(c)},\\
&\qquad\qquad \forall\ (i,J) \in \mathcal{A},\ K \subset J,\ t \in T_c,\ c = 1,\ldots,C,\\
&x^{(t,c)} \in F^{(t,c)}, \qquad \forall\ t \in T_c,\ c = 1,\ldots,C,
\end{aligned} \tag{5.1}$$

where $x^{(t,c)}$ is the vector consisting of $x_{iJj}^{(t,c)}$, $(i,J) \in \mathcal{A}$, $j \in J$, and $F^{(t,c)}$ is the bounded polyhedron of points $x^{(t,c)}$ satisfying the conservation of flow constraints

$$\sum_{\{J|(i,J)\in\mathcal{A}\}} \sum_{j\in J} x_{iJj}^{(t,c)} - \sum_{\{j|(j,I)\in\mathcal{A}, i\in I\}} x_{jIi}^{(t,c)} = \begin{cases} R_{t,c} & \text{if } i = s_c,\\ -R_{t,c} & \text{if } i = t,\\ 0 & \text{otherwise,}\end{cases} \qquad \forall\ i \in \mathcal{N},$$

and non-negativity constraints

$$x_{iJj}^{(t,c)} \ge 0, \qquad \forall\ (i,J) \in \mathcal{A},\ j \in J.$$

In this formulation, $y_{iJK}^{(c)}$ represents the average rate of packets that are injected on hyperarc (i,J) and received by exactly the set of nodes K

(which occurs with average rate z_{iJK}) and that are allocated to connection c.

For simplicity, let us consider the case where $C = 1$. The extension to $C > 1$ is conceptually straightforward and, moreover, the case where $C = 1$ is interesting in its own right: whenever each multicast group has a selfish cost objective, or when the network sets arc weights to meet its objective or enforce certain policies and each multicast group is subject to a minimum-weight objective, we wish to establish single efficient multicast connections.

Let

$$b_{iJK} := \frac{\sum_{\{L \subset J | L \cap K \neq \emptyset\}} z_{iJL}}{z_{iJ}},$$

which is the fraction of packets injected on hyperarc (i, J) that are received by a set of nodes that intersects K. Problem (5.1) is now

$$\begin{aligned} &\text{minimize } f(z) \\ &\text{subject to } z \in Z, \\ &\qquad \sum_{j \in K} x^{(t)}_{iJj} \leq z_{iJ} b_{iJK}, \qquad \forall\ (i, J) \in \mathcal{A},\ K \subset J,\ t \in T, \\ &\qquad x^{(t)} \in F^{(t)}, \qquad \forall\ t \in T. \end{aligned} \tag{5.2}$$

In the lossless case, problem (5.2) simplifies to the following problem:

$$\begin{aligned} &\text{minimize } f(z) \\ &\text{subject to } z \in Z, \\ &\qquad \sum_{j \in J} x^{(t)}_{iJj} \leq z_{iJ}, \qquad \forall\ (i, J) \in \mathcal{A},\ t \in T, \\ &\qquad x^{(t)} \in F^{(t)}, \qquad \forall\ t \in T. \end{aligned} \tag{5.3}$$

As an example, consider the network depicted in Figure 5.1, which consists only of point-to-point arcs. Suppose that the network is lossless, that we wish to achieve multicast of unit rate from s to two sinks, t_1 and t_2, and that we have $Z = [0, 1]^{|\mathcal{A}|}$ and $f(z) = \sum_{(i,j) \in \mathcal{A}} z_{ij}$. An optimal solution to problem (5.3) is shown in the figure. We have flows $x^{(1)}$ and $x^{(2)}$ of unit size from s to t_1 and t_2, respectively, and, for each arc (i, j), $z_{ij} = \max(x^{(1)}_{ijj}, x^{(2)}_{ijj})$, as we expect from the optimization. For a simple arc (i, j), it is unnecessary to write $x^{(1)}_{ijj}$ for the component of flow $x^{(1)}$ on the arc; we can simply write $x^{(1)}_{ij}$, and we shall do so as appropriate. Under this abbreviated notation, we have $z_{ij} = \max(x^{(1)}_{ij}, x^{(2)}_{ij})$.

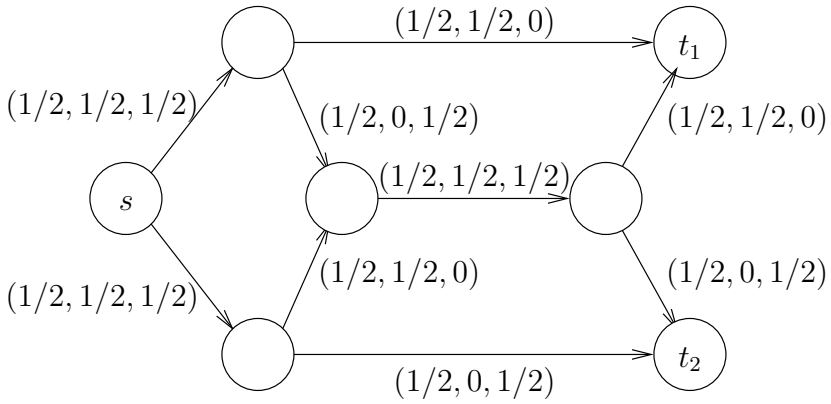


Fig. 5.1. A network of lossless point-to-point arcs with multicast from s to $T = \{t_1, t_2\}$. Each arc is marked with the triple $(z_{ij}, x_{ij}^{(1)}, x_{ij}^{(2)})$. Reprinted with permission from [101].

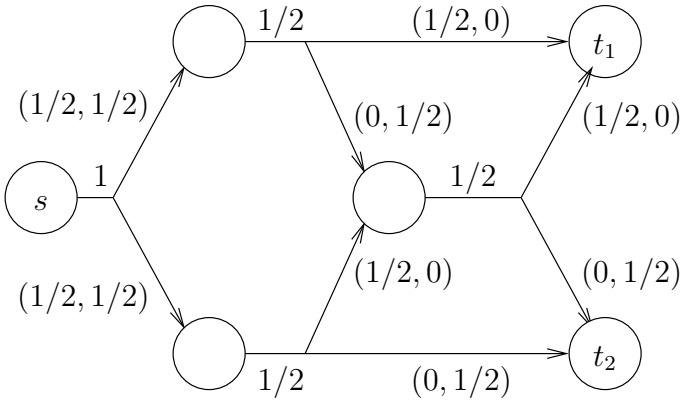


Fig. 5.2. A network of lossless broadcast arcs with multicast from s to $T = \{t_1, t_2\}$. Each hyperarc is marked with z_{iJ} at its start and the pair $(x_{iJ_j}^{(1)}, x_{iJ_j}^{(2)})$ at its ends.

The same multicast problem in a routed packet network would entail minimizing the number of arcs used to form a tree that is rooted at s and that reaches t_1 and t_2—in other words, solving the Steiner tree problem on directed graphs [115]. The Steiner tree problem on directed graphs is well-known to be NP-complete, but solving problem (5.3) is not. In this case, problem (5.3) is in fact a linear optimization problem. It is a linear optimization problem that can be thought of as a fractional

relaxation of the Steiner tree problem [154]. This example illustrates one of the attractive features of the coded approach: it allows us avoid an NP-complete problem and instead solve its fractional relaxation.

For an example with broadcast arcs, consider the network depicted in Figure 5.2. Suppose again that the network is lossless, that we wish to achieve multicast of unit rate from s to two sinks, t_1 and t_2, and that we have $Z = [0,1]^{|\mathcal{A}|}$ and $f(z) = \sum_{(i,J)\in\mathcal{A}} z_{iJ}$. An optimal solution to problem (5.3) is shown in the figure. We still have flows $x^{(1)}$ and $x^{(2)}$ of unit size from s to t_1 and t_2, respectively, but now, for each hyperarc (i,J), we determine z_{iJ} from the various flows passing through hyperarc (i,J), each destined toward a single node j in J, and the optimization gives $z_{iJ} = \max(\sum_{j\in J} x^{(1)}_{iJj}, \sum_{j\in J} x^{(2)}_{iJj})$.

Neither problem (5.2) nor (5.3) as it stands is easy to solve. But the problems are very general. Their complexities improve if we assume that the cost function is separable and possibly even linear, i.e., if we suppose $f(z) = \sum_{(i,J)\in A} f_{iJ}(z_{iJ})$, where f_{iJ} is a convex or linear function, which is a very reasonable assumption in many practical situations. For example, packet latency is usually assessed with a separable, convex cost function and energy, monetary cost, and total weight are usually assessed with separable, linear cost functions.

The complexities of problems (5.2) and (5.3) also improve if we make some assumptions on the form of the constraint set Z, which is the case in most practical situations.

A particular simplification applies if we assume that, when nodes transmit in a lossless network, they reach all nodes in a certain region, with cost increasing as this region is expanded. This applies, for example, if we are interested in minimizing energy consumption, and the region in which a packet is reliably received expands as we expend more energy in its transmission. More precisely, suppose that we have separable cost, so $f(z) = \sum_{(i,J)\in\mathcal{A}} f_{iJ}(z_{iJ})$. Suppose further that each node i has M_i outgoing hyperarcs $(i, J^{(i)}_1), (i, J^{(i)}_2), \ldots, (i, J^{(i)}_{M_i})$ with $J^{(i)}_1 \subsetneq J^{(i)}_2 \subsetneq \cdots \subsetneq J^{(i)}_{M_i}$. (We assume that there are no identical arcs, as duplicate arcs can effectively be treated as a single arc.) Then, we assume that $f_{iJ^{(i)}_1}(\zeta) < f_{iJ^{(i)}_2}(\zeta) < \cdots < f_{iJ^{(i)}_{M_i}}(\zeta)$ for all $\zeta \geq 0$ and nodes i.

Let us introduce, for $(i,j) \in \mathcal{A}' := \{(i,j) | (i,J) \in A, J \ni j\}$, the

variables

$$\hat{x}_{ij}^{(t)} := \sum_{m=m(i,j)}^{M_i} x_{iJ_m^{(i)}j}^{(t)},$$

where $m(i,j)$ is the unique m such that $j \in J_m^{(i)} \setminus J_{m-1}^{(i)}$ (we define $J_0^{(i)} := \emptyset$ for all $i \in \mathcal{N}$ for convenience). Now, problem (5.3) can be reformulated as the following problem, which has substantially fewer variables:

$$\begin{aligned}
&\text{minimize} \sum_{(i,J)\in\mathcal{A}} f_{iJ}(z_{iJ}) \\
&\text{subject to } z \in Z \\
&\sum_{k\in J_{M_i}^{(i)}\setminus J_{m-1}^{(i)}} \hat{x}_{ik}^{(t)} \le \sum_{n=m}^{M_i} z_{iJ_n^{(i)}}, \qquad \forall\, i \in \mathcal{N},\, m = 1,\ldots,M_i,\, t \in T, \\
&\hat{x}^{(t)} \in \hat{F}^{(t)}, \qquad \forall\, t \in T,
\end{aligned} \tag{5.4}$$

where $\hat{F}^{(t)}$ is the bounded polyhedron of points $\hat{x}^{(t)}$ satisfying the conservation of flow constraints

$$\sum_{\{j|(i,j)\in\mathcal{A}'\}} \hat{x}_{ij}^{(t)} - \sum_{\{j|(j,i)\in\mathcal{A}'\}} \hat{x}_{ji}^{(t)} = \begin{cases} R_t & \text{if } i = s, \\ -R_t & \text{if } i = t, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall\, i \in N,$$

and non-negativity constraints

$$0 \le \hat{x}_{ij}^{(t)}, \qquad \forall\, (i,j) \in \mathcal{A}'.$$

Proposition 5.1 *Suppose that $f(z) = \sum_{(i,J)\in\mathcal{A}} f_{iJ}(z_{iJ})$ and that $f_{iJ_1^{(i)}}(\zeta) < f_{iJ_2^{(i)}}(\zeta) < \cdots < f_{iJ_{M_i}^{(i)}}(\zeta)$ for all $\zeta \ge 0$ and $i \in \mathcal{N}$. Then problem (5.3) and problem (5.4) are equivalent in the sense that they have the same optimal cost and z is part of an optimal solution for (5.3) if and only if it is part of an optimal solution for (5.4).*

Proof Suppose (x,z) is a feasible solution to problem (5.3). Then, for

all $(i,j) \in \mathcal{A}'$ and $t \in T$,

$$\begin{aligned}
\sum_{m=m(i,j)}^{M_i} z_{iJ_m^{(i)}} &\geq \sum_{m=m(i,j)}^{M_i} \sum_{k \in J_m^{(i)}} x^{(t)}_{iJ_m^{(i)}k} \\
&= \sum_{k \in J_{M_i}^{(i)}} \sum_{m=\max(m(i,j),m(i,k))}^{M_i} x^{(t)}_{iJ_m^{(i)}k} \\
&\geq \sum_{k \in J_{M_i}^{(i)} \setminus J_{m(i,j)-1}^{(i)}} \sum_{m=\max(m(i,j),m(i,k))}^{M_i} x^{(t)}_{iJ_m^{(i)}k} \\
&= \sum_{k \in J_{M_i}^{(i)} \setminus J_{m(i,j)-1}^{(i)}} \sum_{m=m(i,k)}^{M_i} x^{(t)}_{iJ_m^{(i)}k} \\
&= \sum_{k \in J_{M_i}^{(i)} \setminus J_{m(i,j)-1}^{(i)}} \hat{x}^{(t)}_{ik}.
\end{aligned}$$

Hence $(\hat{x}, z)$ is a feasible solution of problem (5.4) with the same cost.
Now suppose $(\hat{x}, z)$ is an optimal solution of problem (5.4). Since $f_{iJ_1^{(i)}}(\zeta) < f_{iJ_2^{(i)}}(\zeta) < \cdots < f_{iJ_{M_i}^{(i)}}(\zeta)$ for all $\zeta \geq 0$ and $i \in \mathcal{N}$ by assumption, it follows that, for all $i \in \mathcal{N}$, the sequence $z_{iJ_1^{(i)}}, z_{iJ_2^{(i)}}, \ldots, z_{iJ_{M_i}^{(i)}}$ is given recursively, starting from $m = M_i$, by

$$z_{iJ_m^{(i)}} = \max_{t \in T} \left\{ \sum_{k \in J_{M_i}^{(i)} \setminus J_{m-1}^{(i)}} \hat{x}^{(t)}_{ik} \right\} - \sum_{m'=m+1}^{M_i} z_{iJ_{m'}^{(i)}}.$$

Hence $z_{iJ_m^{(i)}} \geq 0$ for all $i \in \mathcal{N}$ and $m = 1, 2, \ldots, M_i$. We then set, starting from $m = M_i$ and $j \in J_{M_i}^{(i)}$,

$$x^{(t)}_{iJ_m^{(i)}j} := \min \left(\hat{x}^{(t)}_{ij} - \sum_{l=m+1}^{M_i} x_{iJ_l^{(i)}j}, z_{iJ_m^{(i)}} - \sum_{k \in J_{M_i}^{(i)} \setminus J_{m(i,j)}^{(i)}} x^{(t)}_{iJ_m^{(i)}k} \right).$$

It is now not difficult to see that (x, z) is a feasible solution of problem (5.3) with the same cost.
Therefore, the optimal costs of problems (5.3) and (5.4) are the same and, since the objective functions for the two problems are the same, z is part of an optimal solution for problem (5.3) if and only if it is part of an optimal solution for problem (5.4). □

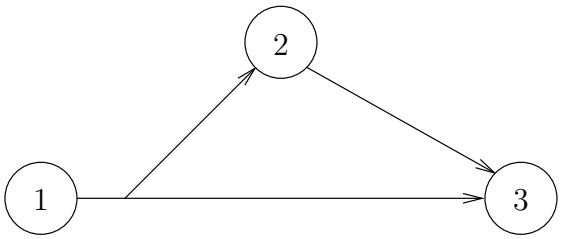


Fig. 5.3. The slotted Aloha relay channel. Reprinted with permission from [93].

5.1.1.2 Example: Slotted Aloha relay channel

This example, which we refer to as the *slotted Aloha relay channel*, relates to multi-hop wireless networks. One of most important issues in multi-hop wireless networks is medium access, i.e., determining how radio nodes share the wireless medium. A simple, yet popular, method for medium access control is slotted Aloha (see, e.g., [13, Section 4.2]), where nodes with packets to send follow simple random rules to determine when they transmit. In this example, we consider a multi-hop wireless network using slotted Aloha for medium access control.

We suppose that the network has the simple topology shown in Figure 5.3 and that, in this network, we wish to establish a single unicast connection of rate R from node 1 to node 3. The random rule we take for transmission is that the two transmitting nodes, node 1 and node 2, each transmit packets independently in a given time slot with some fixed probability. In coded packet networks, nodes are never "unbacklogged" as they are in regular, routed slotted Aloha networks—nodes can transmit coded packets whenever they are given the opportunity. Hence $z_{1(23)}$, the rate of packet injection on hyperarc $(1, \{2, 3\})$, is the probability that node 1 transmits a packet in a given time slot, and likewise z_{23}, the rate of packet injection on hyperarc $(2, 3)$, is the probability that node 2 transmits a packet in a given time slot. Therefore, $Z = [0, 1]^2$, i.e., $0 \leq z_{1(23)} \leq 1$ and $0 \leq z_{23} \leq 1$.

If node 1 transmits a packet and node 2 does not, then the packet is received at node 2 with probability $p_{1(23)2}$, at node 3 with probability $p_{1(23)3}$, and at both nodes 2 and 3 with probability $p_{1(23)(23)}$ (it is lost entirely with probability $1 - p_{1(23)2} - p_{1(23)3} - p_{1(23)(23)}$). If node 2 transmits a packet and node 1 does not, then the packet is received at node 3 with probability p_{233} (it is lost entirely with probability $1 - p_{233}$).

If both nodes 1 and 2 each transmit a packet, then the packets collide and neither of the packets is received successfully anywhere.

It is possible that simultaneous transmission does not necessarily result in collision, with one or more packets being received. This phenomenon is referred to as multipacket reception capability [48] and is decided by lower-layer implementation details. In this example, however, we simply assume that simultaneous transmission results in collision.

Hence, we have

$$z_{1(23)2} = z_{1(23)}(1 - z_{23})p_{1(23)2}, \tag{5.5}$$

$$z_{1(23)3} = z_{1(23)}(1 - z_{23})p_{1(23)3}, \tag{5.6}$$

$$z_{1(23)(23)} = z_{1(23)}(1 - z_{23})p_{1(23)(23)}, \tag{5.7}$$

and

$$z_{233} = (1 - z_{1(23)})z_{23}p_{233}. \tag{5.8}$$

We suppose that our objective is to set up the desired connection while minimizing the total number of packet transmissions for each message packet, perhaps for the sake of energy conservation or conservation of the wireless medium (to allow it to be used for other purposes, such as other connections). Therefore

$$f(z_{1(23)}, z_{23}) = z_{1(23)} + z_{23}.$$

The slotted Aloha relay channel is very similar to the relay channel introduced by van der Meulen [134], and determining the capacity of the latter is one of the famous, long-standing, open problems of information theory. The slotted Aloha relay channel is related to the relay channel (hence its name), but different. While the relay channel relates to the physical layer, we are concerned with higher layers, and our problem is soluble.

The relevant optimization problem to solve in this case is (5.2), and it reduces to

$$\begin{aligned}
&\text{minimize } z_{1(23)} + z_{23} \\
&\text{subject to } 0 \le z_{1(23)}, z_{23} \le 1, \\
&\qquad R \le z_{1(23)}(1 - z_{23})(p_{1(23)2} + p_{1(23)3} + p_{1(23)(23)}), \\
&\qquad R \le z_{1(23)}(1 - z_{23})(p_{1(23)3} + p_{1(23)(23)}) + (1 - z_{1(23)})z_{23}p_{233}.
\end{aligned}$$

Let us assume some values for the parameters of the problem and work through it. Let $R := 1/8$, $p_{1(23)2} := 9/16$, $p_{1(23)3} := 1/16$, $p_{1(23)(23)} :=$

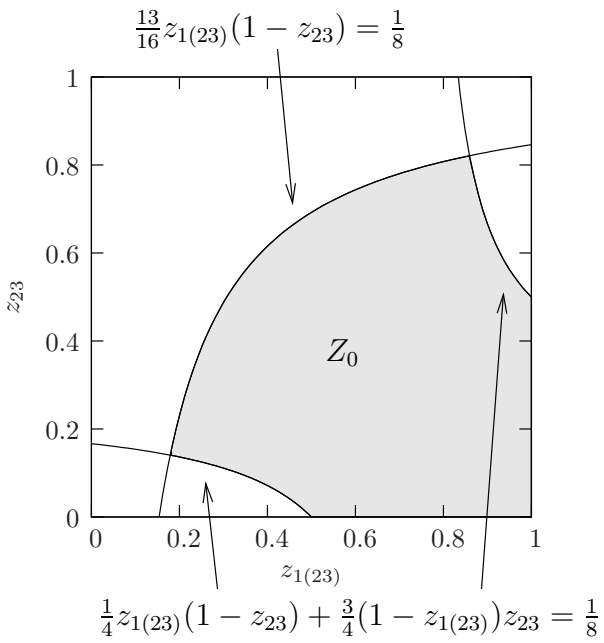


Fig. 5.4. Feasible set of problem (5.9). Reprinted with permission from [93].

3/16, and $p_{233} := 3/4$. Then the optimization problem we have is

$$\begin{aligned} &\text{minimize } z_{1(23)} + z_{23} \\ &\text{subject to } 0 \le z_{1(23)}, z_{23} \le 1, \\ &\qquad \frac{1}{8} \le \frac{13}{16} z_{1(23)}(1 - z_{23}), \\ &\qquad \frac{1}{8} \le \frac{1}{4} z_{1(23)}(1 - z_{23}) + \frac{3}{4}(1 - z_{1(23)}) z_{23}. \end{aligned} \tag{5.9}$$

The feasible set of this problem is shown in Figure 5.4. It is the shaded region labeled Z_0. By inspection, the optimal solution of (5.9) is the lesser of the two intersections between the curves defined by

$$\frac{13}{16} z_{1(23)}(1 - z_{23}) = \frac{1}{8}$$

and

$$\frac{1}{4} z_{1(23)}(1 - z_{23}) + \frac{3}{4}(1 - z_{1(23)}) z_{23} = \frac{1}{8}.$$

We obtain $z^*_{1(23)} \simeq 0.179$ and $z^*_{23} \simeq 0.141$.

The problem we have just solved is by no means trivial. We have taken

a wireless packet network subject to losses that are determined by a complicated set of conditions—including medium contention—and found a way of establishing a given unicast connection of fixed throughput using the minimum number of transmissions per message packet. The solution is that node 1 transmits a packet every time slot with probability 0.179, and node 2 transmits a packet every time slot independently with probability 0.141. Whenever either node transmits a packet, they follow the coding scheme of Section 4.1.

The network we dealt with was, unfortunately, only a small one, and the solution method we used will not straightforwardly scale to larger problems. But the solution method is conceptually simple, and there are cases where the solution to large problems is computable—and computable in a distributed manner. We deal with this topic next.

5.1.1.3 Distributed algorithms

In many cases, the optimization problems (5.2), (5.3), and (5.4) are convex or linear problems and their solutions can, in theory, be computed. For practical network applications, however, it is often important that solutions can be computed in a distributed manner, with each node making computations based only on local knowledge and knowledge acquired from information exchanges. Thus, we seek distributed algorithms to solve optimization problems (5.2), (5.3), and (5.4), which, when paired with the random linear coding scheme of the previous chapter, yields a distributed approach to efficient operation. The algorithms we propose will generally take some time to converge to an optimal solution, but it is not necessary to wait until the algorithms have converged before transmission—we can apply the coding scheme to the coding subgraph we have at any time, optimal or otherwise, and continue doing so while it converges. Such an approach is robust to dynamics such as changes in network topology that cause the optimal solution to change, because the algorithms will simply converge toward the changing optimum.

To this end, we simplify the problem by assuming that the objective function is of the form $f(z) = \sum_{(i,J)\in\mathcal{A}} f_{iJ}(z_{iJ})$, where f_{iJ} is a monotonically increasing, convex function, and that, as z_{iJ} is varied, z_{iJK}/z_{iJ} is constant for all $K \subset J$. Therefore, b_{iJK} is a constant for all $(i, J) \in \mathcal{A}$ and $K \subset J$. We also drop the constraint set Z, noting that separable constraints, at least, can be handled by making f_{iJ} approach infinity as z_{iJ} approaches its upper constraint. These assumptions apply if, at least from the perspective of the connection we wish to establish, arcs

essentially behave independently and the capacities of separate arcs are not coupled.

With these assumptions, problem (5.2) becomes

$$\begin{aligned} &\text{minimize} \sum_{(i,J)\in\mathcal{A}} f_{iJ}(z_{iJ}) \\ &\text{subject to} \sum_{j\in K} x_{iJj}^{(t)} \le z_{iJ} b_{iJK}, \qquad \forall\ (i,J)\in\mathcal{A},\ K\subset J,\ t\in T, \\ &\qquad x^{(t)} \in F^{(t)}, \qquad \forall\ t\in T. \end{aligned} \tag{5.10}$$

Since the f_{iJ} are monotonically increasing, the constraint

$$\sum_{j\in K} x_{iJj}^{(t)} \le z_{iJ} b_{iJK}, \qquad \forall\ (i,J)\in\mathcal{A},\ K\subset J,\ t\in T \tag{5.11}$$

gives

$$z_{iJ} = \max_{K\subset J, t\in T} \left\{ \frac{\sum_{j\in K} x_{iJj}^{(t)}}{b_{iJK}} \right\}. \tag{5.12}$$

Expression (5.12) is, unfortunately, not very useful for algorithm design because the max function is difficult to deal with, largely as a result of its not being differentiable everywhere. One way to overcome this difficulty is to approximate z_{iJ} by replacing the max in (5.12) with an l^m-norm (see [35]), i.e., to approximate z_{iJ} with z'_{iJ}, where

$$z'_{iJ} := \left(\sum_{K\subset J, t\in T} \left(\frac{\sum_{j\in K} x_{iJj}^{(t)}}{b_{iJK}} \right)^m \right)^{1/m}.$$

The approximation becomes exact as $m\to\infty$. Moreover, since $z'_{iJ} \ge z_{iJ}$ for all $m>0$, the coding subgraph z' admits the desired connection for any feasible solution.

Now the relevant optimization problem is

$$\begin{aligned} &\text{minimize} \sum_{(i,J)\in\mathcal{A}} f_{iJ}(z'_{iJ}) \\ &\text{subject to } x^{(t)} \in F^{(t)}, \qquad \forall\ t\in T, \end{aligned}$$

which is no more than a convex multicommodity flow problem. There are many algorithms for convex multicommodity flow problems (see [109] for a survey), some of which (e.g., the algorithms in [9, 12]) are well-suited for distributed implementation. The primal-dual approach to internet congestion control (see [129, Section 3.4]) can also be used to

solve convex multicommodity flow problems in a distributed manner, and we examine this method in Section 5.1.1.4.

There exist, therefore, numerous distributed algorithms for the subgraph selection problem—or, at least, for an approximation of the problem. What about distributed algorithms for the true problem? One clear tactic for finding such algorithms is to eliminate constraint (5.11) using Lagrange multipliers. Following this tactic, we obtain a distributed algorithm that we call the subgradient method. We describe the subgradient method in Section 5.1.1.5.

5.1.1.4 Primal-dual method

For the primal-dual method, we assume that the cost functions f_{iJ} are strictly convex and differentiable. Hence there is a unique optimal solution to problem (5.10). We present the algorithm for the lossless case, with the understanding that it can be straightforwardly extended to the lossy case. Thus, the optimization problem we address is

$$\begin{aligned} &\text{minimize} \sum_{(i,J)\in\mathcal{A}} f_{iJ}(z'_{iJ}) \\ &\text{subject to } x^{(t)} \in F^{(t)}, \qquad \forall\, t \in T, \end{aligned} \tag{5.13}$$

where

$$z'_{iJ} := \left(\sum_{t\in T} \left(\sum_{j\in J} x^{(t)}_{iJj} \right)^m \right)^{1/m}.$$

Let $(y)^+_a$ denote the following function of y:

$$(y)^+_a = \begin{cases} y & \text{if } a > 0, \\ \max\{y, 0\} & \text{if } a \le 0. \end{cases}$$

To solve problem (5.13) in a distributed fashion, we introduce additional variables p and λ and consider varying x, p, and λ in time τ according to the following time derivatives:

$$\dot{x}^{(t)}_{iJj} = -k^{(t)}_{iJj}(x^{(t)}_{iJj}) \left(\frac{\partial f_{iJ}(z'_{iJ})}{\partial x^{(t)}_{iJj}} + q^{(t)}_{ij} - \lambda^{(t)}_{iJj} \right), \tag{5.14}$$

$$\dot{p}^{(t)}_i = h^{(t)}_i(p^{(t)}_i)(y^{(t)}_i - \sigma^{(t)}_i), \tag{5.15}$$

$$\dot{\lambda}^{(t)}_{iJj} = m^{(t)}_{iJj}(\lambda^{(t)}_{iJj}) \left(-x^{(t)}_{iJj} \right)^+_{\lambda^{(t)}_{iJj}}, \tag{5.16}$$

where

$$q_{ij}^{(t)} := p_i^{(t)} - p_j^{(t)},$$
$$y_i^{(t)} := \sum_{\{J|(i,J)\in\mathcal{A}\}} \sum_{j\in J} x_{iJj}^{(t)} - \sum_{\{j|(j,I)\in\mathcal{A}, i\in I\}} x_{jIi}^{(t)},$$

and $k_{iJj}^{(t)}(x_{iJj}^{(t)}) > 0$, $h_i^{(t)}(p_i^{(t)}) > 0$, and $m_{iJj}^{(t)}(\lambda_{iJj}^{(t)}) > 0$ are non-decreasing continuous functions of $x_{iJj}^{(t)}$, $p_i^{(t)}$, and $\lambda_{iJj}^{(t)}$ respectively.

