TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Nachrichtentechnik

Network Coding for the Multiple Access Layer

Danail Traskov

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor–Ingenieurs

genehmigten Dissertation.

Die Dissertation wurde am 22.06.2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 11.10.2010 angenommen.

Preface

The research leading to this thesis has been carried out over the last four years. During this time, I was initially with the University of Illinois at Urbana-Champaign, later at the Technical University Munich, working with my adviser, the late Prof. Ralf Koetter, and periodically visiting with Prof. Muriel Médard at the Massachusetts Institute of Technology. Ralf was the person who introduced me to the skill and art of research and who supported me with great generosity until his untimely passing in February 2009. Muriel has been an immensely helpful mentor throughout. The meetings and discussions with her greatly shaped my thinking about research and have influenced all of the contributions of this thesis. Her abilities and optimism are unbounded and have been a constant inspiration. I sincerely wish to thank Muriel and Ralf, for without them I would not be where I am.

I wish to thank Prof. Michelle Effros for serving on my committee and for providing valuable comments on my work and thesis, and to Prof. Norbert Hanik for serving on my committee and for being very supportive during the last difficult year at our institute. I got to know Prof. Desmond Lun, at that time a graduate student, during my first visit at MIT, and my work has been greatly influenced by his research. I am particularly grateful to Desmond for being an inspiring collaborator and a good friend. I would also like to thank Najak Ratnakar for contributing to various parts of my work and for being always a delightful colleague.

Over the course of my PhD, I have been fortunate to work in inspiring academic environments surrounded by colleagues with whom it has been a pleasure to work and, on occasion, enjoy the time off work. I am grateful to my colleagues in Munich and in Cambridge for the great atmosphere they have created. Looking at the large number of co-authors in my publications, I wish to thank the numerous collaborators who at some point have contributed to my work. Finally, I am most indebted to my family, my father Adrian, my mother Darina, and my brother Boris, for the unconditional love and support in all situations and at all stages. Without them this work would not have been possible.

München, October 2010

Danail Traskov

This research has been supported by the Defense Advanced Research Projects Agency (DARPA) under subcontract #18870740-37362-C issued by Stanford University. It was also supported by the United States Department of the Navy's Space and Naval Warfare Systems Command (SPAWAR) under Contract No. N66001-06-C-2020 through BAE Systems (DARPA CBMANET) and by the European Commission in the framework of the FP7 Network of Excellence in Wireless COMmunications NEWCOM++ (Contract no. 216715).

Contents

1 Introduction **1**

1.1 A Brief Survey of Relevant Work 4

1.2 Outline of the Thesis 6

1.3 Publications Preceding this Thesis 8

2 Preliminaries **9**

2.1 Random Linear Network Coding 9

2.2 Network Coding Subgraph 11

2.3 Interference and Half-Duplex Constraints 12

2.4 Example: Relay channel 15

2.4.1 Scheduling and Routing 16

2.4.2 Scheduling and Network Coding 17

2.4.3 Random Access and Routing 19

2.4.4 Random Access and Network Coding 19

2.4.5 Discussion 20

3 Scheduling for Network Coded Multicast **25**

3.1 The Multicast Rate Region with Scheduling Constraints 27

3.2 On the Complexity of the Scheduling Problem 33

3.3 Decentralized Algorithm 36

3.3.1 Subgradient Optimization on the Dual 37

3.3.2 Solving Subproblems 1 and 2 40

3.4 Simulation Results 41

3.5 Competition for Resources under Interference Conditions 46

3.6 Discussion 48

4 Asynchronous Network Coded Multicast **51**

4.1 Network Model and Optimization Problem 53

4.2 Decentralized Asynchronous Algorithm 55

4.2.1 Block-Coordinate Ascent on the Dual 55

4.2.2 Computing the Gradient 57

4.2.3 Convergence 59

4.3 Performance Evaluation 60

4.4 Discussion 63

5 Joint Scheduling and Instantaneously Decodable Network Coding **65**

5.1 Network Model 68

5.1.1 General Model and Assumptions 68

5.1.2 Queuing Model 70

5.1.3 Network Coding 73

5.1.4 Joint Scheduling and Network Coding 73

5.2 Stability Region 76

5.3 Online Algorithm 77

5.4 Performance Evaluation 78

5.4.1 Polytope Volume Computation 78

5.4.2 Stability Region 79

5.4.3 Online Algorithm 79

5.5 Discussion 80

6 Delay Control in Network Coded Broadcasting **83**

6.1 Network Model and Definitions 85

6.2 Optimization Framework 86

6.3 Algorithms for Solving (6.2) 88

6.3.1 Exhaustive Search 90

6.3.2 Fast Search 93

6.4 Performance Evaluation 93

6.5 Discussion 96

7 Conclusion **97**

7.1 Implementation of Network Coding: Case Studies 99

7.1.1 Case 1: Wireless Video Transmission using Network Coding 99

7.1.2 Case 2: Network Coding for Heterogeneous Networks 100

7.1.3 Case 3: Network Coding for Relaying in LTE-A 101

A Appendix 103

A.1 Analytical Solution of the Linear Program (2.7)-(2.14) 103

A.2 Distributed Maximal Stable Set Algorithm 106

A.3 The Asynchronous Algorithm with Polymatroidal Constraints 108

B Abbreviations 111

Bibliography 113

List of Figures

1.1 Two wireless nodes wish to exchange a pair of packets; the respective radio ranges are indicated by dotted lines. 3

1.2 Network coding versus routing. 4

2.1 An example of a hypergraph. Here, the node set is $\mathcal{N} = \{1, 2, 3, 4\}$ and the hyperarc set is $\mathcal{A} = \{(1, 2), (1, 3), (1, \{2, 3\}), (2, 4), (3, 4)\}$. 14

2.2 The conflict graph corresponding to the hypergraph in Fig. 2.1 15

2.3 The relay network. 16

2.4 The minimum average number of transmissions per packet as a function of the rate R. 22

2.5 Optimal transmission attempt probabilities for network coding as a function of the rate R. 23

3.1 The rate region for the hyperarc $(1, \{2, 3, 4\})$ 30

3.2 A line network with two-hop overhearing. 37

3.3 The conflict graph of the network in Fig. 3.2. 37

3.4 Throughput and power consumption for different scheduling techniques. 42

3.5 The number of configurations which appear in the solution of the optimal scheduling algorithm. We consider schedules with a time sharing coefficient equal to or greater than 0.001. 44

3.6 Maximum throughput of the distributed algorithms as a function of the number of nodes in the network. 45

3.7 Convergence of subgradient optimization for a network with 10 nodes. We report the difference between the optimal throughput R^*, and the throughput after primal recovery at step n. 46

3.8 A wireless network with 12 nodes, one source, and two multicast groups, each consisting of two nodes. 47

3.9 Maximal rate to two groups of multicast receivers and to their union. The average is over 100 random wireless networks. The networks have 12 nodes and two groups, both consisting of two receivers. An example of such a random network is shown in Fig. 3.8. 48

4.1 A two sink multicast from s to t_1 and t_2. 60

4.2 Progress of the asynchronous algorithm (left), synchronous primal-dual algorithm [1] (center) and asynchronous primal-dual algorithm [1] (right). We measure convergence of the flow x, of which the optimal value is 0.3. 61

4.3 (a) Network topology, and (b) corresponding optimal flows. The source is red, the sinks are blue and green. The common flow to both sinks is cyan. Missing links specify a flow of zero. (c) Convergence of the dual function $q(\boldsymbol{p})$ of coordinate ascent (asynchronously) and primal-dual (synchronous rounds); the dual function is normalized to 1. The convergence of the link flows (d) x_1 and (e) x_2. 62

4.4 In the network (a), the edge carrying flow x_1 fails after 150 iterations. In the graph (b), the flow x_2 first converges to the optimal value of 0.4. After the link failure, it converges to the new optimal value of 1. Note that since the links have different convex cost functions, the link flows are not zero or one as in [2]. 63

5.1 The topology graph $\mathcal{G}_t$ of a network with $n = 3$ standard nodes and a relay node 0. Each link represents two directed links, going in opposite directions. 68

5.2 The corresponding queuing network graph $\mathcal{G}_q$ for the network in Fig. 5.1. Directed links indicate possible packet transitions; packets that leave the system (the arrows with solid tips) are directed to an artificial node E, the *system exit*. A subset of the virtual queues X_Q^j can be served jointly in one time slot if they correspond to a stable set in the conflict graph $\mathcal{G}_c$ 72

5.3 The conflict graph $\mathcal{G}_c$ corresponding to the virtual queues at the relay in Fig. 5.2. 74

5.4 Routing stabilizes the network for all rates smaller than $\lambda_r = 0.1448$, network coding extends the stability region to $\lambda_n = 0.1521$, which is an increase of about 5%. In Fig. 5.5 we simulate sample paths of the online backpressure algorithm for the points $\{\lambda_1, \lambda_2\} = \lambda_r \pm 0.0003$, and $\{\lambda_3, \lambda_4\} = \lambda_n \pm 0.0003$. . 80

5.5 The total number of queued packets in the system for different injection rates. In particular, for λ_1 both routing and network coding stabilize the system, for λ_2 and λ_3 only network coding stabilizes the system, while for λ_4 both policies result in an unstable system. 81

6.1 A schematic of Algorithm 1 for finding the optimal network coding solution of (6.2). 91

6.2 Median of decoding delay for the transmission of $K = 100$ packets to $N = 3$ to $N = 100$ receivers. Channel erasures are memoryless and occur with a probability of $p = 0.5$ independently in every link. We compare Algorithm 1, our heuristic, and the random opportunistic algorithm in [3]. 94

6.3 The effect of increasing the number of packets on the computational complexity of Algorithm 1. Our measure for computational complexity is the number of recursions. 95

List of Tables

2.1 Summary of random linear network coding. 10

2.2 The maximal achievable rates (in packets/slot) under the different policies. . . . 21

Kurzfassung

Diese Arbeit untersucht das Mehrfachzugriffsproblem in drahtlosen Netzwerken wenn Netzkodierung verwendet wird. Es wird ein Algorithmus zur Ressourcenallokation hergeleitet, der nicht einzelne Kanten aktiviert, sondern sogenannte Hyperkanten und der dadurch aus den Vorzügen von zufälliger Netzkodierung Nutzen ziehen kann. Es wird sowohl durch theoretische Untersuchungen als auch durch Simulationen gezeigt, dass wenn Mehrfachzugriff und Netzcodierung gemeinsam optimiert werden, ein deutlich höherer Datendurchsatz zu erwarten ist, als wenn beide Probleme separat betrachtet werden. Darauf aufbauend wird ein verteilter heuristischer Algorithmus hergeleitet, der das zu Grunde liegende Optimierungsproblem dezentralisiert löst. Weiterhin wird untersucht in welchem Umfang es nötig ist, das Netzwerk zu synchronisieren, um eine korrekte Konvergenz von verteilten Algorithmen zu gewährleisten. Es wird die Frage behandelt, in wie weit Netzkodierung Vorteile bringt, wenn man mehrere Verbindungen betrachtet und dadurch Netzkodes nicht mehr zufällig sein können. Eine bestimmte Klasse von Kodes - die sogenannten sofort dekodierbaren Kodes - werden analytisch untersucht und es wird durch Simulationen belegt, dass durch ihre Verwendung der Datendurchsatz erhöht werden kann. Schließlich wird auf den Zusammenhang zwischen Datendurchsatz und Verzögerung bei netzkodierter Übertragung eingegangen und es werden Verfahren entwickelt zur Reduktion der Verzögerung.

Abstract

In this work, we address network coding for the multiple access layer in wireless networks. We propose a scheduling technique that activates hyperarcs rather than arcs, as in classical link-based scheduling, and therefore can harness the gains of random network coding. We encapsulate the constraints on valid network configurations in a conflict graph model and formulate a joint optimization problem taking into account both the network coding subgraph and the schedule. By means of simulations, we show that jointly optimizing the network coding subgraph and the transmission schedule leads to a substantial performance improvement. Using Lagrangian relaxation, we decompose the overall problem into two subproblems, a multiple shortest path problem, and a maximum weighted stable set (MWSS) problem. We show that, if we use a greedy heuristic for the MWSS part of the problem, the overall algorithm is completely distributed. We provide extensive simulation results for both the centralized optimal and the decentralized algorithms.

Next, we look at relaxing the assumption of synchronization in the network. We propose an asynchronous algorithm for computing multicast subgraphs, in analogy to the well-known distributed asynchronous Bellman-Ford algorithm for routing. It turns out that asynchronous algorithms require a strictly convex problem formulation, which poses certain restrictions on the network model, most importantly the schedule has to be assumed fixed. We provide extensive simulation results showing fast convergence, despite the lack of any central clock in the network, and robustness with respect to link or node failures.

We then extend network coding to take place across different independent sessions. We propose a framework for joint optimal scheduling of packet transmissions and network coding with the restriction that packets have to be decoded after one hop. We compute the stability region of this scheme and propose an online algorithm that stabilizes every arrival rate vector therein. The

online algorithm requires computation of stable sets in an appropriately defined conflict graph. We show by means of simulations that this inherently hard problem is tractable for some instances and that network coding extends the stability region over routing and leads, on average, to a smaller backlog.

Finally, we look at the relationship between throughput and delay for network coded transmissions in erasure broadcast channels. We present a systematic framework for the minimization of decoding delay under instantaneous decoding constraints. The underlying problem is NP-hard, but we provide a customized and efficient algorithm for finding the optimal solution. We illustrate how this optimal algorithm can be converted to a heuristic with very small computational complexity.

1 Introduction

The advent of network coding, in the early years of the new millennium, awoke the entire community of computer networking from a dogmatic slumber. Ever since computers were connected to exchange data, nobody had questioned the implicit assumption that the flow of information satisfies the same rules as the flow of a liquid through a network of pipes. In a sequence of works by Ahlswede et al. [2], Li et al. [4], and Koetter and Médard [5], it was shown that this simple analogy falls short of characterizing the nature of information and of capturing the rich set of operations within which we can manipulate it. It soon became clear that the notion of network coding is not only of theoretical interest, but moreover can have a profound impact on the design of communication networks and promises significant gains - in terms of performance as well as better and more robust architectures [6, 7].

Network coding can be applied to wireline and wireless networks and shows gains in both scenarios [1]. Our focus will be largely on wireless networks. The benefits of network coding are particularly significant if the underlying medium transmits by broadcast, is unreliable, and operates in a regime where resources are scarce. Moreover, this is also precisely the setup, where classical routing techniques cease to work reliably. Under such conditions, improving network performance becomes critical to the overall operation.

