Foundations and Trends® in
Networking
Vol. 2, No. 2 (2007) 135–269

DOI: 10.1561/1300000013

now
the essence of knowledge

Network Coding Applications

Christina Fragouli[1] and Emina Soljanin[2]

[1] *École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, christina.fragouli@epfl.ch*
[2] *Bell Laboratories, Alcatel-Lucent, USA, emina@research.bell-labs.com*

Abstract

Network coding is an elegant and novel technique introduced at the turn of the millennium to improve network throughput and performance. It is expected to be a critical technology for networks of the future. This tutorial deals with wireless and content distribution networks, considered to be the most likely applications of network coding, and it also reviews emerging applications of network coding such as network monitoring and management. Multiple unicasts, security, networks with unreliable links, and quantum networks are also addressed. The preceding companion deals with theoretical foundations of network coding.

1

Introduction

The emergence of network coding has brought about a metamorphosis in thinking about network communication, with its simple but important premise that in communication networks, we can allow nodes to not only forward but also process the incoming independent information flows.

Today, ten years after the emergence of the first example of network coding the butterfly network, a lot is already known about network coding, in particular for the case of network multicast. Network multicast refers to simultaneously transmitting the same information to multiple receivers in the network. The fascinating fact that the original network coding theorem brought was that the conditions necessary and sufficient for unicast at a certain rate to each of these receiver are also necessary and sufficient for multicast at the same rate, provided the intermediate network nodes are allowed to combine and process different information streams.

In the first part of the tutorial [24], we examined in detail the case of network multicast, mainly from a theoretical point of view. We argued that network coding can and has been studied within a number of different theoretical frameworks, in several research commu-

nities, most notably Information Theory and Computer Science. The choice of framework a researcher makes most frequently depends on his/her background and preferences. However, one may also argue that each network coding issue (e.g., code design, throughput benefits, complexity) should be put in the framework in which it can be studied the most naturally and efficiently.

The goal of the second part of the tutorial, is to depart from the multicast scenario, and discuss how ideas from network coding can have impact on a number of new applications.

Today, more and more researchers and engineers ask what network coding is, what its benefits are, and how much it costs to design and operate networks implementing network coding. At this point, we do not have complete answers to these questions even in the case of network multicast. For example, the minimum network operating costs required to achieve maximum throughput are not known in general in terms of the code alphabet size and the number of routers required to code. (Multicast in networks with two sources and arbitrary number of receivers is almost completely understood.)

Even less is known in the arguably practically more important case of multiple unicasts, where we do not have a good understanding of the theoretical limits and on how to achieve them. Today, not even the throughput benefits of coding have been completely characterized. Although there are directed graph instances where the network coding throughput increase is proportional to the number of nodes in the graph, we are yet to find an undirected graph instance where network coding offers any benefits. Another transmission scenario for which benefits of coding are not fully understood are networks with non-uniform demands. Studying general traffic patterns is complicated from the fact that optimal solutions may require exponential alphabet sizes and nonlinear operations. We discuss such issues in Section 5.

Also, work is just beginning to address the problem of disseminating correlated information over network coded systems, and more generally the problem of distributed source coding. Such connections between source coding and network coding is one of the topics that we will not cover in this tutorial.

The Microsoft's Avalanche system has sparked the interest in using network coding for content distribution. Various tests and measurements have been carried out on experimental P2P systems, and results together with numerous observed advantages of using network coding were reported (see Section 3). Fine-tuning this approach for specific applications, such as video on demand, and developing a theory that would completely support the experimental evidence, is still missing.

Network coding allows to take advantage of the broadcasting capabilities of the shared wireless medium to provide benefits in terms of bandwidth, transmission power, and delay, as we will argue in Section 4. Clearly to warranty the deployment of such techniques, the required processing of data within the network needs to have low complexity and power consumption. MIT's COPE demonstrated that even when coding operations are confined to simple binary additions obeying some additional constraints, there are still gains to be had in terms of throughput and efficiency of MAC layer protocols. The first approaches on wireless network coding ignored the interference of multiple broadcast transmissions at a receiver. One can show that such strategies can incur significant losses in terms of achievable rates. Physical layer network coding was a first attempt to remedy this. Very recently, a linear deterministic model was developed that captures the interactions between the signals in a wireless network, and was shown that for such models one can obtain an information-theoretic max-flow min-cut result. Wireless and sensor networks provide vast opportunities for applications of network coding, and numerous and diverse problems are beginning to receive attention, ranging from techniques such as cross layer design over issues such as fairness and delay, to untuned radios and distributed storage.

Another line of recent work deals with networks in which some edges are in a certain way compromised. The information carried by such edges may be deleted, altered, or observed by an adversary whose information gain we would like to limit. Information may also be lost due to channel errors. Usually, no assumption is made on the choice of such edges, but their number is limited. Network codes can be designed for such networks, although some throughput has to be sacrificed to accommodate for compromised edges. The maximum achievable throughput

is known for most of such scenarios, and it depends on the size of the affected edge set. Algorithms for designing error-correcting, attack resilient, and secure network codes have also been proposed, and we discuss some of them in Sections 6 and 7. Very recently an elegant approach to error correction was introduced based on the use of subspaces. For some cases however, more related to security, the codes we have today require huge alphabet size and are in general too complex to implement. Thus design of practical codes is of interest. In general, combining network coding with security is an area with many interesting open questions. Information theoretic tools have also been useful here to characterize achievable rates, for, up to now, specific sets of networks with lossy links.

These days we are beginning to see network coding ideas being put to use in problems other than increasing throughput in networks with multiple users. There is evidence that network coding may be beneficial for active network monitoring, as well as passive inference of link loss rates. Interestingly, in a system employing randomized network coding, the randomly created linear combinations implicitly carry information about the network topology, that we can exploit toward diverse applications. Use of network coding techniques can help to increase rates of multicast switches, leverage the efficiency of databases, and reduce on-chip wiring. We briefly discuss such applications in Section 9.

The network coding butterfly has even reached quantum information theorists (see Section 8). If we recall that in multicast communications networks, large throughput gains are possible with respect to their (physical) transportation or fluid counterparts because classical information can be processed in a way that physical entities cannot, an interesting question to ask is whether anything can be gained by allowing processing of quantum information at nodes in quantum networks. Although physical carriers of quantum information can be processed in certain ways determined by the laws of quantum mechanics, two operations essential in classical information networking, replication (cloning) and broadcasting, are not possible. However, approximate and probabilistic cloning as well as different types of compression of quantum states are possible, and have been used in attempts to find a quantum counterpart of network coding.

The reason that network coding continues to be a very active field is clearly due to the benefits it promises to offer. As we mentioned earlier, we discuss in Section 4 how network coding can help to better exploit shared resources such as wireless bandwidth, and to conserve scarce resources, such as battery life. Moreover, it can offer benefits in terms of *reliability* against channel errors and *security*, as we discuss in Sections 6 and 7, respectively. Although all these are important, perhaps the most interesting benefits of network coding might manifest in situations where the topology dynamically changes, and operation is restricted to distributed algorithms that do not employ knowledge about the network environment. This is the topic of the following Section 2.

We hope that the research effort in the area of network coding will continue to increase, bringing new exciting results and applications, and making the results described in this tutorial very fast outdated.

2

Decentralized Network Operation

An interesting property of network operation using network coding is that, for some traffic scenarios, network coding effectively allows the nodes of the network to achieve the optimal performance while operating in a decentralized fashion. This finds immediate application in dynamically changing environments, where centralized network management and control has a prohibitive complexity, and thus network nodes need to operate in a distributed fashion without using knowledge of the overall network configuration. Such applications occur in peer-to-peer (P2P) content distribution networks, that we examine next in Section 3, and in wireless networks, where the network configuration may change because nodes move, turn on and off, and roam out of range, that we examine in Section 4.

In this section, we first examine in more detail this property and show it is directly implied by the information theoretic proof of the main theorem in network coding. We then present the coupons collector problem, that is a direct manifestation of this property, and can be used to study a number of different communication problems. The coupons collector problem nicely captures for example the problem of content distribution over P2P networks such as BitTorrent, where

multiple copies of N file fragments exist within a network, and a copy of the file is acquired by collecting these fragments. We close the section by briefly discussing distributed information dissemination protocols, and network coding over random graphs.

2.1 A Distributed Protocol for the Min-cut Max-Flow Theorem

Let $G = (V, E)$ be a graph with the set of vertices V and the set of edges $E \subset V \times V$. Consider a node $S \in V$ that wants to transmit information to a node $R \in V$.

From the max-flow min-cut theorem (see part I), we know that if the min-cut between S and R equals h, then information can be send from S to R at a maximum rate of h. To route this information, if for example we have a graph with unit capacity edges, we would need to find h edge disjoint paths. Once these paths are identified, each node in such a path would forward information to the specific node that is next in the path. If the network topology changes, we would need to find new paths, and redefine the intermediate nodes operation.

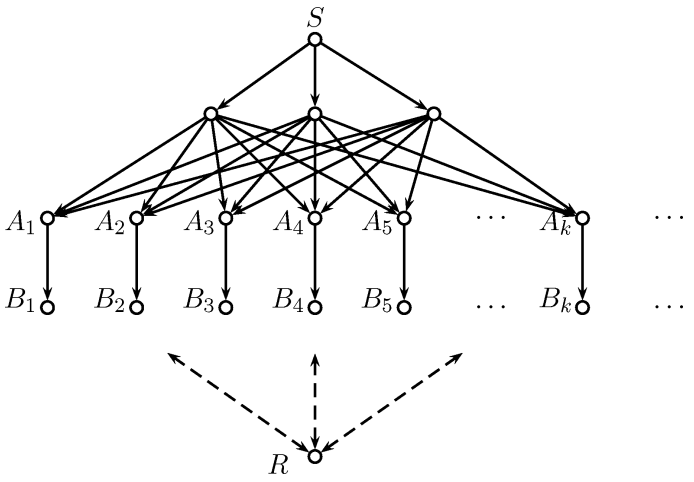
Consider now operating this network using randomized network coding. From the information theoretic proof of the main theorem (see part I), we know that, if all nodes in the network do *exactly the same operation*, randomly combine their incoming flows and transmit them to their outgoing edges, no matter what is their position in the network and what the network topology between the source and the destination is, we can achieve the min-cut rate.

In other words, *even if we have a random network* between the source and the destination, and we know nothing of its structure, provided the min-cut to the receiver is maintained, and allowing all nodes to operate in exactly the same fashion, allows the receiver to get information at the min-cut rate. This is not possible in the case of routing, where information would be routed differently, depending on the network structure.

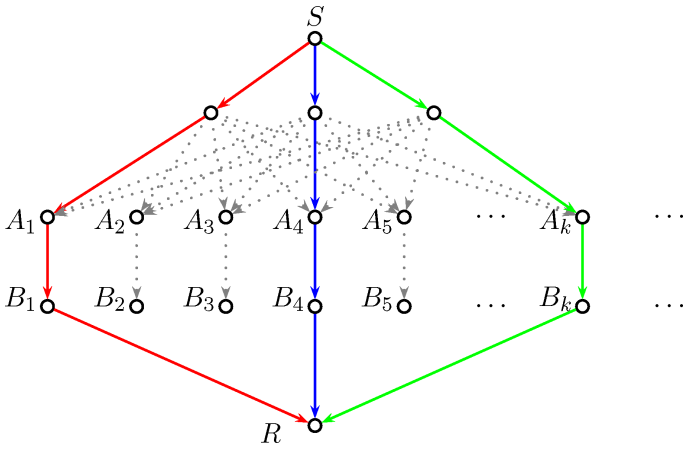
Additionally, all nodes in the network with min-cut equal to h would inherently receive information at this rate. That is, we do not need to differentiate the network operation, depending on whether we have one

or multiple receivers, provided that the transmitted rate is smaller or equal to the smallest min-cut of each intended receiver.

The network depicted in Figure 2.1 illustrates these points. Assume that at every time-slot, the receiver gets connected to a randomly chosen set of three B-nodes. The min-cut between the source and the receiver is always equal to three.



Case 1: The receiver R connects to a random set of three B-nodes.



Case 2: The receiver R connects to a specific set of three B-nodes.

Fig. 2.1 A random graph where at every time-slot, the receiver gets connected to a randomly chosen set of three B-nodes.

With routing, depending on which specific set of B-nodes the receiver gets connected to, we would need to select an appropriate routing scheme, a set of edge-disjoint paths, as for example for the graph realization in Figure 2.1-Case 2. This set of paths would change at every time-slot. With randomized network coding, we can send the same linear combinations of the source symbols for all time slots and graph realizations, and still ensure that the receiver receives information at rate equal to three. That is, nodes A can perform exactly the same operation, irrespective of the graph realization.

In fact, the network in Figure 2.1 provides an alternative proof for the coupons collector problem,[1] that we are going to describe in the next section and prove using simple combinatorics. In this sense, we can think of the benefits network coding offers for the coupons collector problem as a corollary of the main theorem in network coding.

2.2 Collecting Coupons and Linear Combinations

In the coupon collector problem, a coupon (e.g., a beanie baby) is drawn from a set of size N uniformly at random, and its copy is placed inside a box of some commodity (e.g., a happy meal box). An entire edition of boxes is created in this way and placed on the market (e.g., 2004 McDonald's Happy Meal Beanie Giveaway). A collector buys boxes in order to collect the N coupons. He may have a limited budget or limited time to spend acquiring his collection. Consequently, he is interested to know the following:

(1) The sample size S_r necessary for acquisition of r distinct coupons, and, in particular, the sample size necessary for acquisition of all N coupons.
(2) The waiting time W_r to acquire the rth distinct element.
(3) The collection size C_s (the number of different coupons) in a sample of size s. This quantity is of particular interest when the buyer's resources (e.g., time, money) are limited and he can acquire only so many coupon containing boxes.

[1] Simply think of the h unit rate sources as coupons, and of the A_iB_i edges as boxes.

Note that S_r, W_r, and C_s are random variables; we next compute their probability distributions. It is easy to see that once $r-1$ distinct coupons have been collected, the random process of acquiring the rth coupon is a Bernoulli process. In each trial the probability of success (drawing the rth distinct coupon) is

$$p_r = 1 - \frac{r-1}{N} \geq \frac{1}{N}, \quad 1 \leq r \leq N.$$

Note that the larger the number of already acquired coupons the smaller the probability of success. To acquire the rth coupon, our collector thus has to make on the average

$$E\{W_r\} = \sum_{n=1}^{\infty} k \cdot p_r (1-p_r)^{k-1} = \frac{1}{p_r} = \frac{N}{N-r+1}$$

draws. Again note that the larger the number of already acquired coupons the longer the time to wait to acquire a new one.

If the number of draws made (boxes bought) to acquire the rth coupon is W_r, the sample size at the point of acquiring the rth coupon is $S_r = W_1 + W_2 + \cdots + W_r$. Thus, the average sample size necessary for acquisition of r distinct coupons is

$$\begin{aligned} E\{S_r\} &= E\{W_1 + W_2 + \cdots + W_r\} \\ &= N\left(\frac{1}{N} + \frac{1}{N-1} + \cdots + \frac{1}{N-r+1}\right) \gtrapprox N \log \frac{N+1}{N-r+1}, \end{aligned}$$

where the bound is just the left Riemann sum of $1/x$ on the interval $[N-r, N]$, and becomes tight for large N. Thus the average waiting time to acquire N distinct coupons is bounded as

$$E\{S_N\} \gtrapprox N \log(N+1) \Rightarrow E\{S_N\} = N \log N + \Theta(N). \tag{2.1}$$

When a collector has time or money constraints which will permit him to acquire a sample of at most s coupons, he naturally wants to know how many different coupons C_s he has in this sample. To compute the expected value of C_s, we note that the probability of not having a particular coupon in a sample of size s is $\big[(N-1)/N\big]^s$, and therefore,

$$E\{C_s\} = N\big[1 - (1-1/N)^s\big] \gtrapprox N\big(1 - e^{-s/N}\big). \tag{2.2}$$

Suppose now that we are to collect not coupons or any physical commodity but information that can be duplicated, merged, or in general, processed in a way that physical entities cannot. More precisely, suppose that the goal is to collect N numbers $x_1, \ldots, x_N$ which are elements of some finite field $\mathbb{F}_q$. Suppose that a coupon collector is given a choice of two strategies to acquire the numbers. The first strategy is to draw a number with replacement, that is, the classical coupon collection procedure. In the second strategy, as a result of each draw a random linear combination over $\mathbb{F}_q$ of the numbers is made and the coefficients and the result of combining are revealed to the collector. Thus with each draw, the collector acquires a linear equation that the unknowns $x_1, \ldots, x_N$ have to satisfy, and his goal is to acquire a set of N linearly independent equations. If a draw results in a vector of equation coefficients that is linearly independent of those already acquired, we will say that the vector (coupon) was innovative. Similarly to the classical coupon collector problem, we next compute the probability distributions of the random variables S_r^c and W_r^c, where they now refer to innovative rather than distinct coupons, and superscript c indicates that coding is used.

We first compute the distribution for the expected waiting time W_r to acquire the rth innovative coupon. Again, once $r-1$ innovative coupons have been collected, the random process of acquiring the rth equation is a Bernoulli process. In each trial, any of the q^N vectors in $\mathbb{F}_q^N$ can be drawn. The trial is successful if the drawn vector is not one of the q^{r-1} that belong to the $(r-1)$-dimensional space spanned by the $r-1$ already acquired innovative vectors. Therefore,

$$p_r^c = 1 - \frac{q^{r-1}}{q^N} \geq 1 - \frac{1}{q}, \quad 1 \leq r \leq N.$$

Although it is still true that the larger the number of already acquired innovative coupons the smaller the probability of success, there is a nontrivial lower bound on this probability independent of n which can be made arbitrarily close to 1 by increasing the field size q. To acquire the rth innovative coupon, our collector has to make on average

$$E\{W_r^c\} = \sum_{n=1}^{\infty} k \cdot p_r (1-p_r)^{k-1} = \frac{1}{p_r} = \frac{q^N}{q^N - q^{r-1}} < \frac{q}{q-1}$$

draws. We can now compute the average sample size necessary for acquisition of r distinct coupons as

$$\begin{aligned} E\{S_r^c\} &= E\{W_1 + W_2 + \cdots + W_r\} \\ &= \frac{q^N}{q^N} + \frac{q^N}{q^N - q} + \cdots + \frac{q^N}{q^N - q^{r-1}}, \end{aligned}$$

and bound this quantity as

$$r < E\{S_r^c\} < r \cdot \frac{q}{q-1}.$$

Thus the average waiting time to acquire N innovative coupons (which is sufficient to compute the unknowns $x_1, \ldots, x_N$) is

$$E\{S_N^c\} \approxeq N \quad \text{as } q \to \infty. \tag{2.3}$$

The power of coding is seen by comparing this result with its counterpart (2.1) for the classical coupon collector problem. Again, $E\{S_N\}$ can be made arbitrarily close to N by increasing the field size q. Note, however, that the achieved reduction of the average waiting time comes at the price of increased complexity in having to solve a system of linear equations over a larger field.

Increased computational complexity is not the only drawback of using coding for information acquisition. When the collector's resources allow him to buy only N or fewer coupons, then he may not be able to recover any of the numbers $x_1, \ldots, x_N$ over any finite field. On the other hand, in the classical case (2.2), a sample size of N coupons will result in $N\big(1 - e^{-1}\big) > N/2$ distinct coupons on the average.

2.3 Gossip Algorithms for Information Dissemination

Gossip algorithms are used to disseminate information in a distributed fashion, and, as we will discuss, are closely connected with the coupons collector problem.

Consider a network represented as a graph $G = (V, E)$ with $n = |V|$ vertices. Each vertex has a message that it wants to disseminate to all other nodes in the network, using a gossip algorithm. The algorithm consists of two parts. The *gossip mechanism* determines how nodes establish a communication connection. The *gossip protocol* determines what information the nodes exchange during their communication.

(1) *Gossip mechanism:* Nodes communicate with each other in *rounds*. At each round, each node i selects with probability P_{ij} a node j of its neighbors. *Push* algorithms have node i send one message to node j, while *pull* algorithms have node i receive one message from node j.
(2) *Gossip protocol:* We discuss two cases.

   (a) Routing: each node required to transmit at round t selects uniformly at random one from the messages it has received up to round $t-1$ and sends it.
   (b) Random network coding: each node required to transmit at round t sends a uniform at random linear combination over $\mathbb{F}_q$ of the messages it has received up to round $t-1$.

The figure of merit of these algorithms is speed of dissemination: how many rounds are required so that all nodes receive all n messages, with high probability.

In the case where the communication graph is complete (every node has all other nodes as neighbors), and $P_{ij} = \frac{1}{n-1}$ for all i and j, the problem of disseminating the messages can be reduced to the coupons collector problem. Indeed, consider for example the pull algorithm, and the case of routing. Each specific node i acts as the coupon collector that, at each round, is going to receive uniformly at random one of the $n-1$ coupons that its neighbors have. Thus, each node will on the average need $(n-1)\log(n-1) + \Theta(1)$ rounds in order to receive all n messages. Consider now use of network coding. It is easy to show that the dissemination can be completed after $\Theta(n)$ rounds. A similar order benefit can be shown in the case of expander graphs. However, recent work has shown that for example over the square lattice graph, where each node has four neighbors, network coding can only offer constant factor benefits, as we discuss in Section 4.

In general, we can relate the speed of dissemination to properties of the network graph. Let

$$T(\delta) = \inf\{t : Pr(\text{any one of the receivers cannot decode}) < \delta\}$$

Also, define Φ_k as

$$\Phi_k = \min_{S \subset V, 0<|S|\leq k} \frac{\sum_{i\in S, j\notin S} P_{ij}}{|S|}$$

and

$$\hat{\mu} = \sum_{k=1}^{n} \frac{k}{\Phi_k}.$$

We give without proof the following theorem.

Theorem 2.1. Using random network coding for information dissemination allows to achieve

$$T(\delta) = O\left(\frac{\hat{\mu}}{n} \log \frac{1}{\delta}\right).$$

2.4 Network Coding Over Random Graphs

Consider a random graph $G = (V, E)$ that has a fixed set of vertices, but the set of edges E changes according to some random process, for example, during each time-slot. The difficulty of using network coding over a random graph for multicasting lies in that the min-cut between the source and the receivers might change during the different graph realizations. Up to this point,[2] we do not have yet an exact characterization of what rates we can achieve toward receivers that have different min-cut; moreover, once we characterize such rates, we would also need to find coding schemes that achieve them. In particular, it is not clear how well a decentralized protocol would perform.

However, in several well-behaved random graph models, the min-cut values tend to concentrate around an average value with high probability. In such cases, and assuming that at each time-slot we get a independent random graph realization, we can combine coding at the source (see part I, Section 3.6.3) and randomized network coding, to transmit to the receivers at a rate very close to the average min-cut value, asymptotically over time, and with probability of error going to zero (averaged over the receivers).

[2] Current date Summer 2007.

Notes

The coupon collectors problem is one of the most popular topics in discrete probability, and its description can be found in many standard textbooks on probability (e.g., [21, Ch. 9]) or algorithms (e.g., [61, Ch. 3]). The observation that network coding offers benefits for this problem was originally made by Deb et al. [12]. Information dissemination using network coding was studied by Mosk-Aoyamam and Shah [60]. Results for network coding performance over random graphs were investigated for example by Ramamoorthy et al. [66].

3

Content Distribution

Content Distribution (CD) on the Internet refers to the delivery of digital data such as text and multimedia files, software, and streaming audio and video to a large number of users in a network. Today it constitutes the vast majority of Internet traffic. The traditional approach to large scale content distribution relies on client–server systems in which the clients directly request and receive the content from the servers. To minimize the response time to clients' requests as well as to maximize the number of processed requests, a Content Distribution Network (CDN) geographically distributes a collection of server surrogates that cache pages normally maintained in some set of backend servers. Commercial CDNs (e.g., Akamai and Digital Island) provide this service for many popular commercial sites (e.g., CNN and The New York Times).

Peer-to-peer (P2P) networks offer an alternative distributed and cooperative architecture for content distribution. These are non-hierarchical computer networks that rely on the computing power and bandwidth of all the participants in the network rather than in a relatively low number of servers and client–server relationships. In a P2P network, all nodes receiving information assist in further distribution

of the content to other nodes by acting themselves as servers. Thus the network is inherently scalable: the nodes that join the network bring in not only new demands but also additional computing power and bandwidth. Moreover, there no longer exists a single point of failure (server crashing) in the system, as the information is distributed among multiple nodes throughout the network.

Having thus addressed the problem of ensuring sufficient network resources, P2P networks still face the challenge of how to optimally use these resources in a decentralized and low complexity manner. For example, optimal information routing is a difficult problem in such large scale networks that change very fast over time (with thousands of nodes joining and leaving within a short span of time) and where, moreover, nodes only have local information about the network.

BitTorrent is an example of a P2P system that uses swarming techniques to simultaneously disseminate different fragments of a file among peers. Acquiring a file by collecting its fragments can be to a certain extent modeled by the classic coupon collector problem, which indicates some problems such systems may have. For example, probability of acquiring a novel fragment drops rapidly with the number of those already collected. In addition, as the number of peers increases, it becomes harder to do optimal scheduling of distributing fragments to receivers. One possible solution is to use a heuristic that prioritizes exchanges of *locally rarest* fragments. But, such fragments often fail to match those that are *globally rarest*. The consequences include, among others, slower downloads and stalled transfers.

The Microsoft Secure Content Distribution (MSCD), also known as Avalanche, is an example of a P2P system attempting to alleviate such problems using network coding. Instead of distributing the original file fragments, peers produce linear combinations of the fragments they already hold. Such combinations are distributed together with a tag that describes the coefficients in the combination. When a peer has enough linearly independent combinations of the original fragments, it can decode and build the original file. Many of the advantages of Avalanche over BitTorrent can be understood from our previous discussion on collecting numbers as coupons vs. collecting numbers through their linear combinations.

We continue this section by first outlining the BitTorrent solution for P2P content distribution, since Avalanche adopts a number of ideas from this solution. We then explain how Avalanche works, its advantages over BitTorrent, and challenges it still faces.

3.1 BitTorrent Solution for P2P Content Distribution

3.1.1 Topology Management

For each file to be distributed, an overlay network of peers (nodes) is formed. Besides the general peers, there are some special nodes in this distribution network. These are the *registrar*, which enables peer discovery and keeps track of the topology, the *logger*, which aggregates peers' and registrar's trace messages, and the *seeds*, which are peers that have acquired the complete content.

The origin of the file (source) is the first node in the network. To join the network, peers contact the registrar, and are connected to a small number (usually 4–8) neighbors. The neighbors for each arriving node are chosen uniformly at random among already participating nodes, which accept the solicited connection unless they have already reached their maximum number of neighbors. Each node keeps local topological information, namely, the identity of the neighbors it is directly connected to. The registrar keeps track of the list of active peers. A peer may join and leave the distribution network at any time.

To mitigate formation of isolated peer islands (clusters), nodes periodically drop one neighbor and reconnect to a new one, asking the registrar to randomly select the new neighbor from the active peers list.

3.1.2 Content Propagation

The source splits the file to be distributed into N blocks. Each peer in possession of a block acts as a server for that block. The functionality of a peer consists of two parts: network transport and content management. Network transport maintains the local topology as described above. Content management maintains the actual content transfer.

Which of its blocks a peer transfers is decided based for example on one of the following common content propagation strategies:

- a random block in the beginning of the distribution, and
- a local rarest after a few blocks have been downloaded, or
- a global rarest in small P2P networks.

3.2 Microsoft Secure Content Distribution

Microsoft Secure Content Distribution (MSCD), that used to be called Avalanche, follows BitTorrent to a large extent. It follows, for example, the same basic principles for topology management. Here, however, randomized network coding is employed for content distribution, as is described in the following.

3.2.1 Content Propagation

The source splits the file to be distributed into N blocks (1000–2000 in experiments). These blocks are grouped into *generations* (segments), where each generation contains h blocks. Only packets in the same generation are allowed to be linearly combined. Each packet is assumed to consist of symbols over a finite field $\mathbb{F}_q$. This simply means that, for $q = 2^m$, a packet of length L bits is considered to contain L/m symbols over $\mathbb{F}_{2^m}$, where sets of m bits are treated as one symbol over $\mathbb{F}_{2^m}$. Typically, L is 1400×8 bits, and m is 8 or 16. Note that linear combining occurs symbol wise: each symbol in a packet is multiplied by the same number in $\mathbb{F}_q$ prior to adding to other (multiplied) packets.

The source transmits linear combinations of the original h file packets in a generation, while peers recursively and uniformly at random combine their collected packets and create new coded packets that they propagate through the network. Thus each coded packet carries a linear combination of the original file packets. This linear combination is described by the (global) *coding vector* of size h symbols over $\mathbb{F}_q$, that specifies which linear combination each coded packet carries. The generation tag and the coding vector are appended in the packet header. For generation size h, the overhead caused by coding is h/L. For a generation of size 50, the typical overhead is $50/1400 \approx 3\%$.