Proposition 5.2 *The algorithm specified by Equations (5.14)–(5.16) is globally, asymptotically stable.*

Proof We prove the stability of the primal-dual algorithm by using the theory of Lyapunov stability (see, e.g., [129, Section 3.10]). This proof is based on the proof of Theorem 3.7 of [129].

The Lagrangian for problem (5.13) is as follows:

$$\begin{aligned} L(x,p,\lambda) = & \sum_{(i,J)\in\mathcal{A}} f_{iJ}(z'_{iJ}) \\ & + \sum_{t\in T}\left\{\sum_{i\in\mathcal{N}} p_i^{(t)}\left(\sum_{\{J|(i,J)\in\mathcal{A}\}}\sum_{j\in J} x_{iJj}^{(t)} - \sum_{\{j|(j,I)\in\mathcal{A}, i\in I\}} x_{jIi}^{(t)} - \sigma_i^{(t)}\right)\right. \\ & \qquad \left. - \sum_{(i,J)\in\mathcal{A}}\sum_{j\in J} \lambda_{iJj}^{(t)} x_{iJj}^{(t)}\right\}, \end{aligned} \tag{5.17}$$

where

$$\sigma_i^{(t)} = \begin{cases} R_t & \text{if } i = s, \\ -R_t & \text{if } i = t, \\ 0 & \text{otherwise.} \end{cases}$$

Since the objective function of problem (5.13) is strictly convex, it has a unique minimizing solution, say $\hat{x}$, and Lagrange multipliers, say $\hat{p}$ and $\hat{\lambda}$, which satisfy the following Karush-Kuhn-Tucker conditions:

$$\frac{\partial L(\hat{x},\hat{p},\hat{\lambda})}{\partial x_{iJj}^{(t)}} = \left(\frac{\partial f_{iJ}(z'_{iJ})}{\partial x_{iJj}^{(t)}} + \left(\hat{p}_i^{(t)} - \hat{p}_j^{(t)}\right) - \hat{\lambda}_{iJj}^{(t)}\right) = 0, \quad \forall\ (i,J)\in\mathcal{A},\ j\in J,\ t\in T, \tag{5.18}$$

$$\sum_{\{J|(i,J)\in\mathcal{A}\}}\sum_{j\in J}\hat{x}_{iJj}^{(t)}-\sum_{\{j|(j,I)\in\mathcal{A},i\in I\}}\hat{x}_{jIi}^{(t)}=\sigma_i^{(t)},\qquad\forall\ i\in\mathcal{N},\ t\in T, \tag{5.19}$$

$$\hat{x}_{iJj}^{(t)}\geq 0\qquad\forall\ (i,J)\in\mathcal{A},\ j\in J,\ t\in T, \tag{5.20}$$

$$\hat{\lambda}_{iJj}^{(t)}\geq 0\qquad\forall\ (i,J)\in\mathcal{A},\ j\in J,\ t\in T, \tag{5.21}$$

$$\hat{\lambda}_{iJj}^{(t)}\hat{x}_{iJj}^{(t)}=0\qquad\forall\ (i,J)\in\mathcal{A},\ j\in J,\ t\in T. \tag{5.22}$$

Using equation (5.17), we see that $(\hat{x},\hat{p},\hat{\lambda})$ is an equilibrium point of the primal-dual algorithm. We now prove that this point is globally, asymptotically stable.

Consider the following function as a candidate for the Lyapunov function:

$$V(x,p,\lambda)=\sum_{t\in T}\left\{\sum_{(i,J)\in\mathcal{A}}\sum_{j\in J}\left(\int_{\hat{x}_{iJj}^{(t)}}^{x_{iJj}^{(t)}}\frac{1}{k_{iJj}^{(t)}(\sigma)}(\sigma-\hat{x}_{iJj}^{(t)})d\sigma\right.\right.$$
$$\left.\left.+\int_{\hat{\lambda}_{iJj}^{(t)}}^{\lambda_{iJj}^{(t)}}\frac{1}{m_{iJj}^{(t)}(\gamma)}(\gamma-\hat{\lambda}_{iJj}^{(t)})d\gamma\right)+\sum_{i\in\mathcal{N}}\int_{\hat{p}_i^{(t)}}^{p_i^{(t)}}\frac{1}{h_i^{(t)}(\beta)}(\beta-\hat{p}_i^{(t)})d\beta\right\}.$$

Note that $V(\hat{x},\hat{p},\hat{\lambda})=0$. Since, $k_{iJj}^{(t)}(\sigma)>0$, if $x_{iJj}^{(t)}\neq\hat{x}_{iJj}^{(t)}$, we have

$$\int_{x_{iJj}^{(t)}}^{x_{iJj}^{(t)}}\frac{1}{k_{iJj}^{(t)}(\sigma)}(\sigma-\hat{x}_{iJj}^{(t)})d\sigma>0.$$

This argument can be extended to the other terms as well. Thus, whenever $(x,p,\lambda)\neq(\hat{x},\hat{p},\hat{\lambda})$, we have $V(x,p,\lambda)>0$.

Now,

$$\dot{V}=\sum_{t\in T}\left\{\sum_{(i,J)\in\mathcal{A}}\sum_{j\in J}\left[\left(-x_{iJj}^{(t)}\right)_{\lambda_{iJj}^{(t)}}^{+}(\lambda_{iJj}^{(t)}-\hat{\lambda}_{iJj}^{(t)})\right.\right.$$
$$\left.-\left(\frac{\partial f_{iJ}(z'_{iJ})}{\partial x_{iJj}^{(t)}}+q_{iJj}^{(t)}-\lambda_{iJj}^{(t)}\right)\cdot(x_{iJj}^{(t)}-\hat{x}_{iJj}^{(t)})\right]$$
$$\left.+\sum_{i\in\mathcal{N}}(y_i^{(t)}-\sigma_i^{(t)})(p_i^{(t)}-\hat{p}_i^{(t)})\right\}.$$

Note that

$$\left(-x_{iJj}^{(t)}\right)_{\lambda_{iJj}^{(t)}}^{+}(\lambda_{iJj}^{(t)}-\hat{\lambda}_{iJj}^{(t)})\leq-x_{iJj}^{(t)}(\lambda_{iJj}^{(t)}-\hat{\lambda}_{iJj}^{(t)}),$$

since the inequality is an equality if either $x_{iJj}^{(t)} \leq 0$ or $\lambda_{iJj}^{(t)} \geq 0$; and, in the case when $x_{iJj}^{(t)} > 0$ and $\lambda_{iJj}^{(t)} < 0$, we have $(-x_{iJj}^{(t)})^{+}_{\lambda_{iJj}^{(t)}} = 0$ and, since $\hat{\lambda}_{iJj}^{(t)} \geq 0$, $-x_{iJj}^{(t)}(\lambda_{iJj}^{(t)} - \hat{\lambda}_{iJj}^{(t)}) \geq 0$. Therefore,

$$
\begin{aligned}
\dot{V} \leq \sum_{t\in T} \Bigg\{ & \sum_{(i,J)\in\mathcal{A}} \sum_{j\in J} \Big[-x_{iJj}^{(t)}(\lambda_{iJj}^{(t)} - \hat{\lambda}_{iJj}^{(t)}) \\
& \qquad - \left(\frac{\partial f_{iJ}(z'_{iJ})}{\partial x_{iJj}^{(t)}} + q_{iJj}^{(t)} - \lambda_{iJj}^{(t)} \right) \cdot (x_{iJj}^{(t)} - \hat{x}_{iJj}^{(t)}) \Big] \\
& \quad + \sum_{i\in\mathcal{N}} (y_i^{(t)} - \sigma_i^{(t)})(p_i^{(t)} - \hat{p}_i^{(t)}) \Bigg\} \\
= {} & (\hat{q} - q)'(x - \hat{x}) + (\hat{p} - p)'(y - \hat{y}) \\
& + \sum_{t\in T} \Bigg\{ \sum_{(i,J)\in\mathcal{A}} \sum_{j\in J} \Big[-\hat{x}_{iJj}^{(t)}(\lambda_{iJj}^{(t)} - \hat{\lambda}_{iJj}^{(t)}) \\
& \qquad - \left(\frac{\partial f_{iJ}(z'_{iJ})}{\partial x_{iJj}^{(t)}} + \hat{q}_{iJj}^{(t)} - \hat{\lambda}_{iJj}^{(t)} \right) \cdot (x_{iJj}^{(t)} - \hat{x}_{iJj}^{(t)}) \Big] \\
& \quad + \sum_{i\in\mathcal{N}} (\hat{y}_i^{(t)} - \sigma_i^{(t)})(p_i^{(t)} - \hat{p}_i^{(t)}) \Bigg\} \\
= {} & \sum_{t\in T} \sum_{(i,J)\in\mathcal{A}} \sum_{j\in J} \left(\frac{\partial f_{iJ}(\hat{z}'_{iJ})}{\partial \hat{x}_{iJj}^{(t)}} - \frac{\partial f_{iJ}(z'_{iJ})}{\partial x_{iJj}^{(t)}} \right) (x_{iJj}^{(t)} - \hat{x}_{iJj}^{(t)}) - \lambda'\hat{x},
\end{aligned}
$$

where the last line follows from Karush-Kuhn-Tucker conditions (5.18)–(5.22) and the fact that

$$
\begin{aligned}
p'y &= \sum_{t\in T} \sum_{i\in\mathcal{N}} p_i^{(t)} \left(\sum_{\{J|(i,J)\in\mathcal{A}\}} \sum_{j\in J} x_{iJj}^{(t)} - \sum_{\{j|(j,I)\in\mathcal{A}, i\in I\}} x_{jIi}^{(t)} \right) \\
&= \sum_{t\in T} \sum_{(i,J)\in\mathcal{A}} \sum_{j\in J} x_{iJj}^{(t)} (p_i^{(t)} - p_j^{(t)}) = q'x.
\end{aligned}
$$

Thus, owing to the strict convexity of the functions $\{f_{iJ}\}$, we have $\dot{V} \leq -\lambda'\hat{x}$, with equality if and only if $x = \hat{x}$. So it follows that $\dot{V} \leq 0$ for all $\lambda \geq 0$, since $\hat{x} \geq 0$.

If the initial choice of λ is such that $\lambda(0) \geq 0$, we see from the primal-dual algorithm that $\lambda(\tau) \geq 0$. This is true since $\dot{\lambda} \geq 0$ whenever $\lambda \leq 0$.

Thus, it follows by the theory of Lyapunov stability that the algorithm is indeed globally, asymptotically stable. □

The global, asymptotic stability of the algorithm implies that no matter what the initial choice of (x, p) is, the primal-dual algorithm will converge to the unique solution of problem (5.13). We have to choose λ, however, with non-negative entries as the initial choice. Further, there is no guarantee that $x(\tau)$ yields a feasible solution for any given τ. Therefore, a start-up time may be required before a feasible solution is obtained.

The algorithm that we currently have is a continuous time algorithm and, in practice, an algorithm operating in discrete message exchanges is required. To discretize the algorithm, we consider time steps $n = 0, 1, \ldots$ and replace the derivatives by differences:

$$x_{iJj}^{(t)}[n+1] = x_{iJj}^{(t)}[n] - \alpha_{iJj}^{(t)}[n] \left(\frac{\partial f_{iJ}(z'_{iJ}[n])}{\partial x_{iJj}^{(t)}[n]} + q_{ij}^{(t)}[n] - \lambda_{iJj}^{(t)}[n] \right), \tag{5.23}$$

$$p_i^{(t)}[n+1] = p_i^{(t)}[n] + \beta_i^{(t)}[n](y_i^{(t)}[n] - \sigma_i^{(t)}), \tag{5.24}$$

$$\lambda_{iJj}^{(t)}[n+1] = \lambda_{iJj}^{(t)}[n] + \gamma_{iJj}^{(t)}[n] \left(-x_{iJj}^{(t)}[n] \right)^+_{\lambda_{iJj}^{(t)}[n]}, \tag{5.25}$$

where

$$q_{ij}^{(t)}[n] := p_i^{(t)}[n] - p_j^{(t)}[n],$$
$$y_i^{(t)}[n] := \sum_{\{J|(i,J)\in\mathcal{A}\}} \sum_{j\in J} x_{iJj}^{(t)}[n] - \sum_{\{j|(j,I)\in\mathcal{A}, i\in I\}} x_{jIi}^{(t)}[n],$$

and $\alpha_{iJj}^{(t)}[n] > 0$, $\beta_i^{(t)}[n] > 0$, and $\gamma_{iJj}^{(t)}[n] > 0$ are step sizes. This discretized algorithm operates in synchronous rounds, with nodes exchanging information in each round. It is expected that this synchronicity can be relaxed in practice.

We associate a processor with each node. We assume that the processor for node i keeps track of the variables p_i, $\{x_{iJj}\}_{\{J,j|(i,J)\in\mathcal{A}, j\in J\}}$, and $\{\lambda_{iJj}\}_{\{J,j|(i,J)\in\mathcal{A}, j\in J\}}$. With such an assignment of variables to processors, the algorithm is distributed in the sense that a node exchanges information only with its neighbors at every iteration of the primal-dual algorithm. We summarize the primal-dual method in Figure 5.5.

(i) Each node i initializes $p_i[0]$, $\{x_{iJj}[0]\}_{\{J,j|(i,J)\in\mathcal{A},j\in J\}}$, and $\{\lambda_{iJj}[0]\}_{\{J,j|(i,J)\in\mathcal{A},j\in J\}}$ such that $\lambda_{iJj}[0] \geq 0$ for all (J,j) such that $(i,J) \in \mathcal{A}$ and $j \in J$. Each node i sends $p_i[0]$, $\{x_{iJj}[0]\}_{j\in J}$, and $\{\lambda_{iJj}[0]\}_{j\in J}$ over each outgoing hyperarc (i,J).

(ii) At the nth iteration, each node i computes $p_i[n+1]$, $\{x_{iJj}[n+1]\}_{\{J,j|(i,J)\in\mathcal{A},j\in J\}}$, and $\{\lambda_{iJj}[n+1]\}_{\{J,j|(i,J)\in\mathcal{A},j\in J\}}$ using equations (5.23)–(5.25). Each node i sends $p_i[n+1]$, $\{x_{iJj}[n+1]\}_{j\in J}$, and $\{\lambda_{iJj}[n+1]\}_{j\in J}$ over each outgoing hyperarc (i,J).

(iii) The current coding subgraph $z'[n]$ is computed. For each node i, we set

$$z'_{iJ}[n] := \left(\sum_{t\in T}\left(\sum_{j\in J} x^{(t)}_{iJj}[n]\right)^m\right)^{1/m}$$

for all outgoing hyperarcs (i,J).

(iv) Steps (ii) and (iii) are repeated until the sequence of coding subgraphs $\{z'[n]\}$ converges.

Fig. 5.5. Summary of the primal-dual method.

5.1.1.5 Subgradient method

We present the subgradient method for linear cost functions; with some modifications, it may be made to apply also to convex ones. Thus, we assume that the objective function f is of the form

$$f(z) := \sum_{(i,J)\in\mathcal{A}} a_{iJ} z_{iJ},$$

where $a_{iJ} > 0$.

Consider the Lagrangian dual of problem (5.10):

$$\begin{aligned} &\text{maximize } \sum_{t\in T} q^{(t)}(p^{(t)}) \\ &\text{subject to } \sum_{t\in T}\sum_{K\subset J} p^{(t)}_{iJK} = a_{iJ} \qquad \forall\ (i,J)\in\mathcal{A}, \\ &\qquad\qquad p^{(t)}_{iJK} \geq 0, \qquad \forall\ (i,J)\in\mathcal{A},\ K\subset J,\ t\in T, \end{aligned} \tag{5.26}$$

where

$$q^{(t)}(p^{(t)}) := \min_{x^{(t)}\in F^{(t)}} \sum_{(i,J)\in\mathcal{A}}\sum_{j\in J}\left(\sum_{\{K\subset J|K\ni j\}} \frac{p^{(t)}_{iJK}}{b_{iJK}}\right) x_{iJj}. \tag{5.27}$$

In the lossless case, the dual problem defined by equations (5.26) and (5.27) simplifies somewhat, and we require only a single dual variable

$p^{(t)}_{iJJ}$ for each hyperarc (i, J). In the case that relates to optimization problem (5.4), the dual problem simplifies more still, as there are fewer primal variables associated with it. Specifically, we obtain, for the Lagrangian dual,

$$\begin{aligned} \text{maximize} \quad & \sum_{t\in T} \hat{q}^{(t)}(p^{(t)}) \\ \text{subject to} \quad & \sum_{t\in T} p^{(t)}_{iJ^{(i)}_m} = s_{iJ^{(i)}_m}, \qquad \forall\ i \in \mathcal{N},\ m = 1, \ldots, M_i, \\ & p^{(t)}_{iJ} \geq 0, \qquad \forall\ (i, J) \in \mathcal{A},\ t \in T, \end{aligned} \tag{5.28}$$

where

$$s_{iJ^{(i)}_m} := a_{iJ^{(i)}_m} - a_{iJ^{(i)}_{m-1}},$$

and

$$\hat{q}^{(t)}(p^{(t)}) := \min_{\hat{x}^{(t)}\in\hat{F}^{(t)}} \sum_{(i,j)\in\mathcal{A}'} \left(\sum_{m=1}^{m(i,j)} p^{(t)}_{iJ^{(i)}_m} \right) \hat{x}^{(t)}_{ij}. \tag{5.29}$$

Note that, by the assumptions of the problem, $s_{iJ} > 0$ for all $(i, J) \in \mathcal{A}$.

In all three cases, the dual problems are very similar, and essentially the same algorithm can be used to solve them. We present the subgradient method for the case that relates to optimization problem (5.4)—namely, the primal problem

$$\begin{aligned} \text{minimize} \quad & \sum_{(i,J)\in\mathcal{A}} a_{iJ} z_{iJ} \\ \text{subject to} \quad & \sum_{k\in J^{(i)}_{M_i}\setminus J^{(i)}_{m-1}} \hat{x}^{(t)}_{ik} \leq \sum_{n=m}^{M_i} z_{iJ^{(i)}_n}, \\ & \qquad\qquad \forall\ i \in \mathcal{N},\ m = 1, \ldots, M_i,\ t \in T, \\ & \hat{x}^{(t)} \in \hat{F}^{(t)}, \qquad \forall\ t \in T \end{aligned} \tag{5.30}$$

with dual (5.28)—with the understanding that straightforward modifications can be made for the other cases.

We first note that problem (5.29) is, in fact, a shortest path problem, which admits a simple, asynchronous distributed solution known as the distributed asynchronous Bellman-Ford algorithm (see, e.g., [13, Section 5.2]).

Now, to solve the dual problem (5.28), we employ subgradient optimization (see, e.g., [10, Section 6.3.1] or [108, Section I.2.4]). We start with an iterate $p[0]$ in the feasible set of (5.28) and, given an iterate $p[n]$

for some non-negative integer n, we solve problem (5.29) for each t in T to obtain $x[n]$. Let

$$g^{(t)}_{iJ^{(i)}_m}[n] := \sum_{k \in J^{(i)}_{M_i} \setminus J^{(i)}_{m-1}} \hat{x}^{(t)}_{ik}[n].$$

We then assign

$$p_{iJ}[n+1] := \arg\min_{v \in P_{iJ}} \sum_{t \in T} (v^{(t)} - (p^{(t)}_{iJ}[n] + \theta[n] g^{(t)}_{iJ}[n]))^2 \tag{5.31}$$

for each $(i, J) \in \mathcal{A}$, where P_{iJ} is the $|T|$-dimensional simplex

$$P_{iJ} = \left\{ v \,\middle|\, \sum_{t \in T} v^{(t)} = s_{iJ},\, v \geq 0 \right\}$$

and $\theta[n] > 0$ is an appropriate step size. In other words, $p_{iJ}[n+1]$ is set to be the Euclidean projection of $p_{iJ}[n] + \theta[n] g_{iJ}[n]$ onto P_{iJ}.

To perform the projection, we use the following proposition.

Proposition 5.3 *Let $u := p_{iJ}[n] + \theta[n] g_{iJ}[n]$. Suppose we index the elements of T such that $u^{(t_1)} \geq u^{(t_2)} \geq \ldots \geq u^{(t_{|T|})}$. Take $\hat{k}$ to be the smallest k such that*

$$\frac{1}{k} \left(s_{iJ} - \sum_{r=1}^{t_k} u^{(r)} \right) \leq -u^{(t_{knn+1})}$$

or set $\hat{k} = |T|$ if no such k exists. Then the projection (5.31) is achieved by

$$p^{(t)}_{iJ}[n+1] = \begin{cases} u^{(t)} + \frac{s_{iJ} - \sum_{r=1}^{t_{\hat{k}}} u^{(r)}}{kn} & \text{if } t \in \{t_1, \ldots, t_{\hat{k}}\}, \\ 0 & \text{otherwise.} \end{cases}$$

Proof We wish to solve the following problem.

$$\begin{aligned} &\text{minimize} \sum_{t \in T} (v^{(t)} - u^{(t)})^2 \\ &\text{subject to } v \in P_{iJ}. \end{aligned}$$

First, since the objective function and the constraint set P_{iJ} are both convex, it is straightforward to establish that a necessary and sufficient condition for global optimality of $\hat{v}^{(t)}$ in P_{iJ} is

$$\hat{v}^{(t)} > 0 \Rightarrow (u^{(t)} - \hat{v}^{(t)}) \geq (u^{(r)} - \hat{v}^{(r)}), \qquad \forall\, r \in T \tag{5.32}$$

(see, e.g., [10, Section 2.1]). Suppose we index the elements of T such that $u^{(t_1)} \geq u^{(t_2)} \geq \ldots \geq u^{(t_{|T|})}$. We then note that there must be an index k in the set $\{1, \ldots, |T|\}$ such that $v^{(t_l)} > 0$ for $l = 1, \ldots, k$ and $v^{(t_l)} = 0$ for $l > kn + 1$, for, if not, then a feasible solution with lower cost can be obtained by swapping around components of the vector. Therefore, condition (5.32) implies that there must exist some d such that $\hat{v}^{(t)} = u^{(t)} + d$ for all $t \in \{t_1, \ldots, t_k n\}$ and that $d \leq -u^{(t)}$ for all $t \in \{t_{kn+1}, \ldots, t_{|T|}\}$, which is equivalent to $d \leq -u^{(t_{k+1})}$. Since $\hat{v}^{(t)}$ is in the simplex P_{iJ}, it follows that

$$kd + \sum_{t=1}^{t_k} u^{(t)} = s_{iJ},$$

which gives

$$d = \frac{1}{k}\left(s_{iJ} - \sum_{t=1}^{t_k} u^{(t)}\right).$$

By taking $k = \hat{k}$, where $\hat{k}$ is the smallest k such that

$$\frac{1}{k}\left(s_{iJ} - \sum_{r=1}^{t_k} u^{(r)}\right) \leq -u^{(t_{k+1})},$$

(or, if no such k exists, then $\hat{k} = |T|$), we see that we have

$$\frac{1}{\hat{k}-1}\left(s_{iJ} - \sum_{t=1}^{t_{k-1}} u^{(t)}\right) > -u^{(t_k)},$$

which can be rearranged to give

$$d = \frac{1}{\hat{k}}\left(s_{iJ} - \sum_{t=1}^{t_k} u^{(t)}\right) > -u^{(t_k)}.$$

Hence, if $v^{(t)}$ is given by

$$v^{(t)} = \begin{cases} u^{(t)} + \frac{s_{iJ} - \sum_{r=1}^{t_{\hat{k}}} u^{(r)}}{\hat{k}} & \text{if } t \in \{t_1, \ldots, t_{\hat{k}}\}, \\ 0 & \text{otherwise,} \end{cases} \tag{5.33}$$

then $v^{(t)}$ is feasible and we see that the optimality condition (5.32) is satisfied. Note that, since $d \leq -u^{(t_{k+1})}$, equation (5.33) can also be written as

$$v^{(t)} = \max\left(0, u^{(t)} + \frac{1}{\hat{k}}\left(s_{iJ} - \sum_{r=1}^{t_{\hat{k}}} u^{(r)}\right)\right). \tag{5.34}$$

□

The disadvantage of subgradient optimization is that, whilst it yields good approximations of the optimal value of the Lagrangian dual problem (5.28) after sufficient iteration, it does not necessarily yield a primal optimal solution. There are, however, methods for recovering primal solutions in subgradient optimization. We employ the following method, which is due to Sherali and Choi [125].

Let $\{\mu_l[n]\}_{l=1,\ldots,n}$ be a sequence of convex combination weights for each non-negative integer n, i.e., $\sum_{l=1}^{n} \mu_l[n] = 1$ and $\mu_l[n] \geq 0$ for all $l = 1, \ldots, n$. Further, let us define

$$\gamma_{ln} := \frac{\mu_l[n]}{\theta[n]}, \quad l = 1, \ldots, n,\ n = 0, 1, \ldots,$$

and

$$\Delta\gamma_n^{\max} := \max_{l=2,\ldots,n} \{\gamma_{ln} - \gamma_{(l-1)n}\}.$$

Proposition 5.4 *If the step sizes $\{\theta[n]\}$ and convex combination weights $\{\mu_l[n]\}$ are chosen such that*

(i) $\gamma_{ln} \geq \gamma_{(l-1)n}$ *for all* $l = 2, \ldots, n$ *and* $n = 0, 1, \ldots$,
(ii) $\Delta\gamma_n^{\max} \to 0$ *as* $n \to \infty$, *and*
(iii) $\gamma_{1n} \to 0$ *as* $n \to \infty$ *and* $\gamma_{nn} \leq \delta$ *for all* $n = 0, 1, \ldots$ *for some* $\delta > 0$,

then we obtain an optimal solution to the primal problem from any accumulation point of the sequence of primal iterates $\{\tilde{x}[n]\}$ given by

$$\tilde{x}[n] := \sum_{l=1}^{n} \mu_l[n]\hat{x}[l], \quad n = 0, 1, \ldots. \tag{5.35}$$

Proof Suppose that the dual feasible solution that the subgradient method converges to is $\bar{p}$. Then, using (5.31), there exists some m such that for $n \geq m$

$$p_{iJ}^{(t)}[n+1] = p_{iJ}^{(t)}[n] + \theta[n]g_{iJ}^{(t)}[n] + c_{iJ}[n]$$

for all $(i, J) \in \mathcal{A}$ and $t \in T$ such that $\bar{p}_{iJ}^{(t)} > 0$.
Let $\tilde{g}[n] := \sum_{l=1}^{n} \mu_l[n]g[l]$. Consider some $(i, J) \in \mathcal{A}$ and $t \in T$. If

$\bar{p}_{iJ}^{(t)} > 0$, then for $n > m$ we have

$$\begin{aligned}\tilde{g}_{iJ}^{(t)}[n] &= \sum_{l=1}^{m} \mu_l[n] g_{iJ}^{(t)}[l] + \sum_{l=m+1}^{n} \mu_l[n] g_{iJ}^{(t)}[l] \\ &= \sum_{l=1}^{m} \mu_l[n] g_{iJ}^{(t)}[l] + \sum_{l=m+1}^{n} \frac{\mu_l[n]}{\theta[n]} (p_{iJ}^{(t)}[n+1] - p_{iJ}^{(t)}[n] - c_{iJ}[n]) \\ &= \sum_{l=1}^{m} \mu_l[n] g_{iJ}^{(t)}[l] + \sum_{l=m+1}^{n} \gamma_{ln} (p_{iJ}^{(t)}[n+1] - p_{iJ}^{(t)}[n]) \\ &\quad - \sum_{l=m+1}^{n} \gamma_{ln} c_{iJ}[n].\end{aligned} \tag{5.36}$$

Otherwise, if $\bar{p}_{iJ}^{(t)} = 0$, then from equation (5.34), we have

$$p_{iJ}^{(t)}[n+1] \geq p_{iJ}^{(t)}[n] + \theta[n] g_{iJ}^{(t)}[n] + c_{iJ}[n],$$

so

$$\tilde{g}_{iJ}^{(t)}[n] \leq \sum_{l=1}^{m} \mu_l[n] g_{iJ}^{(t)}[l] + \sum_{l=m+1}^{n} \gamma_{ln} (p_{iJ}^{(t)}[n+1] - p_{iJ}^{(t)}[n]) - \sum_{l=m+1}^{n} \gamma_{ln} c_{iJ}[n]. \tag{5.37}$$

It is straightforward to see that the sequence of iterates $\{\tilde{x}[n]\}$ is primal feasible, and that we obtain a primal feasible sequence $\{z[n]\}$ by setting

$$\begin{aligned}z_{iJ_m^{(i)}}[n] &:= \max_{t \in T} \left\{ \sum_{k \in J_{M_i}^{(i)} \setminus J_{m-1}^{(i)}} \tilde{x}_{ik}^{(t)}[n] \right\} - \sum_{m'=m+1}^{M_i} z_{iJ_{m'}^{(i)}}[n] \\ &= \max_{t \in T} \tilde{g}_{iJ_m^{(i)}} - \sum_{m'=m+1}^{M_i} z_{iJ_{m'}^{(i)}}[n]\end{aligned}$$

recursively, starting from $m = M_i$ and proceeding through to $m = 1$. Sherali and Choi [125] showed that, if the required conditions on the step sizes $\{\theta[n]\}$ and convex combination weights $\{\mu_l[n]\}$ are satisfied, then

$$\sum_{l=1}^{m} \mu_l[n] g_{iJ}^{(t)}[l] + \sum_{l=m+1}^{n} \gamma_{ln} (p_{iJ}^{(t)}[n+1] - p_{iJ}^{(t)}[n]) \to 0$$

as $k \to \infty$; hence we see from equations (5.36) and (5.37) that, for k

sufficiently large,

$$\sum_{m'=m}^{M_i} z_{iJ_{m'}^{(i)}}[n] = -\sum_{l=m+1}^{n} \gamma_{ln} c_{iJ_m^{(i)}}[n].$$

Recalling the primal problem (5.30), we see that complementary slackness with $\bar{p}$ holds in the limit of any convergent subsequence of $\{\tilde{x}[n]\}$. □

The required conditions on the step sizes and convex combination weights are satisfied by the following choices ([125, Corollaries 2–4]):

(i) step sizes $\{\theta[n]\}$ such that $\theta[n] > 0$, $\lim_{n\to 0} \theta[n] = 0$, $\sum_{n=1}^{\infty} \theta_n = \infty$, and convex combination weights $\{\mu_l[n]\}$ given by $\mu_l[n] = \theta[l]/\sum_{k=1}^{n} \theta[k]$ for all $l = 1, \ldots, n$, $n = 0, 1, \ldots$;
(ii) step sizes $\{\theta[n]\}$ given by $\theta[n] = a/(b + cn)$ for all $n = 0, 1, \ldots$, where $a > 0$, $b \geq 0$ and $c > 0$, and convex combination weights $\{\mu_l[n]\}$ given by $\mu_l[n] = 1/n$ for all $l = 1, \ldots, n$, $n = 0, 1, \ldots$; and
(iii) step sizes $\{\theta[n]\}$ given by $\theta[n] = n^{-\alpha}$ for all $n = 0, 1, \ldots$, where $0 < \alpha < 1$, and convex combination weights $\{\mu_l[n]\}$ given by $\mu_l[n] = 1/n$ for all $l = 1, \ldots, n$, $n = 0, 1, \ldots$.