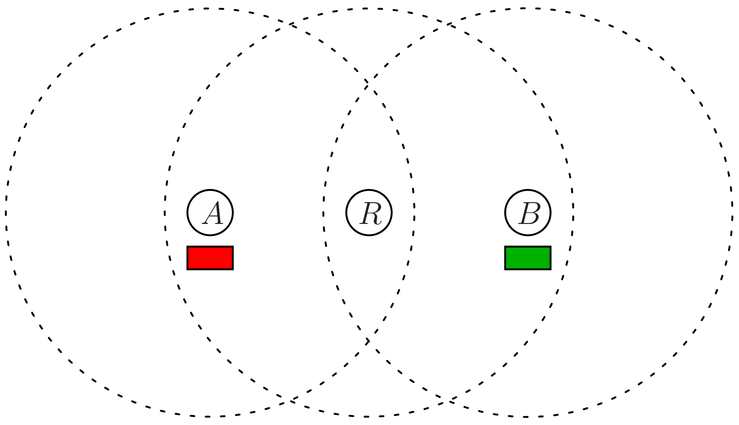
In wireless networks, transmissions are often unreliable. In addition, neighbors often overhear packets not intended for them. Network coding is especially effective in broadcast media, where each transmission is overheard by all neighbors rather than only the single neighbor for whom the message was intended. The question that we shall be addressing from various perspectives throughout this thesis is about the relationship between network coding and the other layers in the network, especially the multiple access (MAC) layer. It is possible to apply network coding in place of routing and leave the other layers completely unaffected. We will argue that, where possible, it is much more desirable to design the overall network with network coding in mind. This allows us not only to capitalize on the increased throughput and robustness that network coding supplies, but also to benefit from the structural gains that network coding offers.

To illustrate what we mean by increased throughput and structural gains, consider the following simple example depicted in Fig. 1.1. There, two wireless nodes A and B need to exchange a pair of packets. They are not in mutual radio range, however the relay R is in radio range of both and can facilitate the exchange. In traditional routing this would take four steps, as indicated in Fig. 1.2(a). With network coding, as shown in Fig. 1.2(b), once the relay has received both A's and B's packets, it can broadcast the binary XOR of the pair. Then A and B can recover the packets that they need by XOR-ing again the mixed packet with the ones they hold[1]. This reduces the number of transmissions to three.

We see in this example how network coding improves performance - it reduces the required bandwidth by 25%, as well as the energy consumption by the same factor, if we assume that

[1]Interestingly, the "trick" that comes to our aid here has long been known under the name *XOR-swap* in assembly programming and has been used for a completely different purpose - to swap the contents of two variables without the need of a temporary variable. Typically, swapping the contents of memory cells A and B requires a third cell C and the following instructions: $C \leftarrow A; \quad A \leftarrow B; \quad B \leftarrow C$. The XOR-swap needs no temporary variable and can be implemented as follows: $A \leftarrow A \oplus B; \quad B \leftarrow A \oplus B; \quad A \leftarrow A \oplus B$.

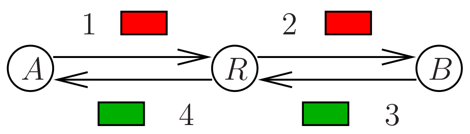
Figure 1.1 Two wireless nodes wish to exchange a pair of packets; the respective radio ranges are indicated by dotted lines.



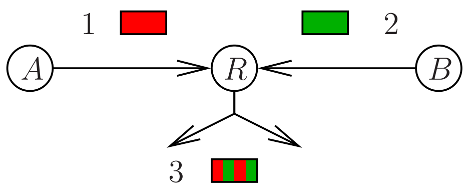
all packet transmissions require the same amount of power. To understand the consequences for the network architecture, consider the required relative bandwidths in the case of routing and in the case of network coding. For routing we have $BW_A = BW_B = 0.25$, and $BW_R = 0.5$, where the total bandwidth is normalized to 1. For network coding, correspondingly, we have $BW_A = BW_B = BW_C = \frac{1}{3}$; in this example, network coding equalizes the bandwidth demands of neighboring nodes [8]. In practice, many MAC-protocols, and in particular 802.11, are *locally fair* and assign equal shares of bandwidth to competing neighbors. Therefore, if network coding equalizes bandwidth demands of neighbors, then a locally fair underlying MAC is expected to perform better. The combination of these two effects, the throughput gain due to coding and the structural gain due to the improved collaboration with the multiple access mechanism, have been demonstrated to dramatically improve network performance [7]. The work [8] has contributed to our understanding of the structural gains of network coding in this setup - in fact, the MAC gain is responsible for most of the observed throughput increase.

The previous example illustrates that when network coding and the underlying multiple access mechanism are well-matched, good network performance results. Since optimal network codes guarantee performance no worse than the best possible routing solution, it is tempting to replace routing with coding in existing network solutions without much thought about the

Figure 1.2 Network coding versus routing.



(a) With routing, four steps are needed for the exchange.



(b) With network coding, three steps are sufficient.

broader network characteristics. We argue that network codes should be adapted to the networks in which they are employed and demonstrate the benefit of joint design with the other layers of the network. Concretely, we look at the following problems:

- How can we schedule wireless broadcast transmissions to achieve the highest possible network coding gain?
- To what extent can we relax the assumption of synchronous updates in the network?
- How can we use coding across different sessions in a practical and local way, but still arrive at a rigorous performance analysis?

1.1 A Brief Survey of Relevant Work

In this section, we will provide a very brief survey of work relevant to the thesis. As the field has grown very quickly, our survey cannot be complete. For a more comprehensive introduction to network coding, see [9].

The notion of network coding was introduced in the seminal work of Ahlswede et al. [2], where it was established that network coding is sufficient to achieve the min-cut bound in multicast networks. This is in sharp contrast to routing, which even in wireline and lossless networks does not achieve the min-cut in general. Li et al. [4] looked at linear network codes and showed that the multicast capacity can always be achieved by linear codes. The work of Koetter and Médard

[5], where the authors developed an algebraic formulation of the network coding problem and used it to derive necessary and sufficient conditions for the feasibility of both multicast and multiple unicast connections. This work provided critical tools later used in the immense body of work on decentralized network coding and optimization.

Code construction algorithms for the multicast were proposed by Ho et al. [6] and Jaggi et al. [10]. The algorithm in [6], referred to as *random linear network coding*, builds on the work in [5]. In [6], nodes form linear combinations of the packets they have stored in memory; coefficients are chosen independently and randomly, thus yielding a fully decentralized and with high probability capacity achieving algorithm. The work in [10] describes polynomial-time global code construction algorithms. The multicast network coding problem is largely solved, though there is still room for advancement. For example, decoding complexity for random linear network coding is high since it requires matrix inversion, which runs in time $O(n^3)$.

Unfortunately, far less is known about code construction for multiple unicast sessions. For example, the capacity region of several simultaneous and independent point-to-point sessions is unknown. The difficulty is caused in part by the insufficiency of linear network coding to achieve capacity [11]. Despite that, there are useful engineering applications. In Traskov et al. [12], the authors present a centralized linear programming solution that searches for coding opportunities in the network. Ho et al. [13] also provide a constructive approach to the multiple unicast problem. These ideas were extended in Eryilmaz et al. [14], where the authors propose dynamic and online stabilizing algorithms for the problem.

Since centralized design is infeasible for many applications, decentralized algorithms are required. The COPE protocol proposed by Katti et al. [7] opportunistically takes advantage of *local* coding structures.

Here, packets are combined into a single transmission if the intended recipient either knows or can overhear the packets necessary for decoding. Its basic idea is based on the example discussed in the previous section, where owing to overheard packets, several transmissions can be combined into one. It is interesting to note that the large reported gains cannot be explained by the ability of network coding to reduce the number of transmissions *alone*. Instead, the more significant factor is the interaction between network coding, which tends to equalize bandwidth demands of neighboring nodes, and the locally fair 802.11 multiple access mechanism [8].

This observation motivates our investigation of the interplay between multiple access and network coding for wireless networks. Although, the literature on wireless network coding is extensive, the papers looking explicitly at multiple access issues are few. Lun et al. [1] proposed posing network coding as a problem of minimizing resources in a network, assuming that all network transmissions are orthogonal and therefore interference-free. In Wu et al. [15], the authors consider the impact of interference, focusing on minimizing power consumption. If multiple access problems are explicitly considered, this is typically done by attempting to find “good” transmission schedules according to heuristic rules. The most popular such technique is to select valid network configurations that are *maximal*, in the sense that no more transmissions can be scheduled without causing a collision, as in e.g. Sagduyu et al. [16]. One of the goals of this work is to to study channel access and wireless network coding jointly rather than independently.

1.2 Outline of the Thesis

The remainder of thesis is organized as follows.

In Chapter 2, we introduce the network coding scheme and the network model, and we provide an example to illustrate different possible approaches to the multiple access problem for coded and uncoded networks.

In Chapter 3, we address the multiple access problem for coded networks and propose a scheduling technique that activates hyperarcs rather than arcs, as in classical scheduling approaches. We encapsulate the constraints on valid network configurations in a conflict graph model and formulate a joint optimization problem taking into account both the network coding subgraph and the schedule. Using Lagrangian relaxation, we decompose the overall problem into two subproblems, a multiple shortest paths problem and a maximum weighted stable set (MWSS) problem. We show that, if we use a greedy heuristic for the MWSS part of the problem, the overall algorithm is completely distributed. Our simulation results indicate that the optimal algorithm improves performance by up to a factor of two compared to widely used techniques such as orthogonal or two-hop-constrained scheduling. The decentralized algorithm is shown to buy its distributed operation with some throughput losses. Experimental results on randomly generated networks suggest that these losses are not large. We also look at the power consumption of our scheme and quantify the trade-off between power and bandwidth efficiency.

In Chapter 4, we propose an asynchronous algorithm for computing multicast subgraphs. The algorithm is analogous to the well-known distributed asynchronous Bellman-Ford algorithm for routing. Our central idea is to apply a block-coordinate ascent algorithm to the dual of the problem. The resulting algorithm is fully asynchronous. However, it leads to certain other constraints on the formulation that we discuss. We provide extensive simulation results showing fast convergence despite the lack of any central clock in the network and robustness with respect to link or node failures.

In Chapter 5, we look at network coding across different users with the restriction that packets have to be decoded after one hop. We compute the stability region of this scheme and propose an online algorithm that stabilizes every arrival rate vector within the stability region. The online algorithm requires computation of stable sets in an appropriately defined conflict graph. We show by means of simulations that this inherently hard problem is tractable for some instances. We also show that network coding extends the stability region over routing and leads, on average, to a smaller backlog.

In Chapter 6, we are concerned with designing feedback-based adaptive network coding schemes with the aim to minimize decoding delay in each transmission. As in Chapter 5, we impose the instantaneous decoding constraint and propose efficient algorithms for finding the optimal solution within this class of network codes. We verify the delay and computational complexity of our techniques through simulations.

In Chapter 7, we conclude the thesis and provide a brief perspective on future work.

1.3 Publications Preceding this Thesis

Parts of the material presented in this thesis appear in published papers [17–23] and in as yet unpublished paper [24].

2

Preliminaries

In this chapter, we review some necessary technical details, describe the network model and motivate the joint approach to multiple access and network coding using a simple example. We begin with discussing the network coding scheme.

2.1 Random Linear Network Coding

In Chapters 3 and 4, we apply the random linear network coding approach, proposed by Ho et al. [6, 25]. It is relatively easy to implement and can be included in existing protocol stacks without the need for a complete redesign [26]. The coding scheme is summarized in Table 2.1. Appending the encoding coefficients in the header [26] incurs an overhead of $N \log_2 q$ bits. This overhead is negligible if the payload of the packets is sufficiently large. Alternatively, Koetter and Kschischang have proposed a method [27] that reduces the overhead associated with headers. Motivated by non-coherent communications, their approach is considerably more complex.

Input:

- A source node, a set of multicast sinks, and intermediate nodes such that the resulting network is connected.
- Packets $p_1, \dots, p_N$ that the source wants to transmit.

Multicast source:

- The source node forms linear combinations $q_j = \sum_{i=1}^{N} \alpha_i p_i$, where the coefficients α_i are drawn uniformly at random from a finite field $GF(q)$.
- The vector of encoding coefficients $[\alpha_1, \dots, \alpha_N]$ is appended to the packets prior to their transmission.

Intermediate nodes:

- When an intermediate node *receives* a packet, it stores it in its memory.
- To *transmit* a packet, it forms a linear combination from the packets in its memory $q_1, \dots, q_K$, with (new) random coefficients β_i drawn from $GF(q)$.
- As all operations in the network are linear, any packet can be represented as a linear combination of packets $p_1, \dots, p_N$. The vector of coefficients used in this linear representation - called the *global encoding vector* - is appended to the packet prior to its transmission.

Multicast sinks:

- Each sink stores received packets in its memory. When it has received at least N packets, it attempts Gaussian elimination on the global encoding vectors of the received packets. If it is successful, it recovers the packets $p_1, \dots, p_N$.

Table 2.1: Summary of random linear network coding.

Random linear network coding is optimal in the sense that it achieves the min-cut bound from the source to *every* multicast sink [6]; more precisely, if the field size q over which we code is sufficiently high, the decoding error probability approaches zero. The maintenance of min-cut conditions between the set of senders and every receiver individually is always a necessary condition for feasibility; under network coding it is also *sufficient*. Network coding can be applied with only minor modifications to wireline or wireless networks.

It is not obvious how we can guarantee that enough linearly independent (or *innovative*) packets reach the sinks such that the collection of global encoding vectors is non-singular. To ensure this, we have to carefully choose which intermediate nodes inject coded packets in the network and at what rates. That is, we have to select a subnetwork that, fully utilized, can support the desired connection. Moreover, often the goal is to satisfy the connection and at the same time to minimize resource consumption in the network. This is the problem that we refer to as *subgraph optimization*. It turns out that these problems, the subgraph optimization and the network code construction, can be addressed separately without loss of optimality [1]. Therefore, from now on, we focus entirely on the subgraph optimization problem.