Nodes collect packets with linearly independent coding vectors. A packet is *innovative* for a node v if its coding vector is not in the span of the coding vectors of the packets already available at v. A node with h linearly independent packets of the same generation can decode the $h \times L/m$ symbols of the original file in that generation. To decode the original symbols, for each symbol position i, $i = 1, \ldots, L/m$, the node solves the following system of linear equations:

$$\mathbf{C}\boldsymbol{x}_i = \boldsymbol{b}_i, \quad i = 1, \ldots, L/m, \tag{3.1}$$

where

- $\boldsymbol{x}_i$ is the $h \times 1$ vector of the unknown content symbols at position i,
- $\mathbf{C}$ is the $h \times h$ matrix whose rows are the coding vectors of the h linearly independent packet, and
- $\boldsymbol{b}_i$ is the $h \times 1$ vector of the packets' symbols at position i.

3.2.2 Propagation Protocols

Peers who need to receive additional packets register demands with their neighbors. A peer with a new packet and registered demands acts as a sender, and peers who have registered demands with the sender become receivers. The transmission of packets proceeds as follows. The sender generates random coefficients that it intends to use for linear combining of its packets to form a new packet to be sent to the receiver. Based on these coefficients and the coding vectors of its packets, the sender then computes the corresponding coding vector of the new packet, and sends only this vector to the receiver. The receiver, if it finds the coding vector innovative, requests the packet from the sender, who only then linearly combines its packets using the previously generated coefficients, and sends the resulting packet to the receiver who requested it.

In another approach, it is arranged that the peers know the coding vectors of their neighbors. For each neighbor, a node computes the rank of the matrix consisting of his and the neighbor's coding vectors. This way, a node can learn not only who of its neighbors could provide

innovative packets but also how many innovative packets each neighbor could provide, and, thus, he can generate his requests accordingly.

Recall that in BitTorrent, the sender transmits either random, local rarest, or global rarest of his packets. In Avalanche, on the other hand, the sender does not estimate its packets' levels of representation in the neighborhood or in the network, but generates a random linear combination of all its packets.

3.2.3 Advantages

Avalanche exhibits a number of advantages over P2P distribution systems which do not use coding. Many of them can be understood from our previous discussion on collecting numbers as coupons vs. collecting numbers through their linear combinations. We have seen that the latter collecting strategy (which amounts to coding) guarantees a much faster acquisition of packets. In addition to the throughput increase, coding provides benefits in a number of P2P networks' phenomena arising because of the dynamics of peer participation, that is, random arrivals, departures, and connectivity. To help appreciate such benefits, we will use as an example the P2P distribution network depicted in Figure 3.1, that consists of two clusters. The source S has h packets, $\{x_1, \ldots, x_h\}$, to distribute to all nodes. Consider for example node C, that receives packets from its parent nodes A and B. With network coding, node C will, with high probability, receive innovative packets from both its parents. Without network coding, on the other hand, to ensure that node C receives innovative information from its parents, we need to ensure that nodes A and B collect disjoint subsets of source packets, which would require centralized control.

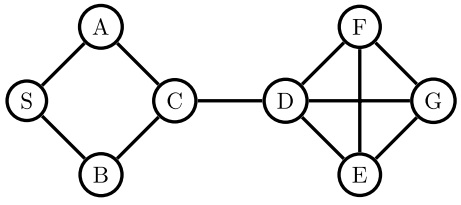


Fig. 3.1 The source S has h packets to distribute over a P2P network.

We next discuss a number of coding benefits in P2P networks. Some of them can be understood keeping the simple picture of the network in Figure 3.1 in mind.

3.2.3.1 Reliability

In P2P networks, packets get lost because of node departures and transmission losses. Without coding, some fragments of the file that is being distributed become *rare*. Such fragments can be lost forever, if the few nodes that possess them leave the distribution network. With coding, because of linear combining, the original file fragments are represented in a larger number of packets, and thus the probability of any particular fragment becoming rare is reduced.

Figure 3.2 examines whether peers are able to complete downloading a file, in the case where the server leaves after distributing one full copy of the file and 5% redundant packets. Peers leave immediately after downloading the file. The simulation results compare three systems: one that employs network coding (NC), one where the server

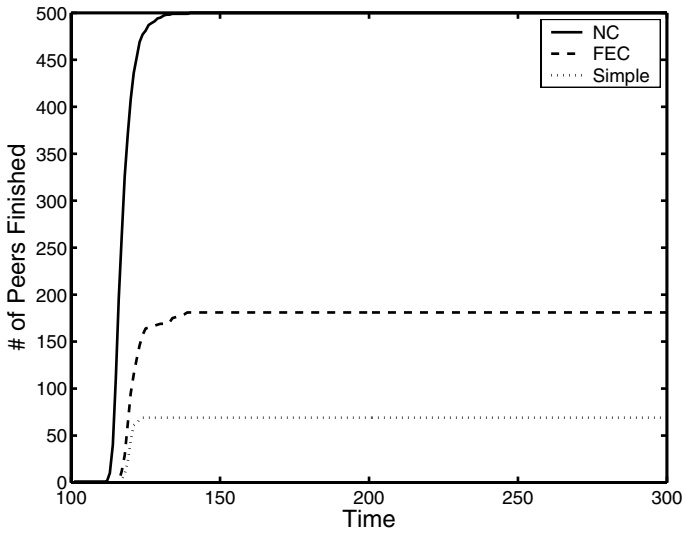


Fig. 3.2 Number of nodes that are able to finish downloading a file, in a system where the server leaves after transmitting the whole file and 5% redundant packets. Total number of peers is 500. This figure is provided by courtesy of the authors in [27].

sends packet encoded with a Forward Error Correction (FEC) scheme but peers simply forward packets, and a "simple" scheme where data is send uncoded by the server. Note that, unlike network coding, in the other two schemes the majority of nodes are not able to complete downloading.

3.2.3.2 Free Riding

Free riders are peers that use the network resources without contributing their own. BitTorrent mitigates this problem by the so called *tit-for-tat* mechanism, which allows download at a rate proportional to the rate of upload. Such strategies penalize new users joining the distribution network, who, naturally, may not have innovative information for their neighbors. Coding helps alleviate this situation, since a coded packet will be with higher probability useful to a larger number of nodes.

3.2.3.3 Churn

Churn in P2P networks, in general, refers to the independent arrivals and departures of numerous peers, and is a source of several problems. For example, imagine a scenario in which node A has downloaded 80% of the file when a new node B joins the network, and requests to get packets from their common neighbors. As a result, packets that A already has get downloaded again through some commonly shared paths, leading to increased delay for node A. Using network coding reduces this problem, since all transmitted packets can be made to carry information useful to a large number nodes.

3.2.3.4 Nodes Behind Walls

Network Address Translation (NAT) devices and firewalls which do not allow incoming connections create *unreachable peers*. Unreachable peers cannot exchange content with each other and do not share content which they advertise. A packet advertised by such nodes may become rare, because the packet is not requested from the source, under the false assumption it is available. As a result, both the download

performance of the nodes behind walls and the overall system throughput is not as large as it could be based on the total existing resources in the network.

However, it has been observed in experiments with network coding that the throughput under this way reduced connectivity is fairly close to that achieved with full connectivity, and that Avalanche systems perform surprisingly well even with a large number of unreachable peers.

3.2.3.5 Clustering

Clusters arise in P2P connectivity graphs because of the way these networks are formed. Reconfiguration then becomes minimal because clustered peers have a large number of neighbors and will not accept new requests for connection. In such scenarios, connectivity and bandwidth are ample within a cluster but very limited between different clusters. It has been observed in experiments that coded systems perform much better than uncoded in these circumstances. The reason is that without coding, some packets are transmitted multiple times over the cuts between clusters, thus wasting precious capacity that could have been used to transmit coded packets likely to be innovative for a larger number of nodes.

3.2.3.6 Passive Topology Management

The performance of P2P networks depends critically on the good connectivity of the overlay topology, and as mentioned earlier can be significantly affected by the formation of clusters.

Randomized rewiring, where nodes are periodically allocated new neighbors chosen randomly among the active peers, offers a simple solution to this problem. However, it results in a fixed average number of reconnections per node independently of how good or bad the formed network topology is. Thus to achieve a good, on the average, performance in terms of breaking clusters, it entails a much larger number of rewiring and requests to the registrar than required, and unnecessary topology changes. Topology rewirings incur delay, require authentication, and increase vulnerability to security attacks.

For P2P networks employing randomized network coding, we can use the structure of the exchanged packets to passively infer cluster formation. Such information can be used to break clusters, using the minimum number of topology rewirings. In particular, it can be used to build algorithms that (i) identify and re-connect only nodes whose re-wiring leads to breaking of clusters, (ii) use a variable number of reconnections, adapting (without centralized knowledge) to the requirements of the network topology, and (iii) are peer-initiated and scalable.

The example in Figure 3.1 can be used to illustrate the basic approach. When node D receives a coded packet from node C, it will forward a linear combination of the packets it has already collected to nodes E, F, and G. Now each of the nodes F and E, once it receives the packet from node D, it also attempts to send a coded packet to node G. But these packets will not bring new information to node G, because they will belong in the linear span of coding vectors that node D has already received. More formally, the coding vectors nodes D, E, and F will collect will effectively span the same subspace; thus the coded packets they will offer to node G to download will belong in significantly overlapping subspaces and will thus be redundant. Node G can use this passively collected information, to initiate a topology change, by contacting for example the registrar, or by locally exchanging messages with its neighbors.

3.2.4 Challenges

3.2.4.1 Encoding/Decoding Complexity

With network coding, peers must have the capability to perform arithmetics over finite fields in real time, in order to examine whether a packet is innovative as well as to further encode and forward, or decode data. In particular, peers will perform the following:

- $\mathcal{O}(h^2)$ operations in $\mathbb{F}_{2^m}$ for linear combining of packets within each generation of size h.
- $\mathcal{O}(m^2)$ binary operations for multiplications and inversions over $\mathbb{F}_{2^m}$.

- $\mathcal{O}(h^3)$ operations over $\mathbb{F}_{2^m}$ for matrix inversions via Gaussian elimination.

Moreover, if while decoding or encoding the peer happens to have small memory, the speed of these operations may be further limited by the I/O devices speed.

3.2.4.2 Security

Coding across packets makes the overall system more vulnerable to poisoning attacks from malicious peers. To see this, consider the system of equations (3.1) for some $1 \leq i \leq L/m$. A node will correctly decode the original file symbols at position i (i.e., recover symbols $\boldsymbol{x}_i$ in (3.1)), only if all h equations are trustworthy (i.e., values of $\boldsymbol{b}_i$ and $\boldsymbol{C}$ in (3.1) are correct). Even if only one of the equations is false, the solution may result in false values for all h symbols in the vector $\boldsymbol{x}_i$. Consequently, the network becomes more vulnerable to DoS attacks. In addition, since multiple packets get combined, a single undetected corrupted packet can spread the infection very fast through the network. Therefore, it is important to put in place mechanisms that verify that the received packets are not corrupt.

In current P2P systems, the source cryptographically signs its data packets to ensure their integrity, for example by hashing the packets content and distributing the hashes through a trusted vein. This method will not work when network coding is used, where peer nodes create new encoded packets. To ensure the packets' integrity in network coded P2P networks, one approach uses special homomorphic hash functions. These have the property that the hash function corresponding to a linear combination of packets can be computed from the hash functions of the individual packets. This is a computationally demanding approach, and some other lower complexity solutions have also been proposed.

3.2.4.3 Peer Heterogeneity

The nodes in distribution systems have diverse capabilities. In terms of the link capacity, most of the users are slow, that is, they access

the network through e.g., dial-up and ADSL connections. On the other hand, the users accessing the network from university and corporate systems are fast. In a P2P environment, fast nodes are allowed to have more neighbors to fully utilize their extra capacity. However, the higher the number of neighbors feeding a fast node, the harder it is for the large set of neighbors to efficiently provide useful data in an uncoordinated fashion. Naturally, network coding is expected to perform better in these circumstances, and this has been confirmed in experiments.

However, nodes have heterogeneous not only access capacities, but also computational power, battery life, content quality or demands. For example, a user may be willing to spend only a limited time in the distribution looking to acquire only the beginning of a movie, or only its low resolution version. Coding, in general, may be counterproductive in such scenarios. To see that, recall our information dissemination study in the previous section. When the collector's resources allow him to buy only h or fewer linear combinations (packets), then he may not be able to recover any of the numbers $x_1, \ldots, x_h$ over any finite field. On the other hand, in the classical case, a sample size of h coupons will result in more than $h/2$ distinct coupons on the average.

Notes

There is vast literature on content distribution networks. See, for example [37] for comprehensive information on many aspects of CDNs and [80] for a survey on the technical aspects of replication. Surveys on P2P systems can be found in e.g., [2, 5]. Some P2P systems have already been successfully deployed on the Internet; see [29] for a description of Gnutella, [62] for Napster, and [10] BitTorrent. Several P2P systems have been proposed in academia with varying goals and architectures [9, 49, 59, 69, 74, 82, 93]. The MS Avalanche system and use of network coding for content distribution has been proposed by Gkatnsidis and Rodriguez [27], and various tests and measurements of this system are described in [28]. These papers also present some approaches to handle certain security issues in Avalanche. A practical scheme, based on homomorphic hashing, that enables a downloader to perform on-the-fly

verification of erasure-encoded blocks is described in [48]. Passive inference of clusters and topology management for network coded P2P systems has been proposed by Jafarisiavoshani et al. [40]. An alternative approach to uniformly at random combining of packets for P2P distributed storage systems has recently been proposed by Dimakis et al. [13].

4

Network Coding for Wireless Networks

Two main features that distinguish wireless from wireline networks are the shared medium and time variability. By increasing its transmission power, a node will eventually reach sufficient signal-to-noise ratio to be able to transmit to every other node in the wireless network, but it will at the same time create significant interference to other nodes. Moreover, channels vary over time because of, for example, fading or node mobility. Finally, resources such as computational power or battery life are often limited in wireless networks.

Most networking protocols operating today use the wireless medium to create point-to-point connections, and do not exploit the broadcasting capabilities of wireless transmission. The main focus of network design has been on simplicity of operation and scalability. On the other hand, recent work on information theory of, for example, relay networks shows that significant savings of physical resources can be made by broadcasting with appropriate coding schemes. However, the complexity of the proposed coding schemes (such as random hashing) is very high, and these schemes, in general, do not scale gracefully with the size of the network.

Network coding which exploits the broadcasting nature of wireless medium is considered to be a technologically sensible step from the state of the art wireless systems toward practical information theoretic based solutions. Indeed, algebraically superimposing signals is a form of hashing, which allows to take advantage of the broadcasting nature of the wireless medium; it is also simple enough to be implemented in practice. In a sense, network coding over wireless has a large overlap as an area with network information theory as well as network algorithms. Ad-hoc wireless and sensor networks are expected to offer one of the first practical applications for network coding, as network environments with less rigid protocols.

The work in wireless networks started by illustrating benefits network coding can offer by taking advantage of the broadcasting capabilities of the shared wireless medium to provide benefits in terms of bandwidth, transmission power, and delay, as well as adaptability to dynamically changing environments. We discuss such benefits for a number of traffic patterns and example network configurations in Sections 4.1–4.4. MIT's COPE demonstrated that even when coding operations are confined to simple binary additions obeying some additional constraints, there are still gains to be had in terms of throughput and efficiency of MAC layer protocols.

These first approaches treat interference at a receiver as detrimental to the systems performance, and rely on appropriately scheduling the broadcast transmissions to minimize it. One can show that such strategies can incur significant losses in terms of achievable rates. Physical layer network coding, described in Section 4.5, was a first attempt to remedy this in a heuristic fashion. Very recently, a linear deterministic model was developed that captures the interactions between the signals in a wireless network, and was shown that for such models one can obtain an information-theoretic max-flow min-cut result, which we briefly describe in Section 4.6.

Wireless and sensor networks provide vast opportunities for applications of network coding, and numerous and diverse problems are beginning to receive attention. We discuss some sensor network applications in Section 4.7, and conclude the section by briefly discussing challenges and open problems.

4.1 Energy Efficiency

Energy efficiency of transmission schemes directly affects the battery life of wireless nodes, and is therefore a critical design parameter for wireless ad-hoc and sensor networks. We here discuss how combining broadcasting with coding may lead to more energy efficient schemes. We have already seen one such example in [24], Section 1 (see also Figure 4.9), where network coding was used to make each broadcast transmission maximally useful to all receiving nodes.

4.1.1 Cost LP Formulation for Broadcasting

Optimizing a multicast session for energy efficiency can be formulated as a cost minimizing linear program, whose solution can be found in polynomial time.

The first step is to model the broadcasting constraint using a network represented as a graph. To do so, we can replace every vertex v of the graph with two vertices v_I and v_O and connect them by an edge directed from v_I to v_O, as depicted in Figure 4.1. The $\texttt{In}(v)$ incoming edges to v end in v_I, and the $\texttt{Out}(v)$ outgoing edges of v, originate in v_O. All edges of the graph have unit capacity. The next step is to

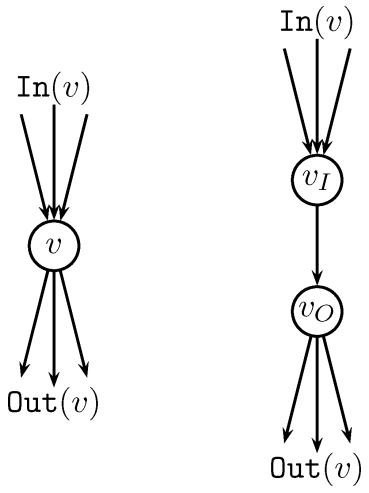


Fig. 4.1 Modeling broadcast transmissions on a graph.

associate cost with using the edge (v_I, v_O) to model the energy expenditure during transmission at node v. We can now directly use on this modified graph the Network Coding Multicast with Cost LP described in [24], Section 3.5.

4.1.2 Benefits for All-to-All Transmission

We next discuss the benefits network coding offers in the wireless network scenario in which each node is a source that wants to transmit information to all other nodes. In particular, we calculate the benefits for the ring and square grid networks. Such all-to-all communication is traditionally used during discovery phases, for example by routing protocols. More recently, it has been described as a key mechanism for application layer communication in intermittently connected ad-hoc networks. It is also directly related to the problem of content distribution. As our figure of merit, we use energy efficiency defined as the number of transmissions required for an information unit to reach all nodes in the network.

Theorem 4.1. Consider a fixed ad-hoc wireless network where each node's broadcast is successfully received by a constant number of neighbors. For the application of all-to-all transmission, network coding offers constant energy efficiency benefits.

Proof. Let n be the number of nodes in the network, and $N_{\max}$ the maximum number of neighbors a node can have within its transmission radius. The proof follows from two observations:

(1) There exists a routing scheme (not necessarily optimal) that achieves the goal in n^2 broadcasts. This is because each of the n nodes needs to broadcast a message to its neighbors at most once for each of the n messages that are to be disseminated.
(2) Any network coding scheme will require at least $n^2/N_{\max}$ transmissions. This is because each of the n nodes needs to receive n innovative transmissions, and each broadcast transmission brings innovative information to at most $N_{\max}$ nodes. □

For canonical configurations, such as networks where nodes are placed on a lattice, we can exactly calculate the benefits in terms of energy efficiency that network coding can offer. We illustrate this claim by two examples: the ring network and the rectangular grid network.

Theorem 4.2. Consider a ring network where n nodes are placed on equal distances on a circle, and each node can successfully broadcast information to its two closest neighbors. Let T_{nc} and T_w denote the minimum possible number of transmissions required for an information unit to reach all nodes in the network, with and without network coding, respectively. Then

(1) $T_w \geq n - 2$, and there are schemes that achieve the lower bound,
(2) $T_{nc} \geq (n-1)/2$, and there are schemes that achieve the lower bound.

Thus, $\lim_{n\to\infty} \frac{T_{nc}}{T_w} = \frac{1}{2}$.
The results do not change if we increase the transmission range.

An example of such a network for $n = 8$ is depicted in Figure 4.2.

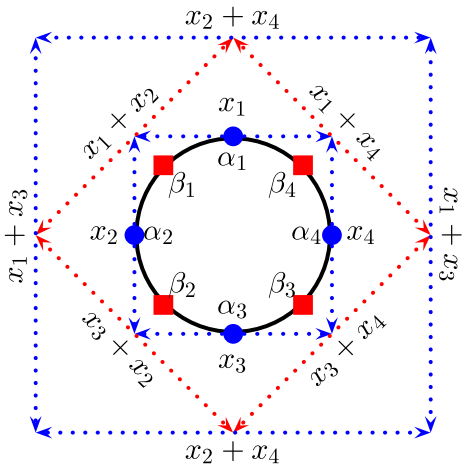


Fig. 4.2 Circular network with $n = 8$ nodes.

Proof. To prove claim (1), we can consider w.l.g a single source transmitting to $n-1$ receivers. When limited to forwarding, the first transmission reaches two receivers. Each additional transmission can contribute one unit of information to one receiver. A simple flooding algorithm achieves the lower bound.

To prove claim (2), we notice that since a node can successfully broadcast to its two nearest neighbors, each broadcast transmission can transfer at most one innovative packet (information unit) to two receivers. We have $n-1$ receivers to cover and thus the best energy efficiency we may hope for is $(n-1)/2$ per information unit. The following scheme achieves the lower bound. Assume that n is an even number. Partition the n nodes in two sets $A = \{\alpha_1, \ldots, \alpha_{\frac{n}{2}}\}$ and $B = \{\beta_1, \ldots, \beta_{\frac{n}{2}}\}$ of size $n/2$ each, such that every node in A has as nearest neighbors two nodes in B, as depicted in Figure 4.2. It is sufficient to show that we can broadcast one information unit from each node in set A to all nodes in sets A and B using $T_{nc} = n/2$ transmissions per information unit. We can then repeat this procedure symmetrically to broadcast the information from the nodes in B. Let $\{x_1, \ldots, x_{\frac{n}{2}}\}$ denote the information units associated with the nodes in A. The following algorithm operates in $n/4$ steps, where in each step first nodes in A transmit and nodes in B receive and then nodes in B transmit and nodes in A receive.

Network Coding for Circular Grid

Step k:

- Phase 1:

  If $k = 1$, each $\alpha_i \in A$ transmits its information symbol x_i.

  If $k > 1$, each $\alpha_i \in A$ transmits the sum of the two information symbols it received in phase 2, step $k-1$.

- Phase 2:

  Each $\beta_i \in B$ transmits the sum of the two information symbols it received in phase 1, step k.

At step k, Phase 1, each node in B is going to receive two new information symbols from the two sources that are $2k-1$ nodes away along

the circle.[1] In Phase 2, each node in A is going to receive two information units from the sources that are $2k$ nodes away. Since the network coding algorithm concludes in at most $n/4$ steps and ensures that each broadcast transmission brings new information to two receivers, the result follows. □

Energy efficiency benefits for square grid networks can be calculated by using similar arguments.

Theorem 4.3. Consider the optimal routing and network coding strategies that minimize the number of transmissions for the all-to all problem over a square grid network with n nodes where each node can successfully broadcast information to its four nearest neighbors. Then

$$\lim_{n\to\infty} \frac{T_{nc}}{T_w} = \frac{3}{4}.$$

It is interesting to note that, a very simple randomized scheduling, such as the one described below, achieves the optimal network coding performance.

Distributed Network Coding for the Square Grid Network

- Iteration 1:
  Each node broadcasts the information symbol it produces to its four closest neighbors.
- Iteration k:
  Each node transmits a linear combination of the source symbols that belongs in the span of the coding vectors the node has received in previous iterations.

Note that, the benefits calculated in Theorem 4.3 assume perfect centralized scheduling, for the case where we do not use network coding.

[1] For simplicity of notation, we assume that all indices are mod $\frac{n}{2}$. Also note that for $n-1$ odd we cannot achieve $(n-1)/2$ transmissions but $n/2$, however this does not affect the order of the result.

4.1.3 Distributed Algorithms For All-to-All Transmissions

Here we describe a very simple protocol for all-to-all transmissions for arbitrarily fixed networks where nodes do not have any a priori knowledge about the network topology. For example, we may assume that the network nodes are randomly placed on the surface of a disk. To account for these factors, and given the randomized nature of our network topology, we use a protocol in analogy to probabilistic routing that forwards packets with a certain probability, according to a *forwarding factor* d_v different for every node v, as described in the following Algorithm.

Forwarding Factor d_v

Each node maintains a send counter s, that is initially equal to zero.

- For each source symbol that originates at a node v, the node increases s by $\max(1, \lfloor d_v \rfloor)$, and it further increases s by one with probability $p = d_v - \max(1, \lfloor d_v \rfloor)$ if $p > 0$.
- Similarly, when a node v receives an innovative symbol, it increases s by $\lfloor d_v \rfloor$, and it further increases s by one with probability $p = d_v - \max(1, \lfloor d_v \rfloor)$ if $p > 0$.
- If $s \geq 1$, a node attempts to broadcast a linear combination over the span of the received coding vectors. Each transmission reduces the send counter s by one.
- In addition to updating the send counter, nodes can also keep track of received non-innovative packets. For each c non-innovative packets a node receives, the send counter s is decremented by one.

If we have some topological information, we can set the value of d_v to help to adapt to irregularities of the network topology. A heuristic choice that resulted in good performance is to set v's forwarding factor inversely proportional to the number $N(v)$ of 1-hop neighbors, that is, $d_v = k/|N(v)|$, for k some constant.

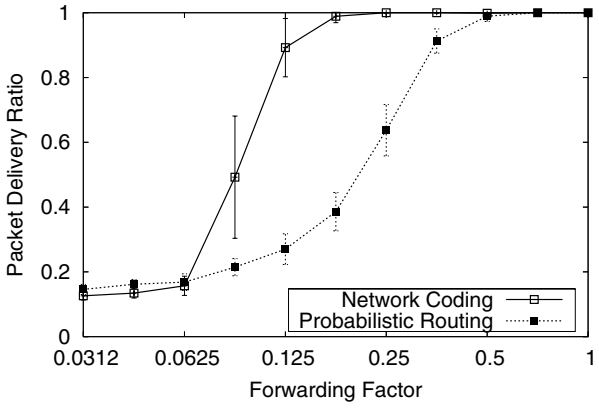


Fig. 4.3 Packet delivery ratio for probabilistic network coding and routing. All nodes use the same forwarding factor. With network coding, the transition to high packet delivery ratios occurs for a much lower forwarding factor, and thus, a smaller number of transmissions.

Figure 4.8 shows simulation results for a network with $n = 144$ nodes that are placed uniformly at random on a simulation area of 1500 m $\times$ 1500 m. The node transmission range is 250 m. Finite field operations are over $\mathbb{F}_{2^8}$. The MAC layer is an idealized version of IEEE 802.11 with perfect collision avoidance. At each time unit, a schedule is created by randomly picking a node and scheduling its transmission if all of its neighbors are idle. This is repeated until no more nodes are eligible to transmit.

4.2 Fairness and Delay

An inherent characteristic of wireless networks is the time variability of the received signal quality due to fading and interference. The random fluctuations at the physical layer are perceived as packet erasures at higher layers, and may result in variability of the reception rates over short time periods. Varying reception rates are seldom tolerated by real-time applications, and often require involved scheduling schemes to ensure fairness over short time periods. Reducing the variability of packet delivery rates may also serve to decrease the problem of window closing, for instance in TCP.

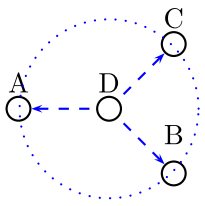


Fig. 4.4 Nodes A, B, and C receive information from node D.

Combining network coding and broadcasting over such time varying environments may "smooth over" the rates that the receivers experience over short time periods. We illustrate this point through the following simple example shown in Figure 4.4. Basestation D has three independent streams of packets $x_A = \{x_A^i\}$, $x_B = \{x_B^i\}$, and $x_C = \{x_C^i\}$ to transmit to nodes A, B, and C, respectively, that are within its broadcast radius. In wireless systems today, independent information streams are transmitted in orthogonal dimensions, using time, frequency or code division multiple access schemes. Assume, for example, that the base-station uses timesharing to sequentially broadcast information to each of the receivers. A broadcast transmission successfully reaches each receiver according to independent Bernoulli distributions with probability p. The basestation has no a priori channel information, but the receivers can acknowledge successful reception.

Table 4.1 shows a possible scenario. After six time slots node A has received no packet from the stream it is interested in, node B receives one packet and node C receives two packets. Note that during the first time slot the basestation broadcasts a packet destined to node A: although node A does not receive it, node B does receive it but has no use for it. The network coding solution,[2] also included in the same table, capitalizes on such situations: the basestation transmits linear combinations such that each transmission offers some benefit to all nodes that receive it.

[2] The network coding solution can also be thought of as a form of superposition coding for a broadcast channel.

Table 4.1 Basestation D broadcasts information destined to nodes A, B, and C over iid erasure channels with probability of erasure 0.5. The network coding solution leads to the same aggregate throughput, but more evenly divided among the receivers.