Moreover, for all three choices, we have $\mu_l[n+1]/\mu_l[n]$ independent of l for all n, so primal iterates can be computed iteratively using

$$\begin{aligned}\tilde{x}[n] &= \sum_{l=1}^{n} \mu_l[n]\hat{x}[l] \\ &= \sum_{l=1}^{n-1} \mu_l[n]\hat{x}[l] + \mu_n[n]\hat{x}[n] \\ &= \phi[n-1]\tilde{x}[n-1] + \mu_n[n]\hat{x}[n],\end{aligned}$$

where $\phi[n] := \mu_l[n+1]/\mu_l[n]$.

This gives us our distributed algorithm. We summarize the subgradient method in Figure 5.6. We see that, although the method is indeed a distributed algorithm, it again operates in synchronous rounds. Again, it is expected that this synchronicity can be relaxed in practice.

5.1.1.6 Application: Minimum-transmission wireless unicast

We have now discussed sufficient material to allow us to establish coded connections. But is it worthwhile to do so? Surely, coding should not be used for all network communications; in some situations, the gain from coding is not sufficient to justify the additional work, and packets

(i) Each node i computes s_{iJ} for its outgoing hyperarcs and initializes $p_{iJ}[0]$ to a point in the feasible set of (5.28). For example, we take $p_{iJ}^{(t)}[0] := s_{iJ}/|T|$. Each node i sends s_{iJ} and $p_{iJ}[0]$ over each outgoing hyperarc (i, J).

(ii) At the nth iteration, use $p^{(t)}[n]$ as the hyperarc costs and run a distributed shortest path algorithm, such as distributed Bellman-Ford, to determine $\hat{x}^{(t)}[n]$ for all $t \in T$.

(iii) Each node i computes $p_{iJ}[n+1]$ for its outgoing hyperarcs using Proposition 5.3. Each node i sends $p_{iJ}[n+1]$ over each outgoing hyperarc (i, J).

(iv) Nodes compute the primal iterate $\tilde{x}[n]$ by setting

$$\tilde{x}[n] := \sum_{l=1}^{n} \mu_l[n]\hat{x}[l].$$

(v) The current coding subgraph $z[n]$ is computed using the primal iterate $\tilde{x}[n]$. For each node i, we set

$$z_{iJ_m^{(i)}}[n] := \max_{t \in T} \left\{ \sum_{k \in J_{M_i}^{(i)} \setminus J_{m-1}^{(i)}} \tilde{x}_{ik}^{(t)}[n] \right\} - \sum_{m'=m+1}^{M_i} z_{iJ_{m'}^{(i)}}[n]$$

recursively, starting from $m = M_i$ and proceeding through to $m = 1$.

(vi) Steps (ii)–(v) are repeated until the sequence of primal iterates $\{\tilde{x}[n]\}$ converges.

Fig. 5.6. Summary of the subgradient method.

should simply be routed. In this section, we describe an application where coding is worthwhile.

We consider the problem of minimum-transmission wireless unicast—the problem of establishing a unicast connection in a lossy wireless network using the minimum number of transmissions per packet. This efficiency criterion is the same as that in Section 5.1.1.2; it is a generic efficiency criterion reflects the fact that sending packets unnecessarily wastes both energy and bandwidth.

There are numerous approaches to wireless unicast; we consider five, three of which (approaches (i)–(iii)) are routed approaches and two of which (approaches (iv) and (v)) are coded approaches:

(i) **End-to-end retransmission:** A path is chosen from source to sink, and packets are acknowledged by the sink, or destination node. If the acknowledgment for a packet is not received by the source, the packet is retransmitted. This represents the situation

where reliability is provided by a retransmission scheme above the arc layer, e.g., by the transmission control protocol (TCP) at the transport layer, and no mechanism for reliability is present at the arc layer.

(ii) **End-to-end coding:** A path is chosen from source to sink, and an end-to-end forward error correction (FEC) code, such as a Reed-Solomon code, an LT code [92], or a Raptor code [102, 126], is used to correct for packets lost between source and sink. This is the Digital Fountain approach to reliability [18].

(iii) **Arc-by-arc retransmission:** A path is chosen from source to sink, and automatic repeat request (ARQ) is used at the arc layer to request the retransmission of packets lost on every arc in the path. Thus, on every arc, packets are acknowledged by the intended receiver and, if the acknowledgment for a packet is not received by the sender, the packet is retransmitted.

(iv) **Path coding:** A path is chosen from source to sink, and every node on the path employs coding to correct for lost packets. The most straightforward way of doing this is for each node to use an FEC code, decoding and re-encoding packets it receives. The drawback of such an approach is delay. Every node on the path codes and decodes packets in a block. A way of overcoming this drawback is to use codes that operate in more of a "convolutional" manner, sending out coded packets formed from packets received thus far, without decoding. The random linear network coding scheme of Section 4.1 is such a code. A variation, with lower complexity, is described in [111].

(v) **Full coding:** In this case, paths are eschewed altogether, and we use our solution to the efficient operation problem. Problem (5.2) is solved to find a subgraph, and the random linear coding scheme of Section 4.1 is used. This represents the limit of achievability provided that we are restricted from modifying the design of the physical layer and that we do not exploit the timing of packets to convey information.

We consider the following experiment: Nodes are placed randomly according to a uniform distribution over a square region whose size is set to achieve unit node density. In the network, transmissions are subject to distance attenuation and Rayleigh fading, but not interference (owing to scheduling). So, when node i transmits, the signal-to-noise ratio (SNR) of the signal received at node j is $\gamma d(i,j)^{-\alpha}$, where γ is an

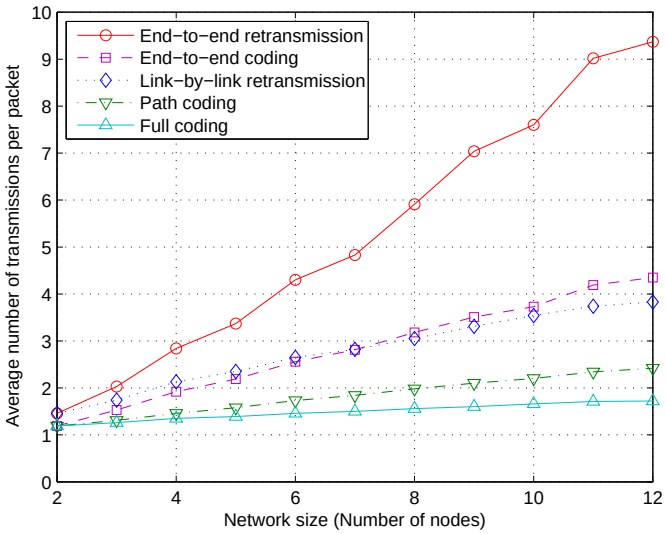


Fig. 5.7. Average number of transmissions per packet as a function of network size for various wireless unicast approaches. Reprinted with permission from [101].

exponentially-distributed random variable with unit mean, $d(i,j)$ is the distance between node i and node j, and α is an attenuation parameter that was taken to be 2. A packet transmitted by node i is successfully received by node j if the received SNR exceeds β, i.e., $\gamma d(i,j)^{-\alpha} \geq \beta$, where β is a threshold that was taken to be $1/4$. If a packet is not successfully received, then it is completely lost. If acknowledgments are sent, acknowledgments are subject to loss in the same way that packets are and follow the reverse path.

The average number of transmissions required per packet using the various approaches in random networks of varying size is shown in Figure 5.7. Paths or subgraphs were chosen in each random instance to minimize the total number of transmissions required, except in the cases of end-to-end retransmission and end-to-end coding, where they were chosen to minimize the number of transmissions required by the source node (the optimization to minimize the total number of transmissions in these cases cannot be done straightforwardly by a shortest path algorithm). We see that, while end-to-end coding and arc-by-arc retransmission al-

ready represent significant improvements on end-to-end retransmission, the coded approaches represent more significant improvements still. By a network size of nine nodes, full coding already improves on arc-by-arc retransmission by a factor of two. Moreover, as the network size grows, the performance of the various schemes diverges.

Here, we discuss performance simply in terms of the number of transmissions required per packet; in some cases, e.g., congestion, the performance measure increases super-linearly in this quantity, and the performance improvement is even greater than that depicted in Figure 5.7. We see, at any rate, that coding yields significant gains, particularly for large networks.

5.1.2 Computation-constrained coding

In the previous section, we assumed that all nodes in the network are capable of coding, and we focused on the problem of minimizing resources that can be expressed as a function of the coding subgraph. But what if the computation required for coding is itself a scarce resource? This is a concern, for example, in currently-deployed networks that only have routing capability—each node that needs to be upgraded to add coding capability will incur some cost. In the computation-constrained case, we wish to restrict coding to only a subset of the network nodes, trading off resources for transmission with resources for computation.

The computation-constrained problem is, in general, a hard one. Simply determining a minimal set of nodes where coding is required in a given subgraph is NP-hard [81], suggesting that heurstics are necessary for multicast connections involving more than a few sinks. When there are only a small number of sinks, however, an optimal solution can be found using a flow-based approach. This approach, due to Bhattad et al. [17], partitions flows not only into sinks, but into *sets* of sinks. Thus, for a multicast with sinks in the set T, we not only have a flow $x^{(t)}$ for each $t \in T$, but we have a flow $x^{(T')}$ for each $T' \subset T$. The flow formulation by Bhattad et al. involves a number of variables and constraints that grows exponentially with the number of sinks, so it is feasible only when $|T|$ is small, but, under this restriction, it allows optimal solutions to be found.

When dealing with a larger number of sinks, suboptimal heuristics provide a solution. Kim et al. [81] have proposed an evolutionary approach based on a genetic algorithm that shows good empircal performance.

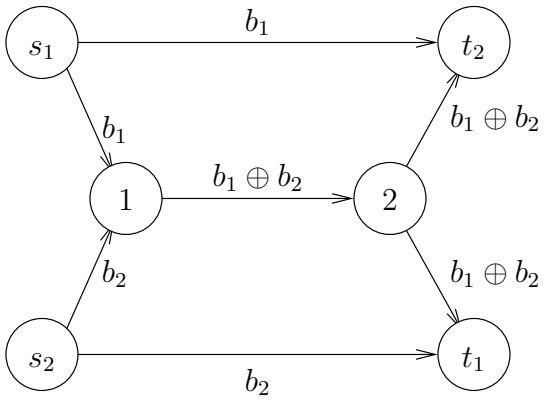


Fig. 5.8. The modified butterfly network. In this network, every arc represents a directed arc that is capable of carrying a single packet reliably.

5.1.3 Inter-session coding

Optimal inter-session coding, as we have mentioned, is very difficult. In fact, even good inter-session coding is very difficult. One of the few methods for finding non-trivial inter-session coding solutions was proposed by Koetter and Médard [85]; their method searched only within a particular, limited class of linear codes and, even then, its complexity scaled exponentially in the size of the network. Since we must maintain an eye on practicability, we must not be too ambitious in our search for inter-session codes.

As discussed in Section 3.5.1, a modest approach that we can take is the following: Since our most familiar examples of inter-session coding are the modified butterfly network (Figure 1.2) and modified wireless butterfly network (Figure 1.4), we seek direct extensions of the inter-session coding opportunities exemplified by these two cases.

For starters, let us consider the lossless wireline case. We show the modified butterfly network again in Figure 5.8. Without intersession coding, we would require two unit-sized flows $x^{(1)}$ and $x^{(2)}$, originating and ending at s_1 and t_1 and s_2 and t_2, respectively. There is only one possible solution for each of the two flows:

$$x^{(1)} = (x^{(1)}_{s_1 1}, x^{(1)}_{s_2 1}, x^{(1)}_{s_1 t_2}, x^{(1)}_{12}, x^{(1)}_{s_2 t_1}, x^{(1)}_{2t_2}, x^{(1)}_{2t_1}) = (1,0,0,1,0,0,1),$$
$$x^{(2)} = (x^{(2)}_{s_1 1}, x^{(2)}_{s_2 1}, x^{(2)}_{s_1 t_2}, x^{(2)}_{12}, x^{(2)}_{s_2 t_1}, x^{(1)}_{2t_2}, x^{(2)}_{2t_1}) = (0,1,0,1,0,1,0).$$

This solution, as we know, is not feasible because it violates the capacity constraint on arc $(1,2)$. Without inter-session coding, the total rate of

packet injections on arc $(1, 2)$ would be two, which violates its capacity of one. We also know, however, that a simple inter-session code where packets from each of the two sessions are XORed at node 1 resolves this situation by reducing the rate of packet injections on arc $(1, 2)$ from two to one and increasing the rate of packet injections on arc (s_1, t_2) and (s_2, t_1) from zero to one. If we can formulate the effect of this code as flow equations, then we can hope to develop a flow-based approach for systematically finding inter-session coding opportunities of the type exemplified by the modified butterfly network.

Such a flow formulation was developed by Traskov et al. [132]. In the formulation, three variables are introduced for each coding opportunity: p, the poison variable, q, the antidote request variable, and r, the antidote variable. The poison p represents the effect of coding two sessions together with an XOR. The poison on arc (i, j), p_{ij}, is strictly negative if the arc carries poisoned flow, i.e., it carries packets XORed from two separate sessions; otherwise, it is zero. Such a flow is "poisoned" because the XORed packets, by themselves, are not useful to their intended destinations. The antidote r represents the extra "remedy" packets that must be set so that the effect of the poison can be reversed, i.e., so that the XORed packets can be decoded to recover the packets that are actually useful. The antidote on arc (i, j), r_{ij}, is strictly positive if the arc carries remedy packets; otherwise, it is zero. The antidote request q is essentially an imaginary variable in that it need not correspond to any real physical entity, It could, however, correspond to actual protocol messages requesting remedy packets to be sent. The antidote request connects the coding node to the nodes from which remedy packets are sent, thus making a cycle from p, q, and r and facilitating a flow formulation. The antidote request on arc (i, j), q_{ij}, is strictly negative if the arc carries antidote requests; otherwise, it is zero.

The Traskov et al. flow formulation is best understood using an example. We take, as our example, the modified butterfly network and, in Figure 5.9, we show the poison, antidote request, and antidote variables for this network. We have two of each variable: one, $1 \to 2$, relates to the impact of coding on flow two, while the other, $2 \to 1$, relates to the impact of coding on flow one. Note that $p(1 \to 2)$, $q(1 \to 2)$, and $r(1 \to 2)$ form a cycle, as do $p(2 \to 1)$, $q(2 \to 1)$, and $r(2 \to 1)$.

This formulation, once extended to a general lossless wireline network that allows coding at all nodes, yields the following formulation of the

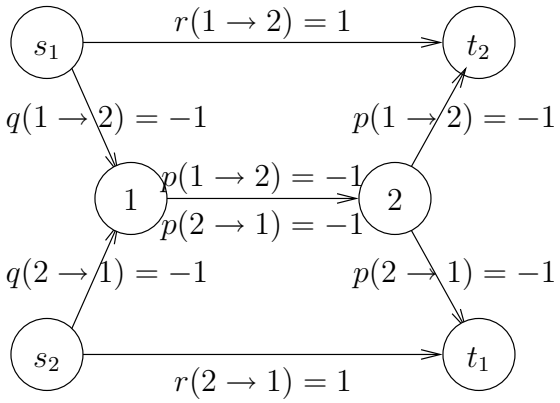


Fig. 5.9. The modified butterfly network with poison, antidote request, and antidote variables shown. Reprinted with permission from [132].

subgraph selection problem:

$$
\begin{aligned}
&\text{minimize } f(z) \\
&\text{subject to } z \in Z, \\
&\sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij}^{(c)} - \sum_{\{j|(j,i)\in\mathcal{A}\}} x_{ji}^{(c)} = \begin{cases} R_c & \text{if } i = s_c, \\ -R_c & \text{if } j = t_c, \\ 0 & \text{otherwise,} \end{cases} \\
&\qquad\qquad \forall\ i \in \mathcal{N},\ c = 1, \ldots, C, \\
&x \geq 0 \\
&x \in \mathcal{T}(z),
\end{aligned}
\tag{5.38}
$$

where $\mathcal{T}(z)$ for a given z is the set of x satisfying, for some $\{p_{ij}(c \to d, k)\}$, $\{q_{ij}(c \to d, k)\}$, and $\{r_{ij}(c \to d, k)\}$, the following equalities and

inequalities:

$$\sum_{\{j|(j,i)\in\mathcal{A}\}} (p_{ji}(c\to d,k)+q_{ji}(c\to d,k)+r_{ji}(c\to d,k))$$
$$= \sum_{\{j|(i,j)\in\mathcal{A}\}} (p_{ij}(c\to d,k)+q_{ij}(c\to d,k)+r_{ij}(c\to d,k)),$$
$$\forall\ i,k\in\mathcal{N},\ c,d=1,\ldots,C,$$

$$\sum_{\{j|(j,i)\in\mathcal{A}\}} p_{ji}(c\to d,k) - \sum_{\{j|(i,j)\in\mathcal{A}\}} p_{ij}(c\to d,k) \begin{cases}\geq 0 & \text{if } i=k,\\ \leq 0 & \text{otherwise,}\end{cases}$$
$$\forall\ i,k\in\mathcal{N},\ c,d=1,\ldots,C,$$

$$\sum_{\{j|(j,i)\in\mathcal{A}\}} q_{ji}(c\to d,k) - \sum_{\{j|(i,j)\in\mathcal{A}\}} q_{ij}(c\to d,k) \begin{cases}\leq 0 & \text{if } i=k,\\ \geq 0 & \text{otherwise,}\end{cases}$$
$$\forall\ i,k\in\mathcal{N},\ c,d=1,\ldots,C,$$

$$p_{ij}(d\to c,i)=p_{ij}(c\to d,i),$$
$$\forall\ j\in\{j|(i,j)\in\mathcal{A}\},\ c,d=1,\ldots,C,$$

$$\sum_{c=1}^{C}\left\{x_{ij}(c)+\sum_k\sum_{d>c}p_{ij}^{\max}(c,d,k)+\sum_k\sum_{d\neq c}r_{ij}(c\to d,k)\right\}\leq z_{ij},$$
$$\forall\ (i,j)\in\mathcal{A},$$

$$x_{ij}(d)+\sum_k\sum_c(p_{ij}(c\to d,k)+q_{ij}(d\to c,k))\geq 0,$$
$$\forall\ (i,j)\in\mathcal{A},\ d=1,\ldots,C,$$

$$p\leq 0,\quad r\geq 0,\quad s\leq 0,$$

where

$$p_{ij}^{\max}(c,d,k)\triangleq\max(p_{ij}(c\to d,k),p_{ij}(d\to c,k)).$$

In this formulation, multicast is not explicitly considered—only inter-session coding for multiple unicast sessions is considered. It allows for packets from two separate sessions to be XORed at any node and for remedy packets to be sent and decoding to be performed at any nodes at which these operations are valid. It does not allow, however, for poisoned flows to be poisoned again. That this formulation is indeed correct, given these restrictions, is shown in [132].

This formulation can be straightforwardly extended to the case of lossless wireless networks, thus allowing us to extend the modified wire-

less butterfly network in the same way. The precise equations that are obtained are given in [42].

5.2 Queue-length-based approaches

Queue-length-based, or back-pressure, algorithms were first introduced in [131, 6] for multicommodity flow problems, i.e. multiple unicast network problems without coding. The basic idea can be summed up as follows. Each node i keeps track of the number $U_i^{(c)}$ of packets of each unicast session c. It is convenient to think of $U_i^{(c)}$ as the length of a queue of packets $Q_i^{(c)}$ maintained by node i for each session c. Each queue has a potential that is an increasing function of its length. At each step, packet transmissions are prioritized according to the potential difference across their corresponding start and end queues, so as to maximize the total potential decrease in the network, subject to network constraints. In the simplest version, the potential is equal to the queue length, and transmissions are prioritized according to the queue length difference, i.e., for a given arc (i, j), packets of session $\arg\max_c(U_i^{(c)} - U_j^{(c)})$ have priority for transmission. This policy gives rise to queue length gradients from each session's source to its sink. We can draw an analogy with pressure gradients and think of packets as "flowing down" these gradients.

Different versions and extensions of the basic back-pressure algorithm have been proposed for finding asymptotically optimal solutions for various types of multicommodity flow problems with different objectives. For instance, back-pressure algorithms with potential given by queue lengths are proposed for dynamic control of routing and scheduling in time-varying networks, in [131] and other subsequent works. Such approaches are extended to the problem of minimum-energy routing in [107]. In [7] a back-pressure algorithm with an exponential potential function is proposed as a low-complexity approximation algorithm for constructing a solution to a feasible multicommodity flow problem.

The back-pressure approach can be generalized to optimize over different classes of network codes. The underlying idea is the introduction of an appropriately defined system of *virtual queues* and/or *virtual transmissions*. The back-pressure principle of maximizing total potential decrease at each step, subject to network constraints, can then be applied to obtain control policies for network coding, routing and scheduling based on the virtual queue lengths. Extensions to various types of net-

work problems with different objectives can be obtained analogously to the multicommodity routing case.

5.2.1 Intra-session network coding for multiple multicast sessions

In this section, we consider a dynamically varying network problem with a set $\mathcal{C}$ of multicast sessions. Each session $c \in \mathcal{C}$ has a set $\mathcal{S}_c$ of source nodes whose data is demanded by a set $\mathcal{T}_c$ of sink nodes. We describe an extension of the basic queue-length-based approach to the case of multiple multicast sessions with intra-session network coding, i.e. only packets from the same session are coded together.

5.2.1.1 Network model

We consider a lossless network comprised of a set $\mathcal{N}$ of $N = |\mathcal{N}|$ nodes, with a set $\mathcal{A}$ of communication arcs between them that are fixed or time-varying according to some specified processes. There is a set of multicast sessions $\mathcal{C}$ sharing the network. Each session $c \in \mathcal{C}$ is associated with a set $\mathcal{S}_c \subset \mathcal{N}$ of source nodes, and an arrival process, at each source node, of exogenous session c packets to be transmitted to each of a set $\mathcal{T}_c \subset \mathcal{N} \backslash \mathcal{S}_c$ of sink nodes. We denote by τ_{max} the maximum number of sinks of a session.

Time is assumed to be slotted, with the time unit normalized so that time slots correspond to integral times $\tau = 0, 1, 2, \ldots$. For simplicity, we assume fixed length packets and arc transmission rates that are restricted to integer numbers of packets per slot. We assume that the channel states, represented by a vector $\underline{S}(\tau)$ taking values in a finite set, are fixed over the duration of each slot τ, and known at the beginning of the slot. For simplicity of exposition we assume that the exogenous packet arrival and channel processes are independent and identically distributed (i.i.d.) across slots. The queue-length-based policy described below applies also in the case of stationary ergodic processes. The analysis below can be generalized to the latter case using an approach similar to that in [106]†.

We consider both wired and wireless networks. In our model, for the wired case the network connectivity and the arc rates are explicitly specified. For wireless networks, the network connectivity and arc transmission rates depend on the signal and interference powers and the

† Groups of M timeslots are considered as a "super timeslot", where M is sufficiently large that the time averages of the channel and arrival processes differ from their steady-state values by no more than a given small amount.

channel states. We assume that the vector of transmit powers $\underline{P}(\tau)$ at each time τ takes values in a given set Π and is constant over each slot. We also assume we are given a rate function $\underline{\mu}(\underline{P}, \underline{S})$ specifying the vector of instantaneous arc rates $\underline{\mu}(\tau) = (\mu_{iJ}(\tau))$ as a function of the vector of transmit powers $\underline{P}(\tau)$ and channel states $\underline{S}(\tau)$.

In this section, all arc, source and flow rates are in packets per unit time. We assume upper bounds μ_{max}^{out} and μ_{max}^{in} on the total flow rate into and out of a node respectively.

5.2.1.2 Network coding and virtual queues

We use the approach of distributed random linear network coding with coefficient vectors, described in Section 2.5.1.1. For simplicity, we do not explicitly consider batch restrictions in the descriptions and analysis of the policies in this section. Thus, the results represent an upper bound on performance that is approached only asymptotically (in the batch size and packet length). If the policies are operated across multiple batches (the only change from the policy described below being an additional restriction not to code across batches), there is some capacity loss which decreases with increasing batch size and depends on the detailed source and channel statistics.

Recall from Section 2.3 that for network coding within a multicast session, a solution is given by a union of individual flow solutions for each sink. Here, we define *virtual queues* to keep track of the individual sinks' flow solutions as follows.

Each node i conceptually maintains, for each sink t of each session c, a virtual queue $Q_i^{(t,c)}$ whose length $U_i^{(t,c)}$ is the number of session c packets queued at node i that are intended for sink t. A single physical session c packet corresponds to a packet in the virtual queue $Q_i^{(t,c)}$ of each sink t for which it is intended. For instance, in the butterfly network of Figure 5.10, each physical session c packet at source node s is intended to be multicast to the two sink nodes t_1, t_2, and so corresponds to one packet in each virtual queue $Q_s^{(t_1,c)}$ and $Q_s^{(t_2,c)}$. Each packet in a virtual queue corresponds to a distinct physical packet; thus there is a one-to-many correspondence between physical packets and packets in virtual queues.

A packet in a virtual queue $Q_i^{(t,c)}$ can be transferred over an arc (i, j) to the corresponding virtual queue $Q_j^{(t,c)}$ at the arc's end node j; this is called a virtual transmission. With network coding, for any subset $\mathcal{T}' \subset \mathcal{T}_c$ of a session's sinks, a single physical transmission of a packet on an arc (i, j) can simultaneously accomplish, for each sink $t \in \mathcal{T}'$, one virtual

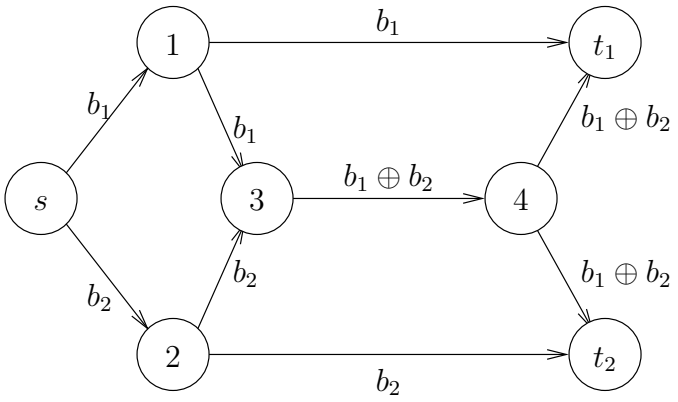


Fig. 5.10. The butterfly network with a single multicast session c, one source node s and two sink nodes t_1, t_2.

transmission from $Q_i^{(t,c)}$ to $Q_j^{(t,c)}$. The physically transmitted packet is a random linear coded combination of the physical packets corresponding to the virtually transmitted packets. In the case of a wireless broadcast transmission from a node i to a set of nodes J, although the nodes in J all receive the transmitted packet, they update their virtual queues selectively, according to control information included in the packet, such that each constituent virtual transmission is point-to-point, i.e. from one queue $Q_i^{(t,c)}$ to one queue $Q_j^{(t,c)}$ at some end node $j \in J$, which may differ for different sinks t. Thus, there is conservation of virtual packets (virtual flows); we can draw an analogy with the no-coding case where physical packets (physical flows) are conserved. An illustration is given in Figure 5.11 of a physical broadcast transmission which accomplishes two virtual transmissions, for a multicast session with two sinks.