2.2 Network Coding Subgraph

We consider wireless networks in slotted time; in any slot a node can either broadcast one constant-length packet or stay idle. In what follows, all rates have the unit packets/slot. We model a wireless network, and in particular broadcasting, by a hypergraph (a generalization of a graph) which is defined as follows:

Definition 1 *A hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ is a set of nodes $\mathcal{N}$ and a collection of hyperarcs $\mathcal{A}$. A hyperarc $(i, J) \in \mathcal{A}$ is a generalization of an edge, where $i \in \mathcal{N}$ and $J \subset \mathcal{N}$.*

If node i injects a packet on hyperarc J, it is received by some subset $K \subseteq J$, possibly K being the empty set $\emptyset$. Let $A_{iJ}(\tau)$ be the counting process describing packet injections on hyperarc

J and $A_{iJK}(\tau)$ be the counting processes accounting for the packets received *precisely* by the subsets K. We have $\sum_{K \subseteq J} A_{iJK}(\tau) = A_{iJ}(\tau)$. We assume that the injection processes we use are stationary and ergodic and therefore their time averages $\lim_{\tau \to \infty} \frac{A_{iJ}(\tau)}{\tau}$ exist with probability 1 and are finite. We use z_{iJ} to denote this limit. Similarly, we define $\lim_{\tau \to \infty} \frac{A_{iJK}(\tau)}{\tau} = z_{iJK}$. With these assumptions $z_{iJ} = \sum_{K \subseteq J} z_{iJK}$ is the average packet injection rate on hyperarc J. We shall assume that the underlying process is memoryless and

$$p_{iJK} = \frac{z_{iJK}}{z_{iJ}}, \tag{2.1}$$

is the probability that a packet injected on J is received precisely by the subset K. This can take into account that transmissions experience erasures, which may be due to distance attenuation, shadowing, or fading. We call the vector $\mathbf{z} = (z_{iJ})_{(i,J) \in \mathcal{A}}$ the *network coding subgraph*. In wireless networks, the network coding subgraph is further constrained to lie in the multiple access rate region of the network. This is - by a time sharing argument - a convex set, albeit with a possibly high description complexity. We discuss the multiple access constraints, such as half-duplex transceivers and interference, in detail in the next section.

2.3 Interference and Half-Duplex Constraints

Consider a wireless network represented by a set of nodes $\mathcal{N}$ and for each node $i \in \mathcal{N}$ a set of neighbors $N(i) \subset \mathcal{N}$. We assume that when i transmits all nodes in $N(i)$ are in radio range and can potentially receive or experience interference[1] from i.

From the neighborhood relation we construct a hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ with $\mathcal{N}$ corresponding to the set of nodes in the network. For each node i we introduce $2^{|N(i)|} - 1$ hyperarcs, (i, J) where J ranges over all subsets of $N(i)$ excluding the empty set.

[1] A popular and slightly more general model is to assume that a node can receive from i if it is contained in a set $N_1(i)$ but is subject to interference if it belongs to a superset $N_2(i) \supset N_1(i)$. Our framework can be extended to take into account such a setup. However, for the sake of a simpler notation, we abide with the above model.

Which sets of hyperarcs can transmit simultaneously without a conflict depends on the system model of the network. In networks with primary interference, e.g. spread-spectrum systems, we restrict each node to receive from at most one other node. In networks with secondary interference, we have the additional constraint that a node can only successfully receive if all other neighbors are silent. In addition, we assume half-duplex transceivers.

We shall call a set of conflict-free hyperarcs a transmission set or valid configuration, formally

Definition 2 *We say that hyperarcs (i_1, J_1) and (i_2, J_2), do* not *conflict if:*

1) $i_1 \neq i_2$,

2) $i_1 \notin J_2$, $i_2 \notin J_1$, *and*

for networks with primary interference

3a) $J_1 \cap J_2 = \emptyset$, or

alternatively for networks with secondary interference

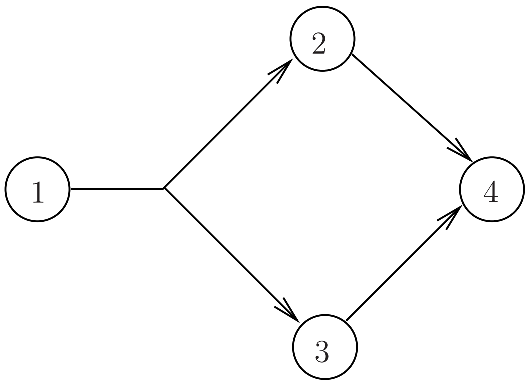
3b) $J_1 \cap N(i_2) = \emptyset$, and $J_2 \cap N(i_1) = \emptyset$.

For both the primary and the secondary interference model, the definitions are symmetric in their arguments and therefore give rise to an undirected graph representing the scheduling conflicts between pairs of hyperarcs. We construct the conflict graph as follows.

Definition 3 *The conflict graph $\mathcal{G}$ of a hypergraph $\mathcal{H}$ is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ corresponding to the set of all hyperarcs. Two hyperarcs are adjacent if they conflict.*

We can define a valid configuration of hyperarcs as a set of nodes in the conflict graph without any conflicting pair, i.e. a valid configuration is a stable set.

Figure 2.1 An example of a hypergraph. Here, the node set is $\mathcal{N} = \{1, 2, 3, 4\}$ and the hyperarc set is $\mathcal{A} = \{(1, 2), (1, 3), (1, \{2, 3\}), (2, 4), (3, 4)\}$.



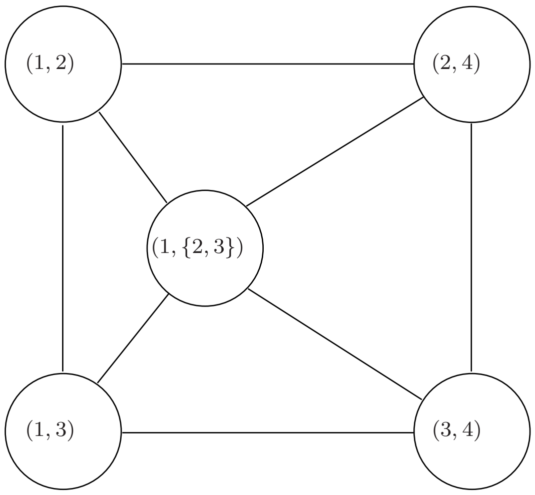
Definition 4 *A stable set S of an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a set of nodes any two of which are nonadjacent. Its incidence vector is a column vector of length $|\mathcal{V}|$, defined as*

$$\chi_v^S = \begin{cases} 1 & \textit{if } v \in S, \\ 0 & \textit{otherwise.} \end{cases} \tag{2.2}$$

A maximal stable set is one that is not contained in any other stable set. A maximum stable set is a stable set of largest cardinality. The stability number $\alpha(\mathcal{G})$ of a graph is the cardinality of the maximum stable set. The stable set polytope $P_{STAB}(\mathcal{G})$ is the convex hull of the incidence vectors of all stable sets of $\mathcal{G}$.

Example To illustrate the notation, consider the hypergraph in Fig. 2.1 and its corresponding conflict graph in Fig. 2.2. In this particular network, both the primary and the secondary interference models give rise to the same conflict graph. The conflict graph has a node for each hyperarc. Thus $\mathcal{V} = \{(1, 2), (1, 3), (1, \{2, 3\}), (2, 4), (3, 4)\}$. The stable set polytope for this example is the convex hull of the incidence vectors of the three stable sets $\{(1, 2), (3, 4)\}$, $\{(1, 3), (2, 4)\}$, and $(1, \{2, 3\})$, and the origin. $\square$

Figure 2.2 The conflict graph corresponding to the hypergraph in Fig. 2.1

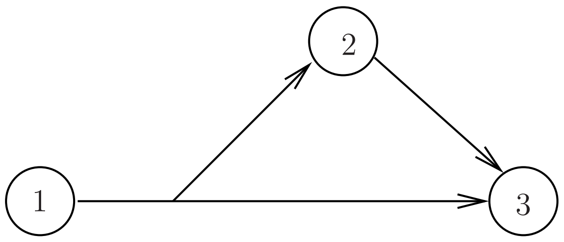


2.4 Example: Relay channel

The purpose of this example is to illustrate four possible approaches to the multiple access problem and the subgraph optimization problem in wireless multi-hop networks. Consider the network in Fig. 2.3, which was discussed in [28] in combination with random access and therefore called the *slotted Aloha relay channel*. This is reminiscent of the classical relay channel from information theory [29], which deals with the physical layer capacity of this three-terminal network. In contrast, the problem we approach is the efficient transmission of already packetized data, which is a problem of higher layers rather than a physical layer problem. We can efficiently solve this problem with network coding. The goal is to establish a unicast connection of rate R from node 1 to node 3. According to the hypergraph model, we have two directed hyperarcs $(1,\{2,3\})$ and $(2,3)$. We assume half-duplex constraints, i.e. a node cannot transmit and receive at the same time. Furthermore, owing to interference, when nodes 1 and 2 transmit simultaneously, the packets collide and *both* are lost. Even in the absence of interference, packets can be lost due to erasures - this is modelled by the reception probabilities p_{iJK}.

We consider two strategies for multiple access control: scheduleding, which effectively elim-

Figure 2.3 The relay network.



inates interference, and random access, where nodes transmit randomly with some probability in every slot. We also consider two strategies for the subgraph optimization problem: routing and network coding. This leads to four different combinations, which we discuss next in detail.

2.4.1 Scheduling and Routing

Routing assumes that we fix a path and send all packets along this path[2]. In our example, we can either use the direct path between node 1 and 3, or transmit via the relay node 2. If we use the relay, it cannot transmit and listen at the same time. To find the better of the two source-destination paths to route over, we compare the following alternatives

Path $1 \to 2 \to 3$**:** We use link $(1, 2)$, which has a success probability of $p_{1\{23\}2} + p_{1\{23\}\{23\}}$, in fraction $\alpha, \alpha \in [0, 1]$, of the time slots and correspondingly link $(2, 3)$, which has a success probability of p_{233}, in fraction $1 - \alpha$ of the time slots. Then, since we need both link throughputs to be equal

$$\alpha(p_{1\{23\}2} + p_{1\{23\}\{23\}}) = (1 - \alpha)p_{233}. \tag{2.3}$$

We conclude that the optimal time sharing coefficient is

$$\alpha^* = \frac{p_{233}}{p_{1\{23\}2} + p_{1\{23\}\{23\}} + p_{233}}, \tag{2.4}$$

[2]It is also possible to use *multi-path routing* and route along several paths. This can lead to better performance in general, but this is not the case for the simple network that we consider.

and the optimal throughput is

$$R^* = \frac{(p_{1\{23\}2} + p_{1\{23\}\{23\}})p_{233}}{p_{1\{23\}2} + p_{1\{23\}\{23\}} + p_{233}}. \tag{2.5}$$

Path $1 \to 3$**:** In this case, the throughput is simply

$$R^* = p_{1\{23\}3} + p_{1\{23\}\{23\}}. \tag{2.6}$$

How we compensate for the erasures depends on whether the network has an ARQ-mechanism; if it does not, then we have to use a forward-error-correction (FEC) code. Assume we apply FEC and select the path via the relay. Then, we are forced to choose between two alternatives, both undesirable: If we use an FEC-code end-to-end, we lose throughput, because the end-to-end-FEC has to compensate the higher erasure rate of the pair of links. If we use the FEC link-by-link, we do not lose throughput but incur the delay of decoding and re-encoding at the relay [30]. Random linear netwok coding, in contrast, achieves the maximal path throughput without decoding at intermediate nodes. Hence, it has advantages even when the "network" is a simple pair of links.

2.4.2 Scheduling and Network Coding

Owing to the half-duplex and interference constraints, in an interference-free schedule for the relay network, at most one hyperarc transmits in every slot. A schedule can therefore be parameterized by one parameter, α, the fraction of time slots in which a packet injection occurs on $(1, \{2, 3\})$. Correspondingly, $1 - \alpha$ is the fraction of time slots in which a packet injection occurs on $(2, 3)$. We wish to determine the maximal rate R^* that can be achieved. We do this by solving a linear program; its formulation will be discussed in greater detail in Chapter 3.

$$\text{maximize}\, R$$

subject to

$$\begin{aligned}
R &\leq x_{12} + x_{13} && (2.7)\\
x_{12} &\leq \alpha\left(p_{1\{23\}2} + p_{1\{23\}\{23\}}\right) && (2.8)\\
x_{13} &\leq \alpha\left(p_{1\{23\}3} + p_{1\{23\}\{23\}}\right) && (2.9)\\
x_{12} + x_{13} &\leq \alpha\left(p_{1\{23\}2} + p_{1\{23\}3} + p_{1\{23\}\{23\}}\right) && (2.10)\\
x_{23} &\leq (1-\alpha)\, p_{233} && (2.11)\\
x_{12} &= x_{23} && (2.12)\\
0 \leq \alpha &\leq 1 && (2.13)\\
x_{12}, x_{13}, x_{23} &\geq 0. && (2.14)
\end{aligned}$$

In this formulation, a variable x_{ij} denotes the flow of innovative packets between nodes i and j. We can solve this LP analytically, by applying Fourier-Motzkin elimination [31, Section 2.8]. The details are described in Appendix A.1. It turns out that we have to distinguish two cases.

Case 1: If $p_{1\{23\}3} + p_{1\{23\}\{23\}} < p_{233}$, then the maximal achievable rate is

$$R^* = \frac{p_{233}(p_{1\{23\}2} + p_{1\{23\}3} + p_{1\{23\}\{23\}})}{p_{233} + p_{1\{23\}2}}, \qquad (2.15)$$

with the sender transmitting in the fraction of slots

$$\alpha^* = \frac{p_{233}}{p_{233} + p_{1\{23\}2}}, \qquad (2.16)$$

and the relay in the remaining fraction of $1 - \alpha^*$ slots.

Case 2: If $p_{1\{23\}3} + p_{1\{23\}\{23\}} > p_{233}$, then the maximal achievable rate is

$$R^* = p_{1\{23\}3} + p_{1\{23\}\{23\}}. \qquad (2.17)$$

In this case the relay is not used at all; the sender transmits directly to the destination.

2.4.3 Random Access and Routing

We now turn our attention to the case where the underlying multiple access mechanism is random access. In this case, we need to compute *transmission attempt probabilities* that maximize the end-to-end rate. If we assume that packets are routed along fixed paths, we again have to distinguish between the two possible paths of the network. Using the direct path $1 \to 3$, there is no need for medium access control; the throughput is simply $R^* = p_{1\{23\}3} + p_{1\{23\}\{23\}}$, the success probability of the link. On the other hand, routing via the relay requires us to compute two transmission attempt probabilities z_{12} and z_{23}, for links $(1,2)$ and $(2,3)$, respectively. The throughput of the path is then the minimum of the effective link throughputs [32, Section 4.6]

$$R^* = \min_{z_{12}, z_{23} \in [0,1]} \left\{ (p_{1\{23\}2} + p_{1\{23\}\{23\}}) z_{12}(1-z_{23}), \quad p_{233} z_{23}(1-z_{12}) \right\}, \tag{2.18}$$

and we choose the better of the two path paths to route along.