	Time slot					
	1	2	3	4	5	6
Round Robin Routing Solution						
D transmits	x_A^1	x_B^1	x_C^1	x_A^1	x_B^2	x_C^2
A receives	–	–	x_C^1	–	x_B^2	x_C^2
B receives	x_A^1	–	x_C^1	–	x_B^2	x_C^2
C receives	–	x_B^1	x_C^1	x_A^1	–	x_C^2
Network Coding Solution						
D transmits	x_A^1	x_B^1	x_C^1	$x_A^1 + x_B^1 + x_C^1$	$x_A^1 + 2x_B^1 + 3x_C^1$	$x_A^1 + 4x_B^1 + 5x_C^1$
A receives	–	–	x_C^1	–	$x_A^1 + 2x_B^1 + 3x_C^1$	$x_A^1 + 4x_B^1 + 5x_C^1$
B receives	x_A^1	–	x_C^1	–	$x_A^1 + 2x_B^1 + 3x_C^1$	$x_A^1 + 4x_B^1 + 5x_C^1$
C receives	–	x_B^1	x_C^1	$x_A^1 + x_B^1 + x_C^1$	–	$x_A^1 + 4x_B^1 + 5x_C^1$

Note that we do not get any benefits (or hits) in terms of throughput: this is because, we are effectively now requiring that all receivers receive all information (so three times as much rate), using three times more the channel. The benefit we get is that the rate individual users experience converges faster toward the average value, i.e., one symbol for every six time slots. More formally, let T be a random variable that denotes the total number of transmissions the base-station will need to convey the L packets to node say A, where packets get dropped independently with probability p. From basic probability theory, the base-station will need on the average $E(T) = L/(1-p)$ attempts. As T grows, the observed empirical distribution will converge to the actual distribution. That is, if T_0 denotes the number of erasures after T transmissions,

$$Pr\left\{\left|\frac{T_0}{T} - p\right| > \epsilon\right\} \to 0$$

exponentially fast. If k receivers share the transmission medium, each one of them will observe erasures over kT instead of T time slots, and thus a distribution that converges much faster to the average.

4.3 Adaptability to Dynamically Changing Networks

As we discussed in Section 2, network coding can offer significant benefits in terms of operational complexity in dynamically changing environments, such as wireless networks which constantly change because nodes move, turn on and off or roam out of range. In such environments, we are often restricted to use very simple distributed algorithms to avoid costs of storing and especially updating constantly changing network information. In this section, we discuss some such examples.

We start by considering the all-to-all transmission scenario, but now we assume uniform at random mobility of nodes. We can model this idealized mobility by assuming that time is divided into time slots (or iterations): at the beginning of each time slot, nodes are placed uniformly at random on a unit area disk. Each node can successfully broadcast information within a radius of $\Theta(1/\sqrt{n})$. Consider two scenarios:

Perfect knowledge of the neighborhood: Each node, before transmitting, knows who its neighbors are, and what information they have already successfully received.

No-knowledge: Nodes have no information of their neighbors identity or past history. In the latter case, and the case of forwarding, without loss of generality, we can assume that during each iteration, and at each (possibly new) position, node i always broadcasts x_i. For the case of network coding, each node transmits a random linear combination over some finite field $\mathbb{F}_q$ of the packets it has previously received.

Example 4.1. In Figure 4.4, assume that all nodes are interested in receiving all information, but node D is mobile and does not know the identity of nodes within its transmission range. A possible random mobility pattern is depicted in Figure 4.5. During time-slot 3 node D may not broadcast innovative information to node A. Using network coding alleviates this problem.

Mobility has a significant effect on forwarding. Initially, as nodes randomly move, the information is disseminated faster than in the case of

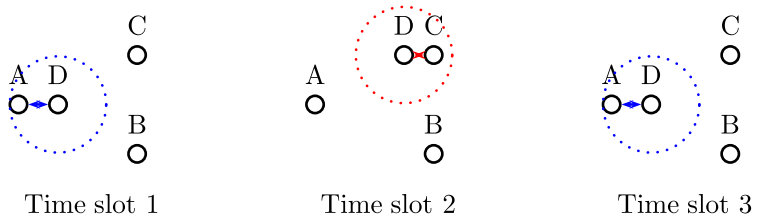


Fig. 4.5 Node D broadcasts information without knowing which nodes are within its range.

a static network. However, because of the assumption that *nodes do not know what information their neighbors have*, as approximately half the nodes collect the information, increasingly frequently, transmissions do not bring new information to the receiving nodes. This phenomenon has been observed in rumor spreading algorithms over networks.

Theorem 4.4. Let T_{nc} and T_w denote the required number of transmissions for the scenario with and without network coding. We assume uniform at random mobility, where time is divided into time-slots (iterations), and at the beginning of each time-slot nodes are placed uniformly at random on a unit area disk. We also assume no knowledge of the network topology. Then on the average,

$$\frac{T_{nc}}{T_w} = \Theta\left(\frac{1}{\log n}\right).$$

Proof Outline: Consider first the case of forwarding, and a particular node j that would like to transmit its message x_j to all other $n-1$ nodes. Construct a bipartite graph in which these $n-1$ nodes constitute one set of vertices. In the other set of vertices we have M nodes v_i, where node v_i corresponds to iteration (time-slot) i and is connected to the neighbors of node j during this iteration. Thus the degree of node v_i is a random variable with average k, here k is the expected number of neighbors for each node. We will here assume that k is a constant.

We are asking, how many right-hand side nodes we need, i.e., which number of iterations M, so that node j transmits its message to all other nodes. This simple analysis has been performed in the context of LT and Raptor codes where it was shown that M should scale as

$\Theta(n \log n)$. It is easy to see that the average number of transmissions we will need equals

$$\Theta(n \log n).$$

In the case of network coding on the other hand, where at each time slot a node transmits uniform at random linear combinations of what it has already collected, $\Theta(n)$ transmissions per node are sufficient. Indeed, if we collect the observations of each node in a matrix, this matrix will with high probability have full rank.

This problem can also be cast as a variation of the coupon collector's problem, where now each box does not contain exactly one coupon, put on the average a constant number of coupons. □

We conclude that:

- Network coding, under the "no-knowledge" assumption achieves the same performance as routing under the "perfect-knowledge" assumption. Thus routing information updates are not needed.
- Under the "no-knowledge" assumption for both network coding and routing, network coding reduces by a factor of $\log n$ the number of required transmissions. In this case, we get benefits in terms of number of transmissions that is proportional to energy efficiency.

The uniform at random mobility model, considered in Theorem 4.4, implies that the composition of the neighborhood of a node is completely uncorrelated from iteration to iteration. In practice, this is true only when the node speed is very high or the packet transmission rate is very low. A less generous mobility implies that less data is transported through the network by node mobility and has instead to be forwarded via intermediate nodes.

Figure 4.6 plots simulation results for a more realistic mobility pattern, the random-waypoint mobility model. In this model, nodes pick a random destination whose location is uniformly distributed in the simulation area as well as a movement speed with which they travel

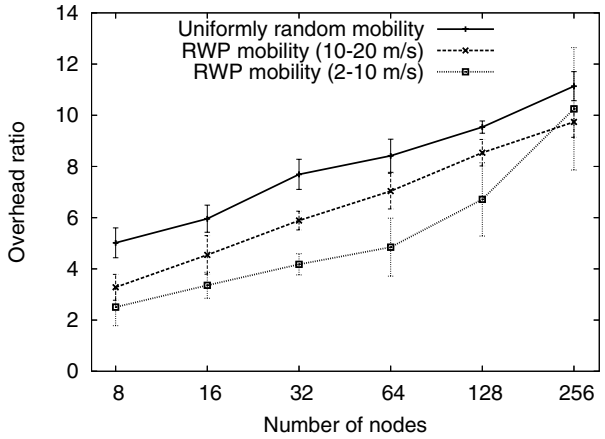


Fig. 4.6 Ratio of flooding overhead to network coding overhead for random waypoint mobility.

until the destination is reached. Our results assume no pause time and movement speeds uniformly distributed between 2 and 10 m/s as well as 10 and 20 m/s, respectively.

We can see that in this case, although network coding still offers benefits that increase with the number of nodes n, the performance gap with routing is smaller. This agrees with our intuition that, when mobility is more restricted, the network coding performance will decrease, because how well the data is "mixed" plays a crucial role for the network coding analysis.

Finally, although our discussion up to now was in the context of ad-hoc wireless networks, similar benefits are also possible in environments where we need to broadcast information to a set of receivers in a distributed manner and without knowledge of the network topology. The following example illustrates one more such case.

Example 4.2 (Broadcasting in Cellular Networks). We consider a cellular network model with m base-stations and n mobile phone receivers. The base-stations have K information units that they want to transmit to all mobiles. We assume that the transmission range is the same for all base-stations, each transmission conveys one unit of

information, and that the coverage areas of the base-stations do not overlap.

In this model base-stations are always active, while nodes are mobile and may turn on and off. A node is active with probability p, and successfully receives information approximately $M(1-p)$ out of M iterations. Thus, if base-stations broadcast using an erasure correcting code of rate $(1-p)$, then each transmission brings useful information at each node. For a node to receive K messages, we need $K/(1-p)$ iterations, on the average.

In the case of forwarding, assume that base-stations randomly select and transmit one of the K messages. Thus each node at each iteration observes one of the messages uniformly at random. We can think of this problem as a balls-in-bins experiment, where the bins are the K messages the node wants to collect, and the balls correspond to the iterations. Using standard balls and bins results we again need on the average $\frac{K \log K}{(1-p)}$ iterations. Thus, network coding offers a $\log n$ benefit.

4.4 COPE: Opportunistic Throughput Benefits

COPE is a proposed system architecture that implements opportunistic algorithms for network coding in wireless networks. Consider a wireless mesh network that needs to accommodate multiple unicast sessions, and where there is no centralized knowledge of the network topology or traffic patterns. The basic idea is that, network nodes might still recognize and exploit opportunities where network coding operations can offer benefits.

Such an example is illustrated in Figure 4.7. Consider two unicast sessions, both of which use node A as a relay in the path from the source to the destination. Assume that D_2 is within the transmission radius of S_1, and similarly D_1 within the transmission radius of S_2. Then when S_1 transmits x_1 to the relay A, this broadcast transmission is also received by D_2. Similarly when S_2 transmits x_2. Now the relay A simply needs to broadcast $x_1 + x_2$, for both destinations to be able to decode their desired information. This example opportunistically packs in the network and reproduces the benefits of the configuration in Figure 4.9.

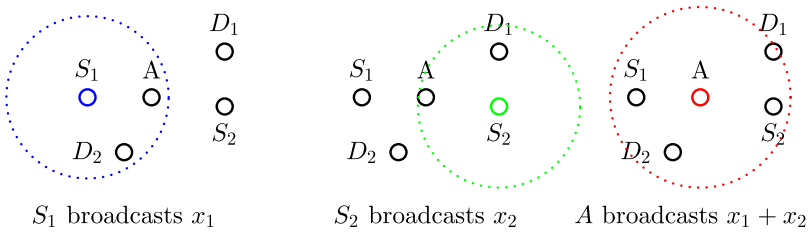


Fig. 4.7 Sources S_1 and S_2 transmit information to their respective destinations D_1 and D_2 through the relay node A.

The approach can be extended in more complex situations, where in general a node A has a list of symbols, e.g., $\{x_1, \ldots, x_m\}$, to broadcast to its neighbors. We assume that all nodes advertise the identity of the packets they have successfully received. Thus node A maintains a list of the symbols that each of its neighbors has. For example, node D_1 may have x_1 and x_2, node D_2, x_3 and x_4, and node D_3, x_1 and x_3. The idea is that based on this information, node A broadcasts a linear combination of symbols such that all neighbors receiving that linear combination can either immediately decode or drop the received packet with no information loss. In our example, node A may choose to transmit $x_1 + x_3$, which enables node D_1 to decode x_3, D_2 to decode x_1, and node D_3 to ignore the packet. Note that this scheme does not require that all nodes necessarily receive innovative information. However, it ensures that decoding always occurs in one hop from the source. Thus no additional delay is caused from nodes having to store packets they cannot immediately use for decoding. Figure 4.8 compares the throughput performance of this algorithm with traditional forwarding over IEEE 802.11.

In addition to increasing throughput, network coding offers additional benefits in these networks by "equalizing" transmission rates and thus helping MAC layer protocols, such as TCP, operate more efficiently. Indeed, such protocols, attempting to be fair, divide equally the bandwidth between competing nodes. However, for the example in Figure 4.7, without network coding node A needs to transmit twice as fast as the remaining nodes to ensure the fastest information flow. Under TCP, node A would not be allowed to that, and thus would become a

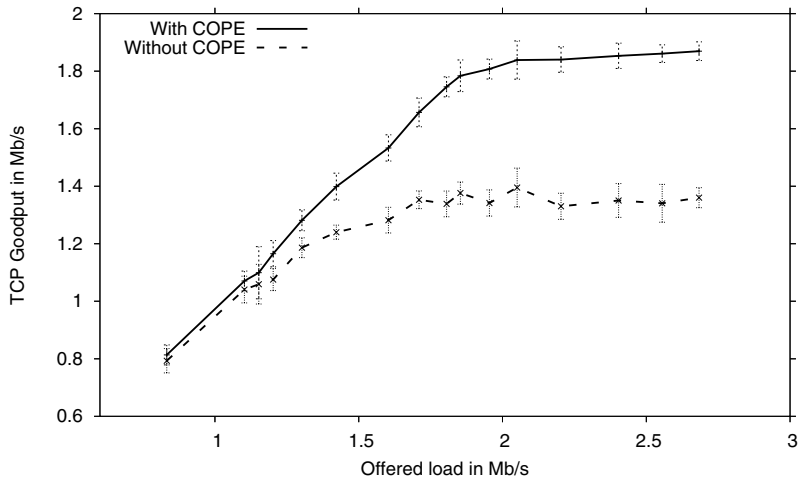


Fig. 4.8 Performance of COPE, courtesy of [44].

bottleneck. Network coding alleviates this problem, since now node A needs to transmit only once, the same as the rest of the nodes. This allows to achieve significantly higher throughput than the example in Figure 4.7 alone would predict.

4.5 Physical Layer Network Coding

In information theory for relay networks, a well-known observation is that relay nodes do not necessarily need to decode the source information, but simply forward to the receivers enough information to help the receivers decode. Physical layer network coding capitalizes on this idea by observing that simultaneous transmissions causing interference do not necessarily need to be treated as noise for each other: a relay could receive the superposition of signals at the physical layer, and simply forward this superposition to the receiver node.

The example in Figure 4.9 illustrates this point. In network coding, the relay linearly combines the received signals x_1 and x_2 and broadcasts the resulting signal $x_1 + x_2$ to both destinations. The linear combining occurs algebraically, once both x_1 and x_2 are received. In physical layer network coding, nodes A and C transmit simultane-

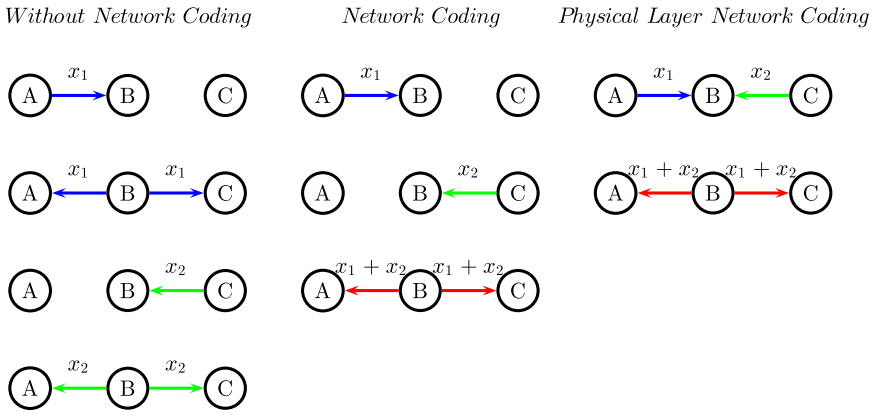


Fig. 4.9 Nodes A and C exchange information via relay B. The network coding approach takes advantage of the natural capability of wireless channels for broadcasting to give benefits in terms of resource utilization.

ously their signals to the relay, much like multiple transmit antennas transmit signals to a common destination. The relay node B observes at the physical layer the superposition of these two coherently arriving electromagnetic waves. This is the signal the relay re-broadcasts, combining in this sense being performed by the physical channel. Heuristic algorithms build based on this example indicate that this approach can indeed be implemented in practice and increase the achievable throughput in wireless ad-hoc networks.

Clearly this simplified example opens a number of theoretical questions and implementation challenges. Theoretical open questions include, how well this approach can perform? what the possible achievable rates over arbitrary networks are? and how we can code taking interference into account? The approach described in the next section promises progress toward answering such questions.

One of the main practical challenges is that wireless signals undergo fading and attenuation. For example, the received baseband signal y at the relay node B in Figure 4.9 can in general be described as

$$y = \gamma_A \sqrt{P_A}\mathbf{x}_1 + \gamma_C \sqrt{P_C}\mathbf{x}_2 + n,$$

where P_A, P_C is the transmitted power and γ_A, γ_C the channel attenuation from nodes A and C, respectively and n is the additive noise.

Each channel attenuation is assumed to be a complex Gaussian with zero mean, and the noise n also a Gaussian process of zero mean and unit variance. Note that the complex channel fading coefficients γ may have quite different magnitudes. This is because the separation of the mobile nodes might result in different path losses in the channels they experience. As a result, quantization at the receiving end of the relay might lead to significant errors.

A second practical challenge is, if we would like to reduce decoding to algebraic operations, we need to map the superimposed signal to algebraic combining. How to achieve this would for example depend on the modulation scheme employed. Finally, we need to achieve synchronization between nodes A and C, to guarantee a correct superposition of the signals at the relay. In practice, propagation delays, clock skews, and variations in the processing time might impact timing.

Physical layer network coding has close connections with the areas of cooperative communication and distributed space–time coding, where similar problems need to be addressed.

4.6 Wireless Information Flow

A systematic approach has recently emerged, that attempts to take into account both broadcasting and interference, in order to calculate achievable rates as well as coding strategies over wireless networks. This approach uses deterministic channels to model the interactions between the signals in the network, and ignores the effect of noise. The argument is that for high SNR, it is these signal interactions that will dominate the performance, and thus the capacity of the deterministic could be very close to that of the noisy network. Thus networks of deterministic channels could be used as approximate models for wireless networks.

The min-cut max-flow theorem and the main theorem in network coding can be thought of as applying to a special network of deterministic channels, where each channel corresponds to a lossless orthogonal link of unit capacity. We will next describe how these theorems can be generalized to other deterministic networks.

Consider a source transmitting information to a destination over a network of deterministic channels, where all network nodes act as

relays. Each node j broadcasts a signal x_j to the other nodes connected to this node. Moreover, it has only one received signal y_j which is a deterministic function of all the signals transmitted by the nodes connected to it.

We can apply the information theoretic cut-set bound to bound the rate R that the source can reliably transmit as

$$R < \max_{p(\{x_j\}_{j\in\mathcal{V}})} \min_{\Omega\in\Lambda_D} I(Y_{\Omega^c}; X_\Omega | X_{\Omega^c}) \tag{4.1}$$

$$\stackrel{(a)}{=} \max_{p(\{x_j\}_{j\in\mathcal{V}})} \min_{\Omega\in\Lambda_D} H(Y_{\Omega^c} | X_{\Omega^c}), \tag{4.2}$$

where Ω is a set of vertices that defines a cut, i.e., Ω contains the source and Ω^c the destination, X_Ω the vector collecting all inputs of vertices in Ω and Y_{Ω^c} the vector collecting all outputs of vertices in Ω^c. $\Lambda_D = \{\Omega : S \in \Omega, D \in \Omega^c\}$ is the set of all source–destination cuts (partitions) and (a) follows since we are dealing with deterministic networks.

It can be shown that by using random mappings at the network nodes of a deterministic network, we can achieve the maximum possible rate with a product distribution:

Theorem 4.5. Given a general deterministic relay network (with broadcast and multiple access), we can achieve all rates R up to

$$\max_{\prod_{i\in\mathcal{V}} p(x_i)} \min_{\Omega\in\Lambda_D} H(Y_{\Omega^c} | X_{\Omega^c}). \tag{4.3}$$

This theorem extends to the multicast case, where we want to simultaneously from S to all destinations in the set $D \in \mathcal{D}$:

Theorem 4.6. Given a general deterministic relay network, we can achieve all rates R from S multicasting to all destinations $D \in \mathcal{D}$ up to

$$\max_{\prod_{i\in\mathcal{V}} p(x_i)} \min_{D\in\mathcal{D}} \min_{\Omega\in\Lambda_D} H(Y_{\Omega^c} | X_{\Omega^c}). \tag{4.4}$$

Note that when we compare (4.3) to the cut-set upper bound in (4.2), we see that the difference is in the maximizing set i.e., we are only able to achieve independent (product) distributions whereas the cut-set optimization is over any arbitrary distribution. In particular, if the network and the deterministic functions are such that the cut-set is optimized by the product distribution, then we would have matching upper and lower bounds.

This indeed happens for example for linear finite-field deterministic networks, where all cut values are simultaneously optimized by independent and uniform distribution of $\{x_i\}_{i\in\mathcal{V}}$. By linear we mean that the input and output of each channel are related by a matrix multiplication. In this case, the optimum value of each cut Ω equals the logarithm of the rank of the transfer matrix $\mathbf{G}_{\Omega,\Omega^c}$ associated with that cut, i.e., the matrix relating the super-vector of all the inputs at the nodes in Ω to the super-vector of all the outputs in Ω^c. The following corollary generalizes the classical max-flow min-cut theorem for wireline networks.

Corollary 4.1. Consider a linear deterministic network with broadcast and multiple access, where operations are over the finite field $\mathbb{F}_q$. A source can transmit information to a receiver at a maximum rate of

$$C = \min_{\Omega\in\Lambda_D} \operatorname{rank}(\mathbf{G}_{\Omega,\Omega^c}) \log q. \tag{4.5}$$

This result can be directly extended to multicasting to a set of receivers, at a rate equal to the minimum capacity between the source and a receiver.

4.7 Sensor Networks

Here, we briefly discuss some first applications of ideas from network coding to sensor networks.

4.7.1 Untuned Radios in Sensor Networks

Consider sensor nodes with "untuned radios", in the sense that each sensor transmits at a randomly chosen (from a finite set) frequency,

and receives from a randomly chosen set of the frequencies. The motivation to consider such sensor nodes is that, tuned radios require use of external to the sensor chips quartz crystals, that are expensive and bulky. Assume a dense deployment of "untuned" nodes in a strip-like network, that aim to provide multi-hop connectivity between a source–destination pair. The randomness in the transmit and receive frequencies of the components of the networks leads to a random network configuration. Using balls and bins arguments, we can calculate the expected min-cut between the source and the destination. As we have also discussed when talking about dynamically changing networks, using randomized network coding we can then have the source transmit at a constant fraction of the average max flow in the graph, and *without apriori knowledge of the network connectivity*, successfully transmit this rate to the destination.

4.7.2 Data Collection in Sensor Networks

We consider a sensor network with n nodes, where each node i has an independent observation x_i. There also exist a set of collector nodes. We want the union of information that these nodes collect to be sufficient to retrieve all x_i. We consider two models. In the first model, the sensor nodes themselves are mobile, while the collector nodes are static. We call this the *mobile nodes model.* In the second model, we have a collector that moves randomly among the nodes, and collects the information. We call this the *moving collector model.*

4.7.2.1 Mobile Node Model

This model corresponds to applications where sensor nodes are placed on mobile objects such as cars or wildlife, that measure statistics to be communicated to base-stations. Assume that sensor nodes transmit at a constant range to other sensor nodes as well as to the base-stations.

In the case of forwarding, we have one more variation of the coupon collector problem, where now we have $\mathcal{C}$ collectors, and we are asking how many boxes should the collectors buy so that the union of their coupons covers all n possibilities. For $\mathcal{C}$ constant with respect to n,

which is the most realistic case, it is easy to see that the results are of the same order. That is, without network coding each collector should collect $\Theta(n \log n)$ transmissions, while use of network coding requires collecting $\Theta(n)$ transmissions.

4.7.2.2 Mobile Collector Model

We consider a sensor network in which n static sensors are placed on the nodes of an $\sqrt{n} \times \sqrt{n}$ square grid. Nodes make independent observations x_i, $1 \leq i \leq n$. We are interested in scenarios where nodes first distribute the information among themselves, so that, if only k sensors survive, they have enough information to reconstruct all observation.

We model this situation through a two-phase scheme as follows: In phase 1, we have $m < n$ rounds, where at each round each sensor node is allowed to broadcast once. We assume that each node transmission is successfully received by its four closest neighbors. Thus at the end of the first phase, each node has successfully received $4m$ broadcast transmissions. In phase 2, there is a mobile collector querying k sensors selected uniformly at random. We can think again of this phase as happening in rounds, where in every round one node is added to the collector's sample. We are asking what is the minimum number of rounds (equivalently, the minimum number of nodes in the sample) to collect all the information. The transmission mechanism used in phase 1 can be either forwarding or network coding.

Clearly, both for the case of forwarding and network coding, a necessary condition for the collected information to be sufficient to retrieve all x_i is that

$$k \geq \frac{n}{4m+1}. \tag{4.6}$$

We will next derive an alternative lower bound, based on the following observation: A necessary condition for successful decoding is that each random variable x_i appears at least once in the collected data of the k sensors.

In the case of forwarding, we assume that at each broadcast round each node forwards with equal probability one of the four messages it has received in the previous round. As a result, after m rounds, each

node will have collected $4m + 1$ of observations x_i from the neighbors within the radius of m. To estimate how many of these observations are different, we note that under our transmission model, at the end of phase 1, each x_i will perform a random walk with m steps, and thus on the average we expect it to reach nodes within distance $\Theta(\sqrt{m})$. We will make here a simplifying assumption that each node will receive all $\Theta(m)$ information symbols x_i within a radius of $\Theta(\sqrt{m})$. Therefore, when forwarding is used in phase 1, a necessary condition for the mobile collector to be successful in phase 2 is that the circles of radius $\Theta(\sqrt{m})$ around the k nodes it selects cover all the sensors.

In the case of network coding, we assume that at each round each node forwards a random linear combination of all the packets in its possession. As a result, after m rounds, each node will have $(4m + 1)$ coded observations, which will depend on the data x_i observed at the $\Theta(m^2)$ neighbors within the radius of m. Therefore, a necessary condition for the mobile collector to be successful in phase 2 is that the circles of radius $\Theta(m)$ around the k nodes it selects cover all the sensors. This ensures that all observations are represented in the collected coded packets.

Consider now a particular node i. The probability that the node is not covered by a randomly placed disk of radius r equals the probability that the center of the disk is not within distance r from node i, that is, the center is not at any of the $\Theta(r^2)$ points within the radius r of node i. This probability is given by

$$1 - \frac{\Theta(r^2)}{n}.$$

After repeating the experiment for k rounds, the probability that a particular node is not covered by any of the k disks equals

$$\left(1 - \frac{\Theta(r^2)}{n}\right)^k,$$

and, thus, the expected number of nodes that are not covered is

$$n\left(1 - \frac{\Theta(r^2)}{n}\right)^k < ne^{-k\Theta(r^2)/n}.$$

The requirement that this number be smaller than 1 gives the constraint

$$k \geq \frac{n}{\Theta(r^2)} \log n. \tag{4.7}$$

For network coding, $r = m$, while for forwarding, $r = \sqrt{m}$. Assume we use $m = \sqrt{n}$. To compare the two collection schemes, we denote by k_f the minimum number of collector's queries in the forwarding based scheme, and by k_c the minimum number of collector's queries in the coding based scheme. Then, applying our two bounds (4.6) and (4.7) we get that, for the case of forwarding (4.7) becomes tighter

$$k_f \geq \Theta(n\sqrt{n}\log n),$$

while for the case of network coding (4.6) is tighter

$$k_c \geq \Theta(n\sqrt{n}).$$

Thus, we obtain

$$k_f \geq k_c \Theta(\log n).$$

4.8 Challenges for Wireless Network Coding

Deployment of network coding requires resources, such as, synchronization and reliability mechanisms, and leveraged functionalities at the network nodes such as operations over finite fields and storage capabilities.

Perhaps more importantly, it also requires designing new protocols and architectures, or adapting the existing infrastructure for wireless networks, so that network coding functionalities are enabled. At what layer should network coding operate, what type of connections should it support, and what mechanisms should be put in place or altered, form a set of currently investigated research problems in networking protocol and architecture design.

For example, current MAC layer protocols such as 802.11, support only unreliable broadcasting. That is, when nodes select to broadcast, there is no mechanism in place for collision avoidance, and there are no acknowledgments collected. It is unclear whether to add this functionality at the MAC layer or at higher layers, and how to implement it without incurring significant complexity.

Other challenges include supporting the demands of specific applications. For example, real-time applications, such as audio and video, may have strict QoS requirements in terms of throughput and delay. These are often conflicting requirements. Achieving the optimal throughput with network coding may require the network nodes to mix n packets, for large values of n. To decode the source information, on the other hand, a node may need to collect n packets, which would cause prohibitive delay. Balancing such requirements might require a cross-layer approach.

Moreover, functionalities such as scheduling and routing need to be reconsidered. For example, for a network that supports multiple unicast sessions, it is unclear how widely the information across different sessions should be mixed, and in which order broadcast transmissions should be scheduled. The following example discusses these challenges and provides one algorithm that achieves good performance in practice.

Example 4.3 (Joint Scheduling and Network Coding) Optimizing parameters such as energy efficiency or delay can in general be expressed as a cost optimization problem. However, if we are interested for example in delay, and we model interference as noise, scheduling transmissions forms an important component of the optimization problem.