Let $w_i^{(c)}$ be the average arrival rate of exogenous session c packets at each node i; $w_i^{(c)} = 0$ for $i \notin \mathcal{S}_c$. Each source node $i \in \mathcal{S}_c$ forms coded source packets at an average rate $r_i^{(c)} = w_i^{(c)} + \epsilon$ for some $\epsilon > 0$, slightly higher than its exogenous packet arrival rate $w_i^{(c)}$. Each coded source packet is formed as an independent random linear combination of previously arrived exogenous packets, and is "added" to each queue $Q_i^{(t,c)}, t \in \mathcal{T}_c$. In order for each sink to be able to decode all source packets, $\frac{w_i^{(c)}}{r_i^{(c)}}$ should be at most the ratio of the total number of packets reaching each sink to the total number of coded source packets. As we

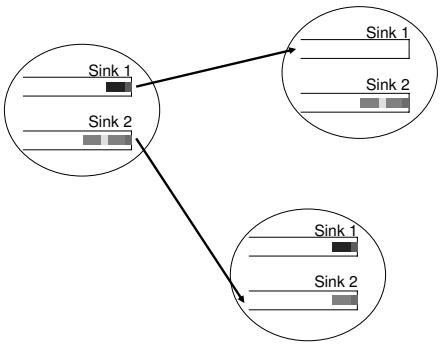


Fig. 5.11. Illustration of a physical broadcast transmission comprising two virtual transmissions. Each oval corresponds to a node. The left node broadcasts a physical packet received by the two right nodes, one of which adds the packet to the virtual queue for sink 1, and the other, to the virtual queue for sink 2.

will see in the next section, for sufficiently large times t, this condition is satisfied and decoding is successful with high probability.†

Let $A_i^{(c)}(\tau)$ be the number of coded session c source packets formed at node i in timeslot τ. Thus we have

$$r_i^{(c)} = E\{A_i^{(c)}(\tau)\}. \tag{5.39}$$

We assume that the second moment of the total number of source packets formed at each node in each timeslot is bounded by a finite maximum value $A_{\max}^2$, i.e.

$$E\left\{\left(\sum_c A_i^{(c)}(\tau)\right)^2\right\} \leq A_{\max}^2, \tag{5.40}$$

which implies

$$E\left\{\sum_c A_i^{(c)}(\tau)\right\} \leq A_{\max}. \tag{5.41}$$

† If we employ batch coding where each batch contains a fixed number of exogenous packets, feedback from the sinks can be used to signal when the sources should stop forming coded packets of each batch. This determines the effective value of ϵ for each batch.

5.2.1.3 Problem and notation

We consider a multiple multicast network problem where the average network capacity is slightly higher than the minimum needed to support the source rates. We would like a control policy for dynamic subgraph selection (i.e. scheduling of packets/transmitters, and routing) which, coupled with random linear network coding, stably supports the given source rates. The problem is formulated more precisely as follows.

Let $U_i^c(\tau)$ be the number of physical session c packets queued at node i at time τ. Stability is defined in terms of an "overflow" function

$$\gamma_i^{(c)}(M) \quad = \quad \limsup_{\tau\to\infty} \frac{1}{\tau} \sum_{\tau'=0}^{\tau} \Pr\{U_i^c(\tau') > M\} \tag{5.42}$$

The session c queue at node i is considered stable if $\gamma_i^c(M) \to 0$ as $M \to \infty$. A network of queues is considered stable iff each individual queue is stable. Since

$$U_i^c(\tau) \le \sum_t U_i^{(t,c)}(\tau) \le \tau_{max} U_i^c(\tau) \quad \forall \quad i, c, \tau$$

the network is stable iff all virtual queues are stable.

Let z_{iJ} denote the average value of the time-varying rate $\mu_{iJ}(\tau)$ of hyperarc (i, J). We use $x_{iJj}^{(t,c)}$ to denote average virtual flow rate, over arc $(i, J) \in \mathcal{A}$, from $Q_i^{(t,c)}$ to $Q_j^{(t,c)}$, $j \in J$. We use $y_{iJ}^{(c)}$ to denote average session c physical flow rate on $(i, J) \in \mathcal{A}$.

For brevity of notation, we use the convention that any term with subscript iJj equals zero unless $(i, J) \in \mathcal{A}, j \in J$, and any term with superscript (t, c) equals zero unless $c \in \mathcal{C}, t \in \mathcal{T}_c$.

Let $\pi_{\underline{S}}$ denote the probability in each slot that the channel states take the value $\underline{S}$. Let Z be the set consisting of all rate vectors $\underline{z} = (z_{iJ})$ that can represented as $\underline{z} = \sum_{\underline{S}} \pi_{\underline{S}} \underline{z}_{\underline{S}}$ for some set of rate vectors $\underline{z}_{\underline{S}}$, each of which is in the convex hull of the set of rate vectors $\{\underline{\mu}_{iJ}(\underline{P}, \underline{S}) | \underline{P} \in \Pi\}$. Z represents the set of all long-term average transmission rates (z_{iJ}) supportable by the network [106, 59].

Let Λ be the set of all rate vectors $(r_i^{(c)})$ such that there exist values

for $(z_{iJ}) \in Z$ and $\{x_{iJj}^{(t,c)}, y_{iJ}^{(t,c)}\}$ satisfying:

$$x_{tJi}^{(t,c)} = 0 \quad \forall\ c,t,J,i \tag{5.43}$$

$$x_{iJj}^{(t,c)} \geq 0 \quad \forall\ i,j,c,t,J \tag{5.44}$$

$$r_i^{(c)} \leq \sum_{J,j} x_{iJj}^{(t,c)} - \sum_{j,I} x_{jIi}^{(t,c)} \quad \forall\ i,c,t \in \mathcal{T}_c, t \neq i \tag{5.45}$$

$$\sum_{j \in J} x_{iJj}^{(t,c)} \leq y_{iJ}^{(c)} \quad \forall\ i,J,c,t \tag{5.46}$$

$$\sum_{c} y_{iJ}^{(c)} \leq z_{iJ} \quad \forall\ i,J \tag{5.47}$$

Equations (5.43)–(5.47) correspond to the feasible set of problem (5.1) in the lossless case, where, rather than a single source node s_c for each session c, there may be multiple source nodes, described by the set $\mathcal{S}_c$. The variables $\{x_{ab}^{(t,c)}\}$ for a (session, sink) pair $(c, t \in \mathcal{T}_c)$ define a flow carrying rate at least $r_i^{(c)}$ from each source node i to t (Inequalities (5.44)–(5.45)), in which virtual flow that is intended for t is not retransmitted away from t (Equation (5.46)).

We describe below a queue-length-based policy that is stable and asymptotically achieves the given source rates for any network problem where $(r_i^{(c)} + \epsilon') \in \Lambda$.† This condition implies the existence of a solution, but we assume that the queue-length-based policy operates without knowledge of the solution variables.

5.2.1.4 Control policies

We consider policies that make control decisions at the start of each time slot τ and operate as follows.

- Power allocation: A vector of transmit powers $\underline{P}(\tau) = (P_{iJ}(\tau))$ is chosen from the set Π of feasible power allocations. This, together with the channel state $\underline{S}(\tau)$, determines the arc rates $\underline{\mu}(\tau) = (\mu_{iJ}(\tau))$, assumed constant over the time slot.
- Session scheduling, rate allocation and network coding: For each arc (i,J), each sink t of each session c is allocated a transmission rate

† This particular problem formulation, which provisions sufficient network capacity to support, for each session c, an additional source rate of ϵ' at each node, affords a simple solution and analysis. We could alternatively use a slightly more complicated formulation which includes for each session c an additional source rate of ϵ' only at the actual source nodes $s \in \mathcal{S}_c$, similarly to that in [67] for the case of correlated sources.

$\mu_{iJj}^{(t,c)}(\tau)$ for each destination node $j \in J$. These allocated rates must satisfy the overall arc rate constraint

$$\mu_{iJ}(\tau) \geq \sum_{c \in \mathcal{C}} \max_{t \in \mathcal{T}_c} \sum_{j \in J} \mu_{iJj}^{(t,c)}(\tau).$$

$\mu_{iJj}^{(t,c)}(\tau)$ gives the maximum rate of virtual transmissions from $Q_i^{(t,c)}$ to $Q_j^{(t,c)}$. Besides this limit on virtual transmissions for pairs of queues over each link, the total number of virtual transmissions out of $Q_i^{(t,c)}$ over all links with start node i is also limited by the queue length $U_i^{(t,c)}(\tau)$ at the start of the time slot. Each session c packet physically transmitted on arc (i, J) is a random linear combination, in $\mathbb{F}_q$, of packets corresponding to a set of virtual transmissions on (i, J), each associated with a different sink in $\mathcal{T}_c$. Thus, the rate allocated to session c on (i, J) is the maximum, over sinks $t \in \mathcal{T}_c$, of each sink t's total allocated rate $\sum_{j \in J} \mu_{iJj}^{(t,c)}(\tau)$.

The following dynamic policy relies on queue length information to make control decisions, without requiring knowledge of the input or channel statistics. The intuition behind the policy is that it seeks to maximize the total weight of virtual transmissions for each time slot, subject to the above constraints.

Back-pressure policy

For each time slot τ, the transmit powers $(P_{iJ}(\tau))$ and allocated rates $(\mu_{iJj}^{(t,c)}(\tau))$ are chosen based on the queue lengths $(U_i^{(t,c)}(\tau))$ at the start of the slot, as follows.

- Session scheduling: For each arc (i, J),
  - for each session c and sink $t \in \mathcal{T}_c$, one end node

$$\begin{aligned} j_{iJ}^{(t,c)*} &= \arg\max_{j \in J} \left(U_i^{(t,c)} - U_j^{(t,c)} \right) \\ &= \arg\min_{j \in J} U_j^{(t,c)} \end{aligned}$$

    is chosen. Let $U_{iJ}^{(t,c)*}$ denote $U_{j_{iJ}^{(t,c)*}}^{(t,c)}$ for brevity.
  - one session

$$c_{iJ}^* = \arg\max_c \left\{ \sum_{t \in \mathcal{T}_c} \max \left(U_i^{(t,c)} - U_{iJ}^{(t,c)*},\ 0 \right) \right\}$$

is chosen. Let

$$w_{iJ}^* = \sum_{t \in \mathcal{T}_{c_{iJ}^*}} \max\left(U_i^{(t,c)} - U_{iJ}^{(t,c)*},\ 0\right) \tag{5.48}$$

be the weight of the chosen session.

- Power control: The state $\underline{S}(\tau)$ is observed, and a power allocation

$$\underline{P}(\tau) = \arg\max_{\underline{P} \in \Pi} \sum_{i,J} \mu_{iJ}(\underline{P}, \underline{S}(\tau)) w_{iJ}^* \tag{5.49}$$

is chosen.

- Rate allocation: For each arc (i, J),

$$\mu_{iJj}^{(t,c)}(\tau) = \begin{cases} \mu_{iJ}(\tau) & \text{if } c = c_{iJ}^*, t \in \mathcal{T}_c, j = j_{iJ}^{(t,c)*} \text{ and } U_i^{(t,c)} - U_j^{(t,c)} > 0 \\ 0 & \text{otherwise} \end{cases} . \tag{5.50}$$

In a network where simultaneous transmissions interfere, optimizing (5.49) requires a centralized solution. If there are enough channels for independent transmissions, the optimization can be done independently for each transmitter.

The stability of the back-pressure policy is shown by comparison with a randomized policy that assumes knowledge of a solution based on the long-term input and channel statistics. We will show that the randomized policy is stable, and that stability of the randomized policy implies stability of the back-pressure policy.

Randomized policy

Assume given values of $(z_{iJ}) \in Z$ and $\{x_{iJj}^{(t,c)}, y_{iJ}^{(t,c)}\}$ satisfying

$$x_{tJi}^{(t,c)} = 0 \quad \forall\ c, t, J, i \tag{5.51}$$

$$x_{iJj}^{(t,c)} \geq 0 \quad \forall\ i, j, c, t, J \tag{5.52}$$

$$r_i^{(c)} + \epsilon' \leq \sum_{J,j} x_{iJj}^{(t,c)} - \sum_{j,I} x_{jIi}^{(t,c)} \quad \forall\ i, c, t \in \mathcal{T}_c, t \neq i \tag{5.53}$$

$$\sum_{j \in J} x_{iJj}^{(t,c)} \leq y_{iJ}^{(c)} \quad \forall\ i, J, c, t \tag{5.54}$$

$$\sum_{c} y_{iJ}^{(c)} \leq z_{iJ} \quad \forall\ i, J \tag{5.55}$$

are given.

The following lemma shows that for any rate vector $(z_{iJ}) \in Z$, power can be allocated according to the time-varying channel state $\underline{S}(\tau)$ such that the time average link rates converge to (z_{iJ}).

Lemma 5.1 *Consider a rate vector $(z_{iJ}) \in Z$. There exists a stationary randomized power allocation policy which gives link rates $\mu_{iJ}(\tau)$ satisfying*

$$\lim_{\tau\to\infty} \frac{1}{\tau} \sum_{0}^{\tau} \mu_{iJ}(\tau') = z_{iJ}$$

with probability 1 for all $(i, J) \in \mathcal{A}$, where, for each time slot τ in which channel state $\underline{S}(\tau)$ takes value $\underline{S}$, the power allocation is chosen randomly from a finite set $\{\underline{P}_{\underline{S},1}, \ldots, \underline{P}_{\underline{S},m}\}$ according to stationary probabilities $\{q_{\underline{S},1}, \ldots, q_{\underline{S},m}\}$.

Proof (Outline) From the definition of Z in Section 5.2.1.3 and by Carathéodory's Theorem (see, e.g. [8]), $(z_{iJ}) = \sum_{\underline{S}} \pi_{\underline{S}} \underline{z}_{\underline{S}}$ for some set of rate vectors $\underline{z}_{\underline{S}}$, each of which is a convex combination of vectors in $\{\underline{\mu}_{iJ}(\underline{P}, \underline{S}) | \underline{P} \in \Pi\}$. The probabilities of the stationary randomized power allocation policy are chosen according to the weights of the convex combinations for each state $\underline{S}$. □

The randomized policy is designed such that

$$E\{\mu_{iJj}^{(t,c)}(\tau)\} = x_{iJj}^{(t,c)}. \tag{5.56}$$

For each time slot τ, transmit powers $(P_{iJ}(\tau))$ and allocated rates $(\mu_{iJj}^{(t,c)}(\tau))$ are chosen based on the given values of (z_{iJ}) and $\{x_{iJj}^{(t,c)}, y_{iJ}^{(t,c)}\}$ as well as the channel state $\underline{S}(\tau)$, as follows.

- Power allocation: The channel state $\underline{S}(\tau)$ is observed, and power is allocated according to the algorithm of Lemma 5.1, giving instantaneous arc rates $\mu_{iJ}(\tau)$ and long-term average rates z_{iJ}.
- Session scheduling and rate allocation: For each arc (i, J), one session $c = c_{iJ}$ is chosen randomly with probability $\frac{y_{iJ}^{(c)}}{\sum_c y_{iJ}^{(c)}}$. Each of its sinks $t \in \mathcal{T}_c$ is chosen independently with probability $\frac{\sum_j x_{iJj}^{(t,c)}}{y_{iJ}^{(c)}}$. Let $\mathcal{T}_{iJ} \subset \mathcal{T}_c$ denote the set of chosen sinks. For each $t \in \mathcal{T}_{iJ}$, one destination node $j = j_{iJ}^{(t,c)}$ in J is chosen with probability $\frac{x_{iJj}^{(t,c)}}{\sum_j x_{iJj}^{(t,c)}}$. The corresponding allocated rates are

$$\mu_{iJj}^{(t,c)}(\tau) = \begin{cases} \frac{\sum_c y_{iJ}^{(c)}}{z_{iJ}} \mu_{iJ}(\tau) & \text{if } c = c_{iJ}, t \in \mathcal{T}_{iJ} \text{ and } j = j_{iJ}^{(t,c)} \\ 0 & \text{otherwise} \end{cases}. \tag{5.57}$$

Theorem 5.1 *If input rates* $(r_i^{(c)})$ *are such that* $(r_i^{(c)} + \epsilon') \in \Lambda$, $\epsilon' > 0$, *both the randomized policy and the back pressure policy stabilize the system with average total virtual queue length bounded as*

$$\sum_{i,c,t} \overline{U_i^{(t,c)}} = \limsup_{\tau\to\infty} \frac{1}{\tau} \sum_{\tau'=0}^{\tau-1} \sum_{i,c,t} E\{U_i^{(t,c)}(\tau')\} \leq \frac{BN}{\epsilon'} \tag{5.58}$$

where

$$B = \frac{\tau_{max}}{2} \left((A_{\max} + \mu_{max}^{in})^2 + (\mu_{max}^{out})^2 \right). \tag{5.59}$$

The proof of this theorem uses the following result:

Theorem 5.2 *Let* $\underline{U}(\tau) = (U_1(\tau), \ldots, U_n(\tau))$ *be a vector of queue lengths. Define the Lyapunov function* $L(\underline{U}(\tau)) = \sum_{j=1}^{n} [U_j(\tau)]^2$. *If for all* τ

$$E\{L(\underline{U}(\tau+1)) - L(\underline{U}(\tau)) | \underline{U}(\tau)\} \leq C_1 - C_2 \sum_{j=1}^{n} U_j(\tau) \tag{5.60}$$

for some positive constants C_1, C_2, *and if* $E\{L(\underline{U}(0))\} < \infty$, *then*

$$\sum_{j=1}^{n} \overline{U_j} = \limsup_{\tau\to\infty} \frac{1}{\tau} \sum_{\tau'=0}^{\tau-1} \sum_{j=1}^{n} E\{U_j(\tau')\} \leq \frac{C_1}{C_2} \tag{5.61}$$

and each queue is stable.

Proof Summing over $\tau = 0, 1, \ldots, T-1$ the expectation of (5.60) over the distribution of $\underline{U}(\tau)$, we have

$$E\{L(\underline{U}(T)) - L(\underline{U}(0))\} \leq TC_1 - C_2 \sum_{\tau=0}^{T-1} \sum_{j=1}^{n} E\{U_j(\tau)\}.$$

Since $L(\underline{U}(T)) > 0$,

$$\frac{1}{T} \sum_{\tau=0}^{T-1} \sum_{j=1}^{n} E\{U_j(\tau)\} \leq \frac{C_1}{C_2} + \frac{1}{TC_2} E\{L(\underline{U}(0))\}.$$

Taking the lim sup as $T \to \infty$ gives (5.61). Each queue is stable since

$$\begin{aligned}\gamma_j(M) &= \limsup_{\tau\to\infty} \frac{1}{\tau}\sum_{\tau'=0}^{\tau} \Pr\{U_j(\tau') > M\} \\ &\leq \limsup_{\tau\to\infty} \frac{1}{\tau}\sum_{\tau'=0}^{\tau} E\{U_j(\tau')\}/M \qquad (5.62)\\ &\leq \frac{C_1}{C_2 M} \ \to\ 0 \quad \text{as } M \to \infty, \qquad (5.63)\end{aligned}$$

where (5.62) holds since $U_j(\tau')$ is nonnegative. □

Proof of Theorem 5.1: The queue lengths evolve according to:

$$\begin{aligned}U_i^{(t,c)}(\tau+1) &\leq \max\left\{U_i^{(t,c)}(\tau) - \sum_{J,j}\mu_{iJj}^{(t,c)}(\tau)\ ,\ 0\right\} \\ &\quad + \sum_{j,I}\mu_{jIi}^{(t,c)}(\tau) + A_i^{(c)}(\tau) \qquad (5.64)\end{aligned}$$

which reflects the policy that the total number of virtual transmissions out of $Q_i^{(t,c)}$ is limited by the queue length $U_i^{(t,c)}(\tau)$.

Define the Lyapunov function $L(\underline{U}) = \sum_{i,c,t}(U_i^{(t,c)})^2$. Squaring (5.64) and dropping some negative terms from the right hand side, we obtain

$$\begin{aligned}[U_i^{(t,c)}(\tau+1)]^2 \leq [U_i^{(t,c)}(\tau)]^2 + \left[\left(A_i^{(c)} + \sum_{j,I}\mu_{jIi}^{(t,c)}\right)^2 + \left(\sum_{J,j}\mu_{iJj}^{(t,c)}\right)^2\right] \\ -2U_i^{(t,c)}(\tau)\left[\sum_{J,j}\mu_{iJj}^{(t,c)} - \sum_{j,I}\mu_{jIi}^{(t,c)} - A_i^{(c)}\right] \qquad (5.65)\end{aligned}$$

where the time dependencies of $\mu_{iJj}^{(t,c)}$ and $A_i^{(c)}$ are not shown for brevity, since these remain constant over the considered time slot.

Taking expectations of the sum of (5.65) over all i, c, t, noting that

$$\begin{aligned}\sum_{i,c,t}\left(\sum_{j,Z}\mu_{iJj}^{(t,c)}\right)^2 &\leq \sum_{i,c}\tau_{max}\left(\max_{t\in\mathcal{T}_c}\sum_{j,Z}\mu_{iJj}^{(t,c)}\right)^2 \\ &\leq \sum_{i}\tau_{max}\left(\sum_{c}\left[\max_{t\in\mathcal{T}_c}\sum_{j,Z}\mu_{iJj}^{(t,c)}\right]\right)^2 \qquad (5.66)\\ &\leq N\tau_{max}\left(\mu_{max}^{out}\right)^2,\end{aligned}$$

and

$$\sum_{i,c,t}\left(A_i^{(c)}+\sum_{j,I}\mu_{jIi}^{(t,c)}\right)^2 \leq \sum_{i,c}\tau_{max}\left(A_i^{(c)}+\max_{t\in\mathcal{T}_c}\sum_{j,I}\mu_{jIi}^{(t,c)}\right)^2$$
$$\leq \sum_{i}\tau_{max}\left(\sum_{c}\left[A_i^{(c)}+\max_{t\in\mathcal{T}_c}\sum_{j,I}\mu_{jIi}^{(t,c)}\right]\right)^2 \quad (5.67)$$
$$\leq N\tau_{max}\left(A_{\max}+\mu_{max}^{in}\right)^2$$

(where the Cauchy-Schwarz inequality is used in steps (5.66) and (5.67)), and using (5.39), (5.40), we obtain the drift expression

$$E\{L(\underline{U}(\tau+1))-L(\underline{U}(\tau))|\underline{U}(\tau)\}\leq 2BN-$$
$$2\sum_{i,c,t}U_i^{(t,c)}(\tau)\left[E\left\{\sum_{J,j}\mu_{iJj}^{ct}-\sum_{j,I}\mu_{jIi}^{(t,c)}\middle|\underline{U}(\tau)\right\}-r_i^{(c)}\right] \quad (5.68)$$

Substituting (5.53) and (5.56) into (5.68) gives

$$E\{L(\underline{U}(\tau+1))-L(\underline{U}(\tau))|\underline{U}(\tau)\}\leq 2BN-2\epsilon'\sum_{i,c,t}U_i^{(t,c)}(\tau) \quad (5.69)$$

where B is defined in (5.59).

Applying Theorem 5.2 gives

$$\sum_{i,c,t}\overline{U_i^{(t,c)}}\leq\frac{BN}{\epsilon'}. \quad (5.70)$$

Thus the randomized policy satisfies the queue occupancy bound (5.58).

For the back pressure policy, $E\{\mu_{iJj}^{(t,c)}(\tau)|\underline{U}(\tau)\}$ is dependent on $\underline{U}(\tau)$. The drift expression (5.71) can be expressed as

$$E\{L(\underline{U}(\tau+1))-L(\underline{U}(\tau))|\underline{U}(\tau)\}\leq 2BN-2\left[D-\sum_{i,c,t}U_i^{(t,c)}(\tau)r_i^{(c)}\right]$$

where

$$D=\sum_{i,c,t}U_i^{(t,c)}(\tau)\left[E\left\{\sum_{J,j}\mu_{iJj}^{(ct)}-\sum_{j,I}\mu_{jIi}^{(t,c)}\middle|\underline{U}(\tau)\right\}\right], \quad (5.71)$$

which is the portion of the drift expression that depends on the policy, can be rewritten as

$$D=\sum_{i,J,j}\sum_{c,t}E\left\{\mu_{iJj}^{(t,c)}|\underline{U}(\tau)\right\}\left(U_i^{(t,c)}(\tau)-U_j^{(t,c)}(\tau)\right).$$

We compare the values of (5.72) for the two policies, giving

$$\begin{aligned} D_{rand} &= \sum_{i,J,j}\sum_{c,t} x_{iJj}^{(t,c)}\left(U_i^{(t,c)}-U_j^{(t,c)}\right) \\ &\le \sum_{j,I}\sum_{c} y_{iJ}^{c}\sum_{t}\max_{j\in Z}\left(U_i^{(t,c)}-U_j^{(t,c)}\right) \\ &\le \sum_{j,I}\sum_{c} y_{iJ}^{c} w_{iJ}^{*} \\ &\le \sum_{j,I} z_{iJ} w_{iJ}^{*} \\ &= \sum_{j,I}\left(\sum_{\underline{S}} \pi_{\underline{S}} z_{iJ}^{\underline{S}}\right) w_{iJ}^{*} \\ &\le \sum_{\underline{S}} \pi_{\underline{S}} \max_{\underline{P}\in\Pi}\sum_{j,I}\mu_{iJ}(\underline{P},\underline{S}) w_{iJ}^{*} \\ &= D_{backpressure} \end{aligned}$$

where the last step follows from (5.49)-(5.50). Since the Lyapunov drift for the back-pressure policy is more negative than the drift for the randomized policy, the bound (5.70) also applies for the back-pressure policy. This completes the proof. □

The queue-length-based policy can be simplified in the wired network case where each arc (i,j) has a destination node set of size 1 and a capacity μ_{ij} that does not depend on $\underline{P}$ or $\underline{S}$.

Back-pressure policy for wired networks

For each time slot τ and each arc (i,j),

- Session scheduling: one session

$$c_{ij}^{*} = \arg\max_{c}\left\{\sum_{t\in\mathcal{T}_c}\max\left(U_i^{(t,c)}-U_j^{(t,c)},\ 0\right)\right\}$$

  is chosen.
- Rate allocation: the maximum rate of virtual transmissions from $Q_i^{(t,c)}$ to $Q_j^{(t,c)}$ is set as

$$\mu_{ij}^{(t,c)}(\tau) = \begin{cases} \mu_{ij} & \text{if } c=c_{ij}^{*}, t\in\mathcal{T}_c, \text{ and } U_i^{(t,c)}-U_j^{(t,c)}>0 \\ 0 & \text{otherwise.} \end{cases} \quad (5.72)$$

- Network coding: each session c packet physically transmitted on arc

(i, j) is a random linear combination, in $\mathbb{F}_q$, of packets corresponding to a set of virtual transmissions on (i, j), each associated with a different sink in $\mathcal{T}_c$.

Theorem 5.1 implies that each sink can receive packets at a rate asymptotically close to the source rate. To retrieve the actual information, each sink must also be able to decode the coded packets. The following theorem shows that the probability that not all sinks are able to decode the information tends to zero exponentially in the coding block length.

Theorem 5.3 *For exogenous arrival rates $w_i^{(c)} = r_i^{(c)} - \epsilon$, if $(r_i^{(c)})$ is strictly interior to Λ, then for sufficiently large time τ, the probability that not every sink is able to decode its session's exogenous packets decreases exponentially in the length of the code.*

Proof As described in Section 2.5, we can draw a parallel between a given sequence $\mathcal{S}$ of packet transmissions and a corresponding static network $\mathcal{G}$ with the same node set $\mathcal{N}$ and with links corresponding to transmissions in $\mathcal{S}$. The following analysis is an extension, based on this correspondence, of the analysis in Section 2.4.2 of random network coding for static networks.

Consider any session c. Let the randomly chosen network coding coefficients associated with the session c packets be represented by a vector $\underline{\xi} = (\xi_1, \ldots, \xi_\nu)$. Consider any sink $t \in \mathcal{T}_c$. It follows from Theorem 5.1 that over some sufficiently large time τ, with high probability there is a virtual flow of $r_i^c\tau - \sum_j U_j^{(t,c)}(\tau) \geq (r_i^{(c)} - \epsilon)\tau$ packets from each session c source node i to t, corresponding to coded combinations of $(r_i^{(c)} - \epsilon)\tau$ exogenous packets. Consider any $(r_i^{(c)} - \epsilon)\tau$ of the packets received by t from each session c source node i. We denote by $d^{(t,c)}(\underline{\xi})$ the determinant, as a polynomial in $\underline{\xi}$, of the matrix whose rows equal the coefficient vectors of these packets. Consider the physical packet transmissions corresponding to this virtual flow, which are transmissions involving queues $Q_j^{(t,c)}$. These physical transmissions would constitute an uncoded physical flow if their originating transmissions from the source nodes were uncoded independent packets and there were no other sinks/virtual flows in the network. We denote by $\tilde{\underline{\xi}}$ the value of $\underline{\xi}$ corresponding to this case, noting that $d^{(t,c)}(\tilde{\underline{\xi}}) = \pm 1$.† Thus, $d^{(t,c)}(\underline{\xi})$ is not identically zero.