2.4.4 Random Access and Network Coding

This problem (introduced and discussed in [28]) requires finding transmission attempt probabilities $z_{1\{23\}}$ and z_{23} that maximize the rate of the connection; the notation $z_{1\{23\}}$ implies that we are now using the hyperarc $(1, \{2,3\})$, instead of the constituting links individually. The optimization problem is as follows [28]:

$$\text{maximize}\, R$$

subject to

$$\begin{aligned} R &\leq z_{1\{23\}}(1-z_{23})(p_{1\{23\}2}+p_{1\{23\}3}+p_{1\{23\}\{23\}}) && (2.19)\\ R &\leq z_{1\{23\}}(1-z_{23})(p_{1\{23\}3}+p_{1\{23\}\{23\}})+z_{23}(1-z_{1\{23\}})p_{233} && (2.20)\\ 0\leq z_{1\{23\}}, z_{23} &\leq 1. && (2.21)\end{aligned}$$

This problem is not convex and as such very difficult to solve in general. Using a time sharing argument, one can argue that, it is equivalent to optimizing over the convex hull of the constraint set. Unfortunately, finding a parameterization of the convex hull is no easier than solving the original non-convex problem. Nevertheless, Riemensberger et al. [33] have successfully applied a specialized version of the branch-and-bound algorithm to find the optimal solution for smaller networks. In [34], a heuristic is considered for finding valid transmission attempt probabilities that support a given throughput. If a certain fixed throughput is found to be feasible, one can then incrementally increase it and check if the new higher throughput is feasible.

2.4.5 Discussion

To compare the four proposed schemes, consider the following fixed transmission success probabilities

$$\begin{aligned} p_{1(23)2} &= 9/16 && (2.22)\\ p_{1(23)3} &= 1/16 && (2.23)\\ p_{1(23)23} &= 3/16 && (2.24)\\ p_{233} &= 3/4. && (2.25)\end{aligned}$$

Table 2.2 shows the resulting maximal rates for the four different mechanisms that were discussed. The results confirm that scheduling always performs better than random access. Net-

Maximal achievable rate	Routing	Network coding
Scheduling	0.375	0.464
Random access	0.25	0.25

Table 2.2: The maximal achievable rates (in packets/slot) under the different policies.

work coding subsumes routing and, therefore, performs at least as well.

Another meaningful comparison between the proposed schemes is to plot their power-efficiency as a function of the rate. If we assume that each packet transmission requires the same amount of energy, then the average number of transmissions needed to deliver a packet end-to-end is a reasonable estimate of the power consumption. Our results are summarized in Fig. 2.4, where we plot the minimal average number of transmissions per packet as a function of the connection rate. For a fixed rate, scheduling always requires fewer transmissions than random access, and network coding reduces the expected transmissions compared with routing. In Fig. 2.4, the curve for routing and random access is defined piecewise. This is the result of a switch of paths as the load increases; for lighter loads, it is better to use the path via the relay $(1 \rightarrow 2 \rightarrow 3)$. For heavier loads, it is better to switch to the direct path $(1 \rightarrow 3)$. We also observe that the maximal achievable rate does not tell the whole story about the merits of a scheme. With random access, the maximal rates with network coding and with routing are the same, but network coding results in a more efficient network operation with fewer packet transmissions for any fixed rate.

For small rates, the difference in performance between random access and scheduling is correspondingly small, since low attempt probabilities will result in fewer collisions and therefore not many packet transmissions will be wasted. This is reflected in Fig. 2.5, which shows the optimal attempt probabilities for network coding with random access. The objective is to minimize the average number of transmissions per packet $z_{1\{23\}} + z_{23}$ for a fixed rate R. For light loads, the transmission attempt probabilities increase almost linearly. As the load becomes heavier, the increase is much more pronounced.

Figure 2.4 The minimum average number of transmissions per packet as a function of the rate R.

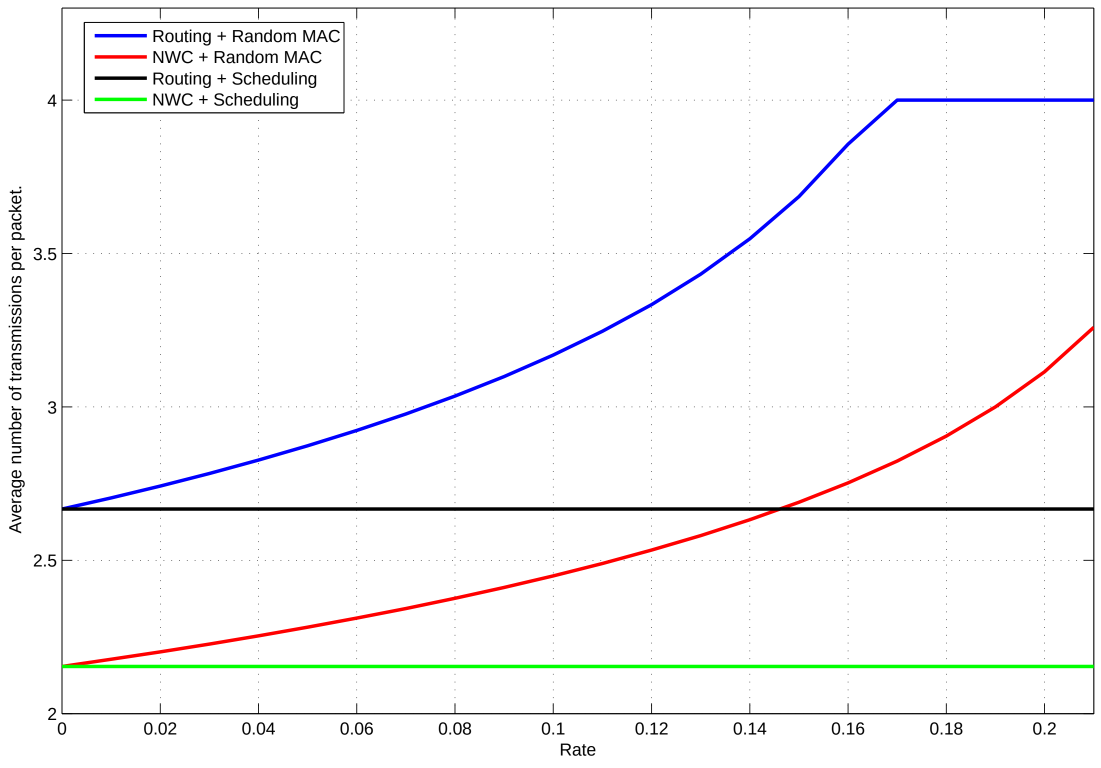
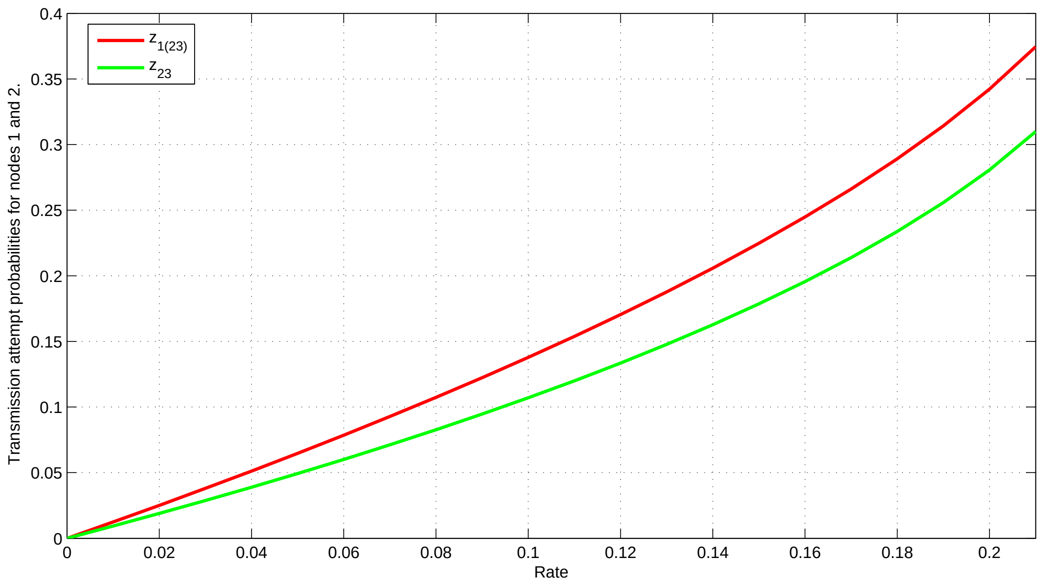


Figure 2.5 Optimal transmission attempt probabilities for network coding as a function of the rate R.



This confirms the well-known behavior and limitations of random access protocols; they are inherently simple and they are easy to implement in a fully decentralized way. However, unless the load is very light, they suffer from limitations in throughput and in efficiency. If the load in the network is light, then the objective of supporting a certain set of connections can be captured by classical routing and channelization schemes; network coding, while beneficial, might not be needed. Under heavy network loads, which is the more relevant operation regime in practice, a lot can be gained by capitalizing on the advantages of network coding regarding throughput and robustness.

It turns out that a sizeable share of the throughput gains of network coding can be harnessed only when transmission scheduling is done in a way that creates coding opportunities; this is the focus of the next chapter. In principle, network coding can work with any underlying multiple access mechanism. We show, however, that a joint approach to scheduling and network coding subgraph optimization is needed if the load of the network is high and bandwidth is scarce. For the simple relay channel example, the underlying transmission schedule was parameterized by just one coefficient α. How this generalizes to larger multi-hop networks is not obvious. In the next chapter, we demonstrate that we can indeed extend the approach to general multi-hop networks and evaluate the resulting gains.

3

Scheduling for Network Coded Multicast

The problem we address in this chapter is to compute an optimal network coding subgraph and a schedule that can support it. Wireless networks are often interference-limited, and efficient operation requires a high frequency reuse within the network. This is achieved by means of scheduling, which carefully allows simultaneous transmissions that do not interfere with each other. Finding an optimal, or even good, subgraph is by no means a simple problem in the presence of half-duplex transceivers and interference. When medium access control assigns every node an orthogonal channel, it is possible to compute the optimal subgraph by solving a linear or convex program [1]. The solution given is distributed. When bandwidth is plentiful, it may be reasonable to orthogonalize the entire network. However, if bandwith is scarce, the resulting throughput will be low.

Owing to the hardness of the general problem, the prevalent approach in the literature is to heuristically construct an interference-free transmission schedule and then to compute an optimal subgraph over this essentially orthogonal network. As an example, in [35] the authors propose a suboptimal collision-free strategy where two nodes cannot transmit simultaneously if they are within two hops. This is a sensible practical solution, but the combination of a carefully optimized network coding subgraph and a more or less ad-hoc medium access strategy may lead to poor performance.

To address this challenge, we suggest a framework where the network coding subgraph and channel access are optimized jointly. We construct a hypergraph that takes into account possible transmissions to every subset of neighbors of a node. Each such subset is represented by a hyperarc. We consider subsets of hyperarcs that can be activated simultaneously without interfering, as opposed to classical link-based scheduling. These constraints are transformed into a conflict graph representation, where the hyperarcs are represented by nodes and the activation constraints are given by edges. A set of hyperarcs can be activated simultaneously if they are not connected by any edge in the conflict graph. This conflict graph encapsulates the combinatorial difficulty of the problem. Finally, we exploit the polymatroid representation of the rate regions associated with valid network configurations to derive a succinct expression for the entire rate region.

Having derived the joint scheduling and network coding algorithm, we seek to distribute the operation across the network, in a way similar to [1]. To that end, we decompose the problem into two subproblems using Lagrangian relaxation. The first subproblem is a multiple shortest path problem, the second and considerably harder subproblem is a maximum weighted stable set problem. Since the latter is NP-hard, we propose an approach that greedily chooses *maximal* stable sets according to appropriately defined weights. This can be done in a decentralized way, giving a distributed algorithm. By means of simulations, we demonstrate that the throughput is close to the optimal performance. By optimal performance, we refer to solving the original optimization problem and optimizing over the entire stable set polytope. We also study the power consumption of our scheduling algorithm and quantify the power vs. bandwidth trade-off. This quantifies, how much power is wasted by permitting collisions as in our approach.

Formulating scheduling as finding “stable” sets or “independent” sets is a common technique. Prior examples include scheduling for routed traffic [36], scheduling in switches [37], network code construction in a wireline setup [38], and scheduling for Banyan networks [39]. For scheduling network coded transmissions in wireless networks, this approach gives rise to a number of novel and interesting observations. Here nodes broadcast coded packets to all neighbors (the wireless broadcast advantage) and transmissions are subject to interference. By guaranteeing a node successful transmission to a subset of its neighbors and at the same time permitting conflicts on the remaining neighbors, we are not seeking to minimize the number of collisions per se. In fact, one can argue that we are scheduling conflicts for the nodes not contained in the activated hyperarc.

To relate our approach to previously published work, note that in contrast to [15], where the authors focus on minimizing power consumption, we consider a wireless network where interference is the limiting factor. Contention resolution by means of clustering has been studied in [40] for networks with CDMA. The scheduling of broadcast transmissions is introduced in [41] in a different context, namely in an attempt to analyze the opportunistic, local combination of packets belonging to multiple unicast connections.

3.1 The Multicast Rate Region with Scheduling Constraints

Multicasting is the transmission of information from a source node s to a subset of network nodes $\mathcal{T}$.