The optimal joint cost optimization and scheduling problem is $\mathcal{NP}$-hard; but in practice, there exist simple algorithms that allow to achieve good performance. One such approach is the following. Let a feasible network realization denote a set of non-conflicting simultaneous node transmissions. Use of realizations effectively disconnects the scheduling problem from the cost optimization problem.

Construct a predetermined finite set of network realizations, that satisfy a set of criteria. For example, we might want to choose realizations so that starting from a given node at any given realization, and changing realizations at each time slot, we can transmit information from this node with all other nodes. That is, the network is connected "over the realizations."

We can now write an LP that optimizes our cost function with respect to the fraction of time we use each realization. Note that this LP is in general suboptimal, because we do not optimize over all possible feasible realizations, but a small pre-selected set.

Notes

There has been a lot of interest in applying ideas from network coding in the context of wireless networks, it is one of the fastest evolving research areas, and here we have briefly summarized some first indicative results. The benefits of combining network coding with broadcasting have been investigated for example by Wu et al. [86] and Fragouli et al. [25]. The LP formulation for energy minimization was proposed by Lun et al. [58]. Fairness and delay over wireless networks were examined by Eryilmaz et al. [19]. Physical layer network coding was proposed by Zhang et al. in [91]. COPE was designed by Katti et al. [44]. Applying network coding to untuned radios is investigated by Petrović et al. [65]. Similar ideas have recently been applied to transportation networks. Network coding for sensor networks is also investigated by Dimakis et al. [14] and Fragouli et al. [26]. Wireless network coding has also been studied using information theoretic tools. For example, Gowaikar et al. looked at the capacity of wireless erasure networks [30], Ratnakar et al. [67] examined broadcasting over deterministic channels, and Avestimehr et al. examined networks of general deterministic channels with broadcasting and interference [3, 4]. Cross layer design has been examined by Sagduyu and Ephremides, see for example [75].

5

Multiple Unicast Sessions

In the first part of this tutorial [24], we discussed the problem of a source transmitting to a receiver and introduced the max-flow min-cut theorem. We then extended this problem to the case of multicasting from one source to multiple receivers, and saw that use of network coding techniques allows to achieve the optimal performance over directed graphs, and can offer throughput benefits over undirected graphs. In this section, we examine a different generalization, from the single source–destination problem to the multiple source–destinations problem, also known as the *multicommodity problem*, where we think of each flow as a commodity.

We start by briefly reviewing results that apply when we employ routing and then examine how network coding can help. In particular, the research effort has focused toward three main directions, which we review in separate sections:

(1) Quantify the benefits network coding can offer for the multicommodity flow problem, over directed and undirected graphs.
(2) Derive tight upper bounds for the achievable rate using network coding.

(3) Design algorithms, and evaluate benefits these algorithms offer in a practical setting.

5.1 The Classical Multicommodity Flow Problem

We will focus our attention on the following problem, known as the *maximum concurrent multicommodity flow problem*. Consider a communication network represented as a graph $G = (V, E)$ where edge $e \in E$ has capacity c_e. The network is shared by k source–destination pairs $\{(S_1, R_1), \ldots, (S_k, R_k)\}$. Source S_i demands to convey an information flow of rate d_i to its destination R_i. From the min-cut, max-flow theorem, we know that any demand $d_i \leq m_i$, where m_i is the min-cut between S_i and R_i, can be satisfied as long as S_i and R_i are the only pair of terminals using the network. However, we may not be able to simultaneously accommodate all demands even if $d_i \leq m_i$, for all i.

We say that *rate r is feasible*, with $0 \leq r \leq 1$, if we can guarantee to each source–destination pair a constant fraction r of its demand. The maximum concurrent multicommodity flow problem asks to maximize the feasible rate r while obeying the edge capacity constraints. Let $P_i = \{P_i^j\}$ denote the set of all paths in G between (S_i, R_i), and let f_i^j be the information flow along path P_i^j. The demand multicommodity flow problem is stated by the following linear program:

Multicommodity Problem — Primal:

maximize r

subject to

$$\sum_{P_i^j \in P_i} f_i^j \geq r\, d_i, \text{ for all } i \text{ (rate feasibility constraints)}$$

$$\sum_{P_i^j \,:\, e \in P_i^j} f_i^j \leq c_e, \text{ for all } e \in E \text{ (capacity constraints)} \tag{5.1}$$

$$f_i^j \geq 0, \ \forall\ i, j$$

This LP has a solution (that is, there is a feasible rate r) if and only if the demands and the edge capacities satisfy a certain condition. This is first observed by a group of Japanese researchers, and is sometimes referred to as the *Japanese theorem*. It should not be confused with the better known Japanese theorem in geometry, which we will at no time use here.

This constraint can be derived from the dual problem. There are two sets of variables in the dual problem corresponding to the two sets of constraints in the primal: $\{m_e\}$, $e \in E$, associated with the edge constraints and $\{\ell_i\}$, $1 \le i \le k$, associated with the rate feasibility constraints. The dual of our demand multicommodity flow problem is stated by the following linear program:

Multicommodity Problem — Dual:

$$\begin{aligned}
&\text{minimize} \sum_{e \in E} c_e m_e \\
&\text{subject to} \\
&\quad \sum_{e \in P_i^j} m_e \ge \ell_i, \text{ for all } i, j \\
&\quad \sum_{i=1}^{k} \ell_i d_i \ge 1 \\
&\quad m_e \ge 0, \text{ for all } e \in E \\
&\quad \ell_i \ge 0, \text{ for all } i
\end{aligned} \tag{5.2}$$

We can think of variable m_e as a length of edge e. Let $\text{dist}_{\{m_e\}}(S_i, R_i)$ denote the shortest path from source S_i to R_i with respect to the length function $\{m_e\}$. The *Japanese theorem* states that the multicommodity flow problem (5.1) has a solution (a feasible rate) if

$$r \sum_{i=1}^{k} d_i \text{dist}_{\{m_e\}}(S_i, R_i) \le \sum_{e \in E} m_e c_e \tag{5.3}$$

for any positive length function $\{m_e\}$.

The dual program (5.2) can be thought of as the LP relaxation of the integer program formulated as (5.2) but with the constraint $m_e \in \{0,1\}$. We can now think of $\{m_e\}$ as an incidence vector of a set $\mathcal{E} \subseteq E$: the edges with value 1 belong in $\mathcal{E}$ and form a cut-set that disconnects at least one source–destination pair.

For this problem, we have what is known as the *sparsity upper bound* $\mathcal{S}$:

$$r \leq \mathcal{S} \triangleq \min_{\mathcal{E}} \frac{\sum_{e \in \mathcal{E}} c_e}{\sum_{\mathcal{E} \text{ separates } S_i \text{ from } R_i} d_i}. \tag{5.4}$$

Sparsity $\mathcal{S}$ gives a very intuitive upper bound on the feasible rate r if we are restricted to employ routing. However, this upper bound can be loose: it is proved that it can be as much as (but not more than) a factor of $\Omega(\log |V|)$ for undirected graphs with $|V|$ vertices. For directed graphs, it can be larger than $\Omega(\log |V|)$ but not more than $\Omega(\sqrt{|V|})$.

The sparsity $\mathcal{S}$ illustrates a fundamental difference between directed and undirected graphs if we allow the use of network coding. For undirected graphs, it upper bounds the rate r achievable with network coding, as we prove in Section 5.3. However, the sparsity bound (5.4) *does not hold* if we allow the use of network coding over *directed* graphs, as the example in Figure 5.1 indicates. Consider the cut set $\mathcal{E} = \{AB\}$ that disconnects each source from its receiver. Then $\mathcal{S} \leq 1/2$. Therefore, $\mathcal{S}$ is not an upper bound on the rate achievable with network

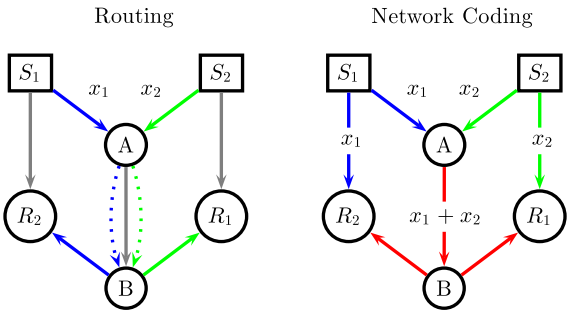


Fig. 5.1 The butterfly network example viewed as a multicommodity flow problem with $k = 2$ source–destination pairs. Edges have unit capacity. S_1 demands to transmit unit rate to R_1, while S_2 demands to transmit unit rate to R_2. In the routing solution the two commodities need to share the edge AB.

coding, which can provide rate one to each receiver. Note that here the network coding solution effectively transforms the multiple unicast to a multicast problem. Another way to think about network coding is that it takes advantage of additional edges in the network to convey some type of independent side information.

5.2 What are the Benefits of Network Coding

There are directed networks with n nodes for which network coding offers benefits of order $\Theta(n)$. On the other hand, we do not know as yet of an undirected network where network coding offers any benefits. If we extend the directed Butterfly network in Figure 5.1, from two to n source destination pairs, we obtain the network in Figure 5.2 for which the benefits of coding are n-fold. If the same network is made undirected, the benefits of coding disappear.

Consider first the directed network case, as shown in Figure 5.2. Source S_i is connected to its own destination R_i only through edge AB. However, each S_i is connected to all other destinations R_j, $j \neq i$. If we assume unit capacity edges, using network coding allows to send rate one to each receiver, by sending the linear combination of all source symbols $x_1 + x_2 + \cdots + x_n$ through edge AB. Routing has all

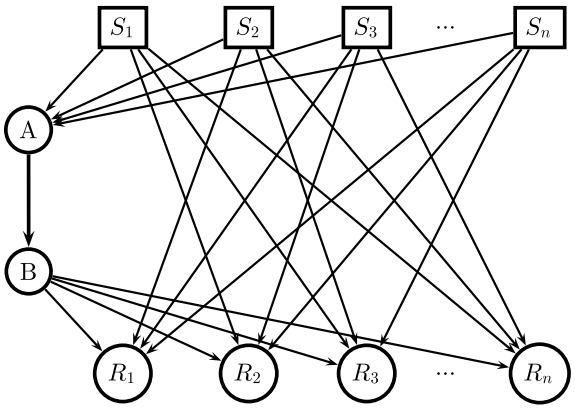


Fig. 5.2 A directed multiple unicast instance with $2n + 2$ nodes and n-fold throughput benefits of network coding.

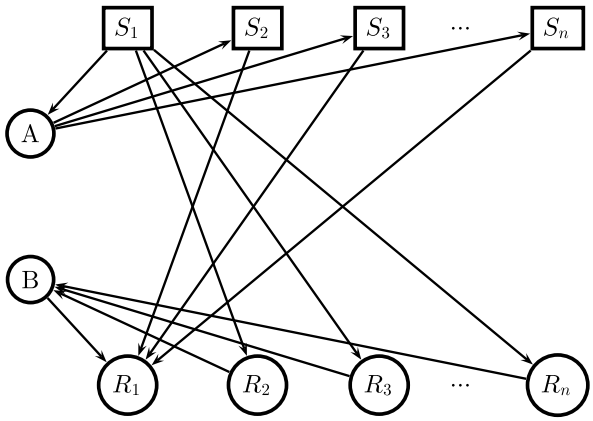


Fig. 5.3 Routing from source S_1 to destination R_1 in an undirected graph instance, where the routing and the network coding solution achieve the same rate.

commodities share the capacity of edge AB, and thus only allows rate $1/n$ for each source destination pair.

Suppose now that all edges in the network are undirected. In this case, an edge can accommodate fractional flows going in arbitrary directions as long as their rates sum to 1. A routing scheme that achieves unit rate throughput for each source–destination pair, delivers the information from source S_i to destination R_i through the $2(n-1)$ rate $1/[2(n-1)]$ flows $S_i \to A \to S_j \to R_i$ and $S_i \to R_j \to B \to R_i$, $1 \leq j \leq n,\ j \neq i$. Routing from source S_1 to destination R_1 is illustrated in Figure 5.3. Note that the routing solution uses fewer resources, namely, does not use edge AB.

It is widely believed that network coding offers no benefits in terms of throughput for the multicommodity flow problem over undirected graphs. This conjecture is supported by a number of examples, that show no benefits from the use of network coding. We will see some such examples in the following section.

5.3 Upper Bounds on the Network Coding Throughput

The state of the art bounds for network coded information flow combine cut-set bounds with information-theoretical tools. To formulate the multicommodity flow problem with network coding, we

model the network $G = (V, E)$ shared by k source–destination pairs $\{(S_1, R_1), \ldots, (S_k, R_k)\}$ as follows. We think of each edge $e \in E$ as a deterministic lossless channel of capacity c_e, and associate a random variable Y_e with the messages through e. Without loss of generality, we assume that source S_i is connected to the network through an infinite capacity edge, and associate a random variable with the messages on that edge. For simplicity, we denote by S_i both the random variable and the edge. Similarly, we assume that each receiver R_i is connected to the network through an infinite capacity edge, and associate random variable T_i with the messages on that edge.

In routing solutions, any two of random variables Y_e, S_i, and T_i are either equal to or different from each other; a network code will, in general, impose more complex functional relationships among them. Consider any graph that implements a valid network code to deliver to each receiver a fraction r of its demand, and where the sources have independent information. The random variables Y_e, S_i, and T_i have to satisfy the following constraints:

(1) Independence of sources:

$$H(S_1, S_2, \ldots, S_k) = \sum_{i=1}^{k} H(S_i).$$

(2) Capacity bounds:

$$H(Y_e) \leq c_e.$$

(3) Correct reception:

$$H(Y, S_i) = H(Y, T_i)$$

for each set for variables Y. For example, by selecting $Y = S_i$ and $Y = T_i$, we get $H(S_i|T_i) = H(T_i|S_i) = 0$, that is, knowledge of T_i removes all uncertainty about S_i.

(4) Achievable rate

$$H(T_i) \geq rd_i$$

between the ith source–destination pair.

The above set of inequalities represents a natural information theoretic characterization of the multicommodity flow problem with network coding. To find tight upper bounds on the rate r for a particular graph, we need to find additional constraints coming from the structure of the graph. This is the challenging part of the problem. We next briefly describe two techniques to find these additional constraints, one based on informational dominance relationships between edge sets, and the other on d-separation conditions on functional dependency graphs. Intuitively, both of these techniques serve to identify what we may call "network coding edge cut-sets."

5.3.1 Informational Dominance

Informational dominance is a relationship between sets of edges in the graph. We say that a set of edges A informationally dominates a set of edges B if the messages transmitted through the edges in A uniquely determine the messages transmitted through the edges in B. In other words, $H(Y_B|Y_A) = 0$, or, equivalently,

$$H(Y_A) = H(Y_A, Y_B), \tag{5.5}$$

where Y_X denotes the vector of the random variables associated with the messages on the edges in the set X. Note that only the graph structure determines whether the relationship of informational dominance exists between two sets of edges or not, whereas a particular coding/routing scheme determines what the relationship is.

We denote by $\mathrm{Dom}(A)$ the set of all edges informationally dominated by A:

$$\mathrm{Dom}(A) \triangleq \{e : e \text{ is informationally dominated by } A\}.$$

From the definition of informational dominance, a set dominates itself: $A \subset \mathrm{Dom}(A)$. We can identify $\mathrm{Dom}(A)$ by a greedy incremental procedure which starts from A. This procedure uses the following properties:

(1) $S_i \in \mathrm{Dom}(A)$ if and only if $T_i \in \mathrm{Dom}(A)$ (because if we know what the source sends we also know what the receiver receives, and vice versa).

(2) Every edge e in $E \setminus \mathrm{Dom}(A)$ is reachable from a source that does not belong in $\mathrm{Dom}(A)$ (because e must get some information that the edges in A do not have, which can only come from some source that does not belong in $\mathrm{Dom}(A)$) .
(3) If S_i is not in $\mathrm{Dom}(A)$, then S_i remains weakly connected to T_i if we remove all edges in A, all edges that do not have a path to T_i, and all edges that are not reachable from any source.

Starting from the set $\tilde{A} = A$ we repeat the above check. If it fails, we add an edge to $\tilde{A}$. If it succeeds, we have identified the maximal set $\mathrm{Dom}(A)$ dominated by A. Having identified this set, for each subset B of edges with $B \subseteq \mathrm{Dom}(A)$, the equality (5.5) holds.

For the following example, we use information dominance relationships to show that network coding does not offer any benefits as compared to routing.

Example 5.1. Consider the directed graph G depicted in Figure 5.4, with three-commodities that have unit rate demands. Applying the sparsity bound to edges e_1 and e_2 we get that routing can at most achieve rate 2/3 (and it is easy to see this upper bound is achievable). To prove that in the case of network coding as well, 2/3 is an upper

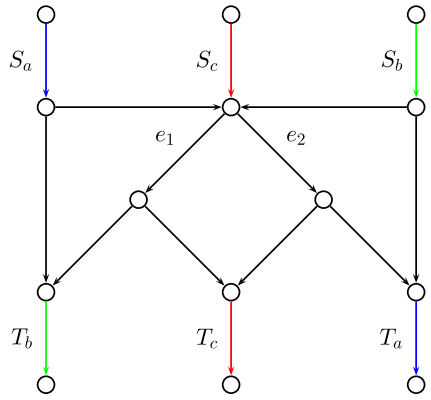


Fig. 5.4 The split-butterfly instance: a three-commodity flow example where network coding does not offer rate benefits as compared to routing.

bound to the optimal rate, we can use the following three information domination relations:

(1) $S_b \in \text{Dom}(\{S_a, Y_{e_1}\}) \Rightarrow H(S_a, Y_{e_1}) = H(S_a, S_b, Y_{e_1})$
(2) $S_a \in \text{Dom}(\{S_b, Y_{e_2}\}) \Rightarrow H(S_b, Y_{e_2}) = H(S_a, S_b, Y_{e_2})$
(3) $S_c \in \text{Dom}(\{S_a, S_b, Y_{e_1}, Y_{e_2}\}) \Rightarrow H(S_a, Y_{e_1}) = H(S_a, S_b, S_c, Y_{e_1}, Y_{e_2})$

We will also use basic entropy properties, such that conditioning reduces entropy, i.e., $H(X|Y) \leq H(X)$. Adding (1) and (2) we get:

$$\begin{aligned} H(S_a, Y_{e_1}) + H(S_b, Y_{e_2}) &= H(S_a, S_b, Y_{e_1}) + H(S_a, S_b, Y_{e_2}) \overset{(a)}{\Rightarrow} \\ H(S_a) + H(Y_{e_1}) + H(S_b) + H(Y_{e_2}) &\geq H(S_a, S_b, Y_{e_1}, Y_{e_2}) + H(S_a, S_b) \overset{(b)}{\Rightarrow} \\ c_{e_1} + c_{e_2} &\geq H(S_a, S_b, S_c, Y_{e_1}, Y_{e_2}) \overset{(c)}{\Rightarrow} \\ r &\leq \frac{c_{e_1} + c_{e_2}}{d_a + d_b + d_c} = \frac{2}{3}, \end{aligned}$$

where (a) holds because $H(X,Y) \leq H(X) + H(Y)$ while $H(X,Y) + H(X,Z) \geq H(X,Y,Z) + H(X)$, (b) holds from independence of sources and (3), and (c) holds from capacity constraints and the demands requests.

Applying informational dominance constraints in undirected graphs, allows to easily see that the sparsity defined in (5.4) is also an upper bound to the best network coding rate. Suppose that the set of edges $\mathcal{E} = \{e_1, \ldots, e_n\}$ is the one for which the minimum in (5.4) is achieved, and that it disconnects ℓ sources, say $S_1, \ldots, S_\ell$, from their receivers. Then, because the graph is undirected, $\mathcal{E} = \{e_1, \ldots, e_n\}$ informationally dominates $\{S_1, \ldots, S_\ell\}$, and thus,

$$H(S_1, \ldots, S_\ell, Y_{e_1}, \ldots, Y_{e_m}) = H(Y_{e_1}, \ldots, Y_{e_m}).$$

Note that in a directed graph, a source S_i may not be informationally dominated by the edges in the cut, because for example condition (3) in the procedure we previously described for information dominance might be satisfied. We also have

$$H(Y_{e_1}, \ldots, Y_{e_n}) \leq \sum_{i=1}^{n} H(Y_{e_i}) \leq \sum_{i=1}^{n} c_{e_i},$$

and

$$H(S_1,\ldots,S_\ell,Y_{e_1},\ldots,Y_{e_m}) \geq H(S_1,\ldots,S_\ell) = \sum_{i=1}^{\ell} H(S_i) \geq r\sum_{i=1}^{\ell} d_i.$$

Therefore,

$$r \leq \frac{H(S_1,\ldots,S_\ell,Y_{e_1},\ldots,Y_{e_m})}{\sum_{i=1}^{\ell} d_i} \leq \frac{\sum_{i=1}^{n} c_{e_i}}{\sum_{i=1}^{\ell} d_i}.$$

5.3.2 *d*-separation Bound

This approach starts with a set of edges $\mathcal{E}_d$ and a set d of connections, and tries to verify whether it holds or not that

$$\sum_{\{i\in d\}} rd_i \leq \sum_{e\in\mathcal{E}_d} c_e. \tag{5.6}$$

The verification process involves building a functional dependency graph (FDG), that expresses how the variables associated with edges, sources, and destination observations of the initial network are related. This graph has as vertices variables. An edge exists from variable X to variable Y if Y is a function of X. We can then iteratively remove edges from this diagram, following an exact procedure described in [47], and verify whether the d sources and destinations become disconnected in an undirected sense. If the verification process succeeds, (5.6) can be used as an upper bound.

Example 5.2. Figure 5.5 depicts what is known as the Okamura-Seymour example. Each edge has unit capacity, and each destination

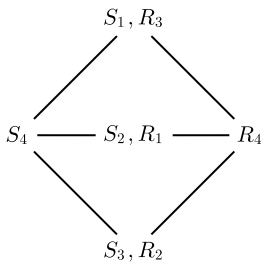


Fig. 5.5 The Okamura-Seymour example.

requires unit rate from its source. For this example, the sparsity equals 1, while both the routing and coding rate are at most 3/4.

5.4 Network Code Design Challenges

A number of heuristic algorithms are proposed in the literature. The basic underlying idea is to perform network coding opportunistically, for example, by embedding the integral butterfly network coding solution in Figure 5.2, in the network structure. This approach can be expressed as a cost optimization linear program, where we divide the sources into pairs, and for every pair of sources we attempt to create flows that mimick the butterfly structure.

The difficulty of designing network codes for multiple unicast sessions is intertwined with the lack of understanding of the theoretical throughput limits and ways to achieve them. The following phenomena observed in connection with solvability of certain multiple unicast graph instances reveal the inherent hardness of this problem when compared to multicast, where such phenomena never arise:

(1) *Exponential alphabet size:* There exist networks with N receivers that only have linear coding solutions over a finite field whose size is exponentially large in N.
(2) *Non-monotonicity of alphabet size:* There exist networks that have linear solutions over alphabets of size p^m but not over alphabets of size $q^t > p^m$ where p and q are primes.
(3) *Nonlinearity:* There exist networks that are solvable using nonlinear operations, but are not solvable using linear operations.
(4) *Non-reversibility:* There exist solvable directed graphs that are not solvable when the roles of sources and receivers are exchanged and the directions of all edges are reversed.

The following example illustrates the last phenomenon.

Example 5.3. Consider the directed graph in Figure 5.6, with unit capacity edges. The sources x, y, and z transmit unit rate to receivers

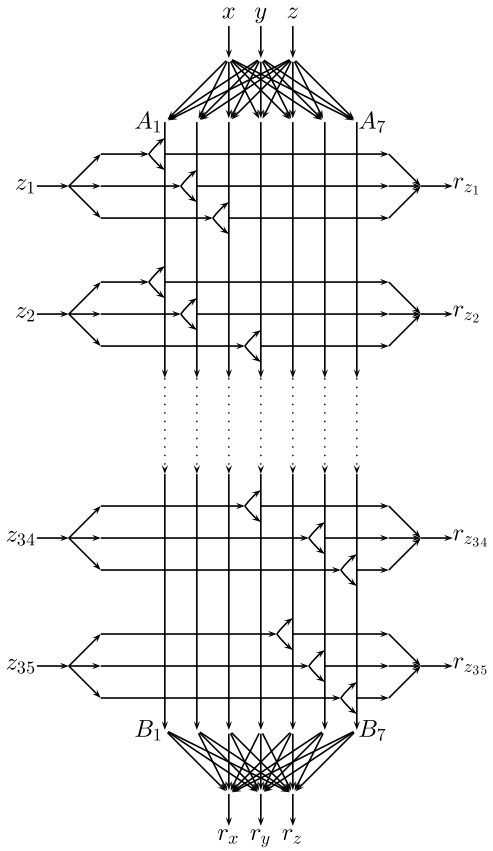


Fig. 5.6 An irreversible network that supports multiple unicast sessions: the sources x, y, and z transmit their information to receivers r_x, r_y, and r_z, respectively, and the sources z_i to their respective receivers r_{z_i}.

r_x, r_y, and r_z, respectively, by sharing the seven perpendicular lines A_iB_i. The $35 = \binom{7}{3}$ sources z_i transmit unit rate to their receivers r_{z_i}, each source using three horizontal lines intersecting with a different subset of the seven A_iB_i lines.

We first describe a binary coding scheme that allows each source to be decoded at its destination. We can view the three bits produced by the sources x, y, and z as a three-bit sequence that takes eight distinct values. We map this sequence to a 7-bit sequence, as described in Table 5.1, so that at most one bit out of the seven takes the value one at a time. We send these 7 bits through the lines A_iB_i to the

Table 5.1 The coding scheme that maps the input bits produced from sources x, y, and z, to the output bits sent along edges A_iB_i in Figure 5.6.

Input bits	000	001	010	...	110	111
Output bits	0000000	0000001	0000010	...	0100000	1000000

destinations r_x, r_y, and r_z. Each of the destinations can decode all three sources x, y, and z.

Consider now a source z_i. The source uses repetition coding, and sends the repeated value through the three horizontal lines to r_{z_i}. Note that this transmission does not affect the information flow in the perpendicular lines, as whatever is added in a line, it is also promptly removed. Now consider the information along A_jB_j acting as noise for the transmission from z_i to r_{z_i}. Because at most one of the three perpendicular lines carries the value one, at most one of the horizontal transmissions will be affected, and thus the destination r_{z_i} observes either 2 times or 3 times the bit value send by z_i. Using majority voting, r_{z_i} can correctly decode the source information. Thus this network is solvable, in the sense that all 38 multiple unicast sessions are served, and we can achieve rate $r = 1$.

Now assume that the role of sources and receivers is reversed, i.e., the receivers reply to their sources. Moreover, assume that the orientation of every single edge of the network is reversed. In this case, the network is no longer solvable. Intuitively, the main reason for this, is that the information inserted by the horizontal lines can no longer be removed, and thus introduces "errors" in the perpendicular lines. Along the perpendicular lines we need to implement a coding scheme of rate at least 3/7, that cannot correct the possible 35 errors the horizontal transmissions may incur.

Notes

The multicommodity flow problem, with use of routing, has been a very active research area; see for example Schrijver [76]. Benefits of network coding over directed graphs were examined by Lehman et al. [50]. The information dominance was introduced by Adler et al. [1], while the d-separation bound by Kramer and Savari [47].

Network examples are provided: for the exponential alphabet size and for the non-monotonicity of alphabet size by Rasala Lehman and Lehman [50]; for nonlinearity by Riis [71] and Dougherty et al. [17]; and for non-reversibility by Riis [70] and Dougherty and Zeger [18]. Constructive algorithms to support multiple unicast sessions were proposed for example by Ho et al. [33].

6

Networks with Errors

Most of the time in this tutorial we assume ideal communications links between network nodes. In this section, we will look into some scenarios when that is not the case, and information carried by network edges may be erased, altered by transmission errors, or the edges themselves may go out of service.

We will restrict our attention to *noisy networks*, in which each edge represents an interference-free, directed, noisy, memoryless channel. We will consider both channel coding, aimed to protect a particular information stream from channel errors, and *network coding*, performed across independent information streams.

We first address the question of channel and network coding separability, and then consider channel coding schemes for line networks. We then move to network coding, and describe how network codes can be designed to make unicast or multicast transmissions over networks resilient to a limited number of permanent link failures (non-ergodic failures) or a limited number of random packet errors.

6.1 Channel and Network Coding Separability

The first natural question to ask about networks with errors is whether network and channel coding can be separated without loss of optimality. We will consider multicasting, and assume that the information-theoretic[1] min-cut between each receiver and the source is equal to h. The answer is affirmative, provided intermediate nodes are allowed to decode and re-encode the information sent by the source without any complexity and/or delay constraints. Indeed, use of a capacity-achieving channel code over each edge of the network allows to transform every link into an error-free channel. Note that the min-cut to each receiver over the lossless network is still h. We can then use the main theorem in network coding to send rate h to each receiver. Thus, assuming possibly unlimited complexity intermediate node processing, network and channel coding separability holds over our class of noisy networks.