† For this uncoded flow case, the coefficient vectors of the $(r_i^{(c)} - \epsilon)\tau$ session c packets received by t form the rows of the identity matrix.

Since the product $\prod_{c,t\in\mathcal{T}_c} d^{(t,c)}(\underline{\xi})$ as a function of the network code coefficients $\underline{\xi}$ is not identically zero, by the Schwartz-Zippel theorem, choosing the code coefficients uniformly at random from a finite field of size q yields a zero value with probability inversely proportional to q. The result follows since q is exponential in the length of the code. □

5.2.2 Inter-session coding

Queue-length-based approaches can also be extended to subgraph selection for simple inter-session network codes such as those described in Section 3.5. In different classes of network coding strategies, different aspects of the coding/routing history of packets restrict whether and how they can be coded/decoded/removed at particular nodes. Using these aspects to define different commodities or queues allows the effect of such restrictions to be propagated to influence control decisions at other nodes. The class of strategies over which we optimize, as well as the complexity and convergence rate, is determined by the choice of commodities and algorithm details.

Queue-length-based algorithms for optimizing over the class of pairwise poison-antidote codes (ref Section 3.5.1) are given in [42, 61]. A common feature of both algorithms is that they make coding decisions by treating an XOR coding operation as a type of virtual transmission that, unlike the virtual transmissions of the previous section, does not occur over a physical network arc. A coding operation is analogous to an actual arc transmission in that it removes one packet from each of a set of start queues and adds one packet to each of a set of end queues. For pairwise poison-antidote coding, there are two start queues corresponding to the packets being coded together, and, depending on the variant of the algorithm, the end queues correspond to the resulting poison packet and/or antidote packets†. At each step, besides prioritizing among physical transmissions over arcs analogously to the previous section, the algorithms also choose, among the coding possibilities at each node, the one with the largest positive potential difference across start and end queues. In [61] there are also virtual decoding transmissions that determine, based on local queue lengths, where each poison packet is decoded. Interested readers are referred to [42, 61] for details.

Simpler algorithms in this vein can be used to optimize over classes of strategies involving the canonical wireless one-hop XOR coding scenarios

† Separate control messages must be sent to the nodes from which the antidote packets are to originate.

of Section 3.5.2.1: for controlling routing/MAC to optimally create and exploit opportunities for coding. Such approaches generalize the COPE protocol which, as discussed in Sections 3.5.2.2-3.5.2.3, assumes given protocols for routing and MAC that do not take coding into account. COPE has been shown experimentally to yield significant performance improvements for UDP sessions over 802.11 wireless networks [77, 78], but it has not been rigorously studied under a theoretical model. Queue-length-based approaches offer one possible approach for distributed optimization over various classes of wireless one-hop XOR codes.

5.3 Notes and further reading

The first papers to broach the subject of subgraph selection in coded packet networks are due to Cui et al. [31], Lun et al. [95, 96, 97, 100, 101], and Wu et al. [140, 141]. These papers all describe flow-based approaches for intra-session coding. Subsequent extensions of this work are plentiful and include [15, 17, 81, 91, 121, 122, 130, 137, 142, 143]. The distributed algorithms that we describe, the primal-dual method and the subgradient method, first appear in [100] and [95], respectively. The flow-based approach for inter-session coding that we discuss is due to Traskov et al. [132]. Another, that deals with COPE-like coding in wireless networks, has recently been put forth by Sengupta et al. [124].

Queue-length-based approaches to subgraph selection in coded packet networks first appear in [67] for the case of intra-session multicast coding. The approach and analysis presented in this chapter, from [59], is based on and generalizes that in [106] for the case of multi-commodity routing. Queue-length-based approaches for inter-session coding are still in their infancy, though they show promise. Some recent work on this topic is described in [42, 61].

An approach to subgraph selection for wireless XOR coding in triangular grid networks is given in [39].

6
Security Against Adversarial Errors

6.1 Introduction

Multicast in decentralized settings, such as wireless ad hoc and peer to peer networks, is seen as a potential application area that can benefit from distributed network coding and its robustness to arc failures and packet losses. In such settings, packets are coded and forwarded by end hosts to other end hosts. It is thus important to consider security against compromised nodes.

Network coding presents new capabilities as well as challenges for network security. One advantage of multicast network coding is that it facilitates the use of a subgraph containing multiple paths to each sink node. Coding across multiple paths offers useful possibilities for information theoretic security against adversaries that observe or control a limited subset of arcs/transmissions in the network. By adding appropriately designed redundancy, error detection or error correction capabilities can be added to a distributed multicast scheme based on random linear network coding, as described in the following. On the other hand, coding at intermediate nodes poses a problem for traditional security techniques. For instance, coded combinations involving an erroneous packet result in more erroneous packets, so traditional error correction codes that deal with a limited proportion of erroneous packets are less effective. Also, traditional signature schemes do not allow for coding at non-trusted intermediate nodes. A homomorphic signature scheme by [22], which is based on elliptic curves, allows nodes to sign linear combinations of packets; under the assumption of the hardness of the computational co-Diffie-Hellman problem on elliptic curves, it prevents forging of signatures and detects corruption of packets. In this chapter

we focus on the problem of detection and correction of adversarial errors in multicast network coding, taking an information theoretic approach.

6.1.1 Notational conventions

We denote matrices with bold uppercase letters and vectors with bold lowercase letters. All vectors are row vectors unless indicated otherwise with a subscript T. We denote by $[\mathbf{x}, \mathbf{y}]$ the concatenation of two row vectors $\mathbf{x}$ and $\mathbf{y}$. For any vector (or matrix) whose entries (rows/columns) are indexed by the arcs of a network, we assume a consistent ordering of the vector entries (matrix rows/columns) corresponding to a topological ordering of the arcs.

6.2 Error correction

We consider network coded multicast on an acyclic graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, with a single source node $s \in \mathcal{N}$ and a set $\mathcal{T} \subset \mathcal{N}$ of sink nodes. The problem is to correct errors introduced on an unknown subset $\mathcal{Z} \subset \mathcal{A}$ of arcs (or packets†), so as to allow reliable communication. The maximum rate at which reliable communication is possible depends on $|\mathcal{Z}|$ and the minimum source-sink cut capacity $m = \min_{t \in \mathcal{T}} R(s, t)$, where $R(s, t)$ is the minimum cut capacity between s and t. We discuss below some theoretical bounds as well as constructions of network error-correcting codes.

6.2.1 Error correction bounds for centralized network coding

6.2.1.1 Model and problem formulation

The case where the network code is centrally designed and known to all parties (source, sinks and adversary) is the most direct generalization from traditional algebraic coding theory to network coding.

The problem formulation here is similar to that of Section 3.2 in that all arcs are assumed to have the same capacity (there can be multiple arcs connecting a pair of nodes), and we are interested in how large the source rate can be relative to the arc capacity. We use the term *network* to refer to a tuple $(\mathcal{G}, s, \mathcal{T})$, or equivalently, $(\mathcal{N}, \mathcal{A}, s, \mathcal{T})$.

Without loss of generality, we assume that the source node has in-degree 0. Since the network is acyclic, we can adopt a delay-free network

† See Section 2.5 for a discussion of the correspondence between the static arc-based and dynamic packet-based network models.

coding model, i.e. the nth symbol of an arc l is transmitted only after $o(l)$ has received the nth symbol of of its input processes. We restrict consideration to scalar network coding, i.e. the nth symbol transmitted on an arc l is a function of the nth symbol of each input process of node $o(l)$, and this function is the same for all n. The transmitted symbols are from an arc alphabet $\mathcal{Y}$ assumed to be equal to $\mathbb{F}_q$ for some prime power q. In these respects the coding model resembles the delay-free scalar linear coding model of Section 2.2. However, we allow arbitrary (possibly nonlinear) coding operations, and allow the source alphabet $\mathcal{X}$ to be different from the arc alphabet $\mathcal{Y}$; instead of a given number of fixed-rate source processes, we have a single source process whose rate $\log|\mathcal{X}|$ we seek to bound relative to the arc capacity $\log|\mathcal{Y}|$.

As in Section 2.2, we can focus on a single symbol for the source and each arc. The coding operation for an arc l is a function $\phi_l : \mathcal{X} \to \mathcal{Y}$ if $o(l) = s$, or $\phi_l : \prod_{k:\, d(k)=o(l)} \mathcal{Y} \to \mathcal{Y}$ if $o(l) \neq s$. The set of coding operations for all network arcs defines a network code $\phi = \{\phi_l : l \in \mathcal{A}\}$. Let X denote the random source symbol, and Y_l the random symbol received by the end node $d(l)$ of arc l.† Let

$$Y_{\mathcal{I}(l)} := \begin{cases} X & o(l) = s \\ \{Y_k : d(k) = o(l)\} & o(l) \neq s \end{cases}$$

denote the set of input symbols of an arc l.

We index and consider the arcs $l \in \mathcal{A}$ in topological order, i.e. lower-indexed arcs are upstream of higher-indexed arcs. For brevity we will refer to the arc and its index interchangeably. If no arc error occurs on l, $Y_l = \phi_l(Y_{\mathcal{I}(l)})$. An arc error is said to occur on l if $Y_l \neq \phi_l(Y_{\mathcal{I}(l)})$.

We say that a *z-error* occurs (in the network) if errors occur on exactly z of the arcs. A network code is said to *correct* a *z-error* if, upon occurrence of the error, each sink in $\mathcal{T}$ is still able to reproduce the source symbol. A network code is *z-error-correcting* if it can correct all z'-errors for all $z' \leq z$. For a set $\mathcal{Z}$ of arcs, if an error occurs on each arc $l \in \mathcal{Z}$ and no errors occur on other arcs, an *$\mathcal{Z}$-error* is said to occur; the set $\mathcal{Z}$ is called the *error pattern* of the error. A *$\mathcal{Z}$-error-correcting* code corrects all $\mathcal{Z}$-errors.

The proofs of the network error correction bounds below use a number of additional definitions. For each arc $l \in \mathcal{A}$, we define the (error-free)

† The transmitted symbol may differ from the received symbol, e.g. if the error is caused by interference on the arc. An error can also be caused by an adversarial or faulty node transmitting a value different from that specified by the network code. The following analysis, which focuses on the received symbol, applies to both cases.

global coding function $\tilde{\phi}_l : \mathcal{X} \to \mathcal{Y}$, where $\tilde{\phi}_l(X) = Y_l$ when all arcs are error-free. $\Gamma_+(Q) := \{(i,j) : i \in Q, j \notin Q\}$ denotes the set of forward arcs of a cut Q; $|\Gamma_+(Q)|$ is called the *size* of the cut.

6.2.1.2 Upper bounds

In this section we present upper bounds on the size of the source alphabet, which are analogs of the classical Hamming and Singleton bounds for point-to-point error-correcting codes. Here we consider arbitrary (possibly nonlinear) coding functions ϕ_l, and define the error value e_l associated with arc l as

$$e_l := \big(Y_l - \phi_l(Y_{\mathcal{I}(l)})\big) \bmod q. \tag{6.1}$$

Note that e_l is defined relative to the values of the arc inputs $Y_{\mathcal{I}(l)}$. This allows us to think of e_l simply as an input of node $d(l)$; for a given code and arc error values, we can inductively determine the arc values Y_l in topological order using

$$Y_l = \big(\phi_l(Y_{\mathcal{I}(l)}) + e_l\big) \bmod q. \tag{6.2}$$

The same result is obtained if we first find the arc values when the network is error-free, then "apply" the arc errors in topological order, i.e. for each arc l for which $e_l \neq 0$, we add e_l to Y_l mod q and change the values of higher-indexed arcs accordingly. Note that an error on arc l does not affect lower-indexed arcs. A (network) error is defined by the vector $\mathbf{e} := (e_l : l \in \mathcal{A}) \in \mathcal{Y}^{|\mathcal{A}|}$; we will refer to an error and its corresponding vector interchangeably.

Theorem 6.1 (Generalized Hamming Bound) *Let $(\mathcal{G}, s, \mathcal{T})$ be an acyclic network, and let $m = \min_{t\in\mathcal{T}} R(s,t)$. If there exists a z-error-correcting code on $(\mathcal{G}, s, \mathcal{T})$ for an information source with alphabet $\mathcal{X}$, where $z \leq m$, then*

$$|\mathcal{X}| \leq \frac{q^m}{\sum_{i=0}^{z} \binom{m}{i}(q-1)^i},$$

where q is the size of arc alphabet $\mathcal{Y}$.

Proof For a given network code ϕ and a set $\mathcal{L}$ of arcs, we denote by $out(\phi, z, \mathcal{L}, x)$ the set of all possible values of the vector $(Y_l : l \in \mathcal{L})$ when the source value is $x \in \mathcal{X}$ and at most z errors occur in the network.

Suppose ϕ is a z-error-correcting code. Consider any cut Q separating the source node s and a sink node t, and any pair of distinct source

values $x, x' \in \mathcal{X}$. To ensure t can distinguish between x and x' for up to z errors, we must have

$$out(\phi, z, \Gamma_+(Q), x) \cap out(\phi, z, \Gamma_+(Q), x') = \emptyset. \tag{6.3}$$

Consider the set $\mathcal{E}$ consisting of $\mathcal{Z}$-errors where $\mathcal{Z} \subset \Gamma_+(Q), |\mathcal{Z}| \leq z$. Let $\mathbf{e}, \mathbf{e}'$ be two distinct errors in $\mathcal{E}$, and let k_0 be the smallest arc index k such that the kth entry of $\mathbf{e}$ and $\mathbf{e}'$ differ. For a fixed source value $x \in \mathcal{X}$, the values of $Y_l, l < k_0$ are the same under both errors $\mathbf{e}, \mathbf{e}'$ while the value of Y_{k_0} differs for the two errors. Thus, the value of $(Y_l : \ l \in \Gamma_+(Q))$ differs for any pair of distinct errors in $\mathcal{E}$. Let $|\Gamma_+(Q)| = j$. Since $|\mathcal{E}| = \sum_{i=0}^{z} \binom{j}{i}(q-1)^i$, we have

$$|out(\phi, z, \Gamma_+(Q), x)| \geq \sum_{i=0}^{z} \binom{j}{i}(q-1)^i. \tag{6.4}$$

From (6.13) and (6.4), since there are only q^j possible values for $(Y_l : l \in \Gamma_+(Q))$, the number of source values is bounded by

$$|\mathcal{X}| \leq \frac{q^j}{\sum_{i=0}^{z} \binom{j}{i}(q-1)^i}.$$

The theorem follows by noting that the bound holds for any source-sink cut. □

Theorem 6.2 (Generalized Singleton Bound) *Let $(\mathcal{G}, s, \mathcal{T})$ be an acyclic network, and let $m = \min_{t \in \mathcal{T}} R(s,t)$. If there exists a z-error-correcting code on $(\mathcal{G}, s, \mathcal{T})$ for an information source with alphabet $\mathcal{X}$, where $m > 2z$, then*

$$\log |\mathcal{X}| \leq (m - 2z) \log q.$$

Proof Suppose $\{\phi_l : \ l \in \mathcal{A}\}$ is a z-error-correcting code for an information source with alphabet $\mathcal{X}$, where $m > 2z$ and

$$|\mathcal{X}| > q^{m-2z}. \tag{6.5}$$

We will show that this leads to a contradiction.
Consider a sink $t \in \mathcal{T}$ for which there exists a cut Q of size m between the source and t. Let $k_1, \ldots, k_m$ be the arcs of $\Gamma_+(Q)$, ordered topologically, i.e. $k_1 < k_2 < \cdots < k_m$. By (6.5), there exist two distinct source symbols $x, x' \in \mathcal{X}$ such that $\tilde{\phi}_{k_i}(x) = \tilde{\phi}_{k_i}(x') \ \forall \ i = 1, 2, \ldots, m-2z$, so we can write

$$\left(\tilde{\phi}_{k_1}(x), \ldots, \tilde{\phi}_{k_m}(x)\right) = (y_1, \ldots, y_{m-2z}, u_1, \ldots, u_z, w_1, \ldots, w_z) \tag{6.6}$$

$$\left(\tilde{\phi}_{k_1}(x'), \ldots, \tilde{\phi}_{k_m}(x')\right) = (y_1, \ldots, y_{m-2z}, u'_1, \ldots, u'_z, w'_1, \ldots, w'_z) \quad (6.7)$$

Suppose the source symbol is x. Let $\mathcal{Z}$ be the set of arcs

$$\{k_{m-2z+1}, \ldots, k_{m-z}\}.$$

We can construct a $\mathcal{Z}$-error that changes the value of $(Y_{k_1}, \ldots, Y_{k_m})$ from its error-free value in (6.6) to the value

$$(y_1, \ldots, y_{m-2z}, u'_1, \ldots, u'_z, w''_1, \ldots, w''_z)\,, \quad (6.8)$$

as follows. We start with the error-free value of $(Y_{k_1}, \ldots, Y_{k_m})$, and apply errors on the arcs of $\mathcal{Z}$ in topological order. First, we apply an error of value $(u'_1 - Y_{k_{m-2z+1}}) \bmod q = (u'_1 - u_1) \bmod q$ on arc k_{m-2z+1}, which causes $Y_{k_{m-2z+1}}$ to change value from u_1 to u'_1. Note that this may change the values of $Y_j, j > k_{m-2z+1}$ but not the values of $Y_j, j < k_{m-2z+1}$. We proceed similarly for arcs $k_{m-2z+i}, i = 2, \ldots, z$, in turn: we apply an error of value $(u'_i - Y_{k_{m-2z+i}}) \bmod q$ and update the values of $Y_j, j > k_{m-2z+i}$ accordingly. The value of $(Y_{k_1}, \ldots, Y_{k_m})$ at the end of this procedure is given by (6.8).

For the source symbol x', we can follow a similar procedure to construct, for the set of arcs $\mathcal{Z}' = \{k_{m-z+1}, \ldots, k_m\}$, a $\mathcal{Z}'$-error that changes the value of $(Y_{k_1}, \ldots, Y_{k_m})$ from its error-free value in (6.7) to the value in (6.8).

Thus, sink t cannot reliably distinguish between the source symbols x and x', which gives a contradiction. □

6.2.1.3 Generic linear network codes

Before developing lower bounds on the source alphabet size in the next section, we introduce the notion of a *generic* linear network code, which will be used in constructing network error-correcting codes that prove the bounds. Intuitively speaking, a generic linear code is a scalar linear code satisfying the following maximal independence property: for every subset of arcs, if their (global) coding vectors can be linearly independent in some network code, then they are linearly independent in a generic linear code. A generic linear code is formally defined as follows.

Definition 6.1 *For an acyclic network $(\mathcal{N}, \mathcal{A}, s, \mathcal{T})$, let a linear network code with n-dimensional coding vectors $\{\mathbf{c}_l : l \in \mathcal{A}\}$, whose entries are elements of a field $\mathbb{F}_q$, be given. Let each node $i \in \mathcal{N}$ be associated with*

a linear subspace W_i *of* $\mathbb{F}_q^n$*, where*

$$W_i = \begin{cases} \mathbb{F}_q^n & i = s \\ span(\{\mathbf{c}_l : d(l) = i\}) & i \in \mathcal{N}\backslash s. \end{cases}$$

The network code is generic if, for any subset of arcs $\mathcal{S} \subset \mathcal{A}$*,*

$$W_{o(l)} \not\subset span(\{\mathbf{c}_k : k \in \mathcal{S}\backslash l\}) \; \forall \; l \in \mathcal{S} \tag{6.9}$$

$$\Rightarrow \quad \textit{Coding vectors } \mathbf{c}_l, l \in \mathcal{S} \textit{ are linearly independent.} \tag{6.10}$$

Note that (6.9) is a necessary condition for (6.10); the definition of a generic code requires the converse to hold.

Generic linear codes satisfy a stronger linear independence requirement compared to multicast linear codes (ref Section 2.2) which require only that each sink node has a full rank set of inputs. Thus, a generic linear code is also a multicast linear code, but a multicast linear code is not in general generic.

Generic linear codes can be constructed using techniques analogous to those we have seen for constructing multicast linear codes. The random linear coding technique introduced for constructing multicast codes in Section 2.4.2 can also construct generic linear network codes with probability asymptotically approaching 1 in the field size, though significantly larger field sizes may be required to achieve similar success probabilities in constructing generic codes, as compared to multicast codes, on a given network. We can also take a centralized deterministic approach similar to that in Section 2.4.1. For any positive integer n and an acyclic network $(\mathcal{N}, \mathcal{A}, s, \mathcal{T})$, the following algorithm constructs a generic linear network code over a finite field F with more than $\binom{|\mathcal{A}|+n-1}{n-1}$ elements. The algorithm is similar to Algorithm 1 for constructing a multicast linear code, in that it sets the coding vectors† of the network arcs in topological order, starting with n virtual arcs connecting a virtual source node s' to the actual source node s.

Note that at step A, there are at most $\binom{|\mathcal{A}|+n-1}{n-1}$ sets $\mathcal{S}$, so it is always possible to find a vector $\mathbf{w}$ satisfying the condition. It can be shown by induction that the network code constructed by Algorithm 2 is always generic. The proof is given in [149].

6.2.1.4 Lower bounds

Next we derive lower bounds on the size of the source alphabet, which generalize the classical Gilbert and Varshamov bounds for point-to-point

† The coding coefficients for an arc can be obtained from the coding vector of the arc.

Algorithm 2: Centralized algorithm for generic linear network code construction

Input: $\mathcal{N}, \mathcal{A}, s, \mathcal{T}, n$
$\mathcal{N}' := \mathcal{N} \cup \{s'\}$
$\mathcal{A}' := \mathcal{A} \cup \{l_1, \ldots, l_m\}$ where $o(l_i) = s', d(l_i) = s$ for $i = 1, \ldots, m$
foreach $i = 1, \ldots, m$ **do** $\mathbf{c}_{l_i} := [\mathbf{0}^{i-1}, 1, \mathbf{0}^{n-i}]$
foreach $l \in \mathcal{A}$ **do** Initialize $\mathbf{c}_l := \mathbf{0}$;
foreach $i \in \mathcal{N}$ *in topological order* **do**
  foreach $l \in \mathcal{O}(i)$ **do**
A choose $\mathbf{c}_l := \mathbf{w} \in \text{span}(\{\mathbf{c}_k : d(k) = i\})$ where
  $\mathbf{w} \notin \text{span}(\{\mathbf{c}_k : k \in \mathcal{S}\})$ for any set $\mathcal{S}$ of $n-1$ arcs in $\mathcal{A} \backslash l$
  such that $\text{span}(\{\mathbf{c}_k : d(k) = i\}) \not\subset \text{span}(\{\mathbf{c}_k : k \in \mathcal{S}\})$

error correcting codes. These bounds give sufficient conditions for the existence of network error-correcting codes with parameters satisfying the bounds.

The proofs are by construction. To construct the network error-correcting codes in this section, we use a generalization of scalar linear network coding where the arc alphabet $\mathcal{Y}$ is a finite field $\mathbb{F}_q$ and the source alphabet $\mathcal{X}$ is a subset of an n-dimensional linear space $\mathbb{F}_q^n$, where we set n equal to the minimum source-sink cut size $m = \min_{t \in \mathcal{T}} R(s,t)$. (For basic scalar linear network coding (ref Section 2.2), the source alphabet is the entire n-dimensional linear space $\mathbb{F}_q^n$, where n is equal to the number of source processes.) For a linear network error-correcting code, $\mathcal{X}$ is a k-dimensional subspace of $\mathbb{F}_q^m$ for some $k \leq m$. We will consider the source values $\mathbf{x} \in \mathcal{X}$ as length-m row vectors with entries in $\mathbb{F}_q$. To distinguish between the set of arc coding operations $\phi = \{\phi_l : l \in \mathcal{A}\}$ (defined in Section 6.2.1.1) and the complete network code, which includes the choice of $\mathcal{X} \subset \mathbb{F}_q^m$ as well, we refer to the former as the *underlying* (scalar linear network) code. In this section, we use a generic linear code as the underlying code ϕ, which can be constructed for a given network as described in the previous section. Our remaining task is to choose $\mathcal{X}$ such that its values can be distinguished under a given set of error events.

The error value e_l associated with an arc l is defined as the difference,

in $\mathbb{F}_q$, between Y_l and $\phi_l(Y_{\mathcal{I}(l)})$. In place of (6.1)-(6.2) we have

$$e_l := Y_l - \phi_l(Y_{\mathcal{I}(l)}) \tag{6.11}$$

$$Y_l = \phi_l(Y_{\mathcal{I}(l)}) + e_l \tag{6.12}$$

where all operations are in $\mathbb{F}_q$. An error is defined by the vector $\mathbf{e} := (e_l : \ l \in \mathcal{A}) \in \mathbb{F}_q^{|\mathcal{A}|}$. Since Y_l is given recursively by (6.12) which is a linear relation, the value of Y_l can be expressed as the sum

$$\tilde{\phi}_l(\mathbf{x}) + \theta_l(\mathbf{e})$$

where $\tilde{\phi}_l$ and θ_l are linear functions, determined by ϕ, whose arguments $\mathbf{x}, \mathbf{e}$ are the source and error values respectively. $\tilde{\phi}_l(\mathbf{x})$, the error-free global coding function for arc l, is given by $\mathbf{x}\mathbf{c}_l^T$ where $\mathbf{c}_l$ is the global coding vector of arc l in the underlying code ϕ.

Consider a set Υ of error patterns, and let Υ^* be the set of all errors whose error pattern is in Υ. A pair of distinct source values $\mathbf{x}, \mathbf{x}'$ is said to be Υ-*separable* at a sink node t if t can distinguish between $\mathbf{x}$ and $\mathbf{x}'$ for any errors in Υ^*. In other words, $\forall \mathbf{e}, \mathbf{e}' \in \Upsilon^*$,

$$(\tilde{\phi}_l(\mathbf{x}) + \theta_l(\mathbf{e}) : \ l \in \mathcal{I}(t)) \neq (\tilde{\phi}_l(\mathbf{x}') + \theta_l(\mathbf{e}') : \ l \in \mathcal{I}(t)). \tag{6.13}$$

A pair $\mathbf{x}, \mathbf{x}'$ is said to be Υ-separable if it is Υ-separable at every sink node.

We wish to translate this condition on separability into a restriction on the set $\mathcal{X}$ of source values. We assume without loss of generality that the number of input arcs at each sink is exactly m, the source-sink minimum cut size. Let $\mathbf{C}_t$ denote the matrix whose columns correspond to the coding vectors $\mathbf{c}_l(\mathbf{x})$ of t's input arcs $l \in \mathcal{I}(t)$. Let $\mathbf{p}_t(\mathbf{e})$ denote the row vector $(\theta_l(\mathbf{e}) : \ l \in \mathcal{I}(t))$. Then (6.13) can be written as

$$\mathbf{x}\mathbf{C}_t + \mathbf{p}_t(\mathbf{e}) \neq \mathbf{x}'\mathbf{C}_t + \mathbf{p}_t(\mathbf{e}').$$

Right-multiplying both sides by $\mathbf{C}_t^{-1}$†, we obtain the following equivalent condition for $\mathbf{x}, \mathbf{x}'$ to be Υ-separable at t: $\forall \mathbf{e}, \mathbf{e}' \in \Upsilon^*$,

$$\mathbf{x} + \mathbf{p}_t(\mathbf{e})\mathbf{C}_t^{-1} \neq \mathbf{x}' + \mathbf{p}_t(\mathbf{e}')\mathbf{C}_t^{-1}.$$

Defining the sets

$$\Xi(\phi, \Upsilon, t) := \{\mathbf{p}_t(\mathbf{e})\mathbf{C}_t^{-1} : \ \mathbf{e} \in \Upsilon^*\} \tag{6.14}$$

$$\Delta(\phi, \Upsilon) := \bigcup_{t \in \mathcal{T}} \{\mathbf{w} = \mathbf{u}' - \mathbf{u} : \ \mathbf{u}, \mathbf{u}' \in \Xi(\phi, \Upsilon, t)\} \tag{6.15}$$

† The inverse exists since the underlying code is generic.

and denoting by $\mathbf{x} + \Delta(\phi, \Upsilon)$ the set $\{\mathbf{x} + \mathbf{w} : \ \mathbf{w} \in \Delta(\phi, \Upsilon)\}$, we have the following lemma.

Lemma 6.1 *(a) A pair of source values $\mathbf{x}, \mathbf{x}' \in \mathbb{F}_q^m$ is Υ-separable if and only if*

$$\mathbf{x}' \notin \mathbf{x} + \Delta(\phi, \Upsilon). \tag{6.16}$$

(b) The network code obtained from the underlying generic code ϕ by restricting the source alphabet to a set $\mathcal{X} \subset \mathbb{F}_q^m$ is a Υ-error-correcting code for the network if and only if the vectors in $\mathcal{X}$ are pairwise Υ-separable.

Let $K = |\mathcal{A}|$ be the number of arcs in the network, and let

$$\Upsilon_j := \{\mathcal{Z} : \ |\mathcal{Z}| = j, \mathcal{Z} \in \Upsilon\} \tag{6.17}$$

be the subset of error patterns in Υ with exactly j arcs.