Definition 5 *A multicast connection is a triple $(s, \mathcal{T}, R)$, with $s \in \mathcal{N}$, $\mathcal{T} \subset \mathcal{N}$, and $R > 0$ denoting the rate of the connection. All multicast sinks request the same information.*

Consider a hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ and a multicast connection of rate R with source $s \in \mathcal{N}$ and sinks $\mathcal{T} \subset \mathcal{N}$. We can apply the following flow formulation from [1] to compute the network coding subgraph. We consider the rate region (which here refers to supportable end-to-end throughputs rather than an information theoretic bound) for a multicast connection[1]. It is not an information theoretic bound because we do not look at the physical layer, but assume packetized data. Furthermore, we assume that collisions lead to loosing the packets. Under these assumptions, the rate region is then the set of rates R subject to the following constraints

$$\sum_{j \in K} x_{iJj}^{(t)} \leq z_{iJ} b_{iJK}, \quad \forall\, (i,J) \in \mathcal{A}, K \subset J, t \in \mathcal{T}, \tag{3.1}$$

$$\sum_{\{J|(i,J)\in\mathcal{A}\}} \sum_{j \in J} x_{iJj}^{(t)} - \sum_{\{j|(j,I)\in\mathcal{A}, i\in I\}} x_{jIi}^{(t)} = \begin{cases} R & i = s, \\ -R & i = t, \\ 0 & \text{else}, \end{cases} \tag{3.2}$$

$$\forall\, i \in \mathcal{N}, t \in \mathcal{T},$$

$$x_{iJj}^{(t)} \geq 0, \quad \forall\, (i,J) \in \mathcal{A}, j \in J, t \in \mathcal{T}, \tag{3.3}$$

$$\mathbf{z} = (z_{iJ}) \in P_{STAB}(\mathcal{G}), \tag{3.4}$$

where we define

$$b_{iJK} = \sum_{\{S \subset J | S \cap K \neq \emptyset\}} p_{iJS}. \tag{3.5}$$

Example To illustrate constraint (3.1), consider a hyperarc $(i, J) = (1, \{2, 3, 4\})$ with the

[1]The extension to multiple multicast connections with intra-session coding is straightforward. We omitted it to simplify exposition and notation.

following reception probabilities p_{iJS}, for $S \subset \{2,3,4\}$

$$\begin{array}{lll} p_{iJ\{2\}} = 0.1, & p_{iJ\{3\}} = 0.2, & p_{iJ\{4\}} = 0.25, \\ p_{iJ\{2,3\}} = 0.2, & p_{iJ\{2,4\}} = 0.15, & p_{iJ\{3,4\}} = 0.05, \\ p_{iJ\{2,3,4\}} = 0.05, & p_{iJ\{\emptyset\}} = 0. & \end{array}$$

We can assign these probabilities arbitrarily, as long as they sum up to one. Therefore, any dependence in the channel erasure probabilities can be accommodated. Let the injection rate be $z_{1\{2,3,4\}} = 1$. Then the rate region for the hyperarc is

$$0 \leq x_{1J2}, x_{1J3}, x_{1J4} \leq 0.5, \tag{3.6}$$

$$x_{1J2} + x_{1J3} \leq 0.75, \tag{3.7}$$

$$x_{1J2} + x_{1J4} \leq 0.8, \tag{3.8}$$

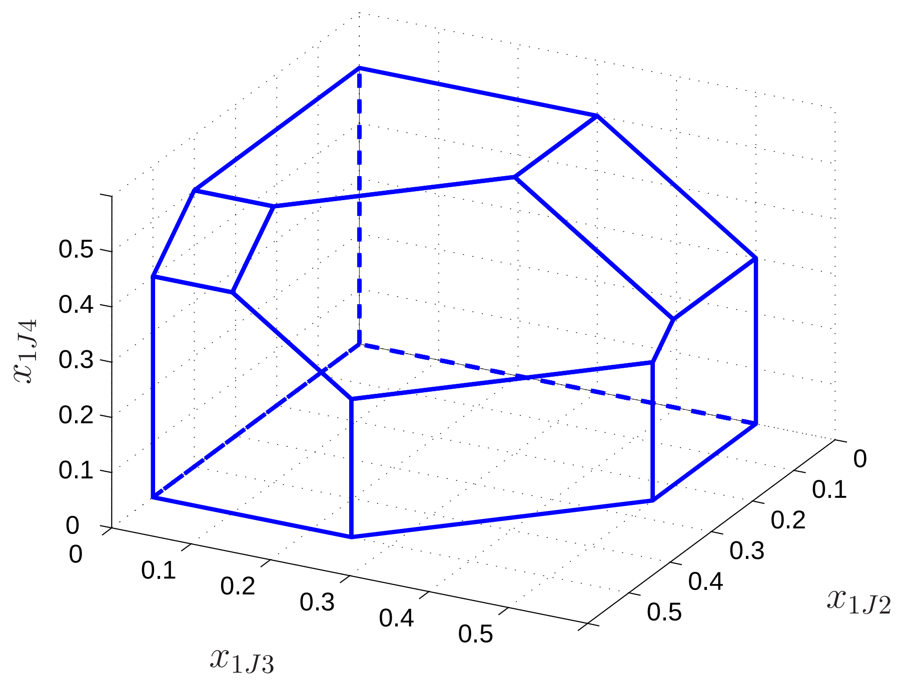
$$x_{1J3} + x_{1J4} \leq 0.9, \tag{3.9}$$

$$x_{1J2} + x_{1J3} + x_{1J4} \leq 1, \tag{3.10}$$

and is plotted in Fig. 3.1. Note that as the joint erasure probabilities p_{iJS} are defined, every link has a marginal erasure probability of 0.5. Thus, if we do not use network coding and time-share between the links, the achievable link rates have to satisfy $x_{1J2} + x_{1J3} + x_{1J4} \leq 0.5$. This illustrates the network coding gain due to the wireless broadcast advantage. □

The last constraint (3.4) explicitly accounts for interference by requiring the network coding subgraph $\mathbf{z}$ to lie in the stable set polytope of the conflict graph. Any vector in the stable set polytope can be written as a convex combination of schedules. Therefore, the demanded rate can be transmitted if the granularity of time slots is sufficiently fine. This is analogous to the Birkhoff-von Neumann decomposition of load matrices for scheduling in switches [37].

Figure 3.1 The rate region for the hyperarc $(1, \{2, 3, 4\})$



We can rewrite the linear program to yield a formulation with substantially fewer variables. For all $i \in \mathcal{N}$ and $j \in N(i)$, let

$$x_{ij}^{(t)} = \sum_{J \subset N(i)} x_{iJj}^{(t)}, \tag{3.11}$$

with the understanding that $x_{iJj}^{(t)} = 0$ if $j \notin J$. Note that on the RHS we sum over all hyperarcs that leave node i. The similar transformation proposed in [1] can only handle collections of hyperarcs with a subset-containment relation. A more general formulation is required in our case, as scheduling depends on activating hyperarcs that are not constrained by a subset relation.

Consider the following formulation in terms of the new variables $x_{ij}^{(t)}$. In the following LP, we maximize the rate of a multicast session. Other objectives, such as minimizing energy consumption subject to a fixed rate, can be easily accommodated.

$$\text{maximize} \quad R$$

subject to:

Capacity constraints:

$$\sum_{j \in K} x_{ij}^{(t)} \leq \sum_{J \subset N(i)} z_{iJ} b_{iJK}, \tag{3.12}$$

$$\forall\, i \in \mathcal{N}, K \subset N(i), t \in \mathcal{T},$$

Flow constraints P_F:

$$\sum_{j \in N(i)} x_{ij}^{(t)} - \sum_{\{j | i \in N(j)\}} x_{ji}^{(t)} = \begin{cases} R & i = s, \\ -R & i = t, \\ 0 & \text{else}, \end{cases} \tag{3.13}$$

$$\forall\, i \in \mathcal{N}, t \in \mathcal{T},$$

$$x_{ij}^{(t)} \geq 0, \quad \forall\, i \in \mathcal{N}, j \in N(i), t \in \mathcal{T}, \tag{3.14}$$

Scheduling constraints:

$$\mathbf{z} \in P_{STAB}(\mathcal{G}). \tag{3.15}$$

Consistently with definition (3.5), b_{iJK} is well defined even if K is not a subset of J.

A variable of the form $x_{ij}^{(t)}$ denotes the flow of *innovative* packets on link (i, j) for sink t, whereas the variable z_{iJ} represents the packet injection rate on hyperarc (i, J). The linear equations in (3.13) establish flows at rate R from the source to all multicast sinks. Constraint (3.12) relates packet injection rates z_{iJ} to the flow of innovative packets. It also implies that the actual link usage is the maximum value of the flows belonging to different multicast sinks traversing it. This is where network coding enters the picture; with routing, the actual link usage would be simply the sum of flows going across. Constraint (3.15) requires the network coding subgraph to lie in the stable set polytope of the conflict graph. This guarantees that we can decompose the subgraph into a convex combination of valid schedules. This is the main difference between our formulation and [1], where the authors have not considered scheduling. The stable set polytope constraint dramatically changes the nature of the problem. Without (3.15) it is nothing but a multi-commodity flow problem with side constraints. As a linear program it is solvable in polynomial time [31]. With the scheduling constraint (3.15), the problem becomes NP-hard due to the encapsulated stable set problem [42].

Lemma 1 *The rate region described by (3.1)-(3.4) is equivalent to the rate region given by the reduced formulation (3.12)-(3.15).*

Proof The new flow conservation constraint (3.13) is just a reformulation in terms of the new flow variables. What we have to show is the equivalence of the constraints (3.12) and (3.1). For a fixed node i, we have a number of outgoing hyperarcs. For each hyperarc (i, J), constraint (3.1) gives us a rate region for the flows $x_{iJj}^{(t)}$. The claim is that the rate region for the sum of these flows, as defined in (3.11), is given by the sum of the inequalities defining their rate regions. The converse is easy to show since

$$\sum_{j\in K} x_{ij}^{(t)} = \sum_{j\in K} \sum_{J\subset N(i)} x_{iJj}^{(t)} \leq \sum_{J\subset N(i)} z_{iJ} b_{iJK}. \tag{3.16}$$

The achievability of these bounds is more difficult to prove. In general, for convex polytopes defined by linear inequalities, the polytope generated by their Minkowski sum is not equal to the one defined by the sum of their individual constraints. In this case, however, we can exploit the special structure of the polytopes in (3.1). They are *polymatroids*, owing to the fact that b_{iJK}, when viewed as a function of K, is a submodular function. From definition (3.5), we can verify that

$$\begin{aligned} b_{iJK} + b_{iJL} &= \sum_{\{S \subset J | S \cap K \neq \emptyset\}} p_{iJS} + \sum_{\{S \subset J | S \cap L \neq \emptyset\}} p_{iJS} \\ &\geq \sum_{\{S \subset J | S \cap (K \cap L) \neq \emptyset\}} p_{iJS} + \\ &\quad \sum_{\{S \subset J | S \cap (K \cup L) \neq \emptyset\}} p_{iJS} \\ &= b_{iJ(K \cap L)} + b_{iJ(K \cup L)}. \end{aligned} \tag{3.17}$$

The inequality is due to the fact that sets of the form $S \subset J$ for which

$$S \cap K \cap L = \emptyset,\ S \cap K \neq \emptyset,\ S \cap L \neq \emptyset \tag{3.18}$$

show up twice on the LHS of (3.17) but only once one on the RHS.

Consider two polymatroids, given by submodular set functions f_1 and f_2 respectively, i.e.

$$P_{f_i} := \left\{ x \in \mathbb{R}_+^{|J|} : \sum_{K \subset J} x_j \leq f_i(K) \quad \forall K \subset J \right\}. \tag{3.19}$$

Since P_{f_1} and P_{f_2} are polymatroids, the convex hull of their Minkowski sum is equivalent to the sum of the inequalities defining them [42, Thm. 44.6, p. 781], i.e.

$$P_{f_1 + f_2} = P_{f_1} + P_{f_2}. \tag{3.20}$$

The result follows since we consider a finite sum of polymatroids. $\square$

3.2 On the Complexity of the Scheduling Problem

Testing stable set polytope membership of the network coding subgraph (3.15) can be difficult. In general, even the question of whether a point belongs to the stable set polytope - in our case whether a network coding subgraph $\mathbf{z}$ can be decomposed into a convex combination of valid schedules - cannot be answered in polynomial time, except for certain special classes of graphs [42]. By the equivalence between optimization and separation, [2] these are the graphs for which a maximum stable set can be computed in polynomial time. Therefore, we are particularly interested in classes of graphs with polynomial stable set algorithms (see e.g. [42] for an extensive survey), as in this case solving the overall problem (3.12)-(3.15) becomes tractable, even for large networks.

A family of graphs with particularly good algorithmic properties is the family of *perfect graphs*. For perfect graphs, there is a polynomial time maximum stable set algorithm [42]. Furthermore, their stable set polytope can be described by clique inequalities, whereas for general graphs these are necessary but not sufficient [42].

Definition 6 *A graph is perfect if and only if for every induced subgraph the clique number equals the chromatic number.*

Proposition 1 *The following characterizations of perfect graphs are equivalent [42]:*

- *The complement of a perfect graph is perfect.*
- *A graph is perfect if and only if it contains no odd holes (induced subgraphs that are cycles of odd length) and antiholes (their complements).*
- $P_{STAB}(\mathcal{G}) = P_{QSTAB}(\mathcal{G})$,

where

$$P_{QSTAB}(\mathcal{G}) = \left\{ x \in \mathbb{R}_+^{|\mathcal{V}|} : \sum_{q \in Q} x_q \leq 1 \; \forall \text{ cliques } Q \subset \mathcal{V} \right\} \tag{3.21}$$

[2]The optimization problem is to maximize a linear function over a polytope P. The corresponding separation problem is to decide whether a point is in P and, if this is not the case, to display a violated constraint. As a consequence of the ellipsoid method, the polynomial time solvability of one of the problems implies the polynomial time solvability of the other [43].

is called the *fractional* stable set polytope. These clique inequalities are often used, even in non-perfect graphs, as upper bounds.

Next, we discuss a family of line networks for which the conflict graph turns out to be perfect. To that end, consider the network in Fig. 3.2. We can think of this example as a line network, where transmitted packets can be heard by the next *two* hops. The hypergraph has nodes $\{1,\ldots,n\}$, and hyperarcs

$$(i, i+1) \quad \text{for} \quad i = 1,\ldots,n-1, \tag{3.22}$$

$$(i, \{i+1, i+2\}) \quad \text{for} \quad i = 1,\ldots,n-2. \tag{3.23}$$

The corresponding conflict graph is shown in Fig. 3.3. Note that we have omitted hyperarcs of the form $(i, i+2)$, as they will never be activated. If a conflict-free transmission from i to $i+2$ is scheduled, then nodes $i+1$ and $i+2$ necessarily have to be silent, and node $i-1$ can not transmit either, owing to interference. That means that scheduling a transmission from i to $i+2$, is equivalent to activating hyperarc $(i, \{i+1, i+2\})$.