However, in a realistic network, intermediate nodes have complexity and delay constraints. In this case, network links can no longer be considered error-free, and as a result, network and channel coding can no longer be separated without loss of optimality. To help us appreciate this phenomenon, we consider the following example:

Example 6.1. Assume that each edge in the network in Figure 6.1 is a Binary Symmetric Channel (BSC) with transition probability $p \in [0, 1/2]$, which we denote as BSC(p). Source A attempts to maximize the rate it can send to receiver F. We will assume that the source and the receiver can invest infinite complexity processing to this task. If intermediate nodes B and C are allowed infinite complexity processing as well, we can achieve rate equal to the min-cut capacity $C = 2(1 - H(p))$, where $H(p)$ is the binary entropy function, by sending independent streams of equally likely bits on paths (AB, BF) and (AC, CF) (two parallel channels).

If, on the other hand, the intermediate nodes are only allowed to process one bit, we can no longer achieve the min-cut capacity. Without

[1] The information-theoretic min-cut is defined as in Cover and Thomas, "Elements of Information Theory."

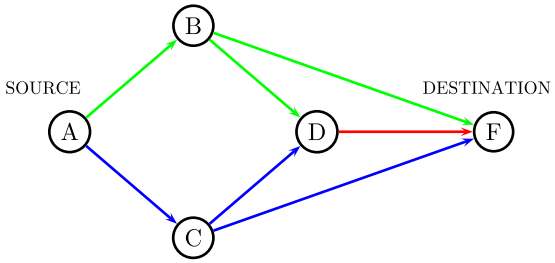


Fig. 6.1 The source node A transmits information to the receiver node F over a lossy network, where each edge is modeled as a BSC.

loss of generality, we can assume nodes B and C simply forward their input bit. At node D, since two bits are available at its input, one bit processing amounts to mapping four input values to two output values. All these mappings are equivalent from symmetry to one of the following functions:

f_1: forwarding one of the incoming bits
f_2: transmitting the binary `XOR` of the incoming bits
f_3: transmitting the binary `AND` of the incoming bits.

Explicitly calculating the end-to-end achievable rates R_1, R_2, and R_3, optimized over all input distributions, shows that each of the functions f_1, f_2, and f_3 can be optimal, depending on the noise parameter p. Figure 6.2 plots the ratio R_2/R_1 and R_3/R_1 for the special case when every edge represents a BSC(p), except BD and CD which we assume to be error-free. For small values of p, linear combining (f_2) outperforms simple forwarding (f_1) and nonlinear processing (f_3). For larger values of p simple forwarding is better, and for very large values of p nonlinear processing (f_3) outperforms linear processing (f_1 and f_2). Moreover, the corresponding optimal input distributions are uniform for f_1 and f_2, and non-uniform for f_3.

This example illustrates that network coding (whether we combine independent information streams), routing (whether we route both or just one of the information streams to node D), and channel coding (what is the optimal input distribution) cannot be optimized separately without loss of optimality.

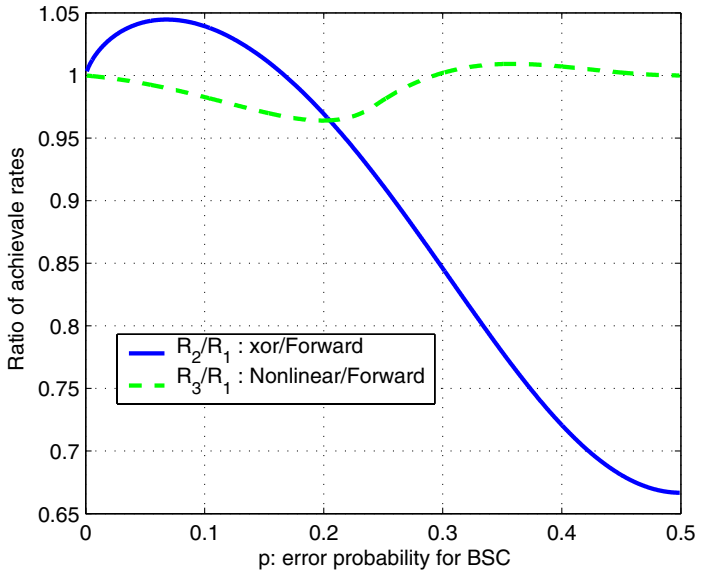


Fig. 6.2 Performance comparison for the network in Figure 6.1 when every edge represents a BSC(p), except BD and CD that are assumed to be error-free.

6.2 Channel Coding Schemes for Packet Erasure Correction

In a network environment, the information sent by a source reaches its final destination through a number of intermediate nodes (relays) rather than directly through a single channel. Packets can be dropped as they traverse the network, because of, for example, congested links or queue delays. Connections that can drop packets are typically modeled by independent memoryless erasure channels.

To protect against packet erasures, networked systems today employ traditional coding schemes for point-to-point connections. These schemes only require end-to-end processing and are oblivious to the network environment. However, as the size of communication networks grows, it becomes less clear if the benefits of the simple end-to-end approach outweigh those of coding schemes that employ intermediate node processing. We will describe some of the coding schemes that employ intermediate node processing and discuss their performance.

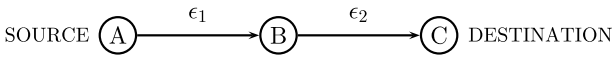


Fig. 6.3 A path between a source A and a receiver C with $L = 2$ edges, where edges correspond to independent memoryless erasure channels.

To illustrate the different approaches to coding at intermediate nodes, we will use the simple network depicted in Figure 6.3 that connects source A to receiver C through the single relay B. Packets are erased with probability ϵ_1 on the channel AB and ϵ_2 on the channel BC. Node A encodes k source symbols[2] to create n_1 coded symbols using a code $\mathcal{C}_1$, and then sends them over the channel AB. Node B will, on average, receive $n_1(1-\epsilon_1)$ coded symbols over n_1 time slots. Node B will send n_2 packets, produced by a code (more generally, processing) $\mathcal{C}_2$. If node B finishes transmitting at time d, where $\max\{n_1, n_2\} \leq d \leq n_1 + n_2$, then node C will receive on average $n_2(1-\epsilon_2)$ packets after d time slots.

We will consider four coding schemes for the two-link line network in Figure 6.3, which naturally apply to line networks with more links, and can be extended to arbitrary graphs and also to other than unicast traffic patterns. The evaluation metrics for the coding schemes will be the following:

(1) *Complexity* of encoding/processing/decoding at nodes A, B, and C in terms of the number of operations required as a function of k, n_1, and n_2.
(2) *Delay* incurred at the intermediate node B defined as the time $(d - k/m)$, where m is the min-cut capacity.
(3) *Memory requirements* in terms of the number of memory elements needed at node B.
(4) *Achievable rate*, i.e., the rate at which information is transmitted reliably from A to C. We say that a coding scheme is optimal in rate if each individual link is used at its capacity. Such schemes achieve the min-cut capacity between the source and the destination.

[2] We will use "packets" and "symbols" interchangeably.

(5) *Adaptability,* that is, whether the coding scheme needs to be designed for specific erasure probabilities ϵ_1 and ϵ_2 or not. Fountain codes, for example, are adaptable in this sense.

Depending on the application, different metrics may become important. From now on, unless stated otherwise, we will assume for simplicity that $\epsilon_1 = \epsilon_2 = \epsilon$, in which case $n_1 = n_2 = n$.

6.2.1 Complete Decoding and Re-encoding

A straightforward scheme is to use a separate code for each link, and have each intermediate node completely decode and re-encode the incoming data. The min-cut capacity can then be achieved by using capacity achieving codes (e.g., Fountain or Reed-Solomon codes) over all links. However, the system suffers a delay of about $k\epsilon/(1-\epsilon)$ time-slots at each intermediate node. Indeed, at node B, we can directly forward the $(1-\epsilon)n$ received coded bits (packets) without delay, and then, after decoding, create and send an additional ϵn bits over the second channel. A complete decoding and re-encoding scheme based on Fountain codes is adaptable and has low complexity. We only need $\mathcal{O}(k\log(k))$ binary operations at each intermediate node to decode and re-encode a Fountain code, and the complete decoding and re-encoding scheme has memory requirements of the order $\mathcal{O}(k)$. The main drawback of the scheme is that it requires each intermediate node to store in memory the entire k packets of information in order to re-encode.

6.2.2 Fixed Systematic Codes

Systematic schemes minimize the memory requirement at the intermediate nodes, but require the knowledge of the erasure probabilities of the links. Consider again the network in Figure 6.3 and assume that we use a systematic (n,k) code (e.g., Fountain or Tornado) for link AB. If we have an MDS systematic code that achieves the capacity of an erasure channel with the erasure probability ϵ, then any $k = n(1-\epsilon)$ symbols will form an information set, and can be treated as systematic. In a systematic scheme, node A sends the n coded symbols, and about $n\epsilon$ of these symbols get erased on AB. Node B forwards the received

$n(1-\epsilon)$ symbols to C (which he treats as systematic), and computes and forwards (about) $n\epsilon = k\epsilon/(1-\epsilon)$ new parity symbols based on those correctly received, which he then transmits in the $n\epsilon$ time slots following the transmission of the systematic bits.

This scheme incurs an average delay of $n\epsilon$, and requires $n\epsilon$ memory elements. The savings in memory, as compared to the complete decoding and re-encoding, are significant when the erasure probability ϵ is small. Although not adaptable to unknown channel parameters, these codes have very low encoding and decoding complexities. Tornado codes for example can be encoded and decoded with $\mathcal{O}(n\log(1/\delta))$ operations, where δ is a constant expressing the (fixed) rate penalty.

6.2.3 Sparse Random Systematic Codes

In this scheme, the non-systematic packets are formed as random (sparse) linear combinations of the systematic ones. More precisely, whenever a new packet is received at B, it is added to the storage space allocated to each of the non-systematic packets independently and with a (small) probability p. We will give without proof the following theorem.

Theorem 6.1. With $p = (1+\delta)\log(\epsilon k)/(\epsilon k)$ for $\delta > 0$, the described systematic random code asymptotically achieves the capacity over the channel BC.

This scheme is an improvement on the one previously described in terms of encoding and decoding complexity.

6.2.4 Random Codes

In this scheme, at each time slot the intermediate node B transmits random linear combinations (over the binary field $\mathbb{F}_2$) of all the packets it has received thus far. The main advantages of this random scheme are its adaptability and optimality in terms of delay. The drawbacks are large memory requirement, and high decoding complexity, which is $\mathcal{O}(k^2 \log k)$ `xor` operations on packets. The following theorem asserts that these random codes achieve capacity.

Theorem 6.2. Given a constant $c > 1$, let $\mathbf{A}$ be a random lower-triangular $(k + c\log(k)) \times k$ binary matrix, where the entries $A_{i,j}$ are zero for $1 \leq i < j \leq k$, and all the other entries are i.i.d. Bernoulli(1/2) random variables. Then

$$\Pr\left[\mathrm{rank}(\mathbf{A}) < \mathrm{k}\right] \leq \frac{1}{2\mathrm{k}^{\mathrm{c}-1}}.$$

6.3 Network Codes for Non-Ergodic Link-Failure Protection

Network coding can be used to gracefully adapt to permanent link failures. Consider a source transmitting information to a single receiver over a network with unit capacity edges, each susceptible to a permanent non-ergodic failure. Assume that the min-cut from the source to the destination is n. We want to guarantee to the receiver rate equal to $k = n - t$, provided that at most t, and no matter which t, edges in the network permanently fail. Of course, such a task is simple, if we are allowed to reroute the information taking into account the link failures.

Use of network coding, however, achieves this goal without having to alter the intermediate network node operations upon observing link failures. Each failed link is detected and assumed to transmit only the zero symbol, which is then processed identically as when the link is in service. The following example illustrates a unicast over a network implementing a failure robust network code.

Example 6.2. Consider the network shown in Fig. 6.4 with the source at node A and the destination at node F. Edges AB and AC have capacity 2 and all other 1. The capacity of the min-cut between the source and the destination is 3. The network code illustrated in the figure enables transmission at rate 2 and robustness to any single link permanent failure.

For a given min-cut n multicast network, we can design a code that guarantees to each receiver rate equal to $k = n - t$ when no more t edges fail. It is straightforward to see how to achieve this

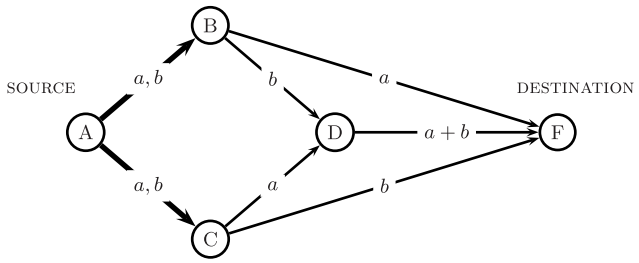


Fig. 6.4 Network coding provides instantaneous recovery when any single link fails. Each failed link is detected and assumed to transmit only the zero symbol. (Edges AB and AC have capacity 2 and all other 1.)

goal by using the algebraic framework for network code design. The basic idea is that we select the local coding vectors in the network so that, no matter which t edges fail, each receiver still has a full rank set of equations to solve. That is, while in the usual multicast scenario if we have N receivers we impose N constraints, requiring that the transfer matrix to each receiver is full rank, we will now simply impose additional constraints. The following example illustrates these points.

Example 6.3. Consider the network[3] in Figure 6.5 where we have two receivers and the min-cut to each receiver equals 3. Assume that the source would like to transmit information to each receiver at a rate equal to three. We would then require that the two transfer matrices $\mathbf{A}_1$, and $\mathbf{A}_2$, from the source to each receiver, are full rank, where

$$\mathbf{A}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha_3 & \alpha_4 \\ \alpha_1\alpha_5 & \alpha_5\alpha_2 + \alpha_6\alpha_3 & \alpha_6\alpha_4 \end{bmatrix}, \quad \text{and}$$

$$\mathbf{A}_2 = \begin{bmatrix} 0 & 0 & 1 \\ \alpha_1 & \alpha_2 & 0 \\ \alpha_1\alpha_5 & \alpha_5\alpha_2 + \alpha_6\alpha_3 & \alpha_6\alpha_4 \end{bmatrix}.$$

[3] We have already encountered this type of network and its subtree decomposition in Section 7, Figure 7.3, in [24].

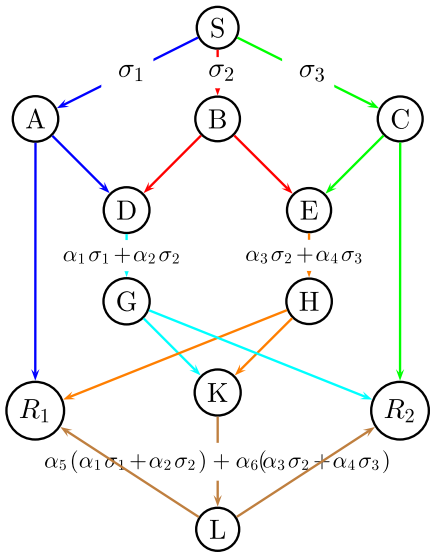


Fig. 6.5 A configuration with two receivers, and min-cut from the source equal to three.

To achieve this, it is sufficient to use the binary alphabet and set $\alpha_1 = \alpha_2 = \cdots = \alpha_6 = 1$.

Assume instead that we would like the source to transmit information at a rate equal to 2, and we want to design a network code so that, no matter which one edge of the network fails, the same network code still functions. Thus the symbols σ_1, σ_2, and σ_3 the source emits would depend upon two information symbols, say τ_1 and τ_2, related through a linear transformation

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{bmatrix} = \underbrace{\begin{bmatrix} \beta_1 & \beta_2 \\ \beta_3 & \beta_4 \\ \beta_5 & \beta_6 \end{bmatrix}}_{\mathbf{B}} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}.$$

If all edges function, the two receivers will observe the transfer matrices $\mathbf{A}_1\mathbf{B}$, and $\mathbf{A}_2\mathbf{B}$, respectively. If exactly one edge fails, receiver R_1 will observe the transfer matrix $\mathbf{A}_1\mathbf{B}$ where some of the numerical values and parameters $\{\alpha_i\}$, $\{\beta_i\}$ are set to zero. For example, if edge SA

fails, the transfer matrix for receiver R_1 equals

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & \alpha_3 & \alpha_4 \\ 0 & \alpha_5\alpha_2 + \alpha_6\alpha_3 & \alpha_6\alpha_4 \end{bmatrix} \mathbf{B}$$

where the numerical value 1 as well as α_1 are set to zero. Similarly for the second receiver. The subtree decomposition, described in Section 3, part I of this tutorial, can help us bound the number of distinct transfer matrices we need consider.

Codes that tolerate a collection of link failure patterns (e.g. all t link failures) can be designed by a straightforward extension of the LIF algorithm for network code design (discussed in Section 5, part I) which, in essence, runs the LIF algorithm in parallel for each failure pattern. Clearly, if the goal is to tolerate any t link failure, the runtime of this algorithm would be $\mathcal{O}\left(\binom{|E|}{t}\right)$ times that of the LIF algorithm, and the required alphabet size would be $\mathcal{O}\left(\binom{|E|}{t}\right)$ times larger than for the ordinary network code (with no robustness requirement) over the same network with E edges.

6.4 Network Codes for Random Packet Error Correction

In a network with random packet errors, each edge is modeled as a channel that can introduce additive errors or erasures. At any given transmission slot, only a limited number t of edges can introduce errors but the choice of these edges is arbitrary. In this section, we briefly describe two approaches to design network error correcting codes for random packet errors: network block codes and network codes based on subspaces.

6.4.1 Network Block Codes

Consider for a moment a simple unicast scenario shown in Figure 6.6. Assume that the source is connected to the destination through h edge-disjoint paths each consisting of unit-capacity edges, and that any t of these edges can introduce errors. Note that an error can

propagate only through the path on which it is introduced. Therefore, the receiver will observe at most t corrupted symbols, and the source can send information at rate k to the receiver by the means of an $(h, k, 2t+1)$ channel block code that maps k information symbols to h coded symbols, and can correct any t random errors. The coded symbols will be sent simultaneously on the h edges, thus using the "space" as the resource to accommodate for the redundancy introduced by the code, rather than time as in classical channel coding where coded symbols are sent sequentially and errors occur at different time slots.

Consider now a multicast scenario where a source is connected to multiple receivers through an arbitrary network with the min-cut between the source and each receiver equal to h. Note that if different sets of h edge disjoint paths between the source and different receivers do not overlap, network coding is not necessary, and the problem reduces to that of Figure 6.6 for each receiver separately, and thus to classical channel coding. Otherwise, we want to design the network code to protect the source message for each receiver.

Definition 6.1. A network code is t-error correcting if it enables each receiver to correctly receive source information as long as the number of edges that introduce errors is smaller than t.

Unlike classical channel codes, network error correction codes use coding operations not only at the source, but also at intermediate nodes of the network. Therefore, an error introduced at a single edge may propagate throughout the network. For simplicity we will assume that both the source and the channel (network) alphabet are $\mathbb{F}_q$. Therefore,

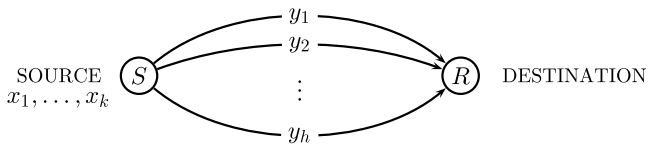


Fig. 6.6 At the source node, k information symbols are encoded into h coded symbols, which are then simultaneously transmitted to the destination.

the source produces q^k messages. If the network code is such that each receiver can correctly decode if errors are introduced in up to t edges, then, for each receiver, the set of q^k h-dimensional vectors at any h-cut corresponding to the q^k different source messages will be an (h, k) classical code that can correct at least t errors. Therefore, the appropriate channel coding bounds connecting parameters q, h, k, and t hold. For example, we have

Sphere Packing (Hamming) Bound:

$$q^k \leq \frac{q^h}{\sum_{k=0}^{t} \binom{h}{k}(q-1)^k}.$$

Singleton Bound

$$q^k \leq q^{h-2t}.$$

If the source alphabet size is $\mathcal{Z}$, we can replace the q on the left-hand side by $|\mathcal{Z}|$. Similarly, if the source alphabet size is $\mathcal{X}$, we can replace the q on the right-hand side by $|\mathcal{X}|$.

The network generalizations of the error vector Hamming weight and the code minimum distance also exist, and the minimum distance has the same operational meaning as in channel coding. In what follows, we will first formally define the minimum network Hamming distance of a code.

We will assume without loss of generality that the source has outdegree exactly equal to h, by possibly adding an artificial node and artificial edges. Let $\boldsymbol{u}$ denote the $k \times 1$ information vector that the source encodes into a $h \times 1$ codeword $\boldsymbol{x}$ by the means of a code $\mathcal{C}$ defined by its $h \times k$ generator matrix $\mathbf{G}$, i.e.,

$$\boldsymbol{x} = \mathbf{G}\boldsymbol{u}.$$

The codeword $\boldsymbol{x}$ will then be multicast through the network by the means of a network code. Receiver R_j observes the $h \times 1$ vector $\boldsymbol{y}_j$ given by

$$\boldsymbol{y}_j = \mathbf{A}_j \boldsymbol{x}$$

where $\mathbf{A}_j$ is the $h \times h$ network code transfer matrix for receiver R_j. As we discussed in Section 3, part I, A_j can be obtained as

$$\mathbf{A}_j = \mathbf{C}_j(\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}, \tag{6.1}$$

where $\mathbf{B}$ is the $|E| \times h$ matrix that reflects the way the source is connected to the network, $\mathbf{C}_j$ is the $h \times |E|$ matrix that reflects the way the receiver R_j is connected to the network, and $\mathbf{A}$ is the $|E| \times |E|$ matrix that contains the information about the network topology and the code. Elements of $\mathbf{A}$ are indexed by network edges, and element (i, j) can have a nonzero value only if the edge i ends where edge j starts. A network code is specified by selecting values for the nonzero elements of $\mathbf{A}$.

Assume now that each network edge introduces an additive error (possibly of value zero). We collect these errors into an $|E| \times 1$ vector $\boldsymbol{z}$. Note that the edges introducing errors act as additional independent sources. Therefore, the vector $\boldsymbol{y}_j$ that receiver R_j observes will depend on both the input vector $\boldsymbol{x}$ and the error vector $\boldsymbol{z}$ as follows:

$$\boldsymbol{y}_j(\boldsymbol{x}, \boldsymbol{z}) = \mathbf{C}_j(\mathbf{I} - \mathbf{A})^{-1}(\mathbf{B}\boldsymbol{x} + \boldsymbol{z}) = \mathbf{A}_j\boldsymbol{x} + \mathbf{F}_j\boldsymbol{z}.$$

Note that the error vector's contribution to the output $\mathbf{F}_j\boldsymbol{z}$ depends on the network code (specified by matrix $\mathbf{A}$) through matrix $\mathbf{F}_j = \mathbf{C}_j(\mathbf{I} - \mathbf{A})^{-1}$.

Let $w_{\mathrm{H}}(\boldsymbol{z})$ denote the Hamming weight of $\boldsymbol{z}$. Clearly, it is not the Hamming weight $w_{\mathrm{H}}(\boldsymbol{z})$ of the error vector $\boldsymbol{z}$ but its contribution to the output $\mathbf{F}_j\boldsymbol{z}$ that directly reduces the distance between possible receiver's observations. We therefore define, for receiver j, the error vector *network* Hamming weight as the minimum Hamming weight of all error vectors which make the same contribution to the output:

$$w^j(\boldsymbol{z}) = \min_{\boldsymbol{z}' : \mathbf{F}_j\boldsymbol{z}' = \mathbf{F}_j\boldsymbol{z}} w_{\mathrm{H}}(\boldsymbol{z}'). \tag{6.2}$$

Following the same line of thought, we define the network Hamming distance between two output vectors $\boldsymbol{y}$ and $\boldsymbol{y}'$ as

$$d^j(\boldsymbol{y}_j, \boldsymbol{y}'_j) = \min_{\boldsymbol{z} : \mathbf{F}_j\boldsymbol{z} = \boldsymbol{y}_j - \boldsymbol{y}'_j} w_{\mathrm{H}}(\boldsymbol{z}).$$

It can be shown that d^j is a metric.

Each receiver implements a minimum distance decoder that, for the received vector $\boldsymbol{y}_j$ finds the codeword $\boldsymbol{x} \in \mathcal{C}$ such that $\mathbf{A}_j\boldsymbol{x}$ is closest to $\boldsymbol{y}_j$ as measured by d^j. The error correcting capabilities of $\mathcal{C}$ on the network implementing a network code that gives rise to $\mathbf{A}_j$ are determined by the network minimum distance of $\mathcal{C}$, defined as

$$d_{\min}(\mathcal{C}) \triangleq \min_j \min_{\boldsymbol{x},\boldsymbol{x}' \in \mathcal{C}: \boldsymbol{x} \neq \boldsymbol{x}'} d^j(\mathbf{A}_j\boldsymbol{x}, \mathbf{A}_j\boldsymbol{x}').$$

As long as the errors are introduced at fewer that $d_{\min}/2$ edges, the received vector will be closer to the error free output corresponding to the transmitted codeword than to the error free output corresponding to any other codeword in the code. Consequently, a minimum distance decoder will make a correct decision. To see that, suppose that $\boldsymbol{x}$ is transmitted and errors are introduced at $\tau < d_{\min}/2$ edges, that is, $w_{\mathrm{H}}(\boldsymbol{z}) = \tau$. From the definition (6.2), we know that $w^j(\boldsymbol{z}) < w_{\mathrm{H}}(\boldsymbol{z})$. Let $\boldsymbol{y}_j$ be the receiver's observation. The network distance of $\boldsymbol{y}_j$ to the error free output $\mathbf{A}_j\boldsymbol{x}$ corresponding to the transmitted sequence $\boldsymbol{x}$ is smaller than $d_{\min}(\mathcal{C})/2$:

$$\begin{aligned} d^j(\boldsymbol{y}_j, \mathbf{A}_j\boldsymbol{x}) &= \min_{\boldsymbol{z}': \mathbf{F}_j\boldsymbol{z}' = \boldsymbol{y}_j - \mathbf{A}_j\boldsymbol{x}} w_{\mathrm{H}}(\boldsymbol{z}') \\ &= \min_{\boldsymbol{z}': \mathbf{F}_j\boldsymbol{z}' = \mathbf{F}_j\boldsymbol{z}} w_{\mathrm{H}}(\boldsymbol{z}') \\ &= w^j(\boldsymbol{z}) < w_{\mathrm{H}}(\boldsymbol{z}) = \tau < d_{\min}(\mathcal{C})/2. \end{aligned}$$

On the other hand the network distance of $\boldsymbol{y}_j$ to the error free output $\mathbf{A}_j\boldsymbol{x}'$ corresponding to some other sequence $\boldsymbol{x}'$ is larger than $d_{\min}(\mathcal{C})/2$:

$$\begin{aligned} d^j(\boldsymbol{y}_j, \mathbf{A}_j\boldsymbol{x}') &> d^j(\mathbf{A}_j\boldsymbol{x}, \mathbf{A}_j\boldsymbol{x}') - d^j(\boldsymbol{y}_j, \mathbf{A}_j\boldsymbol{x})) \\ &\geq d_{\min}(\mathcal{C}) - \tau > d_{\min}(\mathcal{C})/2. \end{aligned}$$

From this analysis, we also see that if errors are introduced at more than one but fewer than $d_{\min}$ edges, the received vector cannot be equal to the error free output corresponding to any codeword of the code. Moreover, if the error introducing edges are known (which effectively makes them erasures), then a minimum distance decoder will identify the transmitted codeword correctly.

Since a code with minimum distance $d_{\min}$ can correct any errors introduced at up to $\tau < d_{\min}/2$ errors, the Singleton bound, stated above, can be expressed as

$$d_{\min} \leq h - k + 1.$$

Codes that can correct erasures introduced at $h - k = d_{\min} - 1$ edges are essentially codes that can tolerate $n - k$ link failures, and can be designed as we discussed in the previous section on robustness to non-ergodic link failures. Such codes are at the same time capable of correcting errors introduced at $\lfloor (h-k)/2 \rfloor$ edges. As before, the algorithm is a straightforward extension of the LIF algorithm for network code design which, in essence, runs the LIF algorithm in parallel for each erasure pattern. Clearly, if the goal is to tolerate any t link failure, the runtime of this algorithm would be $\mathcal{O}\left(\binom{|E|}{d_{\min}-1}\right)$ times that of the LIF algorithm, and the required alphabet size would be $\mathcal{O}\left(\binom{|E|}{d_{\min}-1}\right)$ times larger than for the ordinary network code (with no robustness requirement) over the same network with E edges.

Constructing network error correction codes is very closely related to constructing network codes that protect against Byzantine security attacks, discussed in Section 7.

6.4.2 Network Codes Based on Subspaces

A different approach to network error correction stems from the observation that inputs to the network are $n \times 1$ vectors over $\mathbb{F}_q$. Each receiver observes these input vectors multiplied by an $n \times n$ matrix over $\mathbb{F}_q$. Therefore, if the source sends an arbitrary basis $\{b_1, \ldots, b_\ell\}$ of some subspace U of F_q^n through an error-free network implementing any linear network code, then all nodes will be receiving vectors from U. Consequently, each receiver that observes ℓ linearly independent vectors corresponding to the transmission of $\{b_1, \ldots, b_\ell\}$ will be able to identify the subspace U. Therefore, the source can communicate its messages by mapping them into subspaces of $\mathbb{F}_q^n$.