Theorem 6.3 (Generalized Gilbert-Varshamov Bound) *For any given error pattern set Υ and any positive integer A satisfying*

$$(A-1)|\mathcal{T}| \left(\sum_{j=0}^{K} |\Upsilon_j|(q-1)^j \right)^2 < q^m, \tag{6.18}$$

one can construct an Υ-error-correcting code with source alphabet size $|\mathcal{X}| = A$. For any positive integer k satisfying

$$|\mathcal{T}| \left(\sum_{j=0}^{K} |\Upsilon_j|(q-1)^j \right)^2 < q^{m-k},$$

one can construct a k-dimensional linear Υ-error-correcting code, i.e. $|\mathcal{X}| = q^k$.

Proof We first consider a given underlying code ϕ. By Lemma 6.1, if we can find a set $\mathcal{X} \subset \mathbb{F}_q^m$ such that (6.16) holds for any pair $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, then the network code obtained from ϕ by restricting the source alphabet to $\mathcal{X}$ is an Υ-error-correcting code.

For the first part of the theorem, which generalizes the Gilbert bound, we use a greedy approach similar to that in Gilbert [49]. We show that for any positive integer A satisfying

$$(A-1)|\Delta(\phi, \Upsilon)| < q^m, \tag{6.19}$$

one can construct an Υ-error-correcting code with source alphabet size $|\mathcal{X}| = A$. First, a candidate set $\mathcal{W}$ of source values is initialized as $\mathbb{F}_q^m$. For $i = 1, \ldots, A-1$, at the ith step, an arbitrary vector $\mathbf{x}_i \in \mathcal{W}$ is chosen, $\mathbf{x}_i$ is added to $\mathcal{X}$, and all vectors in the set $(\mathbf{x}_+ \Delta(\phi, \Upsilon)) \cap \mathcal{W}$ are removed from $\mathcal{W}$. This is possible since the number of vectors removed at each step is at most $|\mathbf{x}_i + \Delta(\phi, \Upsilon)| = |\Delta(\phi, \Upsilon)|$, so that at each step $i \leq A-1$, $\mathcal{W}$ has size at least

$$\begin{aligned} |\mathbb{F}_q^m| - i|\Delta(\phi, \Upsilon)| &\geq q^m - (A-1)|\Delta(\phi, \Upsilon)| \\ &> 0 \end{aligned}$$

by condition (6.19). For the set $\mathcal{X}$ constructed by this procedure, any pair $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ satisfies (6.16).

For the second part of the theorem, which generalizes the Varshamov bound, we use an approach similar to that used in Varshamov [135]. We show that for any positive integer k satisfying

$$|\Delta(\phi, \Upsilon)| < q^{m-k}, \tag{6.20}$$

one can construct a k-dimensional linear Υ-error-correcting code, i.e. $|\mathcal{X}| = q^k$. For a linear code, the source alphabet $\mathcal{X}$ is a linear subspace of $\mathbb{F}_q^m$, and the condition that (6.16) holds for any pair $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ is equivalent to the condition that

$$\Delta(\phi, \Upsilon) \cap \mathcal{X} = \{\mathbf{0}\}. \tag{6.21}$$

We construct $\mathcal{X}$ by constructing its parity check matrix $\mathbf{H}$, which is an $(m-k) \times m$ matrix with full row rank such that

$$\mathcal{X} = \{\mathbf{x} : \ \mathbf{x} \in \mathbb{F}_q^m, \mathbf{H}\mathbf{x}^T = \mathbf{0}^T\}. \tag{6.22}$$

Defining

$$\Delta^*(\phi, \Upsilon) := \Delta(\phi, \Upsilon) \backslash \{\mathbf{0}\}, \tag{6.23}$$

we have that (6.21) is equivalent to the condition that

$$\mathbf{H}\mathbf{w}^T \neq \mathbf{0}^T \ \forall \ \mathbf{w} \in \Delta^*(\phi, \Upsilon). \tag{6.24}$$

To construct $\mathbf{H}$, we first partition $\Delta^*(\phi, \Upsilon)$ into subsets $\Delta_1(\phi, \Upsilon), \ldots, \Delta_m(\phi, \Upsilon)$ such that $\Delta_i(\phi, \Upsilon)$ contains all vectors $\mathbf{w} \in \Delta^*(\phi, \Upsilon)$ whose last nonzero entry is the ith entry, i.e.

$$\Delta_i(\phi, \Upsilon) := \{\mathbf{w} \in \Delta^*(\phi, \Upsilon) : \ \mathbf{w} = (w_1, w_2, \ldots, w_i, 0, \ldots, 0), w_i \neq 0\}. \tag{6.25}$$

Let $\mathbf{h}_i^T$ be the column vector corresponding to the ith column of $\mathbf{H}$,

i.e. $\mathbf{H} = [\mathbf{h}_1^T \ldots \mathbf{h}_m^T]$. We set $\mathbf{h}_1^T$ equal to any nonzero vector in $\mathbb{F}_q^{m-k}$.
For $i = 2, 3, \ldots, m$, we recursively set $\mathbf{h}_i^T$ equal to any vector in $\mathbb{F}_q^{m-k} \backslash \mathcal{K}_i(\mathbf{h}_1^T, \ldots, \mathbf{h}_{i-1}^T)$, where

$$\mathcal{K}_i(\mathbf{h}_1^T, \ldots, \mathbf{h}_{i-1}^T) := \left\{ \mathbf{k}^T \in \mathbb{F}_q^{m-k} : \; w_i \mathbf{k}^T + \sum_{j=1}^{i-1} w_j \mathbf{h}_j^T = \mathbf{0}^T \text{ for some } \mathbf{w} \in \Delta_i(\phi, \Upsilon) \right\};$$

this is possible since the number of possible choices for $\mathbf{h}_i^T$ is at least

$$\begin{aligned} |\mathbb{F}_q^{m-k}| - |\mathcal{K}_i(\mathbf{h}_1^T, \ldots, \mathbf{h}_{i-1}^T)| &\geq q^{m-k} - |\Delta_i(\phi, \Upsilon)| \\ &\geq q^{m-k} - |\Delta(\phi, \Upsilon)| \\ &> 0. \end{aligned}$$

By construction, (6.24) is satisfied.
To obtain bounds that are independent of the choice of underlying code ϕ, note that

$$|\Delta(\phi, \Upsilon)| \leq \sum_{t \in \mathcal{T}} |\Xi(\phi, \Upsilon, t)|^2 \tag{6.26}$$

$$\leq |\mathcal{T}| \, |\Upsilon^*|^2 \tag{6.27}$$

where (6.26) follows from the definition of $\Delta(\phi, \Upsilon)$ in (6.15), and (6.27) follows from the definition of $\Xi(\phi, \Upsilon, t)$ in (6.14). Recall that

$$\Upsilon_j := \{\mathcal{Z} : \; |\mathcal{Z}| = j, \mathcal{Z} \in \Upsilon\} \tag{6.28}$$

is the subset of error patterns in Υ with exactly j arcs. Then the number of errors in Υ^* is given by

$$|\Upsilon^*| = \sum_{j=0}^{K} |\Upsilon_j| (q-1)^j. \tag{6.29}$$

The theorem follows from combining (6.19)-(6.20) with (6.26), (6.27) and (6.29). □

We next consider the case where Υ is the collection of subsets of z or fewer arcs, where $z \leq m$. We can use a different proof approach, along with a bound on $|\Delta(\phi, \Upsilon)|$ which tightens the more general bound (6.27) for this case, to obtain the following bound which is tighter than that obtained by simply specializing Theorem 6.3 to this case.

Theorem 6.4 (Strengthened Generalized Varshamov Bound) *For any fixed acyclic network with minimum cut* $m = \min_{t \in \mathcal{T}} R(s, t)$,

$k = m - 2z > 0$, and q sufficiently large, there exists a k-dimensional linear z-error-correcting code for the network.

Proof As in the proof of Theorem 6.3, we construct $\mathcal{X}$ by constructing its $(2z) \times m$ parity check matrix $\mathbf{H}$ (ref (6.22)). We need matrix $\mathbf{H}$ to satisfy (6.24) for the case where Υ is the collection of subsets of z or fewer arcs. For this case, the sets

$$\Delta^*(\phi, \Upsilon),\ \Delta_i(\phi, \Upsilon), 1 \le i \le m,$$

defined in (6.23) and (6.25) respectively, are denoted by

$$\Delta^*(\phi, z),\ \Delta_i(\phi, z), 1 \le i \le m.$$

Each set $\Delta_i(\phi, z), 1 \le i \le m$, is partitioned into $|\Delta_i(\phi, z)|/(q-1)$ equivalence classes each of size $q-1$, such that the vectors in each equivalence class are nonzero scalar multiples of each other. For any particular one of these equivalence classes $\mathcal{Q} \subset \Delta_i(\phi, z)$, a matrix $\mathbf{H}$ satisfies

$$\mathbf{H}\mathbf{w}^T = \mathbf{0}^T \tag{6.30}$$

for all vectors $\mathbf{w} \in \mathcal{Q}$ if and only if it satisfies (6.30) for the vector $(w_1, \ldots, w_{i-1}, 1, 0, \ldots, 0) \in \mathcal{Q}$, or equivalently,

$$\mathbf{h}_i = -\sum_{j=1}^{i-1} w_j \mathbf{h}_j.$$

Thus, there are exactly $q^{2z(m-1)}$ values for $\mathbf{H}$ (corresponding to arbitrary values for $\mathbf{h}_1, \ldots, \mathbf{h}_{i-1}, \mathbf{h}_{i+1}, \ldots, \mathbf{h}_m$, the first $i-1$ of which determine $\mathbf{h}_i$) such that there exists $\mathbf{w} \in \mathcal{Q}$ satisfying (6.30). The number of values for $\mathbf{H}$ such that there exists $\mathbf{w} \in \Delta^*(\phi, z)$ satisfying (6.30) is then at most

$$\sum_{i=1}^{m} q^{2z(m-1)} |\Delta_i(\phi, z)|/(q-1) = q^{2z(m-1)} |\Delta^*(\phi, z)|/(q-1) \tag{6.31}$$

$$\le q^{2z(m-1)} |\Delta(\phi, z)|/(q-1) \tag{6.32}$$

We obtain a bound on $|\Delta(\phi, z)|$ that is tighter than the more general bound (6.27) as follows. From (6.14) and (6.15), we have

$$\Delta(\phi, z) = \bigcup_{t \in \mathcal{T}} \{\mathbf{p}_t(\mathbf{e})\mathbf{C}_t^{-1} - \mathbf{p}_t(\mathbf{e}')\mathbf{C}_t^{-1} :\ w_H(\mathbf{e}) \le z, w_H(\mathbf{e}') \le z\}$$

where w_H denotes Hamming weight. Since $\mathbf{p}_t(\mathbf{e})$ is a linear function in

$\mathbf{e}$, we have

$$\mathbf{p}_t(\mathbf{e})\mathbf{C}_t^{-1} - \mathbf{p}_t(\mathbf{e}')\mathbf{C}_t^{-1} = \mathbf{p}_t(\mathbf{e}-\mathbf{e}')\mathbf{C}_t^{-1},$$

and

$$\{\mathbf{e}-\mathbf{e}' :\ w_H(\mathbf{e}) \le z, w_H(\mathbf{e}') \le z\} = \{\mathbf{d} :\ w_H(\mathbf{d}) \le 2z\}.$$

Thus,

$$\Delta(\phi, z) = \bigcup_{t\in\mathcal{T}} \{\mathbf{p}_t(\mathbf{d})\mathbf{C}_t^{-1} :\ w_H(\mathbf{d}) \le 2z\} \tag{6.33}$$

which gives

$$\begin{aligned} |\Delta(\phi, z)| &\le |\mathcal{T}| \sum_{i=0}^{2z} \binom{K}{i} (q-1)^i \\ &< |\mathcal{T}|(q-1)^{2z} 2^m \\ &\le |\mathcal{T}|(q-1)^{2z} 2^K \end{aligned} \tag{6.34}$$

Using (6.34) we can upper bound (6.32) by

$$q^{2z(m-1)}|\mathcal{T}|2^K(q-1)^{2z-1} < |\mathcal{T}|2^K q^{2zm-1} \tag{6.35}$$

If $q \ge 2^K|\mathcal{T}|$, then (6.35) is upper bounded by q^{2zm}. Since the number of $2z \times m$ matrices over $\mathbb{F}_q$ is q^{2zm}, for $q \ge 2^K|\mathcal{T}|$, there exists some value for $\mathbf{H}$ such that $\mathbf{H}\mathbf{w}^T \ne \mathbf{0}^T\ \forall\ \mathbf{w} \in \Delta^*(\phi, z)$, which gives the desired network code. □

6.2.2 Distributed random network coding and polynomial-complexity error correction

In this section, we consider network error correction in a distributed packet network setting. The model and approach differ from that of the previous section in two ways. First, instead of a centrally-designed network code known in advance by all parties, we use distributed random linear network coding. Second, we allow a fixed amount of overhead in each packet that can be amortized over large packets. We describe, for this setting, constructions of asymptotically optimal network error-correcting codes with polynomial-complexity coding and decoding algorithms.

6.2.2.1 Coding vector approach

We consider multicasting of a batch of r packets from source node s to the set of sink nodes $\mathcal{T}$, using the distributed random coding approach with coding vectors described in Section 2.5.1.1. A non-adversarial packet is formed as a random linear combination†, in $\mathbb{F}_q$, of its input packets, i.e. packets received by its origin node prior to its formation. An adversary can arbitrarily corrupt the coding vector as well as the data symbols of a packet. A packet that is not a linear combination of its input packets is called *adversarial*.

We describe below the construction of a network error-correcting code whose parameters depend on the maximum number z_o of adversarial packets as well as m, the minimum source-sink cut capacity (maximum error-free multicast rate) in units of packets over the batch. The number of source packets in the batch is set as

$$r = m - z_o. \tag{6.36}$$

The proportion of redundant symbols in each packet, denoted ρ, is set as

$$\rho = (z_o + \epsilon)/r \tag{6.37}$$

for some $\epsilon > 0$.

For $i = 1, \ldots, r$, the ith source packet is represented as a length-n row vector $\mathbf{x}_i$ with entries in a finite field $\mathbb{F}_q$. The first $n - \rho n - r$ entries of the vector are independent exogenous data symbols, the next ρn are redundant symbols, and the last r symbols form the packet's coding vector (the unit vector with a single nonzero entry in the ith position). The corresponding information rate of the code is $m - 2z_o - \epsilon - r^2/n$, where the r^2/n term, due to the overhead of including the coding vector, decreases with the length n of the packet. We will show that the probability of error decreases exponentially with ϵ.

We denote by $\mathbf{X}$ the $r \times n$ matrix whose ith row is $\mathbf{x}_i$; it can be written in the block form $\begin{bmatrix} \mathbf{U} & \mathbf{R} & \mathbf{I} \end{bmatrix}$ where $\mathbf{U}$ denotes the $r \times (n - \rho n - r)$ matrix of exogenous data symbols, $\mathbf{R}$ denotes the $r \times \rho n$ matrix of redundant symbols and $\mathbf{I}$ is the $r \times r$ identity matrix.

The $r\rho n$ redundant symbols are obtained as follows. For any matrix $\mathbf{M}$, let $\mathbf{v}_\mathbf{M}^T$ denote the column vector obtained by stacking the columns of $\mathbf{M}$ one above the other, and $\mathbf{v}_\mathbf{M}$ its transpose, a row vector. Matrix $\mathbf{X}$, represented in column vector form, is given by $\mathbf{v}_\mathbf{X}^T = [\mathbf{v}_\mathbf{U}, \mathbf{v}_\mathbf{R}, \mathbf{v}_\mathbf{I}]^T$. Let

† All n symbols of a packet undergo the same random linear coding operations.

$\mathbf{D}$ be an $r\rho n \times rn$ matrix obtained by choosing each entry independently and uniformly at random from $\mathbb{F}_q$. The redundant symbols constituting $\mathbf{v_R}$ (or $\mathbf{R}$) are obtained by solving the matrix equation

$$\mathbf{D}[\mathbf{v_U}, \mathbf{v_R}, \mathbf{v_I}]^T = \mathbf{0} \tag{6.38}$$

for $\mathbf{v_R}$. The value of $\mathbf{D}$ is known to all parties.

Let $\mathbf{y}_u$ denote the vector representing a non-adversarial packet u. If there are no errors in the network, then for all packets u, $\mathbf{y}_u = \mathbf{t}_u\mathbf{X}$ where $\mathbf{t}_u$ is the packet's coding vector. An adversarial packet can be viewed as an additional source packet. The vector representing the ith adversarial packet is denoted $\mathbf{z}_i$. Let $\mathbf{Z}$ denote the matrix whose ith row is $\mathbf{z}_i$.

For the rest of this section, we focus on any one of the sink nodes $t \in \mathcal{T}$. Let w be the number of linearly independent packets received by t; let $\mathbf{Y} \in \mathbb{F}_q^{w\times n}$ denote the matrix whose ith row is the vector representing the ith of these packets. Since all coding operations in the network are scalar linear operations in $\mathbb{F}_q$, $\mathbf{Y}$ can be be expressed as

$$\mathbf{Y} = \mathbf{GX} + \mathbf{KZ} \tag{6.39}$$

where matrices $\mathbf{G} \in \mathbb{F}_q^{w\times r}$ and $\mathbf{K} \in \mathbb{F}_q^{w\times z}$ represent the linear mappings from the source and adversarial packets respectively to the sink's set of linearly independent input packets.

Since the matrix formed by the last r columns of $\mathbf{X}$ is the identity matrix, the matrix $\mathbf{G}'$ formed by the last r columns of $\mathbf{Y}$ is given by

$$\mathbf{G}' = \mathbf{G} + \mathbf{KL}, \tag{6.40}$$

where $\mathbf{L}$ is the matrix formed by the last r columns of $\mathbf{Z}$. In the error-free setting, $\mathbf{G}' = \mathbf{G}$; in the presence of errors, the sink knows $\mathbf{G}'$ but not $\mathbf{G}$. Thus, we rewrite (6.39) as

$$\begin{aligned} \mathbf{Y} &= \mathbf{G}'\mathbf{X} + \mathbf{K}(\mathbf{Z} - \mathbf{LX}) && (6.41)\\ &= \mathbf{G}'\mathbf{X} + \mathbf{E}. && (6.42) \end{aligned}$$

Matrix $\mathbf{E}$ can be intuitively interpreted as the effective error seen by the sink. It gives the difference between the data values in the received packets and the data values corresponding to their coding vectors; its last r columns are all zero.

Lemma 6.2 *With probability at least* $(1 - 1/q)^{|\mathcal{A}|} > 1 - |\mathcal{A}|/q$ *where* $|\mathcal{A}|$ *is the number of arcs in the network, the matrix* $\mathbf{G}'$ *has full column*

rank, and the column spaces of $\mathbf{G}'$ *and* $\mathbf{K}$ *are disjoint except in the zero vector.*

Proof If the adversarial packets were replaced by additional source packets, the total number of source packets would be at most $r + z_o = m$, by (6.36). By Theorems 2.3 and 2.6, with probability at least $(1 - 1/q)^{|\mathcal{A}|}$, random linear network coding in $\mathbb{F}_q$ allows t to decode the original source packets†. This corresponds to $\mathbf{G}$ having full column rank and the column spaces of $\mathbf{G}$ and $\mathbf{K}$ being disjoint except in the zero vector. The result follows by noting from (6.40) that any linear combination of columns of $\mathbf{G}'$ corresponds to a linear combination of one or more columns of $\mathbf{G}$ and zero or more columns of $\mathbf{K}$, which is nonzero and not in the column space of $\mathbf{K}$. □

The decoding process at sink t is as follows. First, the sink determines z, the minimum cut from the adversarial packets to the sink. This is with high probability equal to $w - r$, the difference between the number of linearly independent packets received by the sink and the number of source packets. Next, it chooses z columns of $\mathbf{Y}$ that, together with the columns of $\mathbf{G}'$, form a basis for the column space of $\mathbf{Y}$. We assume without loss of generality that the first z columns are chosen, and we denote the corresponding submatrix $\mathbf{G}''$. Matrix $\mathbf{Y}$, rewritten in the basis corresponding to the matrix $[\mathbf{G}'' \; \mathbf{G}']$, takes the form

$$\begin{aligned} \mathbf{Y} &= [\mathbf{G}'' \; \mathbf{G}'] \begin{bmatrix} \mathbf{I}_z & \mathbf{Y}^Z & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}^X & \mathbf{I}_r \end{bmatrix} \\ &= \mathbf{G}'' \begin{bmatrix} \mathbf{I}_z & \mathbf{Y}^Z & \mathbf{0} \end{bmatrix} + \mathbf{G}' \begin{bmatrix} \mathbf{0} & \mathbf{Y}^X & \mathbf{I}_r \end{bmatrix} \end{aligned} \tag{6.43}$$

where $\mathbf{Y}^Z, \mathbf{Y}^X$ are $z \times (n-z-r)$ and $r \times (n-z-r)$ matrices respectively.

Let $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3$ be the submatrices of $\mathbf{X}$ consisting of its first z columns, the next $n - z - r$ columns of $\mathbf{X}$, and the last r columns respectively.

Lemma 6.3

$$\mathbf{G}'\mathbf{X}_2 = \mathbf{G}'(\mathbf{Y}^X + \mathbf{X}_1\mathbf{Y}^Z) \tag{6.44}$$

Proof Equating (6.41) and (6.43), we have

$$\mathbf{G}'\mathbf{X} + \mathbf{K}(\mathbf{Z} - \mathbf{L}\mathbf{X}) = \mathbf{G}'' \begin{bmatrix} \mathbf{I}_z & \mathbf{Y}^Z & \mathbf{0} \end{bmatrix} + \mathbf{G}' \begin{bmatrix} \mathbf{0} & \mathbf{Y}^X & \mathbf{I}_r \end{bmatrix}. \tag{6.45}$$

† though some of the additional adversarial source packets might not be decodable if the minimum cut between them and t is less than z_o

The column space of $\mathbf{G}''$ is spanned by the column space of $[\mathbf{G}'\,\mathbf{K}]$, so we can rewrite the equation in the form

$$\mathbf{G}'\mathbf{X}+\mathbf{K}(\mathbf{Z}-\mathbf{L}\mathbf{X}) = (\mathbf{G}'\mathbf{M}_1+\mathbf{K}\mathbf{M}_2)\begin{bmatrix}\mathbf{I}_z & \mathbf{Y}^Z & \mathbf{0}\end{bmatrix}+\mathbf{G}'\begin{bmatrix}\mathbf{0} & \mathbf{Y}^X & \mathbf{I}_r\end{bmatrix}. \tag{6.46}$$

From Lemma 6.2, the column spaces of $\mathbf{G}'$ and $\mathbf{K}$ are disjoint except in the zero vector, so we can equate the terms involving $\mathbf{G}'$:

$$\mathbf{G}'\begin{bmatrix}\mathbf{X}_1 & \mathbf{X}_2 & \mathbf{X}_3\end{bmatrix} = \mathbf{G}'\mathbf{M}_1\begin{bmatrix}\mathbf{I}_z & \mathbf{Y}^Z & \mathbf{0}\end{bmatrix} + \mathbf{G}'\begin{bmatrix}\mathbf{0} & \mathbf{Y}^X & \mathbf{I}_r\end{bmatrix}. \tag{6.47}$$

The leftmost z columns of the matrix equation give $\mathbf{X}_1 = \mathbf{M}_1$. Substituting this into the next $n-z-r$ columns, we obtain (6.44). □

Lemma 6.4 *With probability approaching 1 in q, the system of equations (6.38) and (6.44) can be solved simultaneously to recover $\mathbf{X}$.*

Proof We rewrite the system of equations with the matrices $\mathbf{X}_1, \mathbf{X}_2, \mathbf{Y}^X, \mathbf{I}$ expressed in column vector form† $\mathbf{v}_{\mathbf{X}_1}^T, \mathbf{v}_{\mathbf{X}_2}^T, \mathbf{v}_{\mathbf{Y}^X}^T, \mathbf{v}_{\mathbf{I}}^T$.
Let $\mathbf{D} = [\mathbf{D}_1\ \mathbf{D}_2\ \mathbf{D}_3]$, where $\mathbf{D}_1$ comprises the first rz columns of $\mathbf{D}$, $\mathbf{D}_2$ the next $r(n-r-z)$ columns and $\mathbf{D}_3$ the remaining r^2 columns of $\mathbf{D}$. Let

$$\alpha = n - r - z \tag{6.48}$$

and let $y_{i,j}$ denote the (i,j)th entry of matrix $\mathbf{Y}^Z$. We can write the system of equations (6.38) and (6.44) in block matrix form as follows:

$$\mathbf{A}\begin{pmatrix}\mathbf{v}_{\mathbf{X}_1}^T \\ \mathbf{v}_{\mathbf{X}_2}^T\end{pmatrix} = \begin{pmatrix}\mathbf{G}'\mathbf{v}_{\mathbf{Y}^X}^T \\ -\mathbf{D}_3\mathbf{v}_{\mathbf{I}}^T\end{pmatrix} \tag{6.49}$$

where $\mathbf{A}$ is given by

$$\left[\begin{array}{cccc|ccccc}
-y_{1,1}\mathbf{G}' & -y_{2,1}\mathbf{G}' & \dots & -y_{z,1}\mathbf{G}' & \mathbf{G}' & 0 & \dots & \dots & 0 \\
-y_{1,2}\mathbf{G}' & -y_{2,2}\mathbf{G}' & \dots & -y_{z,2}\mathbf{G}' & 0 & \mathbf{G}' & 0 & \dots & 0 \\
-y_{1,3}\mathbf{G}' & -y_{2,3}\mathbf{G}' & \dots & -y_{z,3}\mathbf{G}' & \vdots & 0 & \mathbf{G}' & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & 0 & \ddots & 0 \\
-y_{1,\alpha}\mathbf{G}' & -y_{2,\alpha}\mathbf{G}' & \dots & -y_{z,\alpha}\mathbf{G}' & 0 & 0 & 0 & 0 & \mathbf{G}' \\
\hline
\multicolumn{4}{c|}{\mathbf{D}_1} & \multicolumn{5}{c}{\mathbf{D}_2}
\end{array}\right].$$

† Recall that for any matrix $\mathbf{M}$, $\mathbf{v}_{\mathbf{M}}^T$ denotes the column vector obtained by stacking the columns of $\mathbf{M}$ one above the other.

For $j = 1, \ldots, \alpha$, the jth row of matrices in $\mathbf{A}$ corresponds to the jth column of (6.44), equivalently written as

$$-\mathbf{G}'\mathbf{X}_1\mathbf{Y}^Z + \mathbf{G}'\mathbf{X}_2 = \mathbf{G}'\mathbf{Y}^X.$$

The bottom submatrix $[\mathbf{D}_1\ \mathbf{D}_2]$ of $\mathbf{A}$ corresponds to (6.38). We will show that with probability approaching 1 in q, $\mathbf{A}$ has full column rank (i.e. the columns of $\mathbf{A}$ are linearly independent) which allows (6.49) to be solved.

By Lemma 6.2, with probability approaching 1 in q, the columns of matrix $\mathbf{G}'$, and thus the rightmost αr columns of $\mathbf{A}$, are linearly independent. The upper left submatrix of $\mathbf{A}$ can be zeroed out by column operations involving the right submatrix (rightmost αr columns) of $\mathbf{A}$. The original matrix $\mathbf{A}$ has full column rank iff the matrix resulting from these column operations (or, equivalently, its lower left $r\rho n \times rz$ submatrix, denoted $\mathbf{B}$) has full column rank. Let $\mathbf{d}_k^T, k = 1, \ldots, rz$, denote the kth column of $\mathbf{D}_1$. Consider any fixed value of $\mathbf{Y}^Z$ (whose entries are the $y_{i,j}$ variables), and denote by $\mathbf{b}_k^T, k = 1, \ldots, rz$, the kth column of $\mathbf{B}$. $\mathbf{b}_k^T$ is equal to the sum of $\mathbf{d}_k^T$ and a linear combination, determined by the values of the $y_{i,j}$ variables, of the columns of $\mathbf{D}_2$. Since the entries of $\mathbf{D}_1$ are independently and uniformly distributed in $\mathbb{F}_q$, so are the entries of $\mathbf{B}$. The probability that $\mathbf{B}$ does not have full column rank is $1 - \prod_{l=1}^{rz} \left(1 - 1/q^{r\rho n - l + 1}\right)$, which is upper bounded by $q^{rz - r\rho n}$ for sufficiently large q. Using a union bound over the $q^{\alpha z}$ possible values of $\mathbf{Y}^Z$, the probability that $\mathbf{B}$ does not have full column rank for one or more values of $\mathbf{Y}^Z$ is upper bounded by

$$\begin{aligned} q^{rz - r\rho n + \alpha z} &= q^{rz - n(z_o + \epsilon) + (n - r - z)z} \qquad (6.50) \\ &< q^{-n\epsilon} \end{aligned}$$

where (6.50) follows from (6.37) and (6.48). □

The decoding algorithm's most complex step is solving the system of equations (6.38) and (6.44), or equivalently the matrix equation (6.49) which has dimension $O(nm)$. Thus, the decoding algorithm has complexity $O(n^3 m^3)$.

If instead of an omniscient adversary we assume that the adversary observes only a limited number of packets, or that the source and sinks share a secret channel, then it is possible to achieve a higher communication rate of $m - z_o$. Such non-omniscient adversary models are analyzed in [71, 69].

6.2.2.2 Vector space approach

The vector space approach of Section 2.5.1.2 can be applied directly to the problem of network error and erasure correction.

Distributed random linear network coding on an unknown network is modeled as an *operator channel*, defined as follows.