Lemma 2 *The conflict graph for the family of line networks with two-hop overhearing (see Fig. 3.3) is perfect.*

Proof A class of well-know perfect graphs are the so called interval graphs [44, Proposition 5.1.16]. Interval graphs are those graphs for which there exists a mapping from graph vertices to intervals on the real line such that vertices are adjacent if and only if the corresponding intervals intersect. Let $I_{(i,J)}$ denote the interval corresponding to the conflict graph node (i, J), and consider the following mapping

$$I_{(i,i+1)} = (i, i+2), \tag{3.24}$$

$$I_{(i,\{i+1,i+2\})} = (i, i+3). \tag{3.25}$$

Note that interval $I_{(i,i+1)}$ intersects with $I_{(i-1,i)}$, $I_{(i+1,i+2)}$, $I_{(i-2,\{i-1,i\})}$, $I_{(i-1,\{i,i+1\})}$, $I_{(i,\{i+1,i+2\})}$, and $I_{(i+1,\{i+2,i+3\})}$, and that those are also precisely the adjacency relationships in the conflict graph. Similarly, we can verify that an interval $I_{(i,i+1,i+2)}$ also intersects precicely with those intervals whose corresponding vertices are adjacent. $\square$

Since the conflict graph is perfect, we can optimize over the stable set polytope in polynomial time and as a consequence problem (3.12)-(3.15) is also solvable in polynomial time. Therefore, the joint optimization of subgraph and underlying schedule is efficiently tractable for this family of networks.

The perfection of the conflict graph has a significant implication beyond the polynomial time solvability of the associated optimization problem. It means that we have a succinct description of the scheduling constraints in the form of the clique inequalities (3.21). Furthermore, these constraints are local, in the sense that each node in the conflict graph only needs to know the subgraph induced by its one-hop neighborhood in order to determine the cliques that it belongs to. Such information is usually required anyway in mechanisms that use opportunistic listening, such as [45].

Unfortunately, graph perfection is a rare property and for general networks it is not likely that the resulting conflict graph is perfect. We cannot depend on graph perfection, unless we restrict our attention to special topologies like the above family of line networks. Our goals is to develop algorithms that are relevant in practice. Therefore, we do *not* want to make assumptions on the network structure - it is given to us resulting from some application. We propose an efficient relaxation of the problem in the next section.

3.3 Decentralized Algorithm

We proceed to develop a distributed algorithm that works on arbitrary network topologies. The optimization problem (3.12)-(3.15), as it stands, has to be solved in a centralized fashion. Even

Figure 3.2 A line network with two-hop overhearing.

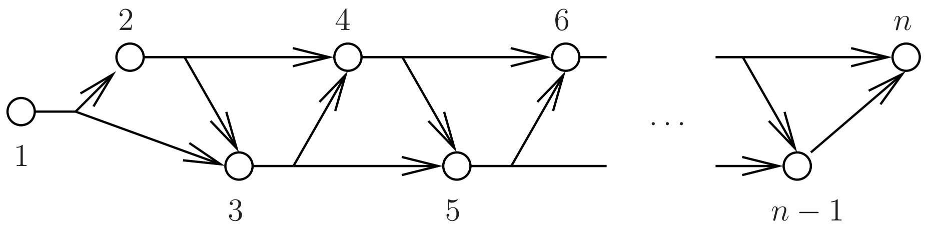
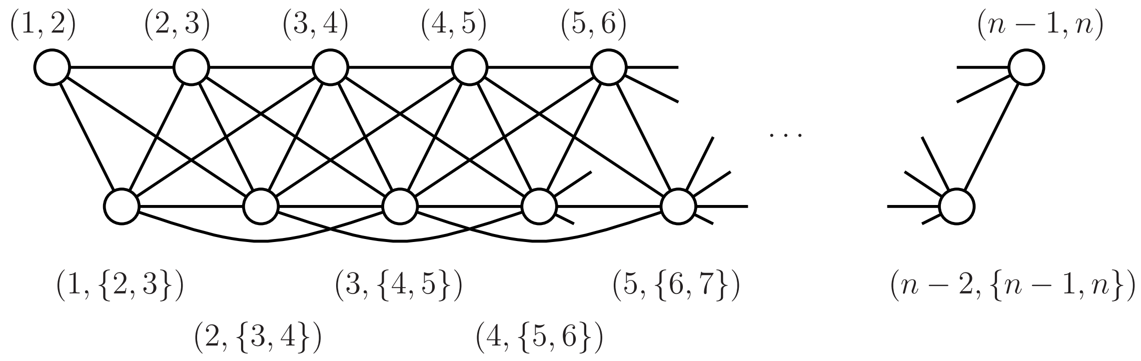


Figure 3.3 The conflict graph of the network in Fig. 3.2.



this is NP-hard in the general case, owing to the combinatorial difficulty encapsulated in P_{STAB}. Our approach is, in short, to relax this constraint and use instead a greedy heuristic for finding stable sets. This yields, as we show, a fully decentralized algorithm.

3.3.1 Subgradient Optimization on the Dual

Consider the Lagrangian dual of problem (3.12)-(3.15), where the capacity constraints (3.12) have been assigned multipliers $\boldsymbol{\lambda} = (\lambda_{iK}^{(t)})$ and moved to the objective function

$$q(\boldsymbol{\lambda}) \quad = \max_{R,\mathbf{x},\mathbf{z}} L(R,\mathbf{x},\mathbf{z},\boldsymbol{\lambda})$$

$$\text{subject to} \qquad (R,\mathbf{x}) \in P_F, \tag{3.26}$$

$$\mathbf{z} \in P_{STAB},, \tag{3.27}$$

where $\mathbf{x} = (x_{ij}^{(t)})$, the flow polytope P_F is defined by constraints (3.13) and (3.14), and the Lagrangian is given by

$$L(R, \mathbf{x}, \mathbf{z}, \boldsymbol{\lambda}) =$$
$$R + \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{N}} \sum_{K\subset N(i)} \lambda_{iK}^{(t)} \left(\sum_{J\subset N(i)} z_{iJ} b_{iJK} - \sum_{j\in K} x_{ij}^{(t)} \right). \tag{3.28}$$

The dual function decomposes into two subproblems, coupled by the Lagrangian multipliers $\lambda_{iK}^{(t)}$, as follows

$$q(\boldsymbol{\lambda}) = \underbrace{\max_{\substack{(R,\mathbf{x})\in P_F \\ R\le 1}} \left(R - \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{N}} \sum_{K\subset N(i)} \lambda_{iK}^{(t)} \sum_{j\in K} x_{ij}^{(t)} \right)}_{\text{Subproblem 1: subgraph optimization}} +$$
$$\underbrace{\max_{\mathbf{z}\in P_{STAB}} \left(\sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{N}} \sum_{K\subset N(i)} \lambda_{iK}^{(t)} \sum_{J\subset N(i)} z_{iJ} b_{iJK} \right)}_{\text{Subproblem 2: scheduling}}. \tag{3.29}$$

Note that in subproblem 1 we have added the constraint $R \le 1$, which is redundant in the primal problem (3.12)-(3.15). To see this, consider the capacity constraints (3.12) on the flow out of the source s. Since the b_{sJK} are, by definition, less than or equal to one and the z_{sJ} sum up to at most one, the rate R cannot be larger than one. We have chosen to add this constraint, in order to avoid dealing with possibly unbounded solutions to subproblem 1. In [1], the authors do not consider scheduling and therefore the scheduling subproblem collapses. In contrast, the performance of our approach critically depends on finding efficient solutions to this subproblem.

To solve the dual problem

$$\min q(\boldsymbol{\lambda})$$
$$\text{subject to} \qquad \boldsymbol{\lambda} \ge 0, \tag{3.30}$$

we apply the projected subgradient algorithm with update rule

$$\boldsymbol{\lambda}[n+1] = \max\left(\boldsymbol{\lambda}[n] - \theta[n]\boldsymbol{\xi}[n],\ 0\right), \tag{3.31}$$

where $\theta[n]$ is a suitable stepsize and $\boldsymbol{\xi}[n] = (\xi_{iK}^{(t)}[n])$ is a subgradient at step n, which we take to be

$$\xi_{iK}^{(t)}[n] = \sum_{J \subset N(i)} \hat{z}_{iJ}[n] b_{iJK} - \sum_{j \in K} \hat{x}_{ij}^{(t)}[n]. \tag{3.32}$$

Here, $\hat{\mathbf{x}}[n]$ and $\hat{\mathbf{z}}[n]$ are the solutions of subproblems 1 and 2, respectively, at step n. We discuss in the next subsection how to obtain solutions to these subproblems in a decentralized way. The subgradient computation (3.32) and the update step (3.31) can be carried out at each node individually since each involves only variables that are associated with a single node.[3]

Subgradient optimization yields iterates $\hat{\mathbf{x}}[n]$ and $\hat{\mathbf{z}}[n]$ that might not be optimal. A well-known technique that yields the primal optimal solution is called primal recovery [46]. The approach takes a weighted average over the sequence of primal solutions. The weights are nonnegative and sum up to one. Constant weights are a simple choice, giving

$$R^* = \frac{1}{N}\sum_{n=1}^{N} \hat{R}[n], \tag{3.33}$$

$$\mathbf{x}^* = \frac{1}{N}\sum_{n=1}^{N} \hat{\mathbf{x}}[n], \tag{3.34}$$

$$\mathbf{z}^* = \frac{1}{N}\sum_{n=1}^{N} \hat{\mathbf{z}}[n]. \tag{3.35}$$

A variety of alternatives appear in the literature. See, for example [1].

If this averaging rule is combined with a subgradient stepsize of the form $\theta[n] = \frac{a}{b+n}$, with $a > 0$, and $b \geq 0$, primal recovery converges to the primal optimal solution [46]. Note that, this recovery rule does not require any additional message exchange, owing to the fact that all

[3]If the step size is agreed upon in advance.

intermediate optimizers are available at the node performing the averaging. (See [47] for a similar approach.)

3.3.2 Solving Subproblems 1 and 2

Rearranging the first part of (3.29), we arrive at

$$\max_{\substack{(R,\mathbf{x})\in P_F \\ R\leq 1}} \left(R - \sum_{t\in\mathcal{T}} \left(\sum_{i\in\mathcal{N}} \sum_{j\in N(i)} x_{ij}^{(t)} p_{ij}^{(t)} \right) \right), \tag{3.36}$$

where we define $p_{ij}^{(t)} = \sum_{K\subset N(i), j\in K} \lambda_{iK}^{(t)}$. Note that $p_{ij}^{(t)}$ is non-negative. Solving (3.36) is equivalent to finding for each sink $t \in \mathcal{T}$, the shortest path with respect to the “lengths” $p_{ij}^{(t)}$ from the source s to sink t. To see this, assume that we have found these paths and have computed the sum of their lengths. If this sum is less than 1, then we achieve the maximum in problem (3.36) by sending flow of rate 1 along each of these paths. On the other hand, if the sum of these lengths is greater than or equal to 1, then sending any flow with a positive rate R leads to a negative cost in (3.36). We therefore achieve a maximal value of zero by setting R and $\mathbf{x}$ to zero. To find these $|\mathcal{T}|$ shortest paths, we can use, for example, the asynchronous distributed Bellman-Ford algorithm [32, Section 5].

We consider now subproblem 2, which can be rewritten as

$$\max_{\mathbf{z}\in P_{STAB}} \sum_{i\in\mathcal{N}} \sum_{J\subset N(i)} z_{iJ} w_{iJ} \tag{3.37}$$

with weights $w_{iJ} = \sum_{K\subset N(i)} \left(b_{iJK} \sum_{t\in\mathcal{T}} \lambda_{iK}^{(t)} \right)$ for each node. This is a standard maximum weight stable set (MWSS) problem, which is NP-hard [48]. We suggest relaxing the *maximum*-WSS constraint and instead find a *maximal* stable set, i.e. a stable set to which no vertex can be added. This problem is much simpler and can be solved in a distributed fashion using the algorithm proposed in [49]. The number of steps needed to terminate the algorithm is twice the stability number of the conflict graph [49, Theorem 1].

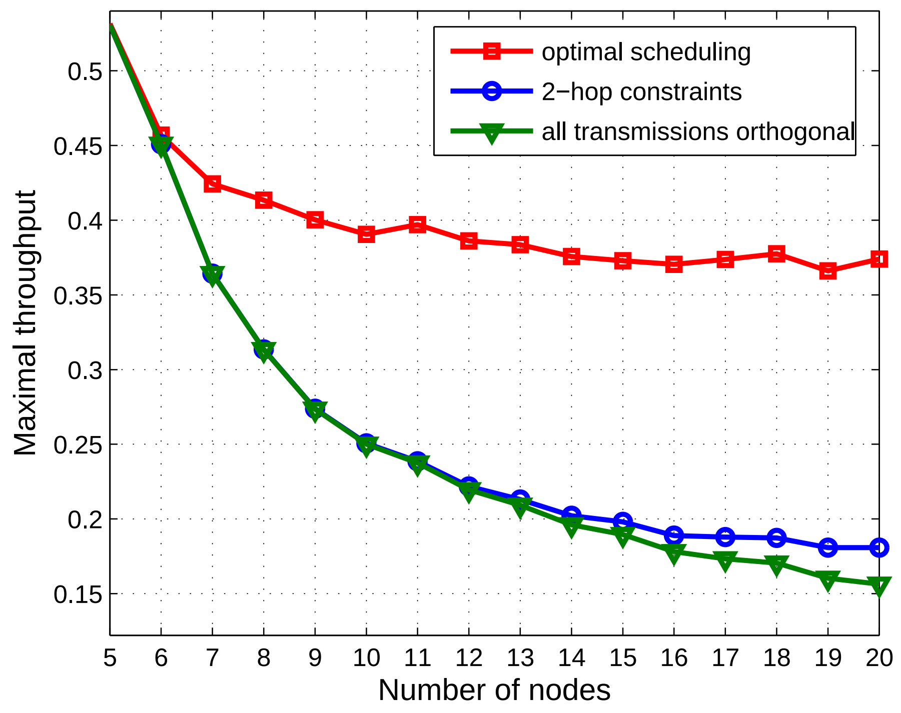
The algorithm operates on the conflict graph and not on the actual network topology. This has the consequence that two adjacent nodes in the conflict graph can represent hyperarcs that are either co-located, or at distance of one or two hops. This is also the number of hops that messages between them have to travel. In particular, if the conflict is due to simultaneous activation constraints on arcs originating at the same node, the hyperarcs are co-located, if the conflict is due to the half-duplex constraint, they are one hop apart and, where the conflict is due to interference, they are at a distance of two hops.