In principle, the source can use subspaces of arbitrary sizes to convey its messages. For simplicity, we will assume that the codewords for the source come from a *Grassmannian*, that is, the set of all ℓ-dimensional

subspaces of the n dimensional space $\mathbb{F}_q^n$, and each codeword is one such subspace. Note that there are

$$\begin{bmatrix} n \\ \ell \end{bmatrix}_q = \frac{(q^n-1)(q^{n-1}-1)\cdots(q^{n-\ell+1}-1)}{(q^\ell-1)(q^{\ell-1}-1)\cdots(q-1)}$$

ℓ-dimensional subspaces of the n-dimensional space $\mathbb{F}_q^n$, where $\begin{bmatrix} n \\ \ell \end{bmatrix}_q$ are known as the *Gaussian coefficients* in combinatorics. It is desirable to choose for the code $\mathcal{C}$ the subspaces from the Grassmannian that are the least likely to be confused with one another after passing through a network with errors and erasures.

A network with errors and erasures implementing a linear network code acts as an operator channel that takes in a vector space, say U, and then either deletes some of its vectors (erasures) or adds some other vectors (errors) or both so that a result is another vector space, say V. More precisely the channel input U and channel output V are related as

$$V = U' + E, \tag{6.3}$$

where, in a network with ρ erasures and t errors, U' is an $(\ell-\rho)$-dimensional subspace of U and E is some t-dimensional subspace of $\mathbb{F}_q^n \backslash U$.

Each receiver implements a minimum distance decoder that upon receiving a subspace outputs the closest codeword (subspace) from the code $\mathcal{C}$ in the following distance measure:

$$D(X,Y) = \dim(X+Y) - \dim(X \cap Y). \tag{6.4}$$

It can be shown that D is a metric. Note that, if the network input U undergoes ρ erasures and t errors, turning the output V according to (6.3), then the distance between U and V is

$$D(U,V) = \ell + t - (\ell - \rho) = t + \rho. \tag{6.5}$$

The minimum distance of $\mathcal{C}$, denoted by $D(\mathcal{C})$ and defined as

$$D(\mathcal{C}) \triangleq \min_{X,Y \in \mathcal{C}: X \neq Y} D(X,Y),$$

allows us to characterize the error/erasure correction capabilities of $\mathcal{C}$. We simply note that, since D is a metric (and thus satisfies the triangle

inequality), if $D(\mathcal{C}) > 2(\rho + t)$, then ρ erasures together with t errors cannot bring the received subspace closer to some other codeword than the one transmitted because of (6.5). Therefore, ρ erasures together with t errors can be corrected by a minimum distance decoder. Counterparts to the Singleton, sphere-packing, and sphere-covering bounds exist for codes based on Grassmannians, as well.

Note that this approach to network error connection is not concerned with the network code, but only with the coding at the source. A similar approach, that uses subspaces as codewords, has also been used for quantum error correction and for communications over unknown wireless channels.

Notes

Separability of channel and network coding was examined by Song et al. [81], Ratnakar and Kramer [68], and Tuninetti and Fragouli [85]. LT codes were proposed by Luby [53], Tornado codes by Luby et al. [54], and Raptor codes by Shokrollahi [78]. Packet level codes that employ intermediate node processing have been proposed by Lun et al. [55, 56]. Additional coding schemes that operate under complexity constraints, such as fixed, finite memory at intermediate nodes have also been investigated by Pakzad et al. [64] and Lun et al. [57]. Non-ergodic failures were studied by Jaggi et al. [41] and Ho et al. [36], and for unicsat by El Rouayheb et al. [73] from which the example shown in Figure 6.4 is taken. Network error correcting codes and bounds have been considered by Yeung and Cai [8, 90], Zhang [92] and Yang et al. [87, 88]. The subspace approach has been recently proposed by Koetter and Kscischang [46]. Coding schemes using this approach are developed by Silva and Kscischang [79].

7

Security

Network coding affects security and reliability of data in both favorable and adverse ways depending on the network scenario and the application. The following toy example will help us appreciate these issues.

Example 7.1. Consider the simple network consisting of two parallel unit-capacity channels, as shown in Figure 7.1. There are two independent unit-rate information sources located at node A and a legitimate user at node D who has paid to receive the information from both sources e.g., watch two movies. This goal can be achieved by either forwarding as in Figure 7.1(a) or coding as in Figure 7.1(b). Consider now an adversary who has access to a single edge in the network. If the transmission is organized as in Figure 7.1(a), the adversary receives complete information of one source. If the transmission is organized as in Figure 7.1(b), the adversary still receives one bit of information about the pair (y_1, y_2) but that information may be useless (as in the movie application).

Consider now an adversary who is capable of not only observing but also modifying the data on a single edge. In that case, the transmission

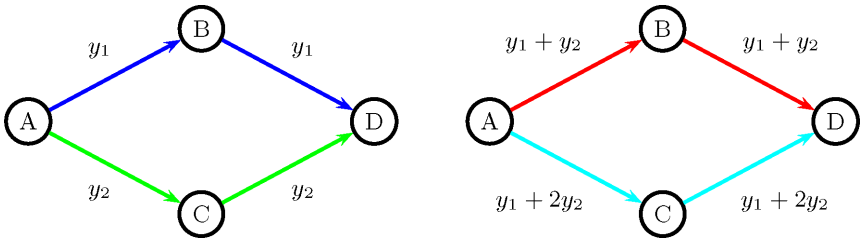


Fig. 7.1 Mixing information streams can both hide meaningful information and make legitimate users more vulnerable.

scheme in Figure 7.1(a) is better since the legitimate user is still able to receive the information from one source.

We will here consider two general cases in which an adversary Calvin has access to a certain number of edges in the network.

In the first case, Calvin is merely eavesdropping. In one scenario, his goal may be to reduce his uncertainty about the information transmitted to the intended receivers. In another scenario, his goal may be to actually decode a fraction of this information from his observation. We will see that linear combining may either assist or hinder Calvin depending on his goal. We will also look into designing network codes that will prevent the eavesdropper from achieving his goal.

In the second case, Calvin can also modify some of the packets he intercepts, i.e., perform a jamming attack. Modifying a certain number of packets in networks which only route information simply results in their incorrect reception, whereas modifying the same number of packets carrying linear combinations of source packets can have a more harmful effect since, if no counter measures are taken, it can result in incorrect decoding of all source packets. We will see how this problem can be controlled by sending some redundant information together with the data through the network.

7.1 Eavesdropping

We here consider multicast networks in which the adversary Calvin can access data on μ links of his choice, and has unlimited computational power. Our goal is to maximize the multicast rate with the constraint

of revealing no information to Calvin. To do so, the source needs to introduce some redundancy to the data it sends, using a wiretap-secure coding scheme. The question is, how does this coding scheme "interacts" with the network code employed by the nodes in the network.

The problem can be mathematically formulated as follows. Assume that the min-cut to each receiver equals n. Let $s = (s_1, s_2, \ldots, s_k)$ be the random variables associated with the k information symbols that the source wishes to send securely, $y = (y_1, y_2, \ldots, y_n)$ the random variables associated with the encoded symbols the source actually multicasts to the receivers and $z = (z_1, z_2, \ldots, z_\mu)$ the random variables associated with the wiretapped symbols that Calvin intercepts.

We distinguish between two levels of security. A scheme is *information theoretically secure* if s is completely determined (decodable) by y, and the uncertainty about s is not reduced by the knowledge of z, that is,

$$H(s|y) = 0 \quad \text{and} \quad H(s|z) = H(s). \tag{7.1}$$

On the other hand, a scheme is *weakly secure* if the uncertainty about a particular s_i is not reduced by the knowledge of z, that is,

$$H(s|y) = 0 \quad \text{and} \quad H(s_i|z) = H(s_i)\ \forall i, \tag{7.2}$$

but possibly $H(s|z) < H(s)$. The following examples illustrate the difference between these two notions of security.

Example 7.2. For the network in Figure 7.1(a), an information secure coding scheme with $n = 2$, $k = 1$, and $\mu = 1$ can be organized as follows. If the source bit s_1 equals 0, then either 00 or 11 is transmitted through the channel with equal probability. Similarly, if the source bit equals 1, then either 01 or 10 is transmitted through the channel with equal probability.

codeword y_1y_2 chosen at random from:	$\{00, 11\}$	$\{01, 10\}$
source bit s_1:	0	1

It is easy to see that knowledge of either y_1 or y_2 does not reduce the uncertainty about s_1, whereas knowledge of both y_1 and y_2 is sufficient to completely determine s_1, namely, $s_1 = y_1 + y_2$.

Example 7.3. Figure 7.1(b) provides an example of weak security, using $y_1 = s_1$ and $y_2 = s_2$. If, for example, s_1 and s_2 take values uniformly at random over $\mathbb{F}_3$, then $H(s_1) = H(s_2) = \log 3$. Assume that Calvin intercepts the value of $y_1 + y_2 = s_1 + s_2$, and attempts to guess s_1. It is easy to see that s_1 may still take all three values in $\mathbb{F}_3$ with equal probability, that is, $H(s_1|s_1 + s_2) = \log 3$. In other words, the probability of error Calvin will make is the same, as if he were attempting to randomly guess the value of s_1.

7.1.1 Information Theoretic Security

For a point-to-point scenario in which the source can transmit n data symbols to the receiver and an adversary can access any μ of those symbols, it is well known that the maximum number of symbols that the source can communicate to the receiver securely in the information theoretic sense is equal to $k = n - \mu$.

This can be achieved using a $[n_1, k_1]$ linear MDS code $\mathcal{C} \subset \mathbb{F}_q^{n_1}$. Recall that the codewords in a $[n_1, k_1]$ linear code are all the vectors in a k_1-dimensional subspace of the n_1-dimensional space. The associated generator matrix $\mathbf{G}$ has dimension $k_1 \times n_1$ while the parity matrix $\mathbf{H}$ has dimension $n_1 - k_1 \times n_1$. The rows of $\mathbf{G}$ form a basis of this subspace $\mathcal{C}$, and the rows of $\mathbf{H}$ form a basis of the orthogonal space, i.e., $\mathbf{H}y^T = 0$ for any codeword $y \in \mathcal{C}$. $\mathcal{C}$ is also an abelian subgroup of the group $\mathbb{F}_q^{n_1}$, and its distinct cosets divide $\mathbb{F}_q^{n_1}$ into $q^{n_1 - k_1}$ sets of size q^{k_1}. Two vectors x_1 and x_2 belong to the same coset, if and only if $x_1 - x_2 \in \mathcal{C}$, and $\mathbf{H}x_1 = \mathbf{H}x_2$ (this last quantity is called the syndrome associated with the coset). An MDS code meets the Singleton bound and thus has minimum distance equal to $d_1 = n_1 - k_1 + 1$. This implies that any two vectors of $\mathbb{F}_q^{n_1}$ in the same coset differ in at least d_1 positions.

To create an information secure network code that transmits k symbols, we can use an MDS code with $n_1 = n$ and $k_1 = n - k$, that has k syndrome values. The k information symbols are taken as a syndrome which specifies a coset. The transmitted word is chosen uniformly at random from the vectors in the specified coset. The decoder

that receives a vector y recovers the information symbols by simply computing the syndrome Hy of the received word.

Assume now that the adversary Calvin learns $\mu = n - k = k_1$ symbols of y. The question is, whether he can infer from this any information on what the syndrome is. The answer is no, because, each of the $q^{n_1-k_1}$ vectors that contain the k_1 symbols Calvin intercepted, belongs in a different one of the $q^{n_1-k_1}$ cosets. Indeed, any two of the $q^{n_1-k_1}$ vectors, say y_1 and y_2, differ in at most $n_1 - k_1$ positions. But the minimum distance equals $d_1 = n_1 - k_1 + 1$, and thus no two such vectors can belong in the same coset.

The code used in Example 7.2 is the $[2,1]$ repetition code over $\mathbb{F}_2$ with parity check matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 1 \end{bmatrix},$$

cosets $\{00,11\}$, $\{01,10\}$, and distance 2.

Now consider a specific network code used to multicast n symbols from an information source to N receivers over an acyclic multicast network $G = (V,E)$ and assume that Calvin wiretaps any μ edges of G. Our next question is whether, using the same network code, the source can multicast $k \leq n - \mu$ symbols securely if it first applies a secure wiretap channel code (as described above) mapping k into n symbols. The answer is in general no, as the following example illustrates.

Example 7.4. Consider the butterfly network shown in Figure 7.2 where we have $n = 2$, $k = 1$, and $\mu = 1$. If the source applies the coding scheme described in Example 7.2, based on the MDS code over $\mathbb{F}_2$ with $\mathbf{H} = \begin{bmatrix} 1 & 1 \end{bmatrix}$, and the usual network as in Figure 7.2(a), the adversary will be able to immediately learn the source bit if he taps into any of the edges BE, EF, ED. This is because the coding vector $\begin{bmatrix} 1 & 1 \end{bmatrix}$ reveals the product of a row of $\mathbf{H}$ and the transmitted word y, i.e., reveals one bit of the syndrome. Therefore, this network code breaks down the secure-wiretap channel code.

However, if the network code is changed so that node B combines its inputs over $\mathbb{F}_3$ and the BE coding vector is $\begin{bmatrix} 1 & \alpha \end{bmatrix}$ where α is a primitive element of $\mathbb{F}_3$ (as in Figure 7.2(b)), the wiretap channel code remains secure, that is, the adversary cannot gain any information by

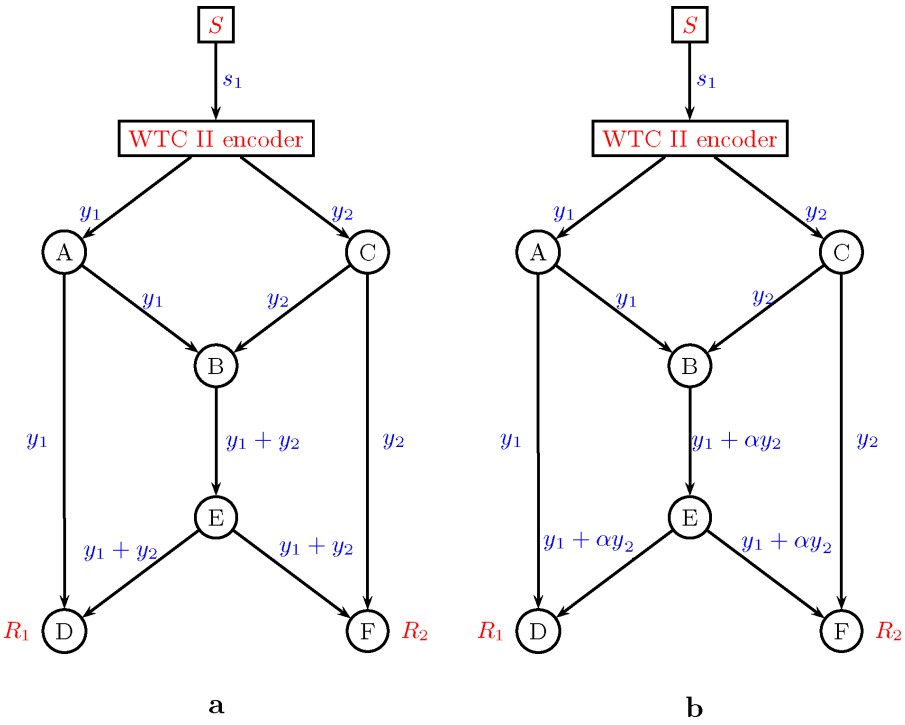


Fig. 7.2 Single-edge wiretap butterfly network with (a) insecure network code and (b) secure network code.

accessing any single link in the network. In general, the wiretap channel code remains secure with any network code whose BE coding vector is linearly independent of $\begin{bmatrix}1 & 1\end{bmatrix}$.

More generally, the source can multicast $k \leq n - \mu$ symbols securely if it first applies a secure-wiretap code based on an MDS code with a $k \times n$ parity check matrix $\mathbf{H}$, and if the network code is such that no linear combination of $\mu = n - k$ or fewer coding vectors belongs to the space spanned by the rows of $\mathbf{H}$. That is, if any μ coding vectors together with the rows of $\mathbf{H}$ form a basis of the n-dimensional space. This guarantees that the symbols sent through the network edges cannot be used to infer information about the syndrome bits.

Information theoretical arguments can also be used to prove this assertion. Let $W \subset E$ denote the set of $|W| = \mu$ edges the wiretapper

chooses to observe, and $\mathbf{C}_W$ denote the matrix whose rows are the coding vectors associated with the observed edges in W. Consider $H(s, y, z)$ with the security requirement $H(s|z) = H(s)$ to obtain

$$\begin{aligned}
&\underbrace{H(s|z)}_{=H(s)} + H(y|s,z) = H(y|z) + \underbrace{H(s|y,z)}_{=0} \\
&\Rightarrow H(y|s,z) = H(y|z) - H(s) \\
&\Rightarrow 0 \leq n - \operatorname{rank}(\mathbf{C}_W) - k.
\end{aligned}$$

Since there is a choice of edges such that $\operatorname{rank}(\mathbf{C}_W) = \mu$, the maximum rate for secure transmission is bounded as

$$k \leq n - \mu.$$

If the bound is achieved with equality, we have $H(y|s,z) = 0$ and consequently, the system of equations

$$\begin{bmatrix} s \\ z \end{bmatrix} = \begin{bmatrix} \mathbf{H} \\ \mathbf{C}_W \end{bmatrix} \cdot y$$

has to have a unique solution for all W for which $\operatorname{rank}(\mathbf{C}_W) = \mu$.

To conclude, we showed that given a fixed wiretap-secure code, we can find a network code that does not affect the security. The reverse procedure is also possible: we can start with a fixed network code, and select an appropriate wiretap-secure code, that is not compromised by the network code.

7.1.2 Weak Security

Again consider a multicast network where the min-cut to each receiver equals n. If we do not insist on information theoretic security, but instead, weak security as defined in (7.2) is sufficient, we can send information to the receivers at rate equal to n, even if Calvin eavesdroppes on $\mu = n - 1$ edges.

The basic idea is very simple. We create y by encoding the source symbols s with an invertible matrix $\mathbf{A}$, i.e., $y = \mathbf{A}s$. Calvin observes $z = \mathbf{B}y = \mathbf{BA}s$, where the $\mu \times n$ matrix $\mathbf{B}$ has as rows the coding vectors on the edges Calvin wiretaps. It is sufficient to select the matrix $\mathbf{A}$

and the network code so that, for all possible matrices $\mathbf{B}$, there does not exist a vector c_i with $c_i\mathbf{BA}s = s_i$, for all i. This is always possible, provided we use coding over a large enough finite field.

To see that this construction guarantees weak security recall that if we start with a uniform distribution on the elements of a finite field, and perform any linear transformation, the distribution on the resulting output will also be uniform. For example, if s_1 and s_2 are uniformly distributed over $\mathbb{F}_2$, so is $y_1 = s_1 + s_2$, and the distribution of s_1 conditioned on y_1 is also uniform. Thus, if the linear combinations Calvin collects do not allow him to decode a source symbol s_i, the probability distribution of s_i conditioned on these observed values is still uniform, and (7.2) holds.

7.2 Byzantine Modification

Byzantine fault broadly refers to any fault in a distributed system coming from a failure in some subset of its components. The name comes from the frame such problems are often put into, referring to Byzantine generals deciding through messengers on a common plan of attack when some of them may be traitors lying about whether they will support a particular plan and what other generals told them. Because it requires that multiple nodes in the network perform prescribed, agreed upon tasks (e.g., linear combining and forwarding), network coding is a distributed process and as such vulnerable to Byzantine attacks. Much as Byzantine generals, the nodes in the network may forward faulty information pretending it is genuine.

A Byzantine attacker can, for example, supply a certain number μ of modified packets to the receiver, who in turn may decode all of the source packets incorrectly. We can distinguish different types of attack, depending on the power of the attacker, the mode of communication between the source and the receiver, and the desired level of protection. For example, the attacker may be able to eavesdrop on all network edges, although he may only jam μ of them, or, he may only be able to eavesdrop on the μ edges it jams. There may exist a secret channel, outside the reach of the attacker, through which the source can send a few bits to the receivers, or not. At the receiver side, we may be

interested in simply detecting whether a Byzantine attack has taken place to discard the corrupted packets, or we may want to also be able to filter some of the information the source was sending while the attack was under way. In all cases, the natural way to protect data is by introducing some type of error correction coding so that the packets carry not only data but also some redundant information.

We will consider a particular type of Byzantine attack in which there is a low-rate secret channel between the source and each receiver, at a multicast network in which the min-cut between the source and each receiver equals n. Each receiver is interested in recovering some information from the source, and the attacker observes all transmissions and can insert μ corrupted packets. In this case, there exist polynomial time algorithms for encoding and decoding that allow the receivers to securely recover information from the source at a rate of $n - \mu$. This clearly is the maximum achievable rate, if the rate of the secret channel goes to zero as compared to the rate supported by the network.

We will consider a practical set-up, where the source produces n packets of length L information symbols each, and network nodes randomly combine and exchange linear combinations of the source packets. A header of length n appended to each packet specifies the linear combination that the packet carries. Let $\mathbf{S}$ be the $n \times (L + n)$ matrix whose rows are the packets that the source sends, and contains the $n \times n$ identity matrix $\mathbf{I}$. Then the receiver observes

$$\mathbf{Y} = \mathbf{CS} + \mathbf{C}_A\mathbf{Z}, \tag{7.3}$$

where $\mathbf{Z}$ is the $\mu \times (L + n)$ matrix whose rows are the packets injected by the attacker (Calvin), and $\mathbf{Y}$ is the received set of of packets. The matrix $\mathbf{C}$ is the $n \times n$ transfer matrix from the source to the receiver, and the matrix $\mathbf{C}_A$ the $n \times \mu$ transfer matrix from Calvin to the receiver.

Because operations at network nodes occur symbolwise, the identity matrix $\mathbf{I}$ contained in $\mathbf{S}$ undergoes the same transform as the original matrix $\mathbf{S}$. Thus $\mathbf{Y}$ contains the matrix

$$\widehat{\mathbf{C}} = \mathbf{CI} + \mathbf{C}_A\mathbf{L}, \tag{7.4}$$

for some matrix $\mathbf{L}$ contained in $\mathbf{Z}$. Combining (7.4) and (7.3) gives

$$\mathbf{Y} = \widehat{\mathbf{C}}\mathbf{S} + \mathbf{C}_A(\mathbf{Z} - \mathbf{L}\mathbf{S}) = \widehat{\mathbf{C}}\mathbf{S} + \mathbf{E}. \tag{7.5}$$

Matrix $\mathbf{E}$ summarizes the effect of the attack. The receiver knows $\mathbf{Y}$ and $\widehat{\mathbf{C}}$, and would like to decode $\mathbf{S}$.

To help the receiver do so, the source operates as follows. First, it randomly selects $n + L$ values over $\mathbb{F}_q$ and creates the Vandermonde parity check matrix $\mathbf{H}$ of dimension $(n + L) \times c$, where c is a design parameter:

$$\mathbf{H} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{c-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{c-1} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_{n+L} & x_{n+L}^2 & \dots & x_{n+L}^{c-1} \end{bmatrix}.$$

The source communicates $\mathbf{H}$ to all receivers using the secret channel. This communication occurs only once.

Additionally, every time it produces the n packets of length L symbols collected in the $n \times (n + L)$ matrix $\mathbf{S}$, it sends through the secure channel to all receivers the hashed matrix $\mathbf{P} = \mathbf{S}\mathbf{H}$. Note that the size of the information transmitted over the secret channel to convey the $n \times c$ matrix $\mathbf{P}$ can be made arbitrarily small as compared to the packet length L.

Now the receiver can take advantage of this extra information. First, it projects $\mathbf{Y}$ on the matrix $\mathbf{H}$

$$\mathbf{Y}\mathbf{H} = \widehat{\mathbf{C}}\mathbf{S}\mathbf{H} + \mathbf{E}\mathbf{H} = \widehat{\mathbf{C}}\mathbf{P} + \mathbf{X} \tag{7.6}$$

to learn the matrix $\mathbf{X} = \mathbf{E}\mathbf{H}$. The columns of this matrix span the same vector space as the columns of $\mathbf{E}$ (we will not prove this), and thus $\mathbf{E} = \mathbf{X}\mathbf{A}$ for some matrix $\mathbf{A}$. Then the receiver can express $\mathbf{Y}$ in (7.5) as

$$\mathbf{Y} = \begin{bmatrix} \widehat{\mathbf{C}} \, \mathbf{X} \end{bmatrix} \begin{bmatrix} \mathbf{S} \\ \mathbf{A} \end{bmatrix}. \tag{7.7}$$

Now it can also be shown that the matrix $[\widehat{\mathbf{C}} \, \mathbf{X}]$ has full column rank. Thus, the receiver can decode by simply solving the system of

linear equations in (7.7). The complexity of the described algorithm is $\mathcal{O}(Ln^2)$.

Notes

The problem of making a linear network code secure in the presence of a wiretap adversary that can look at a bounded number of network edges was first studied by Cai and Yeung [89]. They demonstrated the existence of a code over an alphabet with at least $\binom{|\mathcal{E}|}{k}$ elements which can support the multicast rate of up to $n - k$. Feldman et al. derived trade-offs between security, code alphabet size, and multicast rate of secure linear network coding schemes in [20]. The exposition here follows the work of El Rouayheb and Soljanin [72]. Weakly secure network coding was studied by Bhattad and Narayanan [6]. The Byzantine modification detection in networks implementing random network coding was studied by Ho et al. [35]. The algorithm we presented comes from Jaggi et al. [41], where achievable rates and algorithms for other cases can be found as well.

8

Quantum Networks

We have seen that in multicast communication networks, large throughput gains are possible with respect to their transportation or fluid counterparts because classical (as opposed to quantum) information can be duplicated, merged, or in general, processed in a way that (non-quantum) physical entities cannot. An interesting question to ask is whether anything can be gained by allowing processing of quantum information at nodes in quantum networks.

Consider the quantum counterpart of the network multicast scenario where the sources produce quantum states (e.g., qubits as opposed to bits) which are then sent over quantum noiseless channels to be simultaneously delivered to the receivers. Since quantum states are represented by physical entities, the problem of quantum multicast at first seems nothing more than the multicommodity flow problem of shipping a collection of different commodities through a common network so that the total flow going through each edge in the network does not exceed its capacity.

Although quantum information can be processed in certain ways determined by the laws of quantum mechanics, two operations essential in classical information networking, replication (cloning) and

broadcasting, are not possible. However, approximate and probabilistic cloning as well as compression of quantum states are possible, and have been used in attempts to find a quantum counterpart of network coding. In this section, we first introduce quantum information and allowed processing, and then discuss quantum networks and quantum multicast.

8.1 Quantum Information Systems

In this section, we give mathematical definitions of quantum information carriers, i.e., quantum states, describe physical operations that can be performed on such states, and list a number of state transformations relevant for network coding that cannot be achieved by allowed quantum operations.

8.1.1 Quantum States

Quantum states are, in the simplest case, mathematically represented as unit length column vectors in a d-dimensional complex vector space $\mathcal{H}$. Such quantum states are called *pure*. When $d = 2$, quantum states are called *qubits*. A column vector is denoted by $|\varphi\rangle$, its complex conjugate transpose by $\langle\varphi|$. A pure state is mathematically described by its *density matrix* equal to the outer product $|\varphi\rangle\langle\varphi|$.

In a more complex case, all we know about a quantum state is that it is one of a finite number of possible pure states $|\varphi_i\rangle$ with probability p_i. Such quantum states are called *mixed*. A mixed state is also described by its density matrix which is equal to

$$\rho = \sum_i p_i |\varphi_i\rangle\langle\varphi_i|.$$

Note that a density matrix is a $d \times d$ Hermitian trace-one positive semidefinite matrix. A classical analog to a mixed state can be a multifaced coin which turns up as any of its faces with the corresponding probability.

We often deal with sequences rather than with individual states. The quantum state corresponding to a sequence of length n has a $d^n \times d^n$

density matrix, equal to the tensor product of density matrices corresponding to the states in the sequence.

8.1.2 Quantum Operations

A quantum state ρ can be reversibly transformed to another state $\mathcal{E}(\rho)$ only by a physical process consistent with the lows of quantum theory. Such a process is, in the simplest case, mathematically described as a *unitary* evolution:

$$\mathcal{E}(\rho) = \mathbf{U}\rho\mathbf{U}^\dagger \text{ where } \mathbf{U}\mathbf{U}^\dagger = \mathbf{I},$$

and, in a more general case, as an evolution by a *completely positive, trace-preserving* map:

$$\mathcal{E}(\rho) = \sum_k \mathbf{E}_k \rho \mathbf{E}_k^\dagger \text{ where } \sum_k \mathbf{E}_k^\dagger \mathbf{E}_k = \mathbf{I}.$$

It is envisioned that a quantum computer (like a classical) would implement such evolutions by using universal quantum gates. An example of a two-qubit quantum gate is the `XOR`:

$$\texttt{XOR} : |x, y\rangle \rightarrow |x, x \oplus y\rangle \quad U_{\texttt{XOR}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{8.1}$$

where $x, y \in \{0, 1\}$ and $\oplus$ denotes the binary addition.

The above operations are reversible, and thus, in information theoretic sense, we can consider them lossless. Quantum measurements, which we describe next, are irreversible (lossy) quantum operations.