Definition 6.2 *An operator channel associated with ambient space W is a channel with input and output alphabets equal to the set $\mathcal{P}(W)$ of all subspaces of W. The input V and output U of the channel are related by*

$$U = g_k(V) \oplus E$$

where g_k is an erasure operator that projects V onto a random k-dimensional subspace of V, and $E \in \mathcal{P}(W)$ is the error vector space.

This casts the problem as a point-to-point channel coding problem.

This channel coding formulation admits a Reed-Solomon like code construction, where the basis for the transmitted vector space is obtained by evaluating a linearized message polynomial. We consider $\mathbb{F} = \mathbb{F}_{2^m}$ as a vector space of dimension m over $\mathbb{F}_q$. Let $\mathbf{u} = (u_0, u_1, \ldots, u_{j-1}) \in \mathbb{F}^j$ be the source symbols, and

$$f(x) := \sum_{i=0}^{j-1} u_i x^{q^i}$$

the corresponding linearized polynomial. Let $A = \{\alpha_1, \ldots, \alpha_l\}$ be a set of $l \geq j$ linearly independent elements of $\mathbb{F}$ spanning an l-dimensional vector space $\langle A \rangle$ over $\mathbb{F}_q$. The ambient space W is given by $\{(\alpha, \beta) : \alpha \in \langle A \rangle, \beta \in \mathbb{F}\}$ which is regarded as a $(l+m)$-dimensional vector space over $\mathbb{F}_q$. The vector space V transmitted by the source is then the span of the set

$$\{(\alpha_1, f(\alpha_1)), \ldots, (\alpha_l, f(\alpha_l))\}.$$

In [83] is shown that these Reed-Solomon like codes are nearly Singleton Bound-achieving, and admit an efficient decoding algorithm.

6.3 Detection of adversarial errors

In this section we consider information theoretic detection of errors introduced by an adversary who knows the entire message and coding

strategy except for some of the random coding coefficients. Such a situation may arise, for instance, if an adversary compromises a sink node that was originally an intended recipient of the source message.

Suppose the adversary sends z erroneous adversarial packets. Each sink receives packets that are random linear combinations of these $r + z$ packets. In the error correction case, the minimum overhead (i.e. proportion of redundant information) depends on the number of arcs/transmissions controlled by the adversary as a proportion of the source-sink minimum cut. In the error detection case, there is no minimum overhead; it can be traded off flexibly against the detection probability and coding field size. An error detection scheme can be used for low overhead monitoring during normal conditions when no adversary is known to be present, in conjunction with a higher overhead error correction scheme activated upon detection of an adversarial error.

Error detection capability is added to random linear coding by including a flexible number of hash symbols in each packet. With this approach, a sink node can detect adversarial modifications with high probability. The only condition needed, which we will make precise below, is the adversary's incomplete knowledge of the random network code when designing its packets. The adversary can have the same (or greater) transmission capacity compared to the source, even to the extent where every packet received by a sink is corrupted with an independent adversarial packet.

6.3.1 Model and problem formulation

Each packet p in the network is represented by a row vector $\mathbf{w}_p$ of $d + c + r$ symbols from a finite field $\mathbb{F}_q$, where the first d entries are data symbols, the next c are redundant hash symbols and the last r form the packet's (global) coefficient vector $\mathbf{t}_p$. The hash symbols in each exogenous packet are given by a function $\psi_d : \mathbb{F}_q^d \to \mathbb{F}_q^c$ of the data symbols. The coding vector of the ith exogenous packet is the unit vector with a single nonzero entry in the ith position.

Let row vector $\mathbf{m}_i \in \mathbb{F}_q^{(c+d)}$ represent the concatenation of the data and hash symbols for the ith exogenous packet, and let $\mathbf{M}$ be the matrix whose ith row is $\mathbf{m}_i$. A packet p is *genuine* if its data/hash symbols are consistent with its coding vector, i.e. $\mathbf{w}_p = [\mathbf{t}_p\mathbf{M}, \mathbf{t}_p]$. The exogenous packets are genuine, and any packet formed as a linear combination of genuine packets is also genuine. *Adversarial packets*, i.e. packets transmitted by the adversary, may contain arbitrary coding vector and

data/hash values. An adversarial packet p can be represented in general by $[\mathbf{t}_p\mathbf{M} + \mathbf{v}_p, \mathbf{t}_p]$, where $\mathbf{v}_p$ is an arbitrary vector $\mathbb{F}_q^{c+d}$. If $\mathbf{v}_p$ is nonzero, p (and linear combinations of p with genuine packets) are non-genuine.

A set $\mathcal{S}$ of packets can be represented as a block matrix $[\mathbf{T}_\mathcal{S}\mathbf{M} + \mathbf{V}_\mathcal{S}|\mathbf{T}_\mathcal{S}]$ whose ith row is $\mathbf{w}_{p_i}$ where p_i is the ith packet of the set. A sink node t attempts to decode when it has collected a *decoding set* consisting of r linearly independent packets (i.e. packets whose coding vectors are linearly independent). For a decoding set $\mathcal{D}$, the decoding process is equivalent to pre-multiplying the matrix $[\mathbf{T}_\mathcal{D}\mathbf{M} + \mathbf{V}_\mathcal{D}|\mathbf{T}_\mathcal{D}]$ with $\mathbf{T}_\mathcal{D}^{-1}$. This gives $\left[\mathbf{M} + \mathbf{T}_\mathcal{D}^{-1}\mathbf{V}_\mathcal{D}|\mathbf{I}\right]$, i.e. the receiver decodes to $\mathbf{M} + \tilde{\mathbf{M}}$, where

$$\tilde{\mathbf{M}} = \mathbf{T}_\mathcal{D}^{-1}\mathbf{V}_\mathcal{D} \tag{6.51}$$

gives the disparity between the decoded packets and the original packets. If at least one packet in a decoding set is non-genuine, $\mathbf{V}_\mathcal{D} \neq \mathbf{0}$, and the decoded packets will differ from the original packets. A decoded packet is *inconsistent* if its data and hash values do not match, i.e. applying the function ψ_d to its data values does not yield its hash values. If one or more decoded packets are inconsistent, the sink declares an error.

The coding vector of a packet transmitted by the source is uniformly distributed over $\mathbb{F}_q^r$; if a packet whose coding vector has this uniform distribution is linearly combined with other packets, the resulting packet's coding vector has the same uniform distribution. We are concerned with the distribution of decoding outcomes conditioned on the adversary's information, i.e. the adversary's observed and transmitted packets, and its information on independencies/dependencies among packets. Note that in this setup, scaling a packet by some scalar element of $\mathbb{F}_q$ does not change the distribution of decoding outcomes.

For given $\mathbf{M}$, the value of a packet p is specified by the row vector $\mathbf{u}_p = [\mathbf{t}_p, \mathbf{v}_p]$. We call a packet p *secret* if, conditioned on the value of $\mathbf{v}_p$ and the adversary's information, its coding vector $\mathbf{t}_p$ is uniformly distributed over $\mathbb{F}_q^r \backslash W$ for some (possibly empty) subspace or affine space $W \subset \mathbb{F}_q^r$. Intuitively, secret packets include genuine packets whose coding vectors are unknown (in the above sense) to the adversary, as well as packets formed as linear combinations involving at least one secret packet. A set $\mathcal{S}$ of secret packets is *secrecy-independent* if each of the packets remains secret when the adversary is allowed to observe the other packets in the set; otherwise it is *secrecy-dependent*. Secrecy-dependencies arise from the network transmission topology, for instance, if a packet p is formed as a linear combination of a set $\mathcal{S}$ of secret packets (possibly with other non-secret packets), then $\mathcal{S} \cup \{p\}$ is secrecy-

dependent. To illustrate these definitions, suppose that the adversary knows that a sink's decoding set contains an adversarial packet p_1 and a packet p_4 formed as a linear combination of a non-genuine adversarial packet p_2 with a genuine packet p_3, and suppose that the adversary does not observe any packets dependent on p_3. Since a decoding set consists of packets with linearly independent coding vectors, the distribution of $\mathbf{t}_{p_4}$, conditioned on the adversary's information and any potential value $k_2\mathbf{v}_{p_2}$ for $\mathbf{v}_{p_4}$, is uniform over $\mathbb{F}_q^r \backslash \{k\mathbf{t}_{p_1} - k_2\mathbf{t}_{p_2} : k \in \mathbb{F}_q\}$. Also, packets p_3 and p_4 are secrecy-dependent.

Consider a decoding set $\mathcal{D}$ containing one or more secret packets. Choosing an appropriate packet ordering, we can express $[\mathbf{T}_\mathcal{D}|\mathbf{V}_\mathcal{D}]$ in the form

$$[\mathbf{T}_\mathcal{D}|\mathbf{V}_\mathcal{D}] = \left[\begin{array}{c|c} \mathbf{A} + \mathbf{B}_1 & \mathbf{V}_1 \\ \hline \mathbf{NA} + \mathbf{B}_2 & \mathbf{V}_2 \\ \hline \mathbf{B}_3 & \mathbf{V}_3 \end{array}\right] \tag{6.52}$$

where for any given values of $\mathbf{B}_i \in \mathbb{F}_q^{s_i \times r}$, $\mathbf{V}_i \in \mathbb{F}_q^{s_i \times (d+c)}$, $i = 1, 2, 3$, and $\mathbf{N} \in \mathbb{F}_q^{s_2 \times s_1}$, the matrix $\mathbf{A} \in \mathbb{F}_q^{s_1 \times r}$ has a conditional distribution that is uniform over all values for which $\mathbf{T}_\mathcal{D}$ is nonsingular. The first $s_1 + s_2$ rows correspond to secret packets, and the first s_1 rows correspond to a set of secrecy-independent packets. $s_2 = 0$ if there are no secrecy-dependencies among the secret packets in $\mathcal{D}$.

6.3.2 Detection probability

In the following we consider decoding from a set of packets that contains some non-genuine packet, which causes the decoded packets to differ from the original exogenous packets. The first part of the theorem gives a lower bound on the number of equally likely potential values of the decoded packets– the adversary cannot narrow down the set of possible outcomes beyond this regardless of how it designs its adversarial packets. The second part provides, for a simple polynomial hash function, an upper bound on the proportion of potential decoding outcomes that can have consistent data and hash values, in terms of $k = \lceil \frac{d}{c} \rceil$, the ceiling of the ratio of the number of data symbols to hash symbols. Larger values for k correspond to lower overheads but lower probability of detecting an adversarial modification. This tradeoff is a design parameter for the network.

Theorem 6.5 *Consider a decoding set $\mathcal{D}$ containing a secrecy-independent subset of s_1 secret (possibly non-genuine) packets, and suppose the decoding set contains at least one non-genuine packet.*
a) The adversary cannot determine which of a set of at least $(q-1)^{s_1}$ equally likely values of the decoded packets will be obtained at the sink. In particular, there will be at least s_1 packets such that, for each of these, the adversary cannot determine which of a set of at least $q-1$ equally likely values will be obtained.
b) Let $\psi : \mathbb{F}_q^k \to \mathbb{F}_q$ be the function mapping $(x_1, \ldots, x_k)$, $x_i \in \mathbb{F}_q$, to

$$\psi(x_1, \ldots, x_k) = x_1^2 + \cdots + x_k^{k+1} \tag{6.53}$$

where $k = \left\lceil \frac{d}{c} \right\rceil$. Suppose the function ψ_d mapping the data symbols $x_1, \ldots, x_d$ to the hash symbols $y_1, \ldots, y_c$ in an exogenous packet is defined by

$$\begin{aligned} y_i &= \psi(x_{(i-1)k+1}, \ldots, x_{ik}) \ \ \forall\, i = 1, \ldots, c-1 \\ y_c &= \psi(x_{(c-1)k+1}, \ldots, x_d) \end{aligned}$$

Then the probability of not detecting an error is at most $\left(\frac{k+1}{q}\right)^{s_1}$.

Proof See Appendix 6.A. □

Corollary 6.1 *Let the hash function ψ_d be defined as in Theorem 6.5b. Suppose a sink obtains more than r packets, including a secrecy-independent set of s secret packets, and at least one non-genuine packet. If the sink decodes using two or more decoding sets whose union includes all its received packets, then the probability of not detecting an error is at most $\left(\frac{k+1}{q}\right)^{s}$.*

Example: With 2% overhead ($k = 50$), code length=7, $s = 5$, the detection probability is at least 98.9%; with 1% overhead ($k = 100$), code length=8, $s = 5$, the detection probability is at least 99.0%.

While this approach works under relatively mild assumptions as described above, it fails in the case where the adversary knows that the genuine packets received at a sink have coding vectors that lie in some w-dimensional subspace $W \subset \mathbb{F}_q^r$, the following strategy allows it to control the decoding outcome and so ensure that the decoded packets have consistent data and hash values.

The adversary ensures that the sink receives w genuine packets with linearly independent coefficient vectors in W, by supplying additional

such packets if necessary. The adversary also supplies the sink with $r-w$ non-genuine packets whose coding vectors $\mathbf{t}_1, \ldots, \mathbf{t}_{r-w}$ are not in W. Let $\mathbf{t}_{r-w+1}, \ldots, \mathbf{t}_r$ be a set of basis vectors for W, and let $\mathbf{T}$ be the matrix whose ith row is $\mathbf{t}_i$. Then the coding vectors of the r packets can be represented by the rows of the matrix

$$\left[\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{K} \end{array}\right] \mathbf{T}$$

where $\mathbf{K}$ is a nonsingular matrix in $\mathbb{F}_q^{w\times w}$. From (6.54), we have

$$\left[\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{K} \end{array}\right] \mathbf{T}\tilde{\mathbf{M}} = \left[\begin{array}{c} \tilde{\mathbf{V}} \\ \hline \mathbf{0} \end{array}\right]$$

$$\begin{aligned} \tilde{\mathbf{M}} &= \mathbf{T}^{-1} \left[\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{K}^{-1} \end{array}\right] \left[\begin{array}{c} \tilde{\mathbf{V}} \\ \hline \mathbf{0} \end{array}\right] \\ &= \mathbf{T}^{-1} \left[\begin{array}{c} \tilde{\mathbf{V}} \\ \hline \mathbf{0} \end{array}\right] \end{aligned}$$

Since the adversary knows $\mathbf{T}$ and controls $\tilde{\mathbf{V}}$, it can determine $\tilde{\mathbf{M}}$.

6.4 Notes and further reading

Yeung and Cai were the first to study error correction in network coding. They developed the bounds on centralized network error correction presented in this chapter [148, 20]. Low-complexity methods for detection and correction of adversarial errors in distributed random network coding were given in Ho et al. [63] and Jaggi et al. [70, 71] respectively. The vector space approach for correction of errors and erasures in distributed random network coding was developed by Koetter and Kschischang [83].

In other related work on network coding security not covered in this section, Charles et al. [22] and Zhao et al. [153] have developed signature schemes for multicast network coding. The problem of ensuring secrecy for multicast network coding in the presence of a wire tap adversary has been considered in [19, 44, 16, 130].

6.A Appendix: Proof of results for adversarial error detection

We first establish two results that are used in the proof of Theorem 6.5. Consider the hash function defined in (6.53). We call a vector $(x_1, \ldots, x_{k+1}) \in \mathbb{F}_q^{k+1}$ *consistent* if $x_{k+1} = \psi(x_1, \ldots, x_k)$.

Lemma 6.5 *At most $k+1$ out of the q vectors in a set*

$$\{\mathbf{u} + \gamma\mathbf{v} : \gamma \in \mathbb{F}_q\},$$

where $\mathbf{u} = (u_1, \ldots, u_{k+1})$ is a fixed vector in $\mathbb{F}_q^{k+1}$ and $\mathbf{v} = (v_1, \ldots, v_{k+1})$ is a fixed nonzero vector in $\mathbb{F}_q^{k+1}$, can be consistent.

Proof Suppose some vector $\mathbf{u} + \gamma\mathbf{v}$ is consistent, i.e.

$$u_{k+1} + \gamma v_{k+1} = (u_1 + \gamma v_1)^2 + \cdots + (u_k + \gamma v_k)^{k+1}$$

Note that for any fixed value of $\mathbf{u}$ and any fixed nonzero value of $\mathbf{v}$, (6.54) is a polynomial equation in γ of degree equal to $1 + \tilde{k}$, where $\tilde{k} \in [1, k]$ is the highest index for which the corresponding $v_{k'}$ is nonzero, i.e. $v_{\tilde{k}} \neq 0, v_{k'} = 0 \; \forall \; k' > \tilde{k}$. By the fundamental theorem of algebra, this equation can have at most $1 + \tilde{k} \leq 1 + k$ roots. Thus, the property can be satisfied for at most $1 + k$ values of γ. □

Corollary 6.2 *Let $\mathbf{u}$ be a fixed row vector in $\mathbb{F}_q^n$ and $\mathbf{Y}$ a fixed nonzero matrix in $\mathbb{F}_q^{n\times(k+1)}$. If row vector $\mathbf{g}$ is distributed uniformly over $\mathbb{F}_q^n$, then the vector $\mathbf{u} + \mathbf{g}\mathbf{Y}$ is consistent with probability at most $\frac{k+1}{q}$.*

Proof Suppose the ith row of $\mathbf{Y}$, denoted $\mathbf{y}_i$, is nonzero. We can partition the set of possible values for $\mathbf{g}$ such that each partition consists of all vectors that differ only in the ith entry g_i. For each partition, the corresponding set of values of $\mathbf{u} + \mathbf{g}\mathbf{Y}$ is of the form $\{\mathbf{u}' + g_i\mathbf{y}_i : g_i \in \mathbb{F}_q\}$. The result follows from Lemma 6.5 and the fact that g_i is uniformly distributed over $\mathbb{F}_q$. □

Proof of Theorem 6.5: Writing $\mathbf{A}' = \mathbf{A} + \mathbf{B}_1$, $\mathbf{T}_{\mathcal{D}}$ becomes

$$\begin{bmatrix} \mathbf{A}' \\ \mathbf{N}(\mathbf{A}' - \mathbf{B}_1) + \mathbf{B}_2 \\ \mathbf{B}_3 \end{bmatrix}$$

From (6.51), we have

$$\begin{bmatrix} \mathbf{A}' \\ \mathbf{N}(\mathbf{A}' - \mathbf{B}_1) + \mathbf{B}_2 \\ \mathbf{B}_3 \end{bmatrix} \tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{A}' \\ -\mathbf{N}\mathbf{B}_1 + \mathbf{B}_2 \\ \mathbf{B}_3 \end{bmatrix} \tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 - \mathbf{N}\mathbf{V}_1 \\ \mathbf{V}_3 \end{bmatrix}$$

which we can simplify to

$$\begin{bmatrix} \mathbf{A}' \\ \mathbf{B}' \end{bmatrix} \tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2' \end{bmatrix} \tag{6.54}$$

by writing

$$\mathbf{B}' = \begin{bmatrix} -\mathbf{N}\mathbf{B}_1 + \mathbf{B}_2 \\ \mathbf{B}_3 \end{bmatrix}, \quad \mathbf{V}_2' = \begin{bmatrix} \mathbf{V}_2 - \mathbf{N}\mathbf{V}_1 \\ \mathbf{V}_3 \end{bmatrix}$$

Since

$$\begin{bmatrix} \mathbf{A}' \\ \mathbf{B}' \end{bmatrix} \text{ is nonsingular} \Leftrightarrow \mathbf{T}_{\mathcal{D}} \text{ is nonsingular},$$

for given values of $\mathbf{B}', \mathbf{V}_1$ and $\mathbf{V}_2'$, matrix $\mathbf{A}' \in \mathbb{F}_q^{s_1 \times r}$ has a conditional distribution that is uniform over the set $\mathcal{A}$ of values for which $\begin{bmatrix} \mathbf{A}' \\ \mathbf{B}' \end{bmatrix}$ is nonsingular.

The condition that the decoding set contains at least one non-genuine packet corresponds to the condition $\mathbf{V}_{\mathcal{D}} \neq \mathbf{0}$. We consider two cases. In each case we show that we can partition the set $\mathcal{A}$ such that at most a fraction $\left(\frac{k+1}{q}\right)^{s_1}$ of values in each partition give decoding outcomes $\mathbf{M}+\tilde{\mathbf{M}}$ with consistent data and hash values. The result then follows since the conditional distribution of values within each partition is uniform.

Case 1: $\mathbf{V}_2' \neq \mathbf{0}$. Let $\mathbf{v}_i$ be some nonzero row of $\mathbf{V}_2'$, and $\mathbf{b}_i$ the corresponding row of $\mathbf{B}'$. Then $\mathbf{b}_i\tilde{\mathbf{M}} = \mathbf{v}_i$.

We first partition $\mathcal{A}$ into cosets

$$\mathcal{A}_n = \{\mathbf{A}_n + \mathbf{r}^T\mathbf{b}_i : \mathbf{r} \in \mathbb{F}_q^{s_1}\}, n = 1, 2, \ldots, \chi$$

where

$$\chi = \frac{|\mathcal{A}|}{q^{s_1}}$$

This can be done by the following procedure. Any element of $\mathcal{A}$ can be chosen as $\mathbf{A}_1$. Matrices $\mathbf{A}_2, \mathbf{A}_3, \ldots, \mathbf{A}_\chi$ are chosen sequentially; for each $j = 2, \ldots, \chi$, $\mathbf{A}_j$ is chosen to be any element of $\mathcal{A}$ not in the cosets $\mathcal{A}_n, n < j$. Note that this forms a partition of $\mathcal{A}$, since the presence of some element c in two sets $\mathcal{A}_n$ and $\mathcal{A}_j$, $n < j$, implies that $\mathbf{A}_j$ is also in $\mathcal{A}_n$, which is a contradiction. It is also clear that each coset has size $\left|\{\mathbf{r} : \mathbf{r} \in \mathbb{F}_q^{s_1}\}\right| = q^{s_1}$.

For each such coset $\mathcal{A}_n$, the corresponding values of $\tilde{\mathbf{M}}$ satisfy, from (6.54),

$$\begin{bmatrix} \mathbf{A}_n + \mathbf{r}^T\mathbf{b}_i \\ \mathbf{B}' \end{bmatrix} \tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2' \end{bmatrix}$$
$$\begin{bmatrix} \mathbf{A}_n \\ \mathbf{B}' \end{bmatrix} \tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{V}_1 - \mathbf{r}^T\mathbf{v}_i \\ \mathbf{V}_2' \end{bmatrix}$$
$$\tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{A}_n \\ \mathbf{B}' \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{V}_1 - \mathbf{r}^T\mathbf{v}_i \\ \mathbf{V}_2' \end{bmatrix}$$

Let $\mathbf{U}$ be the submatrix consisting of the first s_1 columns of $\begin{bmatrix} \mathbf{A}_n \\ \mathbf{B}' \end{bmatrix}^{-1}$. Since $\mathbf{U}$ is nonsingular, we can find a set $\mathcal{J} \subset \{1, \ldots, r\}$ of s_1 indexes that correspond to independent rows of $\mathbf{U}$. Consider sequentially the corresponding rows of $\mathbf{M} + \tilde{\mathbf{M}}$. The set of potential values for each of these s_1 rows, for any given value of the previously considered rows, is or can be partitioned into sets of the form $\{\mathbf{u} + \gamma\mathbf{v}_i : \ \gamma \in \mathbb{F}_q\}$. Applying Lemma 6.5 yields the result for this case.

Case 2: $\mathbf{V}_2' = \mathbf{0}$, i.e. $\mathbf{V}_2 - \mathbf{N}\mathbf{V}_1 = \mathbf{V}_3 = \mathbf{0}$. Then $\mathbf{V}_1 \neq \mathbf{0}$, since otherwise $\mathbf{V}_1 = \mathbf{V}_2 = \mathbf{0}$ and $\mathbf{V}_{\mathcal{D}} = \mathbf{0}$ which would contradict the assumption that there is at least one non-genuine packet.

We partition $\mathcal{A}$ such that each partition consists of all matrices in $\mathcal{A}$ that have the same row space:

$$\mathcal{A}_n = \left\{\mathbf{R}\mathbf{A}_n : \ \mathbf{R} \in \mathbb{F}_q^{s_1 \times s_1}, \ \det(\mathbf{R}) \neq 0\right\}, \ n = 1, 2, \ldots, \chi$$

where

$$|\mathcal{A}_n| = \prod_{i=0}^{s_1 - 1} \left(q^{s_1} - q^i\right), \ \chi = \frac{|\mathcal{A}|}{|\mathcal{A}_n|}$$

This can be done by choosing any element of $\mathcal{A}$ as $\mathbf{A}_1$, and choosing $\mathbf{A}_n, n = 2, \ldots, \chi$ sequentially such that $\mathbf{A}_n$ is any element of $\mathcal{A}$ not in $\mathcal{A}_j, j < n$.

For each $\mathcal{A}_n, n = 1, \ldots, \chi$, the corresponding values of $\tilde{\mathbf{M}}$ satisfy, from (6.54),

$$\begin{bmatrix} \mathbf{R}\mathbf{A}_n \\ \mathbf{B}' \end{bmatrix} \tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{0} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{A}_n \\ \mathbf{B}' \end{bmatrix} \tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{R}^{-1}\mathbf{V}_1 \\ \mathbf{0} \end{bmatrix}$$

$$\tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{A}_n \\ \mathbf{B}' \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{R}^{-1}\mathbf{V}_1 \\ \mathbf{0} \end{bmatrix}$$

Let $\mathbf{U}$ be the submatrix consisting of the first s_1 columns of $\begin{bmatrix} \mathbf{A}_n \\ \mathbf{B}' \end{bmatrix}^{-1}$. We can find an ordered set $\mathcal{J} = \{i_1, \ldots, i_{s_1} : \ i_1 < \cdots < i_{s_1}\} \subset \{1, \ldots, r\}$ of s_1 indexes that correspond to linearly independent rows of $\mathbf{U}$. Let $\mathbf{U}_\mathcal{J}$ and $\mathbf{M}_\mathcal{J}$ be the submatrices of $\mathbf{U}$ and $\mathbf{M}$ respectively consisting of the s_1 rows corresponding to $\mathcal{J}$. Then $\mathbf{U}_\mathcal{J}$ is nonsingular, and the value of the matrix representation of the corresponding decoded packets is uniformly distributed over the set

$$\left\{\mathbf{M}_\mathcal{J} + \mathbf{R}'\mathbf{V}_1 : \ \mathbf{R}' \in \mathbb{F}_q^{s_1 \times s_1}, \ \det(\mathbf{R}') \neq 0\right\} \tag{6.55}$$

Let ν be the rank of $\mathbf{V}_1$. Consider a set of ν independent rows of $\mathbf{V}_1$. Denote by $\mathcal{I}$ the corresponding set of row indexes, and denote by $\mathbf{V}_\mathcal{I}$ the submatrix of $\mathbf{V}_1$ consisting of those rows. We can write

$$\mathbf{V}_1 = \mathbf{L}\mathbf{V}_\mathcal{I}$$

where $\mathbf{L} \in \mathbb{F}_q^{s_1 \times \nu}$ has full rank ν. We define $\mathbf{R}_\mathcal{I} = \mathbf{R}'\mathbf{L}$, noting that

$$\mathbf{R}_\mathcal{I}\mathbf{V}_\mathcal{I} = \mathbf{R}'\mathbf{L}\mathbf{V}_\mathcal{I} = \mathbf{R}'\mathbf{V}_1$$

and that $\mathbf{R}_\mathcal{I}$ is uniformly distributed over all matrices in $\mathbb{F}_q^{s_1 \times \nu}$ that have full rank ν. Thus, (6.55) becomes

$$\left\{\mathbf{M}_\mathcal{J} + \mathbf{R}_\mathcal{I}\mathbf{V}_\mathcal{I} : \ \mathbf{R}_\mathcal{I} \in \mathbb{F}_q^{s_1 \times \nu}, \ \mathrm{rank}(\mathbf{R}_\mathcal{I}) = \nu\right\} \tag{6.56}$$

Denote by $\mathbf{r}_1, \ldots, \mathbf{r}_{s_1}$ the rows of $\mathbf{R}_\mathcal{I}$, and by $\mathbf{R}_n$ the submatrix of $\mathbf{R}_\mathcal{I}$ consisting of its first n rows. We consider the rows sequentially, starting with the first row $\mathbf{r}_1$. For $n = 1, \ldots, s_1$, we will show that conditioned on any given value of $\mathbf{R}_{n-1}$, the probability that the i_nth decoded packet $\mathbf{M}_{i_n} + \mathbf{r}_n\mathbf{V}_\mathcal{I}$ is consistent is at most $\frac{k+1}{q}$. Note that the conditional distribution of $\mathbf{r}_n$ is the same for any values of $\mathbf{R}_{n-1}$ that have the same rank.

Case A: $\mathbf{R}_{n-1}$ has zero rank. This is the case if $n = 1$, or if $n > 1$ and $\mathbf{R}_{n-1} = \mathbf{0}$.

Suppose we remove the restriction $\text{rank}(\mathbf{R}_{\mathcal{I}}) = \nu$, so that $\mathbf{r}_n$ is uniformly distributed over $\mathbb{F}_q^{\nu}$. By Corollary 6.2, $\mathbf{m}_{i_n} + \mathbf{r}_n\mathbf{V}_{\mathcal{I}}$ would have consistent data and hash values with probability at most $\frac{k+1}{q}$. With the restriction $\text{rank}(\mathbf{R}_{\mathcal{I}}) = \nu$, the probability of $\mathbf{r}_n$ being equal to $\mathbf{0}$ is lowered. Since the corresponding decoded packet $\mathbf{m}_{i_n} + \mathbf{r}_n\mathbf{V}_{\mathcal{I}}$ is consistent for $\mathbf{r}_n = \mathbf{0}$, the probability that it is consistent is less than $\left(\frac{k+1}{q}\right)$.