3.4 Simulation Results

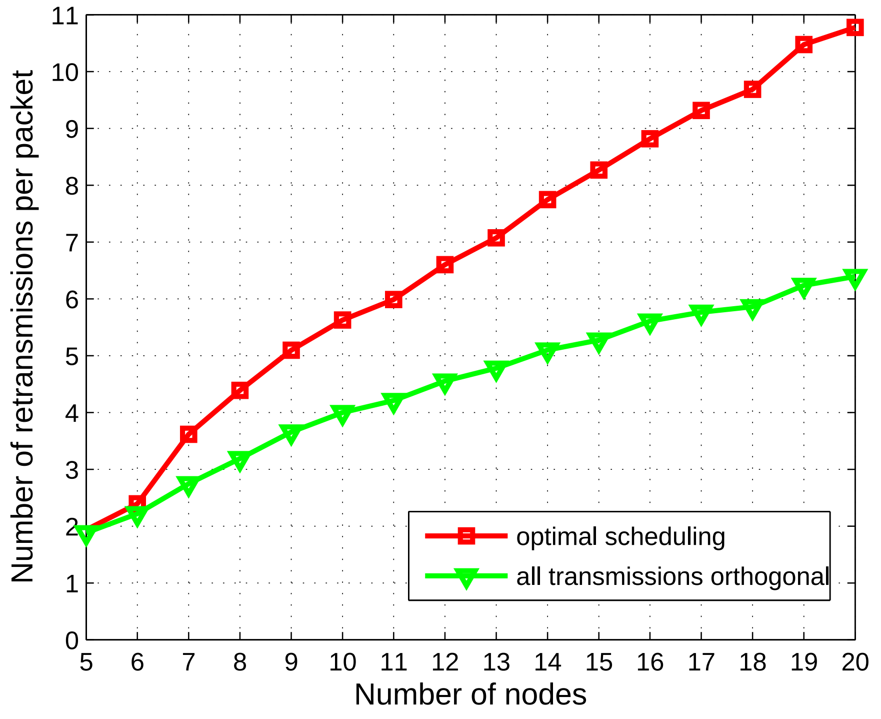
To evaluate the performance of our techniques we conduct simulations over random network topologies. The setup remains the same throughout, the only parameter that changes is the number of nodes in the network; when we add more nodes to the network, we keep the density constant. For each random instance, we assume that a number of nodes are uniformly scattered over a square region with unit node density. Two nodes are in radio range if their distance is below a certain threshold, the radius of connectivity, which we take to be 1.8. The number of neighbors of a node is restricted to 5. We consider the leftmost node to be the sender, multicasting to two receivers, the two rightmost nodes. Transmissions are subject to erasures, which may be due to distance attenuation or fading. When a node transmits, a neighbor at distance d will receive the packet correctly if $\Gamma d^{-2} \geq \beta$ where Γ is a unit mean exponential variable and $\beta = \frac{1}{4}$ is our chosen SNR threshold. Otherwise the packet is lost completely. We assume secondary interference constraints as well as half-duplex transceivers.

In Fig. 3.4(a), we compare the throughput of optimal scheduling, i.e. solving problem (3.12)-(3.15) optimally, with two commonly used scheduling techniques. In the fully orthogonal model [1], all nodes in the network are assigned orthogonal channels, making the network interference-free. In the two-hop constraints model (see e.g. [35] for such a scheduling protocol)

Figure 3.4 Throughput and power consumption for different scheduling techniques.



(a) Maximum rate of a multicast connection with one sender and two receivers as a function of the number of network nodes.



(b) Average number of retransmissions per packet $\frac{\sum_{iJ} z_{iJ}}{R_{\max}}$ for the maximal achievable rate of optimal scheduling as compared to orthogonal scheduling.

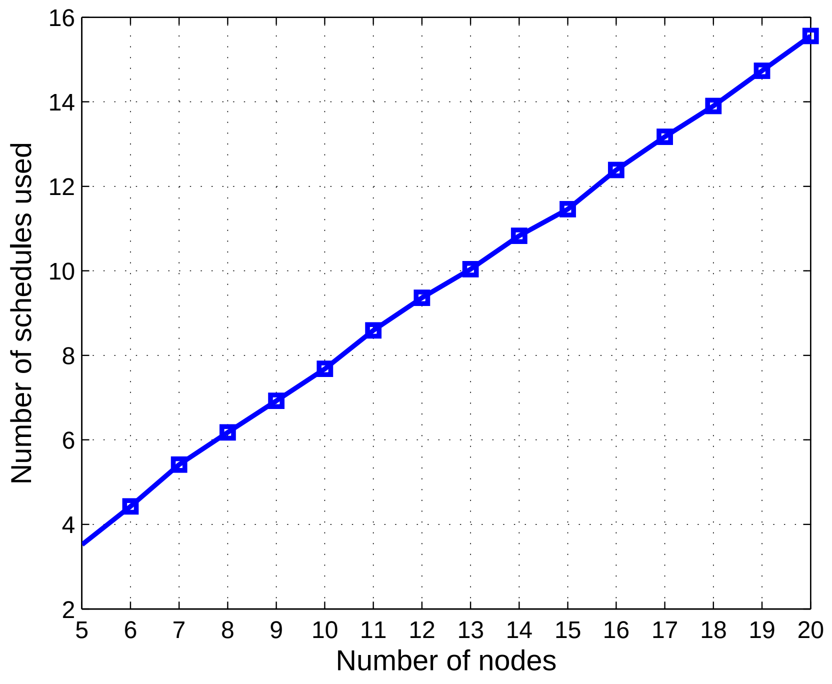
transmissions are scheduled such that, if node i transmits, all nodes in a two-hop neighborhood are silent, eliminating the possibility of a node being in radio range of two simultaneous transmissions. Both the orthogonal and the two-hop constraint model eliminate interference at the expense of suboptimal bandwidth reuse. We see that this is apparently a rather wasteful way of operating an interference limited wireless network. Furthermore, for the networks of moderate size that we consider, the two-hop constraint is almost as restrictive as full orthogonalization. This can be seen in Fig. 3.4(a), where the corresponding curves almost match. Apparently, for the networks that we consider very few nodes are more than two hops apart. Therefore, in most cases only one node at a time can transmit under the two-hop constraint. On the other hand, a significant increase in bandwidth efficiency is possible if we use optimal scheduling.

We investigate the trade-off between bandwidth and power efficiency in Fig. 3.4(b), where we plot the average number of retransmissions per packet for both orthogonal scheduling and optimal scheduling. The number of retransmissions serves as an estimate for the total power expenditure. Both curves show the maximal rate. More precisely, if $R_{\max}$ is the maximal rate for a fixed policy, and z_{iJ} are the injection rates computed for this rate, then the average number of retransmissions per packet is $\frac{\sum_{iJ} z_{iJ}}{R_{\max}}$. It shows how much additional power expenditure is required to obtain the throughput gains in Fig. 3.4(a). Since we permit some simultaneous transmissions rather than eliminating all collisions, we obtain a substantial increase in throughput and bandwidth efficiency. The price we pay is a higher power expenditure due to packets that collide and are therefore lost. We see that optimal scheduling apparently does not lead to an excessive number of collisions or retransmissions.

Fig. 3.5 shows the average number of network configurations comprising the solution of the optimal scheduling algorithm. This is an important measure, as time-sharing over many network configurations leads to higher delays as well as complexity of operation.

The throughput of the decentralized algorithm from Section 3.3, which we call the “decentralized greedy” algorithm, is shown in Fig. 3.6. For comparison, we have plotted orthogonal

Figure 3.5 The number of configurations which appear in the solution of the optimal scheduling algorithm. We consider schedules with a time sharing coefficient equal to or greater than 0.001.

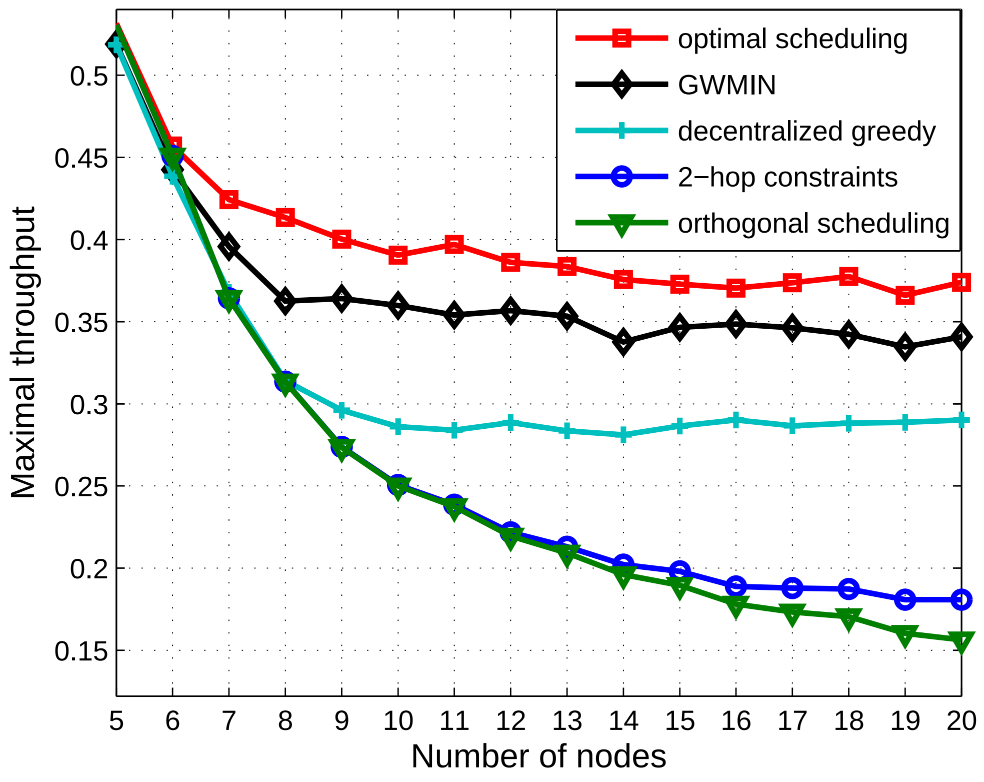


scheduling, two-hop constraint scheduling, and optimal scheduling, the latter for network sizes that are still tractable. In addition, we consider a variation of the decentralized algorithm, which we refer to as GWMIN [50], that differs in how the weights for the stable set computation are chosen. In particular, GWMIN takes weights w_v^*, which are derived from the original weights w_v as follows:

$$w_v^* = \frac{w_v}{|N_{\mathcal{G}}^*(v)| + 1}, \tag{3.38}$$

where $|N_{\mathcal{G}}^*(v)|$ is the *current* number of neighbors of node v in the conflict graph; as nodes leave the conflict graph $|N_{\mathcal{G}}^*(v)|$ decreases. This heuristic was proposed in [50], and performs somewhat better than the decentralized algorithm without weight adjustment. Note, however, that it comes with a higher communication overhead. Every time a node leaves the conflict graph, its adjacent edges are removed and, therefore, the degree of the neighbors changes. After the neighbors have updated their weight, they have to communicate the new weight to

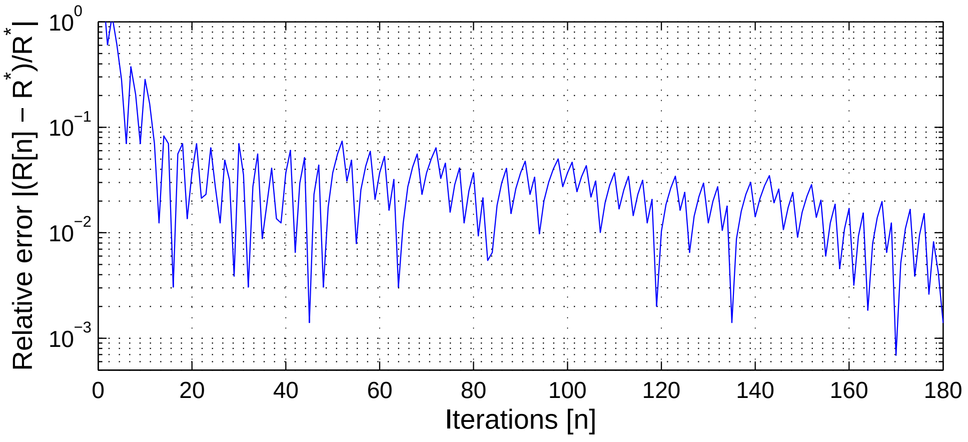
Figure 3.6 Maximum throughput of the distributed algorithms as a function of the number of nodes in the network.



their neighbors before the algorithm can resume. Both algorithms, the decentralized greedy algorithm as well as GWMIN, significantly outperform orthogonal or two-hop scheduling.

In Fig. 3.7 we illustrate the convergence of the subgradient optimization for a network with 10 nodes when the stable set problem is computed optimally in each step. Note that the scale is logarithmic, and therefore the deviation from the optimum is not large after a moderate amount of iterations.

Figure 3.7 Convergence of subgradient optimization for a network with 10 nodes. We report the difference between the optimal throughput R^*, and the throughput after primal recovery at step n.

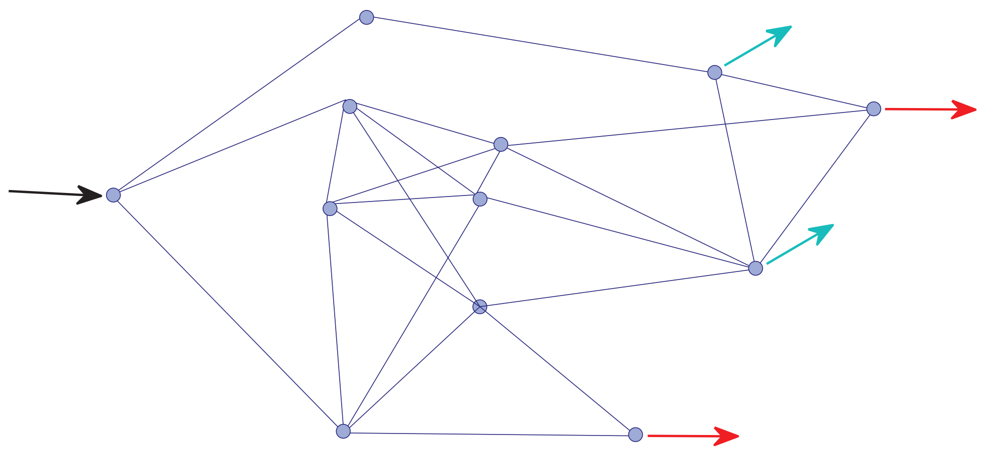


3.5 Competition for Resources under Interference Conditions

Under network coding, the maintenance of min-cut conditions between the set of senders and every receiver individually is a sufficient condition to support the connection. Thus, for a network without interference such as a wireline network, different receivers in a network coded multicast connection do not compete with each other for resources; each receiver with a sufficiently large min-cut to the source can participate. In wireless networks, however, interference from simultaneous transmissions has to be taken into account. Such interference in effect changes the underlying network and thus can create interactions among receivers involved in a single multicast session.