A quantum measurement is a physical process applied to determine the state of the quantum system being measured. Only when the possible states, say $\{|\psi_j\rangle,\ j = 1, \ldots, J\}$, are orthogonal can a quantum measurement be designed to give an unambiguous answer.

The simplest model of quantum measurement is known as the von Neumann's measurement. Mathematically, this type of measurement is defined by a set of pairwise orthogonal projection operators $\{\Pi_i\}$ which

form a complete resolution of the identity, that is, $\sum_i \Pi_i = I$. For input $|\psi_j\rangle$, the *classical* output $\Pi_i|\psi_j\rangle$ happens with probability $|\langle\psi_j|\Pi_i|\psi_j\rangle|^2$.

In a more general case, the pairwise orthogonal projection operators $\{\Pi_i\}$ are replaced by any positive-semidefinite operators $\{E_i\}$ which form a complete resolution of the identity. This type of measurement in known as positive operator-valued measure (POVM).

8.1.3 Cloning, Broadcasting, and Deleting

Since quantum information processing is limited in the way described above, quantum information cannot be cloned, broadcast or deleted in the sense made precise below, unless we are dealing with states with commuting density matrices.

8.1.3.1 The No-Cloning and No-Broadcasting Principles

There is no physical process that leads to an evolution (i.e., quantum process defined in Section 8.1.2)

$$|\phi\rangle \otimes |s\rangle \rightarrow |\phi\rangle \otimes |\phi\rangle,$$

where $|\phi\rangle$ is an arbitrary state and $|s\rangle$ is a fixed state.

However, cloning-like operations known as *approximate* and *probabilistic* cloning are possible, and were instrumental in developing network coding protocols for multiple unicasts. As the names of these operations suggest, an approximate clone is close in some distance measure to the original state, whereas, a probabilistic clone is exactly equal to the original state but not all the time the operation is performed. The generalization of the above claim to mixed states is known as the no-broadcasting principle.

8.1.3.2 The No-Deleting Principle

There is no physical process that leads to an evolution

$$|\phi\rangle \otimes |\phi\rangle \rightarrow |\phi\rangle \otimes |s\rangle,$$

where $|\phi\rangle$ is an arbitrary state and $|s\rangle$ is a fixed state.

8.1.4 Networks with Orthogonal States

Quantum information processing in networks in which sources produce orthogonal quantum states is a straightforward generalization of classical information processing, and all classical network coding results hold for such networks. Because of that, we will only describe the quantum generalization of multicast over the butterfly network, and then, starting with next section, move to networks with non-orthogonal quantum states.

Example 8.1. Consider a quantum counterpart of the butterfly network coding scheme where the sources produce mutually orthogonal quantum states $|0\rangle$ and $|1\rangle$. Let $|b_1\rangle$ and $|b_2\rangle$ (where $b_1, b_2 \in \{0,1\}$) be the qubits produced by sources S_1 and S_2, respectively. If each network node implements a quantum `XOR` gate as defined by (8.1), then the inputs $|x\rangle$ and $|y\rangle$ to the gate at each node and the outputs sent along the outgoing edges are as follows:

NODE	INPUT	OUTPUT
A	$\|x\rangle = \|b_1\rangle$	AB : $\|b_1\rangle$
	$\|y\rangle = \|0\rangle$	AD : $\|b_1\rangle$
B	$\|x\rangle = \|b_1\rangle$	BE : $\|b_1 \oplus b_2\rangle$
	$\|y\rangle = \|b_2\rangle$	
C	$\|x\rangle = \|b_2\rangle$	CB : $\|b_2\rangle$
	$\|y\rangle = \|0\rangle$	CF : $\|b_2\rangle$
D	$\|x\rangle = \|b_1\rangle$	$\|b_1\rangle$
	$\|y\rangle = \|b_1 \oplus b_2\rangle$	$\|b_2\rangle$
E	$\|x\rangle = \|b_1 \oplus b_2\rangle$	ED : $\|b_1 \oplus b_2\rangle$
	$\|y\rangle = \|0\rangle$	EF : $\|b_1 \oplus b_2\rangle$
F	$\|x\rangle = \|b_2\rangle$	$\|b_2\rangle$
	$\|y\rangle = \|b_1 \oplus b_2\rangle$	$\|b_1\rangle$

Therefore, this scheme enables quantum multicast at rate 2.

8.2 Single-Source Quantum Multicast

We now consider a simple quantum multicast scenario as illustrated in Figure 8.1. A quantum source of information S produces quantum states $|\phi\rangle \in \mathcal{H}$ which are to be reproduced at N receiving points $R_1, \ldots, R_N$ simultaneously. The source node is connected to another node A. Since A cannot copy or broadcast the quantum information it receives, the state $|\phi\rangle^{\otimes N} \in \mathcal{H}^{\otimes N}$, prepared at the source, has to be made available at A. The question is whether that state has to be actually transmitted through the quantum channel from S to A if nodes S and A are allowed to perform lossless compression and decompression of quantum information. Recall that if this network is classical, then only a single bit would have to be sent from S to A, which can then duplicate this bit and send it to the receivers. On the other hand, if N identical cars are to be delivered to N customers simultaneously over a network of highways with the same topology as the one shown in the figure, then clearly, N actual cars would have to be made at the source, and the highway between S and A would have to have N parallel lanes. We show that for the scenario of Figure 8.1, a quantum link between S and A that can carry $\Theta(\log N)$ qubits is necessary and sufficient

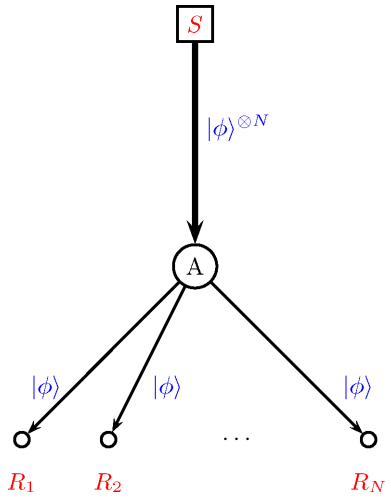


Fig. 8.1 Quantum multicast.

to accommodate multicast of any state $|\phi\rangle \in \mathcal{H}$ produced at the source.

The idea is to look into how states of the form $|\phi\rangle^{\otimes N} \in \mathcal{H}^{\otimes N}$ can be compressed. The smallest subspace of $\mathcal{H}^{\otimes N}$ that contains all vectors of the form $|\phi\rangle^{\otimes N} \in \mathcal{H}^{\otimes N}$ is known as the *symmetric subspace*. We denote this subspace by $\mathcal{SYM}(\mathcal{H}^{\otimes N})$. This subspace is precisely the subspace of $\mathcal{H}^{\otimes N}$ invariant under permutations of the qubits. We will next find a basis of the space $\mathcal{SYM}(\mathcal{H}^{\otimes N})$ and prove that its dimension is polynomial in N, and then show how any N-qubit state $|\phi\rangle^{\otimes N} \in \mathcal{H}^{\otimes N}$ can be compressed into a $\Theta(\log N)$-qubit state in $\mathcal{SYM}(\mathcal{H}^{\otimes N})$ and decompressed from it.

8.2.1 Types and Type Classes

We briefly review types and type classes since they will be instrumental in defining a basis for

$$\mathcal{SYM}(\mathcal{H}^{\otimes N}) = \langle \{|\phi\rangle^{\otimes N} : |\phi\rangle \in \mathcal{H}\} \rangle.$$

Let $\mathcal{X}$ be a finite set. Given a sequence $\boldsymbol{x} = (x_1, \ldots, x_N) \in \mathcal{X}^N$ and a letter $a \in \mathcal{X}$, let $\eta(a|\boldsymbol{x})$ denote the number occurrences of a in $\boldsymbol{x}$.

Definition 8.1. The *type* of a sequence $\boldsymbol{x} \in \mathcal{X}^N$ is the distribution $P\boldsymbol{x}$ given by

$$P\boldsymbol{x}(a) = \frac{1}{N}\eta(a|\boldsymbol{x}) \text{ for every } a \in \mathcal{X}.$$

Conversely, the *type class* of a distribution P is the set T_P^N of all sequences of type P in $\mathcal{X}^N$:

$$\mathsf{T}_P^N = \{\boldsymbol{x} : \boldsymbol{x} \in \mathcal{X}^N \text{ and } P\boldsymbol{x} = P\}.$$

The set consisting of all possible types of sequences $\boldsymbol{x} \in \mathcal{X}^N$ is denoted by $\mathcal{P}_N(\mathcal{X})$. It is easy to show by elementary combinatorics that

$$|\mathcal{P}_N(\mathcal{X})| = \binom{N + |\mathcal{X}| - 1}{|\mathcal{X}| - 1}. \tag{8.2}$$

Note that the number of types is polynomial in N.

8.2.2 A Basis for Vector Space $\langle\{|\phi\rangle^{\otimes N} : |\phi\rangle \in \mathcal{H}\}\rangle$

Let $|\phi\rangle$ be a vector in a d-dimensional complex vector space $\mathcal{H}$ with a basis $\{|e_i\rangle,\ i = 1,\ldots,d\}$:

$$|\phi\rangle = \sum_{i=1}^{d} \alpha_i |e_i\rangle.$$

Let $\mathcal{X} = \{1,\ldots,d\}$ and $\boldsymbol{x} = (x_1,\ldots,x_N) \in \mathcal{X}^N$. Consider the vector $|\phi\rangle^{\otimes n} \in \mathcal{H}^{\otimes n}$:

$$|\phi\rangle^{\otimes n} = \left(\sum_{i=1}^{d} \alpha_i |e_i\rangle\right)^{\otimes N} = \sum_{\boldsymbol{x}\in\mathcal{X}^N} \alpha_{\boldsymbol{x}} |e_{\boldsymbol{x}}\rangle,$$

where

$$\alpha_{\boldsymbol{x}} = \prod_{j=1}^{N} \alpha_{x_j} \quad \text{and} \quad |e_{\boldsymbol{x}}\rangle = |e_{x_1}\rangle \otimes \cdots \otimes |e_{x_N}\rangle.$$

We further have

$$|\phi\rangle^{\otimes n} = \sum_{\boldsymbol{x}\in\mathcal{X}^n} \alpha_{\boldsymbol{x}} |e_{\boldsymbol{x}}\rangle = \sum_{P\in\mathcal{P}_N(\mathcal{X})} \prod_{i=1}^{d} \alpha_i^{NP(i)} \sum_{\boldsymbol{x}\in\mathsf{T}_P^N} |e_{\boldsymbol{x}}\rangle.$$

Consequently, a vector of the form $|\phi\rangle^{\otimes N}$ has $|\mathcal{P}_N(\mathcal{X})|$ degrees of freedom, and the orthonormal basis vectors for $\langle\{|\phi\rangle^{\otimes N} : |\phi\rangle \in \mathcal{H}\}\rangle$ are indexed by types:

$$|E_P\rangle = \frac{1}{\sqrt{|\mathsf{T}_P^N|}} \sum_{\boldsymbol{x}\in\mathsf{T}_P^N} |e_{\boldsymbol{x}}\rangle,\ P \in \mathcal{P}_N(\mathcal{X}). \tag{8.3}$$

Since $|\mathcal{X}| = d$, from (8.2) we get that the number of these vectors is

$$\binom{N+d-1}{d-1}.$$

We have shown that all vectors in the space $\mathcal{H}^{\otimes N}$ (whose dimension is d^N) that have the form $|\phi\rangle^{\otimes N}$ actually belong to a subspace of $\mathcal{H}^{\otimes N}$ whose dimension is only polynomial in N (linear in the binary case $d = 2$). We next exploit this fact to show that instead of sending $|\phi\rangle^{\otimes N}$ along the noiseless quantum channel from S to A in Figure 8.1, it is sufficient (and necessary) to send only $\Theta(\log N)$ qubits to communicate the same quantum information.

8.2.3 Lossless Compression into and Decompression from $\mathcal{SYM}(\mathcal{H}^{\otimes N})$

We describe a simple quantum algorithm to compress into and decompress from $\mathcal{SYM}(\mathcal{H}^{\otimes N})$ using arbitrary single-qubit and two-qubit gates, in addition to arbitrary reversible classical computation. Any of those gates can in turn be implemented by a circuit of gates from any *universal gate set*. For the clarity of the presentation and because generalization is straightforward, from now on we consider the binary case only, i.e., $\mathcal{X} = \{0,1\}$.

For an N-bit binary string $x \in \{0,1\}^N$, let $|x|$ denote its Hamming weight (the number of 1s in the string). Since the type class of x is determined by its Hamming weight, the basis for $\mathcal{SYM}(\mathcal{H}^{\otimes N})$ (as defined by(8.3)) is

$$|E_i\rangle = \frac{1}{\sqrt{\binom{N}{i}}} \sum_{x\in\{0,1\}^N, |x|=i} |x\rangle, \quad 0 \le i \le N.$$

We need only to construct a quantum circuit W that maps $|i\rangle \mapsto |E_i\rangle$. Then we can use the reverse of W to compress into $\mathcal{SYM}(\mathcal{H}^{\otimes N})$ and W itself to decompress. The circuit W would need ancilla states that are initialized as $|0\rangle$ and returned to the same state at the end of the computation.

We first apply the transformation

$$W_1 : \ |i\rangle \otimes |0\rangle^{\otimes N} \mapsto |i\rangle \otimes |E_i\rangle.$$

This can be done, for example, by implementing

$$|i\rangle \otimes |0\rangle^{\otimes N} \mapsto |i\rangle \otimes \left(\frac{1}{\sqrt{\binom{N}{i}}} \sum_{0\le j\le \binom{N}{i}-1} |j\rangle \right)$$

and then mapping j to the jth element in $\{x \in \{0,1\}^N : |x| = i\}$ under some standard ordering. It is easy to implement

$$W_2 : |y\rangle \otimes |E_i\rangle \mapsto |i + y \mod (N+1)\rangle \otimes |E_i\rangle.$$

Hence

$$W_2^\dagger W_1(|i\rangle \otimes |0\rangle^{\otimes N}) = |0\rangle^{\lceil \log_2(N+1)\rceil} \otimes |E_i\rangle.$$

Swapping the qubits completes the construction of W.

8.3 Quantum Network Multicast

We now consider a quantum communication network represented as a directed graph $G = (V, E)$. There are h information sources $S_1, \ldots, S_h$, each producing a qubit per unit time, and N receivers $\mathcal{R} = \{R_1, \ldots, R_N\}$. For each receiver, there are h edge disjoint paths to it, one from each of the h sources. For receiver j, we denote these paths as (S_i, R_j), $i = 1, \ldots, h$. The h information sources need to multicast h qubits simultaneously to all N receivers at rate h. It is easy to see that the capacity of any cut which separates the sources from the receivers should be at least $h \log(N+1)$. We will show that the multicast is possible if each edge of the network has the capacity of carrying $h \log(N+1)$ qubits.

Definition 8.2. We say that source S_i is *present* at edge $e \in E$ if there is a receiver for which the path (S_i, R_j) form source S_i passes through edge e.

Let $\mathcal{I} = \{1, \ldots, h\}$ denote the set used to index the h sources, and $\mathcal{I}(e) \subseteq \mathcal{I}$ denote the index set comprising the labels of sources present at edge $e \in E$. Let ζ denote the maximum number of sources present at an edge in the network:

$$\zeta \triangleq \max_{e \in E} |\mathcal{I}(e)| \leq h. \tag{8.4}$$

Lemma 8.1. Let $r(i; e)$ denote the number of all receivers $R_j \in \mathcal{R}$ for which the paths (S_i, R_j) form source S_i pass through edge e. We have

$$\sum_{i \in \mathcal{I}(e)} r(i; e) \leq N, \quad \forall e \in E. \tag{8.5}$$

Proof. The inequality (8.5) holds because there are N receivers and for each receiver, say R_j, the paths (S_i, R_j), $i \in \mathcal{I}$, are disjoint. □

Theorem 8.2. If in a quantum network with h sources of arbitrary qubits and N receivers the following three conditions hold:

(1) the number of edges in the min-cut between the sources and each receiver is h,
(2) each edge has the capacity to carry $h \log(N+1)$-qubit states,
(3) the nodes have the capability to losslessly compress and decompress quantum information,

then the sources can simultaneously multicast their qubits through the network to all receivers.

Proof. The first condition guarantees that, for each receiver, there are h edge disjoint paths to it, one from each of the h sources. If the quantum information is multicast from the sources to the receivers with no processing at the intermediate nodes (e.g., compression/decompression), then edge e carries the quantum state $\Phi(e)$:

$$\Phi(e) = \bigotimes_{i \in \mathcal{I}(e)} |\phi_i\rangle^{\otimes r(i;e)},$$

where the qubits $|\phi_i\rangle$ are prepared by source S_i and destined to $r(i;e)$ receivers through edge e. States $\Phi(e)$ belong to a subspace of dimension $\prod_{i \in \mathcal{I}(e)} (r(i;e)+1)$ which can be bounded as follows:

$$\prod_{i \in \mathcal{I}(e)} (r(i;e)+1) \leq \left[\frac{\sum_{i \in \mathcal{I}(e)} (r(i;e)+1)}{|\mathcal{I}(e)|} \right]^{|\mathcal{I}(e)|} \tag{8.6}$$

$$\leq (N+1)^{\zeta} \leq (N+1)^{h}, \tag{8.7}$$

where (8.6) follows from the geometric/arithmetic mean inequality, and (8.7) follows from (8.4) and (8.5). □

Note that, even if all sources share the same link to all the receivers (e.g., there are h sources in place of S in Figure 8.1), the capacity of

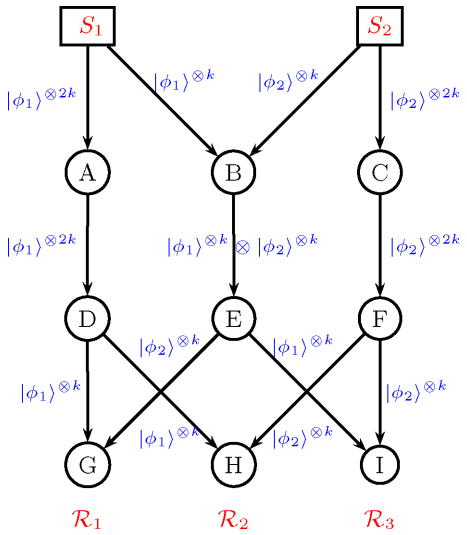


Fig. 8.2 Quantum network multicast from two sources S_1 and S_2 and three groups of receivers $\mathcal{R}_1 = \{R_1, \ldots, R_k\}$, $\mathcal{R}_2 = \{R_{k+1}, \ldots, R_{2k}\}$, and $\mathcal{R}_3 = \{R_{2k+1}, \ldots, R_{3k}\}$ located on the three nodes.

$h \log(N+1)$ is sufficient. This requirement is reduced when the qubits from the h sources are routed through the network in a way which (roughly speaking) favors more homogeneous edge flows. An example of qubit routing is shown for the network in Figure 8.2 with $3k$ receivers. Note that this is the only valid routing for this network. The maximum required edge-capacity is $2 \log k$ qubits on the edge BE.

Notes

A good introduction and reference material for quantum information and computation can be found in books by Nielsen and Chuang [63] and Gruska [31]. Quantum multicast was studied by Shi and Soljanin [77], who showed that lossless compression of special multicast quantum states is possible and significantly reduces the edge capacity requirements of the multicast. Two unicasts on the butterfly network of quantum channels were first considered by Hayashi et al. [32], who showed that two qubits can be sent simultaneously with keeping their fidelity strictly greater than 1/2 if the nodes of the network can use one of the

available algorithms for approximate cloning [7]. This approach was later extended to probabilistic cloning and general graphs by Iwama et al. [38]. General multiple unicasts in quantum networks in which exact or asymptotically exact reproduction of states is required was studied by Leung et al. [51]. They considered all four scenarios of classical communication assistance (none, forward, backward, and two-way) and bipartite networks, and showed that in all these cases, the optimal strategy is routing. More details on types and type classes can be found in [11].

9

Emerging Applications of Network Coding

Since its introduction, the field of network coding has been interdisciplinary, attracting interest from electrical engineers, computer scientists, and mathematicians. Its focus, however, has traditionally been on transmission scenarios where multiple users share the same network resources. These days, network coding, and in general, coding ideas are starting to be successfully used in other applications. In this section, we briefly survey some of the emerging applications of network coding, namely, network monitoring, operation of switches, on-chip communication, and distributed storage.

9.1 Network Monitoring

Distributed Internet applications often use overlay networks that enable them to detect and recover from failures or degraded performance of the underlying Internet infrastructure. To achieve this high-level goal, it is necessary for the nodes in the overlay to monitor, assess, and predict the behavior of Internet paths, and eventually make efficient use of them. Clearly, accurate monitoring at minimum overhead and complexity is of crucial importance for the operation of all networks.

Network characteristics such as topology, packet loss rate, delay, and link failures, are most often inferred based on end-to-end measurements by techniques commonly referred to as *network tomography* in analogy to medical tomography. Active measurement techniques have been proposed that send sequences of probe packets from a set of sources to a set of receivers, and infer link-level metrics of interest from the received packets. Some techniques send probes over unicast paths while others use multicast trees. To cover the entire network, a mesh of paths and/or trees is needed. The bandwidth efficiency of these methods can be measured by the number of probe packets required to estimate the metric of interest with a desired accuracy. It depends both on the choice of paths/trees over which the sequences of probes are sent, as well as on the number of probes in each sequence. There is a tradeoff between bandwidth efficiency and estimation accuracy; it is desirable to improve both while keeping computational complexity low.

In networks where nodes can perform network coding operations, we can exploit these capabilities to improve several aspects of network tomography, as we next show for two problems: link loss monitoring and topology inference.

9.1.1 Link Loss Monitoring

In this problem, a network is represented as a directed graph $G = (V, E)$ and packet losses at each link are modeled by a Bernoulli process. A packet traversing a link $e \in E$ is lost with probability p_e independently of losses at other links. There is a set $S \subseteq V$ of nodes that can act as sources of probe packets, a set $R \subseteq V$ of nodes that can act as receivers of probe packets, and a set of links $L \subseteq E$ being monitored. The goal is to estimate the packet loss probabilities on the links in L by sending probe packets from the nodes in S to the nodes in R. The following example motivates the use of network coding techniques for link loss monitoring.

Example 9.1. Consider the network shown in Figure 9.1 in which packet losses are modeled as described above. Nodes A and B send probes and nodes E and F receive them. We are interested in estimating

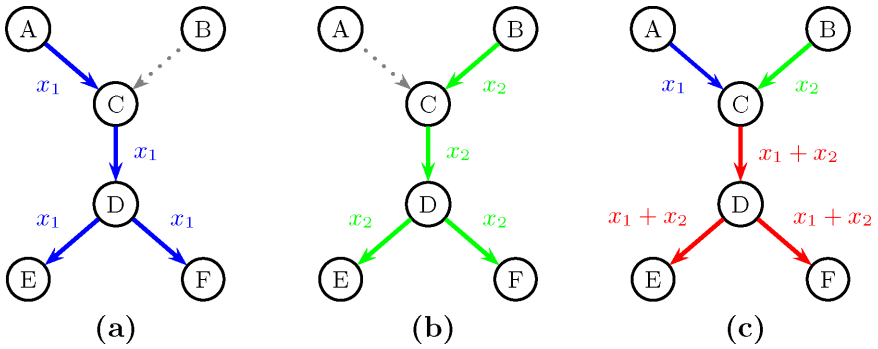


Fig. 9.1 Link loss monitoring example. Nodes A and B are sources, E and F are receivers, C can add (or *xor*) incoming packets, and D copies incoming packet to both outgoing links.

the loss probabilities in all links, namely, p_{AC}, p_{BC}, p_{CD}, p_{DE}, and p_{DF}.

The traditional multicast-based tomography uses two multicast trees rooted at nodes A and B and ending at E and F, as depicted in Figures 9.1(a) and 9.1(b). A number of identical experiments is made; at each experiment, source A sends packet x_1 and source B sends packet x_2. The receivers E and F infer the link loss rates by keeping track of how many times they receive packets x_1 and x_2. Note that, because of the overlap of the two trees, links CD, DE, and DF are used twice in each experiment, leading to inefficient bandwidth usage. Moreover, from this set of experiments, we cannot calculate p_{CD}, and thus edge CD is not *identifiable*. Indeed, by observing the experiments outcome, we cannot distinguish whether packet x_1 is dropped on edge AC or CD; similarly, we cannot distinguish whether packet x_2 is dropped on edge BC or CD.

Network coding techniques can be used to address this problem. The basic idea is the following. We assume that the intermediate nodes C and D can look at the content of the incoming packets and form packet(s) to forward to their outgoing link(s). Node A sends to node C a probe packet with payload that contains the binary string $x_1 = [1 \quad 0]$. Similarly, node B sends probe packet $x_2 = [0 \quad 1]$ to node C. If node C receives only x_1 or only x_2, then it just forwards the received packet to node D; if C receives both packets x_1 and x_2, then it creates a new

packet, with payload their linear combination $x_3 = [1 \quad 1]$, and forwards it to node D; more generally, $x_3 = x_1 \oplus x_2$, where $\oplus$ is the bit-wise XOR operation. Node D sends the incoming packet x_3 to both outgoing links DE and DF. All operations happen in one time slot, where a time slot refers to the time-window node C waits to receive packets x_1 and x_2; this time-window needs to be long enough to account for the difference in the delay that may occur on edges AC and BC.

In each time slot, probe packet x_1 is sent from node A and probe packet x_2 is sent from node B. This constitutes one experiment. Depending on which links packet losses take place, nodes E and F receive x_1, x_2, x_3, or no packet at all. Note that because of the topology of the network, E and F either receive identical packets or one of them receives no packet. The possible outcomes observed at nodes E and F are summarized in the left two columns of Table 9.1. The five right columns in the table show the combination of loss and success events in the links that lead to the observed outcome, where 0 symbolizes the lost packet event and 1 the success. For example, the outcome (x_1,x_1) occurs when events ($AC = 1$, $BC = 0$, $CD = 1$, $DE = 1$, $DF = 1$) take place, which happens with probability $(1 - p_{AC})p_{BC}(1 - p_{CD})(1 - p_{DE})(1 - p_{DF})$. Similarly, we can write the probability of each of the 10 observed events as a function of the link loss probabilities. Note that each of the first nine outcomes uniquely determines the sequence of events that took place on the network links.

Table 9.1 Possible observed probes at nodes E and F, together with the combination of loss (0) and success (1) indicators in the five links corresponding to the observed outcome.

Packets received	Link transmission indicator				
at (E,F)	AC	BC	CD	DE	DF
(x_1,x_1)	1	0	1	1	1
(x_2,x_2)	0	1	1	1	1
(x_3,x_3)	1	1	1	1	1
$(x_1,-)$	1	0	1	1	0
$(x_2,-)$	0	1	1	1	0
$(x_3,-)$	1	1	1	1	0
$(-,x_1)$	1	0	1	0	1
$(-,x_2)$	0	1	1	0	1
$(-,x_3)$	1	1	1	0	1
$(-,-)$	Multiple possible causes				

Repeating the experiment a number of times, we can observe how many times each event in Table 9.1 occurs. We can then use standard Maximum Likelihood (ML) estimation to infer the underlying link loss rates, i.e., estimate p_{AC}, p_{BC}, p_{CD}, p_{DE}, p_{DF}. The ML estimator identifies the link-loss rates that would, with higher probability, result in obtaining our particular set of data. Clearly, the associated estimation accuracy increases with the number of experiments performed.

In general, work in the literature has shown that for the link loss monitoring problem, use of network coding promises the following benefits:

(1) increase the number of (identifiable) links whose loss probability can be inferred from end-to-end measurements;
(2) eliminate the overlap between paths and/or trees needed to cover the entire network, i.e., ensure that a single probe packet traverses each link of the network during each experiment;
(3) use less probes (smallest number of experiments) to achieve a certain estimation accuracy, by intelligently using not only the number, but also the content of received probes;
(4) improve the estimation accuracy when allowed to select which nodes of the network act as sources and which nodes act as receivers;
(5) reduce the complexity of selecting probe packet paths for minimum cost monitoring[1]; and
(6) offer a graceful link-loss monitoring approach over networks that are not trees.

9.1.2 Topology Inference

In networks implementing network coding, the observations at the receivers depend on the information produced by the sources in a way

[1] The problem of selecting multicast trees to identify a set of link-loss rates, while minimizing the network bandwidth usage, is $\mathcal{NP}$-hard; using network coding offers a polynomial time solution for the same problem.

determined by the network code and the network topology.[2] If the internal nodes are instructed to perform XOR operations on their inputs, we can develop algorithms to discover the network topology *deterministically* and without any further active participation by the internal nodes. Consider again the network in Figure 9.1. Assume that links are lossless and that we have access to the leaf nodes A, B, E, and F but do not know the topology of the network. If probes $x_1 = [1 \quad 0]$ and $x_2 = [0 \quad 1]$ are sent from nodes A and B, and nodes E and F receive $x_3 = [1 \quad 1]$, then, based on this observation, we can conclude that there was at least one intermediate node that performed XOR on its inputs coming from A and B.