Case B: $n > 1$ and $\mathbf{R}_{n-1}$ has nonzero rank.

Conditioned on $\mathbf{r}_n$ being in the row space of $\mathbf{R}_{n-1}$, $\mathbf{r}_n = \mathbf{g}\mathbf{R}_{n-1}$ where $\mathbf{g}$ is uniformly distributed over $\mathbb{F}_q^{n-1}$. Since $\mathbf{R}_{n-1}\mathbf{V}_{\mathcal{I}} \neq \mathbf{0}$, by Corollary 6.2, the corresponding decoded packet

$$\mathbf{m}_{i_n} + \mathbf{r}_n\mathbf{V}_{\mathcal{I}} = \mathbf{m}_{i_n} + \mathbf{g}\mathbf{R}_{n-1}\mathbf{V}_{\mathcal{I}}$$

is consistent with probability at most $\frac{k+1}{q}$.

Conditioned on $\mathbf{r}_n$ not being in the row space of $\mathbf{R}_{n-1}$, we can partition the set of possible values for $\mathbf{r}_n$ into cosets

$$\left\{\mathbf{r} + \mathbf{g}\mathbf{R}_{n-1} :\ \mathbf{g} \in \mathbb{F}_q^{n-1}\right\}$$

where $\mathbf{r}$ is not in the row space of $\mathbf{R}_{n-1}$; the corresponding values of the i_nth decoded packet are given by

$$\left\{\mathbf{m}_{i_n} + \mathbf{r}\mathbf{V}_{\mathcal{I}} + \mathbf{g}\mathbf{R}_{n-1}\mathbf{V}_{\mathcal{I}} :\ \mathbf{g} \in \mathbb{F}_q^{n-1}\right\}.$$

Noting as before that $\mathbf{R}_{n-1}\mathbf{V}_{\mathcal{I}} \neq \mathbf{0}$ and applying Corollary 6.2, the i_nth decoded packet is consistent with probability at most $\frac{k+1}{q}$. □

Proof of Corollary 6.1: Suppose two or more different sets of packets are used for decoding. If not all of them contain at least one non-genuine packet, the decoded values obtained from different decoding sets will differ, and indicate an error. Otherwise, suppose all the decoding sets contain at least one non-genuine packet. Consider the sets in turn, denoting by s_i' the number of unmodified packets in the ith set that are not in any set $j < i$. For any particular values of packets in sets $j < i$, we have from Theorem 6.5 that at most a fraction $\left(\frac{k+1}{q}\right)^{s_i'}$ of decoding outcomes for set i have consistent data and hash values. Thus, the overall fraction of consistent decoding outcomes is at most $\left(\frac{k+1}{q}\right)^{\sum_i s_i'} = \left(\frac{k+1}{q}\right)^{s}$. □

Bibliography

S. Acedański, S. Deb, M. Médard, and R. Koetter, "How good is random linear coding based distributed networked storage?" in *Proc. WINMEE, RAWNET and NETCOD 2005 Workshops*, Riva del Garda, Italy, Apr. 2005.

M. Adler, N. J. A. Harvey, K. Jain, R. Kleinberg, and A. R. Lehman, "On the capacity of information networks," *IEEE Transactions on Information Theory*, vol. 52, no. 6, June 2006.

A. Agarwal and M. Charikar, "On the advantage of network coding for improving network throughput," in *Information Theory Workshop*, 2004.

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

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ: Prentice Hall, 1993.

B. Awerbuch and T. Leighton, "A simple local-control approximation algorithm for multicommodity flow," in *Proc. 34th Annu. IEEE Symp. Foundations of Computer Science*, Palo Alto, CA, Nov. 1993, pp. 459–468.

——, "Improved approximation algorithms for the multicommodity flow problem and local competitive routing in dynamic networks," in *STOC '94: Proc. 26th Ann. ACM Symp. Theory of Computing*, Montreal, QC, Canada, May 1994, pp. 487–496.

D. Bertsekas, A. Nedic, and A. E. Ozdaglar, *Convex analysis and optimization*. Athena Scientific, 2003.

D. P. Bertsekas, "A class of optimal routing algorithms for communication networks," in *Proc. 5th Int. Conf. Computer Communication (ICCC '80)*, Atlanta, GA, Oct. 1980, pp. 71–76.

——, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1995.

——, *Network Optimization: Continuous and Discrete Models*. Belmont, MA: Athena Scientific, 1998.

D. P. Bertsekas, E. M. Gafni, and R. G. Gallager, "Second derivative algorithms for minimum delay distributed routing in networks," *IEEE Trans. Commun.*, vol. 32, no. 8, pp. 911–919, Aug. 1984.

D. P. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1992.

D. Bertsimas, I. C. Paschalidis, and J. Tsitsiklis, "On the large deviations behavior of acyclic networks of $G/G/1$ queues," *Ann. Appl. Probab.*, vol. 8,

no. 4, pp. 1027–1069, Nov. 1998.

S. Bhadra, S. Shakkottai, and P. Gupta, "Min-cost selfish multicast with network coding," *IEEE Trans. Inform. Theory*, vol. 52, no. 11, pp. 5077–5087, Nov. 2006.

K. Bhattad and K. R. Nayayanan, "Weakly secure network coding," in *Proc. WINMEE, RAWNET and NETCOD 2005 Workshops*, Riva del Garda, Italy, Apr. 2005.

K. Bhattad, N. Ratnakar, R. Koetter, and K. R. Narayanan, "Minimal network coding for multicast," in *Proc. 2005 IEEE Int. Symp. Information Theory (ISIT 2005)*, Adelaide, Australia, Sept. 2005, pp. 1730–1734.

J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE J. Select. Areas Commun.*, vol. 20, no. 8, pp. 1528–1540, Oct. 2002.

N. Cai and R. W. Yeung, "Secure network coding," in *Proc. 2002 IEEE Int. Symp. Information Theory (ISIT 2002)*, Lausanne, Switzerland, June/July 2002, p. 323.

——, "Network error correction, part II: Lower bounds," *Commun. Inf. Syst.*, vol. 6, no. 1, pp. 37–54, 2006.

T. H. Chan, "On the optimality of group network codes," in *Proc. 2005 IEEE Int. Symp. Information Theory (ISIT 2005)*, Adelaide, Australia, Sept. 2005, pp. 1992–1996.

D. Charles, K. Jain, and K. Lauter, "Signatures for network coding," in *Proc. 2006 Conf. Information Sciences and Systems (CISS 2006)*, Princeton, NJ, Mar. 2006, invited paper.

C. Chekuri, C. Fragouli, and E. Soljanin, "on average throughput and alphabet size in network coding," *IEEE Transactions on Information Theory*, vol. 52, no. 6, June 2006.

H. Chen and A. Mandelbaum, "Discrete flow networks: Bottleneck analysis and fluid approximations," *Math. Oper. Res*, vol. 16, no. 2, pp. 408–446, May 1991.

H. Chen and D. D. Yao, *Fundamentals of Queueing Networks: Performance, Asymptotics, and Optimization*, ser. Applications of Mathematics. New York, NY: Springer, 2001, vol. 46.

P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proceedings of 41st Annual Allerton Conference on Communication, Control, and Computing*, October 2003. [Online]. Available: http://research.microsoft.com/ pachou/pubs/ChouWJ03.pdf

——, "Practical network coding," in *Proc. 41st Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Oct. 2003.

T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY: John Wiley & Sons, 1991.

R. L. Cruz and A. V. Santhanam, "Optimal routing, link scheduling and power control in multi-hop wireless networks," in *Proc. IEEE INFOCOM 2003*, vol. 1, San Francisco, CA, Mar./Apr. 2003, pp. 702–711.

I. Csiszár, "Linear codes for sources and source networks: Error exponents, universal coding," *IEEE Trans. Inform. Theory*, vol. 28, no. 4, pp. 585–592, July 1982.

Y. Cui, Y. Xue, and K. Nahrstedt, "Optimal distributed multicast routing using network coding: Theory and applications," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 2, pp. 47–49, Sept. 2004.

A. F. Dana, R. Gowaikar, R. Palanki, B. Hassibi, and M. Effros, "Capacity of

wireless erasure networks," *IEEE Trans. Inform. Theory*, vol. 52, no. 3, pp. 789–804, Mar. 2006.

S. Deb and M. Medard, "Algebraic gossip: A network coding approach to optimal multiple rumor mongering," *IEEE Transactions on Information Theory, submitted*, 2004.

S. Deb, M. Médard, and C. Choute, "Algebraic gossip: A network coding approach to optimal multiple rumor mongering," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2486–2507, June 2006.

S. Deb and R. Srikant, "Congestion control for fair resource allocation in networks with multicast flows," *IEEE/ACM Trans. Networking*, vol. 12, no. 2, pp. 274–285, Apr. 2004.

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

——, "Networks, matroids, and non-Shannon information inequalities," *IEEE Trans. Inform. Theory*, vol. 53, no. 6, June 2007.

——, "Insufficiency of linear coding in network information flow," *IEEE Trans. Inform. Theory*, vol. 51, no. 8, pp. 2745–2759, Aug. 2005.

M. Effros, T. Ho, and S. Kim, "A tiling approach to network code design for wireless networks," in *Proc. 2006 IEEE Information Theory Workshop (ITW 2006)*, Punta del Este, Uruguay, Mar. 2006, pp. 62–66.

E. Erez and M. Feder, "Convolutional network codes," in *Proc. 2004 IEEE Int. Symp. Information Theory (ISIT 2004)*, Chicago, IL, June/July 2004, p. 146.

——, "Convolutional network codes for cyclic networks," in *Proc. WINMEE, RAWNET and NETCOD 2005 Workshops*, Riva del Garda, Italy, Apr. 2005.

A. Eryilmaz and D. S. Lun, "Control for inter-session network coding," in *Proc. 2007 Information Theory and Applications Workshop (ITA 2007)*, San Diego, CA, Jan./Feb. 2007.

M. Feder, D. Ron, and A. Tavory, "Bounds on linear codes for network multicast," Electronic Colloquium on Computational Complexity, Report TR03-033, May 2003.

J. Feldman, T. Malkin, C. Stein, and R. A. Servedio, "On the capacity of secure network coding," in *Proc. 42nd Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Sept./Oct. 2004.

C. Fragouli and E. Soljanin, "A connection between network coding and convolutional codes," in *Proc. 2004 IEEE Int. Conf. Communications (ICC 2004)*, vol. 2, Paris, France, June 2004, pp. 661–666.

——, "Information flow decomposition for network coding," *IEEE Transactions on Information Theory, submitted*, 2004.

——, "Information flow decomposition for network coding," *IEEE Trans. Inform. Theory*, vol. 52, no. 3, pp. 829–848, Mar. 2006.

S. Ghez, S. Verdú, and S. C. Schwartz, "Stability properties of slotted Aloha with multipacket reception capability," *IEEE Trans. Automat. Contr.*, vol. 33, no. 7, pp. 640–649, July 1988.

E. N. Gilbert, "A comparison of signaling alphabets," *Bell Syst. Tech. J.*, vol. 31, pp. 504–522, May 1952.

C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *Proc. IEEE INFOCOM 2005*, vol. 4, Miami, FL, Mar. 2005, pp. 2235–2245.

R. Gowaikar, A. F. Dana, R. Palanki, B. Hassibi, and M. Effros, "On the capacity of wireless erasure networks," in *Proc. 2004 IEEE Int. Symp. Information Theory (ISIT 2004)*, Chicago, IL, June/July 2004, p. 401.

T. S. Han, "Slepian-Wolf-Cover theorem for networks of channels," *Inf. Control*, vol. 47, pp. 67–83, 1980.

N. J. A. Harvey, R. Kleinberg, C. Nair, and Y. Wu, "A 'chicken & egg' network coding problem," in *Proc. 2007 IEEE Int. Symp. Information Theory (ISIT 2007)*, Nice, France, June 2007.

N. J. A. Harvey, D. R. Karger, and K. Murota, "Deterministic network coding by matrix completion," in *SODA '05: Proc. 16th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2005, pp. 489–498.

T. Ho, D. R. Karger, M. Médard, and R. Koetter, "Network coding from a network flow perspective," in *Proceedings of the IEEE International Symposium on Information Theory*, 2003.

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

T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proceedings of 41st Annual Allerton Conference on Communication, Control, and Computing*, October 2003. [Online]. Available: http://web.mit.edu/trace/www/allerton.pdf

T. Ho and H. Viswanathan, "Dynamic algorithms for multicast with intra-session network coding," in *Proc. 43rd Annual Allerton Conference on Communication, Control, and Computing*, 2005.

——, "Dynamic algorithms for multicast with intra-session network coding," *IEEE Transactions on Information Theory*, submitted, 2006.

T. Ho, "Networking from a network coding perspective," Ph.D. dissertation, Massachusetts Institute of Technology, May 2004.

T. Ho, Y.-H. Chang, and K. J. Han, "On constructive network coding for multiple unicasts," in *Proc. 44th Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Sept. 2006, invited paper.

T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. 2003 IEEE Int. Symp. Information Theory (ISIT 2003)*, Yokohama, Japan, June/July 2003, p. 442.

T. Ho, B. Leong, R. Koetter, M. Médard, M. Effros, and D. R. Karger, "Byzantine modification detection for multicast networks using randomized network coding," in *Proc. 2004 IEEE Int. Symp. Information Theory (ISIT 2004)*, Chicago, IL, June/July 2004, p. 144.

T. Ho, B. Leong, R. Koetter, and M. Medard, "Distributed asynchronous algorithms for multicast network coding," in *Proceedings of First Workshop on Network Coding, Theory and Applications*, 2005.

T. Ho, M. Médard, and M. Effros, "Network coding for correlated sources," in *Proc. 2004 Conf. Information Sciences and Systems (CISS 2004)*, Princeton, NJ, Mar. 2004, invited paper.

T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proc. 41st Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Oct. 2003.

T. Ho and H. Viswanathan, "Dynamic algorithms for multicast with intra-session network coding," in *Proc. 43rd Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Sept. 2005.

S. Jaggi, P. Chou, and K. Jain, "Low complexity algebraic network codes," in *Proc. 2003 IEEE Int. Symp. Information Theory (ISIT 2003)*, Yokohama, Japan, June/July 2003, p. 368.

S. Jaggi and M. Langberg, "Resilient network coding in the presence of eavesdropping byzantine adversaries," in *Proc. IEEE International Symposium on Information Theory*, Nice, France, June 2007.

S. Jaggi, M. Langberg, T. Ho, and M. Effros, "Correction of adversarial errors in networks," in *Proc. 2005 IEEE International Symposium on Information Theory (ISIT 2005)*, Adelaide, Australia, Sept. 2005, pp. 1455–1459.

S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard, "Resilient network coding in the presence of byzantine adversaries," in *Proc. IEEE INFOCOM 2007*, Anchorage, AK, May 2007.

S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. M. G. M. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Inform. Theory*, vol. 51, no. 6, pp. 1973–1982, June 2005.

——, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Inform. Theory*, vol. 51, no. 6, pp. 1973–1982, June 2005.

K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," in *MobiCom '03: Proc. 9th Annu. Int. Conf. Mobile Computing and Networking*, San Diego, CA, 2003, pp. 66–80.

M. Johansson, L. Xiao, and S. Boyd, "Simultaneous routing and power allocation in CDMA wireless data networks," in *Proc. 2003 IEEE Int. Conf. Communications (ICC 2003)*, vol. 1, Anchorage, AK, May 2003, pp. 51–55.

S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Mdard, "Practical network coding for wireless environments," in *Proceedings of 43rd Allerton Conference on Communication, Control, and Computing, Monticello, IL*, September 2005.

S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Médard, "The importance of being opportunistic: Practical network coding for wireless environments," in *Proc. 43rd Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Sept. 2005.

S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 243–254, 2006.

F. P. Kelly, *Reversibility and Stochastic Networks*. Chichester: John Wiley & Sons, 1979.

R. Khalili and K. Salamatian, "On the capacity of erasure relay channel: Multi-relay case," in *Proc. 2005 IEEE Information Theory Workshop (ITW 2005)*, Rotorua, New Zealand, Aug./Sept. 2005.

M. Kim, M. Médard, V. Aggarwal, U.-M. O'Reilly, W. Kim, C. W. Ahn, and M. Effros, "Evolutionary approaches to minimizing network coding resources," in *Proc. IEEE INFOCOM 2007*, Anchorage, AK, May 2007.

M. Kodialam and T. Nandagopal, "Characterizing achievable rates in multi-hop wireless mesh networks with orthogonal channels," *IEEE/ACM Trans. Networking*, vol. 13, no. 4, pp. 868–880, Aug. 2005.

R. Koetter and F. R. Kschischang, "Coding for errors and erasures in random network coding," *IEEE Transactions on Information Theory*, 2007, submitted.

R. Koetter, M. Effros, T. Ho, and M. Médard, "Network codes as codes on graphs," in *Proc. 2004 Conf. Information Sciences and Systems (CISS 2004)*, Princeton, NJ, Mar. 2004.

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

G. Kramer and S. Savari, "Progressive d-separating edge set bounds on network coding rates," in *Proceedings of the International Symposium on Information Theory*, 2005.

S.-Y. R. Li and R. W. Yeung, "On convolutional network coding," 2006.

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

Z. Li and B. Li, "Network coding in undirected networks," in *Proc. 2004 Conf. Information Sciences and Systems (CISS 2004)*, Princeton, NJ, Mar. 2004.

——, "Network coding: The case of multiple unicast sessions," in *Proc. 42nd Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Sept./Oct. 2004.

——, "Efficient and distributed computation of maximum multicast rates," in *Proc. IEEE INFOCOM 2005*, vol. 3, Miami, FL, Mar. 2005, pp. 1618–1628.

M. Luby, "LT codes," in *Proc. 34th Annu. IEEE Symp. Foundations of Computer Science*, Vancouver, Canada, Nov. 2002, pp. 271–280.

D. S. Lun, "Efficient operation of coded packet networks," Ph.D. dissertation, Massachusetts Institute of Technology, June 2006.

D. S. Lun, M. Médard, and M. Effros, "On coding for reliable communication over packet networks," in *Proc. 42nd Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Sept./Oct. 2004, invited paper.

D. S. Lun, M. Médard, T. Ho, and R. Koetter, "Network coding with a cost criterion," in *Proc. 2004 Int. Symp. Information Theory and its Applications (ISITA 2004)*, Parma, Italy, Oct. 2004, pp. 1232–1237.

D. S. Lun, M. Médard, and D. R. Karger, "On the dynamic multicast problem for coded networks," in *Proc. WINMEE, RAWNET and NETCOD 2005 Workshops*, Riva del Garda, Italy, Apr. 2005.

D. S. Lun, M. Médard, and R. Koetter, "Efficient operation of wireless packet networks using network coding," in *Proc. Int. Workshop Convergent Technologies (IWCT) 2005*, Oulu, Finland, June 2005, invited paper.

D. S. Lun, M. Médard, R. Koetter, and M. Effros, "Further results on coding for reliable communication over packet networks," in *Proc. 2005 IEEE Int. Symp. Information Theory (ISIT 2005)*, Adelaide, Australia, Sept. 2005, pp. 1848–1852.

D. S. Lun, P. Pakzad, C. Fragouli, M. Médard, and R. Koetter, "An analysis of finite-memory random linear coding on packet streams," in *Proc. 4th Int. Symp. Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt '06)*, Boston, MA, Apr. 2006.

D. S. Lun, N. Ratnakar, R. Koetter, M. Médard, E. Ahmed, and H. Lee, "Achieving minimum-cost multicast: A decentralized approach based on network coding," in *Proc. IEEE INFOCOM 2005*, vol. 3, Miami, FL, Mar. 2005, pp. 1608–1617.

D. S. Lun, N. Ratnakar, M. Médard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet

networks," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2608–2623, June 2006.

P. Maymounkov, "Online codes," NYU, Technical Report TR2002-833, Nov. 2002.

P. Maymounkov, N. J. A. Harvey, and D. S. Lun, "Methods for efficient network coding," in *Proc. 44th Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Sept. 2006.

M. Médard, M. Effros, T. Ho, and D. R. Karger, "On coding for non-multicast networks," in *Proc. 41st Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Oct. 2003.

R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge: Cambridge University Press, 1995.

M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," *IEEE J. Select. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Jan. 2005.

M. J. Neely, "Energy optimal control for time-varying wireless networks," *IEEE Trans. Inform. Theory*, vol. 52, no. 7, pp. 2915–2934, July 2006.

G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. New York, NY: John Wiley & Sons, 1999.

A. Ouorou, P. Mahey, and J.-P. Vial, "A survey of algorithms for convex multicommodity flow problems," *Manage. Sci.*, vol. 46, no. 1, pp. 126–147, Jan. 2000.

J. G. Oxley, *Matroid Theory*. Oxford Univ. Press, 1992.

P. Pakzad, C. Fragouli, and A. Shokrollahi, "Coding schemes for line networks," in *Proc. 2005 IEEE Int. Symp. Information Theory (ISIT 2005)*, Adelaide, Australia, Sept. 2005, pp. 1853–1857.

J.-S. Park, M. Gerla, D. S. Lun, Y. Yi, and M. Médard, "Codecast: A network-coding-based ad hoc multicast protocol," *IEEE Wireless Commun. Mag.*, vol. 13, no. 5, pp. 76–81, Oct. 2006.

I. C. Paschalidis and Y. Liu, "Large deviations-based asymptotics for inventory control in supply chains," *Oper. Res.*, vol. 51, no. 3, pp. 437–460, May–June 2003.

A. Ramamoorthy, K. Jain, P. A. Chou, and M. Effros, "Separating distributed source coding from network coding," *IEEE Transactions on Information Theory*, vol. 52, no. 6, June 2006.

S. Ramanathan, "Multicast tree generation in networks with asymmetric links," *IEEE/ACM Trans. Networking*, vol. 4, no. 4, pp. 558–568, Aug. 1996.

A. Rasala Lehman and E. Lehman, "Complexity classification of network information flow problems," in *SODA '04: Proc. 15th Annu. ACM-SIAM Symp. Discrete algorithms*, New Orleans, LA, Jan. 2004, pp. 142–150.

N. Ratnakar, R. Koetter, and T. Ho, "Linear flow equations for network coding in the multiple unicast case," in *Proc. DIMACS Working Group Network Coding*, Piscataway, NJ, Jan. 2005.

N. Ratnakar, D. Traskov, and R. Koetter, "Approaches to network coding for multiple unicasts," in *Proc. 2006 International Zurich Seminar on Communications (IZS 2006)*, Zurich, Switzerland, Feb. 2006, pp. 70–73, invited paper.

S. Riis, "Linear versus non-linear Boolean functions in network flow," in *Proc. 2004 Conf. Information Sciences and Systems (CISS 2004)*, Princeton, NJ, Mar. 2004.

J. R. Roche, R. W. Yeung, and K. P. Hau, "Symmetrical multilevel diversity coding," *IEEE Trans. Inform. Theory*, vol. 43, no. 3, pp. 1059–1064, May 1997.

Y. E. Sagduyu and A. Ephremides, "Crosslayer design and distributed MAC and network coding in wireless ad hoc networks," in *Proc. 2005 IEEE Int. Symp. Information Theory (ISIT 2005)*, Adelaide, Australia, Sept. 2005, pp. 1863–1867.

——, "Joint scheduling and wireless network coding," in *Proc. WINMEE, RAWNET and NETCOD 2005 Workshops*, Riva del Garda, Italy, Apr. 2005.

P. Sanders, S. Egner, and L. Tolhuizen, "Polynomial time algorithms for network information flow," in *SPAA '03: Proc. 15th Annu. ACM Symp. Parallel Algorithms and Architectures*, San Diego, CA, June 2003, pp. 286–294.

S. Sengupta, S. Rayanchu, and S. Banerjee, "An analysis of wireless network coding for unicast sessions: The case for coding-aware routing," in *Proc. IEEE INFOCOM 2007*, Anchorage, AK, May 2007.

H. D. Sherali and G. Choi, "Recovery of primal solutions when using subgradient optimization methods to solve Lagrangian duals of linear programs," *Oper. Res. Lett.*, vol. 19, pp. 105–113, 1996.

A. Shokrollahi, "Raptor codes," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.

N. Shulman, "Communication over an unknown channel via common broadcasting," Ph.D. dissertation, Tel Aviv University, July 2003.

N. Shulman and M. Feder, "Static broadcasting," in *Proc. 2000 IEEE Int. Symp. Information Theory (ISIT 2000)*, Sorrento, Italy, June 2000, p. 23.

R. Srikant, *The Mathematics of Internet Congestion Control*. Boston, MA: Brikhäuser, 2004.

J. Tan and M. Médard, "Secure network coding with a cost criterion," in *Proc. 4th Int. Symp. Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt '06)*, Boston, MA, Apr. 2006.

L. Tassiulas and A. F. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automat. Contr.*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

D. Traskov, N. Ratnakar, D. S. Lun, R. Koetter, and M. Médard, "Network coding for multiple unicasts: An approach based on linear optimization," in *Proc. 2006 IEEE Int. Symp. Information Theory (ISIT 2006)*, Seattle, WA, July 2006, pp. 1758–1762.

J. N. Tsitsiklis, "Decentralized detection," in *Advances in Statistical Signal Processing*. Greenwich, CT: JAI Press, 1993, vol. 2, pp. 297–344.

E. C. van der Meulen, "Three-terminal communication channels," *Adv. Appl. Probab.*, vol. 3, pp. 120–154, 1971.

R. R. Varshamov, "Estimate of the number of signals in error-correcting codes," *Dokl. Acad. Nauk SSSR*, vol. 117, pp. 739–741, 1957.

Y. Wu, "A trellis connectivity analysis of random linear network coding with buffering," in *Proc. 2006 IEEE Int. Symp. Information Theory (ISIT 2006)*, Seattle, WA, July 2006, pp. 768–772.

Y. Wu, M. Chiang, and S.-Y. Kung, "Distributed utility maximization for network coding based multicasting: A critical cut approach," in *Proc. 4th Int. Symp. Modeling and Optimization in Mobile, Ad Hoc and Wireless*

Networks (WiOpt '06), Boston, MA, Apr. 2006.

Y. Wu, P. A. Chou, and S. Y. Kung, "Information exchange in wireless networks with network coding and physical-layer broadcast," in *CISS*, 2005.

Y. Wu, P. A. Chou, and S.-Y. Kung, "Information exchange in wireless networks with network coding and physical-layer broadcast," in *Proc. 2005 Conf. Information Sciences and Systems (CISS 2005)*, Baltimore, MD, Mar. 2005.

——, "Minimum-energy multicast in mobile ad hoc networks using network coding," *IEEE Trans. Commun.*, vol. 53, no. 11, pp. 1906–1918, Nov. 2005.

Y. Wu, P. A. Chou, Q. Zhang, K. Jain, W. Zhu, and S.-Y. Kung, "Network planning in wireless ad hoc networks: A cross-layer approach," *IEEE J. Select. Areas Commun.*, vol. 23, no. 1, pp. 136–150, Jan. 2005.

Y. Xi and E. M. Yeh, "Distributed algorithms for minimum cost multicast with network coding," in *Proc. 43rd Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Sept. 2005.

——, "Distributed algorithms for minimum cost multicast with network coding in wireless networks," in *Proc. 4th Int. Symp. Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt '06)*, Boston, MA, Apr. 2006.

L. Xiao, M. Johansson, and S. Boyd, "Simultaneous routing and resource allocation via dual decomposition," *IEEE Trans. Commun.*, vol. 52, no. 7, pp. 1136–1144, July 2004.

X. Yan, R. W. Yeung, and Z. Zhang, "The capacity region for multi-source multi-sink network coding," in *Proc. 2007 IEEE International Symposium on Information Theory (ISIT 2007)*, 2007.

R. W. Yeung, "Multilevel diversity coding with distortion," *IEEE Trans. Inform. Theory*, vol. 41, no. 2, pp. 412–422, Mar. 1995.

——, *A First Course in Information Theory.* Kluwer Academic/Plenum, 2002.

R. W. Yeung and N. Cai, "Network error correction, part I: Basic concepts and upper bounds," *Commun. Inf. Syst.*, vol. 6, no. 1, pp. 19–36, 2006.

R. W. Yeung, S.-Y. R. Li, N. Cai, and Z. Zhang, "Network coding theory: Part I: Single source," *Foundations and Trends in Communications and Information Theory*, vol. 2, no. 4, 2005.

——, "Network coding theory: Part II: Multiple sources," *Foundations and Trends in Communications and Information Theory*, vol. 2, no. 5, 2005.

R. W. Yeung and Z. Zhang, "Distributed source coding for satellite communications," *IEEE Trans. Inform. Theory*, vol. 45, no. 4, pp. 1111–1120, May 1999.

——, "On symmetrical multilevel diversity coding," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 609–621, Mar. 1999.

F. Zhao, T. Kalker, M. Medard, and K. J. Han, "Signatures for content distribution with network coding," in *Proc. 2007 IEEE International Symposium on Information Theory (ISIT 2007)*, 2007.

L. Zosin and S. Khuller, "On directed Steiner trees," in *SODA '02: Proc. 13th Annu. ACM-SIAM Symp. Discrete Algorithms*, San Francisco, CA, Jan. 2002, pp. 59–63.

Index