We next show how we can use this approach to infer the topology of undirected trees where each edge can be used in either direction and the connection between any two vertices is reciprocal. We will restrict our discussion to trees where internal nodes have degree exactly equal to three. Denote by $\mathcal{L} = \{1, 2, \ldots, L\}$ the leaf-vertices (leaves) of the tree. They correspond to end-hosts that can act as sources or receivers of probe packets. The basic idea is to devise a series of experiments such that each successive experiment allows us to further reveal the inner structure of the network. Thus the first experiment partitions the network into subgraphs with unknown structure and reveals network edges and vertices that connect these subgraphs. Each successive experiment further partitions one of the remaining subgraphs with unknown structure in the same manner. We illustrate this approach for a particular network in Example 9.2. The extension to a general algorithm for arbitrary trees with internal nodes of degree three is straightforward. The same ideas can also be easily extended to arbitrary degree trees if we allow packet combining over a larger than the binary field.

Example 9.2. Consider the network in Figure 9.2. Assume that we only know the set of leaves $\{1, 2, 3, 4, 5, 6, 7\}$ and that each node in the tree network has degree 3. In the first experiment, node 1 acts as a

[2] Without network coding, the observations at the receivers are simply equal to the information generated at the sources.

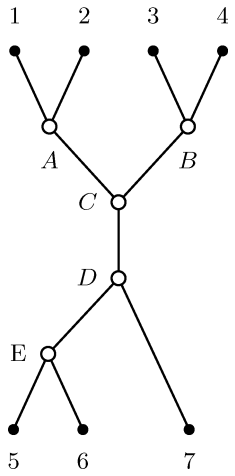


Fig. 9.2 An undirected tree network topology with 7 leaves and 5 intermediate nodes.

source of the probe packet $x_1 = [1 \quad 0]$, node 7 acts as a source of the probe packets $x_2 = [0 \quad 1]$, and the rest of the nodes act as receivers of the probe packets. If an internal node receives a single packet, it sends a copy of the packet to each of its neighbors. If an internal node receives two packets within a predetermined time period, it XORs the packets and sends a copy of the result to each of its neighbors. Thus, node A receives packet x_1, and sends a copy of it to both leaf 2 and to node C. Similarly, node D receives packet x_2, and sends a copy of it to E which in turn sends a copy to both leaf 5 and leaf 6. Probe packets x_1 and x_2 arrive (within a predetermined time window) to node C. Node C creates the packet $x_3 = x_1 \oplus x_2 = [1 \quad 1]$ and forwards x_3 to node B which in turn sends a copy to both leaf 3 and leaf 4.[3]

Our knowledge of the network prior to the above experiment is summarized in Figure 9.3(a). As a result of the experiment, the tree will be divided into three areas, $\mathcal{L}_1$ containing S_1 and the leaves that received probe packet x_1 (in total, $\mathcal{L}_1 = \{1,2\}$). Similarly $\mathcal{L}_2 = \{5,6,7\}$ are nodes containing S_2 and the leaves that received probe packet x_2

[3] Note that we have chosen the directionality of the edges depending on which source reaches the vertex first. If there is variable delay, then the vertex where the packets x_1, x_2 meet could be different, but this does not affect the algorithm.

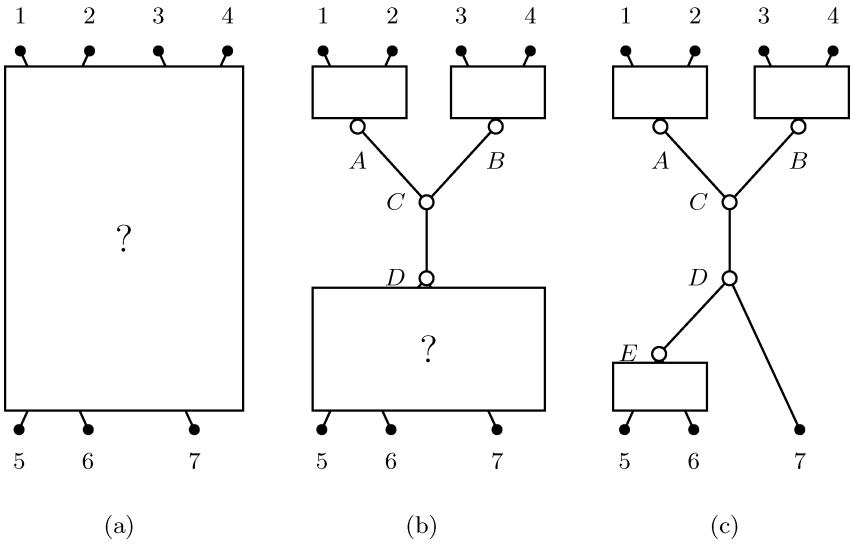


Fig. 9.3 Structure revealed after two iterations.

and $\mathcal{L}_3 = \{3,4\}$ containing the leaves that received probe packets $x_1 \oplus x_2$. From this information, observed at the edge of the network, we can deduce that the network has the structure depicted in Figure 9.3(b).

To infer the structure that connects leaves $\{5,6,7\}$ to node C, we need to perform a second experiment, where we now randomly choose two of these three leaves to act as sources of probe packets. For example, assume that nodes 5 and 6 act as sources S_1 and S_2 of probe packets. Note that any probe packet leaving node D will be multicasted to all the remaining leaves of the network, i.e., nodes $\{1,2,3,4\}$ will observe the same packet. Thus in this sense we can think of node D as a single "aggregate-receiver" for this second experiment, that will observe the common packet received at nodes $\{1,2,3,4\}$. Following the same procedure as before, assuming that packets x_1 and x_2 meet at node E, receivers 7 and $\{1,2,3,4\}$ receive packet $x_3 = x_1 \oplus x_2$. Using this additional information we refine the inferred network structure as depicted in Figure 9.3(c). Since each node in the network has degree three, we can deduce from Figure 9.3(c) the precise topology shown in Figure 9.2.

9.2 Operations of Switches

A switch is a network device with k inputs and n outputs (or ports), that connects inputs to outputs under the constraint that each output is connected to at most one input at a time. Moreover, an input cannot send different information to different outputs at the same time.

Arriving data packets are stored in the input queues of the switch, and register a request to be forwarded to a specific subset of the output ports. To maximize the speed of operation, the switch uses a scheduling algorithm to serve as many packets as possible in parallel.

In practice, the switch serves traffic flows, rather than individual packets. The packets in each flow have a common source and destination set and an associated arrival rate. To accommodate the multiple traffic flows, a technique typically used is *speedup*, which refers to that the switch operates at a faster clock than the incoming and outgoing network links.

Example 9.3. Consider a switch with two inputs and three outputs as shown in Figure 9.4. Assume we want to accomodate four rate 1/2 data flows: at input 1, we have flow A destined to outputs $\{1,2\}$ and flow B to output $\{1\}$, and at input 2, we have flow C destined to outputs $\{2,3\}$ and flow D to output $\{3\}$. This traffic pattern can be accommodated using a speedup factor of two, where the switch operates twice as fast

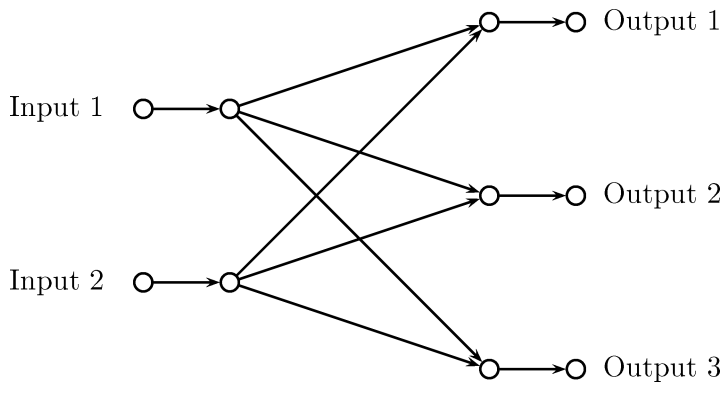


Fig. 9.4 A switch with two inputs and three outputs.

as the network. In the first time-slot, flows A and D can be served, while in the second time-slot flows B and C.

We say that a switch has network coding capabilities, if it is allowed to send linear combinations of the packets waiting in the input queues. For switches that serve multicast patterns, use of network coding allows to simultaneously accommodate a larger number of traffic patterns, and increase the achievable rate region of the switch, as the following example illustrates.

Example 9.4. Consider again the switch in Figure 9.4, but now assume the following four data flows: at input 1, we have one flow A of rate $2/3$ destined to outputs $\{1,2,3\}$; at input 2, we have three rate $1/3$ unicast flows, flow B destined to output $\{1\}$, flow C to output $\{2\}$, and flow D to output $\{3\}$. Note that the multicast flow A from input 1 requests twice the rate of the unicast flows; i.e., requests to send two packets, say $\{P_1, P_2\}$ to the three outputs, for each one packet the unicast flows send. However, each unicast flow from input 2 blocks one of the outputs at each time slot, and thus the multicast packet can be send at best to the other two outputs. This implies that it will take at least two time slots to complete sending each multicast packet, and hence, rate more than $1/2$ is not achievable. Table 9.2 shows a network coding scheme that allows to serve this traffic pattern using simple binary operations. Again, each unicast flow blocks one output at each time slot, but now input 1 employs coding across the packets $\{P_1, P_2\}$ to ensure that all outputs receive useful (innovative) information during each time slot.

To design schedules for multicast switches, we can use graphs that capture the contention among different flow requests. Such graphs are

Table 9.2 Coding scheme applied to packets $\{P_1, P_2\}$ of flow A at input 1 of the switch.

Time slot	Code	Outputs
1	P_1	$\{1,2\}$
2	P_2	$\{2,3\}$
3	$P_1 + P_2$	$\{1,3\}$

known as *conflict graphs*, and can be created as follows. We first split each multicast flow destined to n_1 outputs into n_1 unicast subflows, each destined to a single output. We then form the conflict graph as an undirected graph whose vertices correspond to the subflows and whose edges connect the vertices corresponding to contending subflows. More precisely, a conflict graph has

(1) a vertex for each subflow,
(2) an edge between each pair of vertices whose corresponding subflows arrive at the same input but belong to different flows, and
(3) an edge between each pair of vertices whose corresponding subflows request the same output.

In addition, with each vertex we associate a weight equal to the rate of the flow to which the subflow corresponding to the vertex belongs.

Figure 9.5 shows the conflict graph for the traffic pattern requests in Example 9.4. Note that the multicast flow A is split in three unicast flows A_1, A_2, and A_3. Edges BC, CD, and DB are of type (2), and edges A_1B, A_2C, and A_3D are of type (3).

Clearly, the set of subflows that can be served simultaneously at a given time slot corresponds to an independent set, i.e., a set of vertices no two of which are connected by an edge. To accommodate the different rates requests, we can use different independent sets at different

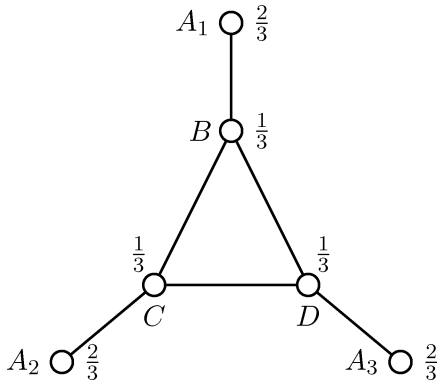


Fig. 9.5 Conflict graph corresponding to Example 9.4.

time-slots. In particular, assume that the conflict graph has m vertices (subflows), and let ϕ be the $m \times 1$ vector collecting the rates a particular schedule allocates to each subflow. If χ_s denotes the incidence vector for an independent set s, and during our schedule we use χ_s for ϕ_s time slots, the rates we will be serving for each subflow can be calculated as

$$\phi = \sum_s \phi_s \chi_s.$$

The achievable rate region corresponds to all possible vectors ϕ. Network coding is used to ensure that the transmitted packets on the subflows are innovative for all outputs. The following example illustrates this approach.

Example 9.5. Consider the traffic pattern requests in Example 9.4 and the corresponding conflict graph in Figure 9.5. We have six subflows, three corresponding to the unicast flows B, C and D, and three A_1, A_2, and A_3 created by splitting the multicast flow A according to the outputs. The solution in Table 9.2 corresponds to using three independent sets of the conflict graph, $s_1 = \{A_1, A_2, D\}$, $s_2 = \{A_2, A_3, B\}$, and $s_3 = \{A_1, A_3, C\}$, each a fraction 1/3 of time. Coding ensures that the subflows A_1, A_2, and A_3, whenever active, bring innovative information to the corresponding outputs.

9.3 On-Chips Communication

Recent work has started investigating potential benefits of applying network coding techniques to VLSI chip design. The design of VLSI chips aims to simplify and minimize the length of on-chip wiring. Network coding can help reach this goal, at the cost of additional coding/decoding logic in the network. This overhead indicates that scenarios where network coding is beneficial may occur in the routing of multicast signals along long routes (buses). The butterfly network (embedded on a chip) can be used as an illustrative example, where, to transmit at rate two to both receivers, we can either perform coding operations at one of the network nodes, or alternatively, use two

parallel edges to replace the edge where the contention for network resources occurs. Experimental results indicate potential benefits in terms of interconnect power, wiring length, and area.

9.4 Distributed Storage

Distributed storage systems store pieces of information entities (e.g., pieces of data files or sensor measurements) throughout the network, usually with the purpose to ensure sufficient storage capacity by using resources of multiple network nodes (e.g., disks at several computers or memory of wireless sensors). The information should be stored reliably over long time periods, i.e., with enough redundancy so that it could be recovered even if several of the computers or sensors were unavailable. Simple replication, a straightforward approach to storing data redundantly, is not very efficient in terms of capacity required to store the redundant information and transmission resources required to collect the distributed pieces of information.

The most efficient way to store a data entity redundantly is by the means of MDS codes (e.g., Reed Solomon), so that different network nodes store different coded symbols. The challenging questions arise when multiple data entities originating at different places in the network are to be simultaneously stored throughout the network. Naturally, it is in such scenarios that network coding ideas are directly applicable and beneficial.

Another line of distributed problems deals with large-scale wireless sensor networks consisting of small devices (nodes) with limited resources (e.g., CPU power, bandwidth, and memory). Because of that, data acquired by sensor nodes have short lifetime, and any processing of such data within the network should have low complexity and power consumption. The goal here is to disseminate data acquired by a small number K of sensor nodes (sources) distributed throughout the network, redundantly to all nodes in the network so that at the end of this process, the K originally acquired pieces of information can be recovered from any collection of nodes of a size that is not much larger than K, with low computational complexity. We already discussed such applications in Section 4. The main advantages of such data

dissemination are prolonged lifetime, increased spatial availability, as well as computationally and spatially easy access to the acquired data. The problem here, which also can be addressed by network coding, is to devise efficient algorithms for the acquired data dissemination that upon completion guarantees that each node stores a coded symbol of an MDS code.

Notes

Network coding for active network monitoring was proposed by Fragouli and Markopoulou [22, 23]. Passive inference of link-loss rates has also been proposed by Ho et al. [34]. More recently, passive inference of topological properties in networks that employ randomized network coding was proposed by Jafarisiavoshani et al. [39]. Network coding for multicast switches was proposed by Sundararajan et al. [45, 83, 84]. The presented examples and achievable rate region description come from [84]. Using network coding to improve on-chip communication was proposed by Jayakumar et al. [42]. A decentralized implementation of Fountain codes for distributed data storage that uses geographic routing was proposed by Dimakis et al. [15, 16]. Another approach that uses random walks with traps to disseminate the source packets throughout the network was proposed by Lin et al. [52]. A technique called growth codes to increase data persistence in wireless sensor networks by increasing the amount of information that can be recovered at the sink was introduced by Kamara et al. [43].

Acknowledgments

This work was in part supported by NSF under award No. CCR-0325673, FNS under award No. 200021-103836/1, the NetRefound FP6 FET Open Project of the European Union under grant number IST-034413 and DIMACS under the auspices of Special focus on Computational Information Theory and Coding. We would also like to acknowledge useful comments from Dr Eleni Drinea.

References

[1] M. Adler, N. J. A. Harvey, K. Jain, R. Kleinberg, and A. R. Lehman, "On the capacity of information networks," *SODA*, pp. 241–250, 2006.

[2] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, vol. 36, no. 4, no. 4, 2004.

[3] S. Avestimehr, S. N. Diggavi, and D. N. Tse, "A deterministic model for wireless relay networks and its capacity," *Information Theory Workshop*, September 2007.

[4] S. Avestimehr, S. N. Diggavi, and D. N. Tse, "Wireless network information flow," *Allerton*, September 2007.

[5] P. Backx, T. Wauters, B. Dhoedt, and P. Demester, "A comparison of peer-to-peer architectures," in *Eurescom Summit 2002*, 2002.

[6] K. Bhattad and K. R. Narayanan, "Weakly Secure Network Coding," in *Proceedings of First Workshop on Network Coding, Theory, and Applications (NetCod'05)*, April 2005.

[7] V. Buzek and M. Hillery, "Quantum copying: Beyond the no-cloning theorem," *Physics Review A*, vol. 54, pp. 1844–1853, 1996.

[8] N. Cai and R. W. Yeung, "Network error correction, II: Lower bounds," *Communication and Information Systems*, vol. 6, pp. 37–54, 2006.

[9] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *ICSI Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[10] B. Cohen, "http://www.bittorrent.com,".

[11] I. Csiszár and J. Körner, *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Budapest, Hungary: Académiai Kiadó, 1986.

[12] S. Deb, M. Médard, and C. Choute, "Algebraic gossip: A network coding approach to optimal multiple rumor mongering," *IEEE/ACM Transactions on Networking*, vol. 14, pp. 2486–2507, June 2006.
[13] A. G. Dimakis, P. B. Godfrey, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *Infocom*, 2007.
[14] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes," in *Symposium on Information Processing in Sensor Networks (IPSN '05)*, April 2005.
[15] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Distributed fountain codes for networked storage," *Acoustics, Speech and Signal Processing, ICASSP 2006*, May 2006.
[16] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE/ACM Transactions on Networking (TON)*, June 2006.
[17] R. Dougherty, C. Freiling, and K. Zeger, "Unachievability of network coding capacity," *IEEE Transactions on Information Theory and IEEE/ACM Transactions on Networking*, vol. 52, pp. 2365–2372, June 2006.
[18] R. Dougherty and K. Zeger, "Nonreversibility and equivalent constructions of multiple-unicast networks," *IEEE Transactions on Information Theory*, vol. 52, pp. 5067–5077, November 2006.
[19] A. Erylmaz, A. Ozdaglar, and M. Médard, "On delay performance gains from network coding," *CISS*, 2006.
[20] J. Feldman, T. Malkin, C. Stein, and R. A. Servedio, "On the Capacity of Secure Network Coding," in *Proceedings of 42nd Annual Allerton Conference on Communication, Control, and Computing*, September 2004.
[21] W. Feller, *An Introduction to Probability Theory and Its Applications*. Vol. 1, Wiley, Third Edition, 1968.
[22] C. Fragouli and A. Markopoulou, "A network coding approach to network monitoring," *Allerton*, 2005.
[23] C. Fragouli, A. Markopoulou, and S. Diggavi, "Active topology inference using network coding," *Allerton*, 2006.
[24] C. Fragouli and E. Soljanin, "Network coding fundamentals," *Foundation and Trends in Networking*, vol. 2, no. 1, pp. 1–133, 2007.
[25] C. Fragouli, J. Widmer, and J.-Y. L. Boudec, "A network coding approach to energy efficient broadcasting: From theory to practice," in *IEEE Infocom*, Barcelona, Spain, April 2006.
[26] C. Fragouli, J. Widmer, and J.-Y. L. Boudec, "On the benefits of network coding for wireless applications," in *Network Coding Workshop*, Boston, 2006.
[27] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *Proceedings of Infocom*, IEEE, 2005.
[28] C. Gkantsidis, J. Miller, and P. Rodriguez, "Comprehensive view of a live network coding P2P system," in *IMC '06: Proceedings of the 6th ACM SIGCOMM on Internet Measurement*, pp. 177–188, New York, NY, USA: ACM Press, 2006.
[29] Gnutella, "http://gnutella.wego.com," 2000.

[30] R. Gowaikar, A. F. Dana, R. Palanki, B. Hassibi, and M. Effros, "On the capacity of wireless erasure networks," *in Proceedings of the IEEE International Symposium on Information Theory*, p. 401, 2004.

[31] J. Gruska, *Quantum Computing*. Mcgraw Hill, 2000.

[32] M. Hayashi, K. Iwama, H. Nishimura, R. Raymond, and S. Yamashita, "Quantum Network Coding," Available: http://arxiv.org/abs/quant-ph/0601088 [online], 2006.

[33] T. Ho, Y. Chang, and K. J. Han, "On constructive network coding for multiple unicasts," *44th Allerton Conference on Communication, Control and Computing*, 2006.

[34] T. Ho, B. Leong, Y. Chang, Y. Wen, and R. Koetter, "Network monitoring in multicast networks using network coding," in *International Symposium on Information Theory (ISIT)*, June 2005.

[35] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. Karger, "Byzantine Modification Detection in Multicast Networks using Randomized Network Coding," in *Proceedings of 2004 IEEE International Symposium on Information Theory (ISIT'04)*, June 2004.

[36] T. Ho, M. Médard, and R. Koetter, "An information-theoretic view of network management," *IEEE Transactions on Information Theory*, vol. 51, pp. 1295–1312, April 2005.

[37] M. Hofmann and L. Beaumont, *Content Networking: Architecture, Protocols, and Practice*. Morgan Kaufmann, 2004.

[38] K. Iwama, H. Nishimura, R. Raymond, and S. Yamashita, "Quantum network coding for general graphs," Available: http://arxiv.org/abs/ quant-ph/0611039 [online], 2006.

[39] M. Jafarisiavoshani, C. Fragouli, and S. Diggavi, "On subspace properties for randomized network coding," in *Information Theory Workshop (ITW)*, July 2007.

[40] M. Jafarisiavoshani, C. Fragouli, S. Diggavi, and C. Gkantsidis, "Bottleneck discovery and overlay management in network coded peer-to-peer systems," *ACM SigComm INM'07*, August 2007.

[41] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard, "Resilient network coding in the presence of Byzantine adversaries," in *INFOCOM*, pp. 616–624, 2007.

[42] N. Jayakumar, K. Gulati, and S. K. A. Sprintson, "Network coding for routability improvement in VLSI," *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2006.

[43] A. Kamra and V. Misra, "Growth Codes: Maximizing Sensor Network Data Persistence," in *Sigcomm06*, Pisa, Italy, 2006.

[44] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," *ACM SIGCOMM*, September 2006.

[45] M. Kim, J. Sundararajan, and M. Médard, "Network coding for speedup in switches," in *ISIT*, June 2007.

[46] R. Koetter and F. Kschischang, "Coding for errors and rrasures in random network coding," *ISIT*, June 2007.

[47] G. Kramer and S. A. Savari, "Edge-cut bounds on network coding rates," *Journal of Network and Systems Management*, vol. 14, 2006.

[48] M. N. Krohn, M. J. Freedman, and D. Mazières, "On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2004.

[49] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, and B. Zhao, "OceanStore: An architecture for global-scale persistent storage," in *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 190–201, ACM Press, 2000.

[50] A. R. Lehman and E. Lehman, "Network coding: Does the model need tuning?," *SODA*, pp. 499–504, 2005.

[51] D. Leung, J. Oppenheim, and A. Winter, "Quantum network communication — the butterfly and beyond," Available: http://arxiv.org/abs/ quant-ph/0608223 [online], 2006.

[52] Y. Lin, B. Liang, and B. Li, "Data persistence in large-scale sensor networks with decentralized fountain codes," in *Proceedings of the 26th IEEE INFOCOM 2007*, Anchorage, Alaska, May 6–12 2007.

[53] M. Luby, "LT Codes," *IEEE Symposium on the Foundations of Computer Science (STOC)*, pp. 271–280, 2002.

[54] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Practical loss-resilient codes," *ACM Symposium on Theory of Computing*, pp. 150–159, 1997.

[55] D. Lun, M. Médard, and M. Effros, "On coding for reliable communication over packet networks," *Allerton Conference on Communication, Control, and Computing*, September–October 2004.

[56] D. S. Lun, "Efficient operation of coded packet networks," PhD thesis, Massachusetts Institute of Technology, June 2006.

[57] D. S. Lun, P. Pakzad, C. Fragouli, M. Medard, and R. Koetter, "An analysis of finite-memory random linear coding on packet streams," *WiOpt '06*, April 2006.

[58] D. S. Lun, N. Ratnakar, R. Koetter, M. Medard, E. Ahmed, and H. Lee, "Achieving minimum-cost multicast: A decentralized approach based on network coding," in *Proceedings of IEEE Infocom*, March 2005.

[59] P. Maymounkov and D. Mazieres, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *International Workshop on Peer-to-Peer Systems (IPTPS'02)*, 2002.

[60] D. Mosk-Aoyamam and D. Shah, "Information dissemination via network coding," *ISIT*, pp. 1748–1752, 2006.

[61] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[62] Napster, "http://www.napster.com," 1999.

[63] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[64] P. Pakzad, C. Fragouli, and A. Shokrollahi, "Coding schemes for line networks," *ISIT*, pp. 1853–1857, September 2005.

[65] D. Petrović, K. Ramchandran, and J. Rabaey, "Overcoming untuned radios in wireless networks with network coding," *IEEE/ACM Transactions on Networking*, vol. 14, pp. 2649–2657, June 2006.

[66] A. Ramamoorthy, J. Shi, and R. D. Wesel, "On the capacity of network coding for random networks," *IEEE Transactions on Information Theory*, vol. 51, pp. 2878–2885, August 2005.

[67] N. Ratnakar and G. Kramer, "The multicast capacity of acyclic, deterministic, relay networks with no interference," *Network Coding Workshop*, April 2005.

[68] N. Ratnakar and G. Kramer, "On the separation of channel and network coding in Aref networks," *ISIT*, pp. 1716–1719, September 2005.

[69] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker, "A scalable content-addressable network," in *SIGCOMM*, pp. 161–172, ACM Press, 2001.

[70] S. Riis, "Reversible and irreversible information networks," submitted.

[71] S. Riis, "Linear versus non-linear Boolean functions in network flow," *CISS*, 2004.

[72] S. E. Rouayheb and E. Soljanin, "On wiretap networks II," in *Proceedings of 2007 International Symposium on Information Theory, (ISIT'07)*, Nice, France, June 2007.

[73] S. E. Rouayheb, A. Sprintson, and C. N. Georghiades, "Simple network codes for instantaneous recovery from edge failures in unicast connections," in *Proceedings of 2006 Workshop on Information Theory and its Applications, (ITA'06)*, San Diego, CA, USA, February 2006.

[74] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp. 329–350, Heidelberg, Germany: Springer, 2001.

[75] Y. E. Sagduyu and A. Ephremides, "Joint scheduling and wireless network coding," *Network Coding Workshop*, 2005.

[76] Schrijver, *Combinatorial Optimization*. Springer, 2003.

[77] Y. Shi and E. Soljanin, "On Multicast in Quantum Networks," in *Proceedings of 40th Annual Conference on Information Sciences and Systems (CISS'06)*, March 2006.

[78] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, pp. 2551–2567, 2006.

[79] D. Silva and F. R. Kschischang, "Using rank-metric codes for error correction in random network coding," *ISIT*, June 2007.

[80] S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. van Steen, "Replication for web hosting systems," *ACM Computing Surveys*, vol. 36, no. 3, no. 3, pp. 291–334, 2004.

[81] L. Song, N. Cai, and R. W. Yeung, "A separation theorem for single source network coding," *IEEE Transactions on Information Theory*, vol. 52, pp. 1861–1871, May 2006.

[82] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 ACM Sigcomm Conference*, pp. 149–160, ACM Press, 2001.

[83] J. Sundararajan, S. Deb, and M. Médard, "Extending the Birkhoff-von Neumann switching strategy to multicast switches," in *Proceedings of the IFIP Networking 2005 Conference*, May 2005.

[84] J. Sundararajan, M. Médard, M. Kim, A. Eryilmaz, D. Shah, and R. Koetter, "Network coding in a multicast switch," *Infocom*, March 2007.

[85] D. Tuninetti and C. Fragouli, "On the throughput improvement due to limited complexity processing at relay nodes," *ISIT*, pp. 1081–1085, September 2005.

[86] Y. Wu, P. A. Chou, and S.-Y. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," *IEEE Transactions on Communications*, vol. 53, no. 11, pp. 1906–1918, November 2005.

[87] S. Yang, C. K. Ngai, and R. W. Yeung, "Construction of linear network codes that achieve a refined Singleton bound," *ISIT*, June 2007.

[88] S. Yang and R. W. Yeung, "Characterizations of network error correction/detection and erasure correction," *Network Coding Workshop*, January 2007.

[89] R. W. Yeung and N. Cai, "Secure Network Coding," in *Proceedings of 2002 IEEE International Symposium on Information Theory (ISIT'02)*, June 2004.

[90] R. W. Yeung and N. Cai, "Network error correction, I: Basic concepts and upper bounds," *Communication and Inforamtion Systems*, vol. 6, pp. 19–35, 2006.

[91] S. Zhang, S. Liew, and P. Lam, "Physical layer network coding," *ACM MobiCom 2006*, pp. 24–29, September 2006.

[92] Z. Zhang, "Network error correction coding in packetized networks," *ITW*, October 2006.

[93] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, no. 1, pp. 41–53, 2004.