To consider the interaction among users, let $\mathcal{T}_1$ denote one group of receivers, and $\mathcal{T}_2$ another set of receivers. First, consider orthogonal scheduling and the multicast connections $(s, \mathcal{T}_1, R_1^*)$, $(s, \mathcal{T}_2, R_2^*)$, and $(s, \mathcal{T}_1 \cup \mathcal{T}_2, R_0^*)$, where for each connection the rate is taken to be the maximal rate that can be supported by the network if this particular connection is present alone. We do not consider coding across the two multicast sessions (also referred to as *inter-session* network

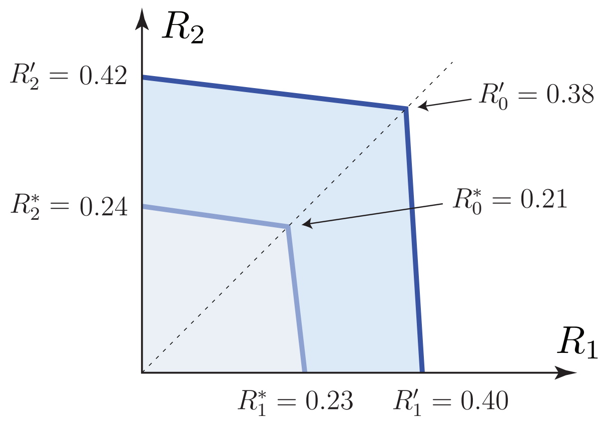
Figure 3.8 A wireless network with 12 nodes, one source, and two multicast groups, each consisting of two nodes.



coding). Inter-session coding may be required in the optimal solution, but we exclude it due to the inherent hardness of the problem. We have plotted the rate points $(R_1^*, 0)$, $(0, R_2^*)$, and (R_0^*, R_0^*) in Fig. 3.9. Note, that at the point (R_0^*, R_0^*), we transmit *the same information* to both multicast groups at rate R_0. The effect that users have on each other through interference is captured by the fact that R_0^* is typically strictly lower than either R_1^* or R_2^*. If the underlying subgraph were not modified through the effect of interference, the minimum cut from the source to the users in $\mathcal{T}_1 \cup \mathcal{T}_2$ would be $\min(R_1^*, R_2^*)$.

Since the realization of the network depends on the schedule, we expect different choices of scheduling to change the rate regions. Indeed, in Fig. 3.9, we show two different rate regions, corresponding to different scheduling policies. For the rate region indexed by * the underlying schedule is assumed to be orthogonal. Note that in Fig. 3.9, we have plotted the “time-sharing region” between the points $(R_1^*, 0)$, $(0, R_2^*)$, and (R_0^*, R_0^*), which is in general not the full rate region, even without inter-session coding. We compare this with our optimal approach, i.e. solving problem (3.12)-(3.15) optimally, (the region indexed by $'$). The resulting rate region significantly expands the rate region obtained by orthogonal scheduling. Another interesting observation is that the rate regions are almost rectangular. That means that a 4-multicast with

Figure 3.9 Maximal rate to two groups of multicast receivers and to their union. The average is over 100 random wireless networks. The networks have 12 nodes and two groups, both consisting of two receivers. An example of such a random network is shown in Fig. 3.8.



random terminals can support almost the same rate as a 2-multicast and suggests that, in our example networks, network coded multicasting scales well with the number of receivers.

3.6 Discussion

We have seen that addressing network coding subgraph optimization and scheduling jointly results in significant performance gains. Scheduling to create network coding opportunities improves the throughput by over a factor of two. Moreover, we have proposed a way to distribute the computation across the network. To arrive at a distributed solution, we used a heuristic, greedy stable set search in place of the full optimization over the stable set polytope. The performance of distributed scheduling is empirically evaluated to be not far from the optimum.

Nevertheless, a central assumption is that in every time slot the updates at all nodes are carried out simultaneously, as if triggered by a central clock signal. In the next chapter, we investigate the consequences of relaxing this restriction and allowing the updates to be carried out *asynchronously*. It turns out that this is possible, although it leads to other constraints on the problem. Unfortunately, these constraints are fundamental and not merely a side effect of the particular formulation. In consequence, it is the network designer's decision to either choose asynchronous operation, or to ensure enough synchronization in the network.

4

Asynchronous Network Coded Multicast

The distributed computation of network connections is highly desirable in practice, as otherwise information about the entire network topology has to be collected at a special central instance which carries out the computation. Once computed, the link and injection rates have to be communicated across the network, leading to significant delays and overhead. If the network changes over time, these rates may be outdated when they reach their destination nodes. Even assuming distributed operation, one main assumption is that in every time slot the updates at all nodes are carried out simultaneously, i.e. the algorithm iterates in synchronous rounds [1, 51]. The contribution of this chapter is to relax this assumption and instead propose an *asynchronous* algorithm for solving the problem. As we show, our approach requires very few restrictions on the update schedule; even a random sequence of node updates will converge as long as each node is chosen with non-zero probability.

Distributed asynchronous algorithms have been proposed and used for routing. The most prominent example is the distributed Bellman-Ford algorithm, which has both a synchronous and an asynchronous variant [32, Section 5.2.4]. This is, to the best of our knowledge, the first asynchronous approach for network coded traffic and the particular optimization problem associated with it [1]. The asynchronous algorithm proposed in [52] addresses the problem of network code design and is orthogonal to our approach, due to the aforementioned separation between coding and resource provision. We are concerned with providing and allocating a minimal set of network resources that guarantees a certain min-cut to all receivers.

The motivations for seeking asynchronous solutions are two-fold. Firstly, in large networks the assumption of having a clock that is available at all nodes is unrealistic or requires a significant amount of communication overhead. The fundamental limits of clock synchronization across a network are discussed in [53] and the references therein. Secondly, network transmissions often have a non-negligible loss or error rate. When the algorithm requires synchronous rounds of updates, such losses of messages can seriously impair convergence. An asynchronous algorithm, such as the one we suggest, can easily deal with lost update messages due to the minimal requirements it poses on the update schedule.

The main idea of our work is to apply a *block-coordinate ascent algorithm* to the dual of the optimization problem that describes the multicast connection. If we want a distributed and asynchronous algorithm, this implies that we can only hope to update one variable block at a time, similar to Gauss-Seidel-type algorithms for solving systems of linear equations [54, Section 1.2]. For such a block-coordinate method to work, it is well-known that we need to impose one major requirement on the function we wish maximize: it has to be *differentiable* [55, Section 2.7]. For the dual objective function of a convex problem to be differentiable, we need the primal to be *strictly* convex [55, Section 6.2]. For this, in turn, we will have to make a few modifications to the optimization problem that we formulated in the previous hapter. There, we formed a linear program and applied the subgradient method - which can handle nondifferentiable functions - to its Lagrangian dual. In this context, however, this is not possible, as we need a strictly convex program to start with. Note that multicast problems with a convex objective function were addressed in [1] and solved with a *primal-dual algorithm*, which is distributed but not asynchronous.

The algorithm we propose, as we show by means of simulations, exhibits fast convergence compared to the primal-dual algorithm in [1] and is very robust with respect to randomly occurring node updates. Even in the presence of link failures, the algorithm continues updating and eventually converges without the need for a restart. It can, therefore, run continuously in the background and automatically adapt to changes in the network.

4.1 Network Model and Optimization Problem

Our starting point is optimization problem (3.1)-(3.3). However, based on the previous discussion we impose the following modifications:

- The objective is not to maximize the rate, but to minimize a strictly convex cost function that penalizes high injection rates.
- We assume the schedule is fixed, i.e. a non-conflicting collection of hyperarcs is given to us. This means dropping the stable set polytope constraint (3.4).
- To simplify notation, we will replace the polymatroidal constraints (3.1) with simple capacity constraints on the arcs, i.e. the flow on arc (i, j) has to lie between 0 and the capacity c_{ij}. In Appendix A.3, we show that the entire analysis remains valid for polymatroidal constraints, the only drawback being a more complicated notation.

With the modification of replacing the polymatroidal constraints by capacity constraints, the hypergraph essentially can be modelled as a simple directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of vertices and $\mathcal{A}$ is the set of directed arcs. Consider the following linear progam for setting up a multicast connection $(s, \mathcal{T}, R)$

$$\text{minimize} \sum_{(i,j)\in\mathcal{A}} f_{ij}(z_{ij})$$

subject to

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(t)} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{(t)} = \sigma_i^{(t)}, \quad \forall i \in \mathcal{N}, t \in \mathcal{T}, \tag{4.1}$$

$$0 \leq x_{ij}^{(t)} \leq z_{ij}, \qquad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}, \tag{4.2}$$

$$z_{ij} \leq c_{ij}, \qquad \forall (i,j) \in \mathcal{A}, \tag{4.3}$$

where we define

$$\sigma_i^{(t)} = \begin{cases} R & i = s \\ -R & i = t \\ 0 & \text{else}. \end{cases} \tag{4.4}$$

We assume the cost functions $f_{ij}(\cdot)$ to be monotonically increasing and *strictly* convex throughout. Also, let $T = |\mathcal{T}|$.

An instance of the problem is described by the network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, the link capacities c_{ij}, the link cost functions f_{ij}, and the multicast session $(s, \mathcal{T}, R)$.

4.2 Decentralized Asynchronous Algorithm

4.2.1 Block-Coordinate Ascent on the Dual

In this section, we apply a block-coordinate ascent algorithm to the dual of convex program (4.1)-(4.3), resulting in decentralized and asynchronous operation. As the f_{ij} are monotonically increasing functions, constraint (4.2) essentially means that $z_{ij} = \max_{t\in\mathcal{T}} x_{ij}^{(t)}$. The $\max$ function is not differentiable everywhere and thus poses a challenge for gradient-type optimization algorithms. One approach is to replace this relation with a *soft*-maximum that is differentiable and that approximates the maximum function. Two common approximations are the *log-sum-exp* function [56] and the l_p-norm [57], given by

$$z'_{ij} = L\log\left(\sum_{t\in\mathcal{T}} \exp\left(x_{ij}^{(t)}/L\right)\right) \tag{4.5}$$

and

$$z''_{ij} = \left(\sum_{t\in\mathcal{T}} \left(x_{ij}^{(t)}\right)^p\right)^{1/p}, \tag{4.6}$$

respectively. Both functions converge to the maximum function, for $L \to 0$ and for $p \to \infty$, respectively. Although they are convex and differentiable, they are not *strictly* convex. This can be seen by setting $x_{ij}^{(t)} = x_{ij}^{(1)}, \quad \forall t \in \mathcal{T}$. For the *log-sum-exp* function, this leads to $z'_{ij} = L\log T + x_{ij}^{(1)}$, which is linear and not strictly convex. Replacing z_{ij} with z'_{ij}, we can define a modified cost function F_{ij} according to

$$F_{ij}(\boldsymbol{x_{ij}}) = f_{ij}\left(L\log\left(\sum_{t\in\mathcal{T}} \exp\left(x_{ij}^{(t)}/L\right)\right)\right), \tag{4.7}$$

where $\boldsymbol{x_{ij}} = \left(x_{ij}^{(1)}, ..., x_{ij}^{(T)}\right)$. Here, $F_{ij}(\boldsymbol{x_{ij}})$ is (strictly) convex if f_{ij} is (strictly) convex and monotonically increasing. With this transformation, the problem is reformulated as a standard convex multi-commodity flow problem, where the flows are coupled only via the cost function and the constraints are separable [58, Section 8.3]. The primal optimization problem takes the form

$$\text{minimize} \sum_{(i,j)\in\mathcal{A}} F_{ij}(\boldsymbol{x_{ij}})$$

subject to

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(t)} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{(t)} = \sigma_i^{(t)}, \qquad \forall i \in \mathcal{N}, t \in \mathcal{T}, \tag{4.8}$$

$$0 \le x_{ij}^{(t)} \le c_{ij}, \qquad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}. \tag{4.9}$$

Introducing a Lagrange multiplier $p_i^{(t)}$ for every constraint in (4.8), we form the Lagrangian

$$\begin{aligned} L(\boldsymbol{x},\boldsymbol{p}) = & \sum_{(i,j)\in\mathcal{A}} F_{ij}(\boldsymbol{x_{ij}}) + \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}} p_i^{(t)} \left(\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(t)} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{(t)} - \sigma_i^{(t)} \right) & (4.10) \\ = & \sum_{(i,j)\in\mathcal{A}} \left(F_{ij}(\boldsymbol{x_{ij}}) + \sum_{t\in\mathcal{T}} \left(x_{ij}^{(t)} \left(p_i^{(t)} - p_j^{(t)} \right) \right) \right) - \sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}} p_i^{(t)} \sigma_i^{(t)}. & (4.11) \end{aligned}$$

Note that the capacity constraints (4.9) are not dualized but kept explicitly. The dual function value $q(\boldsymbol{p})$ at a price vector $\boldsymbol{p}$ is

$$q(\boldsymbol{p}) = \sum_{(i,j)\in\mathcal{A}} g_{ij}(\boldsymbol{p_i} - \boldsymbol{p_j}) - \sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}} p_i^{(t)} \sigma_i^{(t)},$$

where g_{ij} is defined as

$$g_{ij}(\boldsymbol{p_i} - \boldsymbol{p_j}) = \inf_{0 \le \boldsymbol{x_{ij}} \le c_{ij}} \left\{ F_{ij}(\boldsymbol{x_{ij}}) + \sum_{t \in \mathcal{T}} \left(x_{ij}^{(t)} \left(p_i^{(t)} - p_j^{(t)} \right) \right) \right\}, \tag{4.12}$$

and $\boldsymbol{p_i} = \left(p_i^{(1)}, ..., p_i^{(T)} \right)$. The solution of the dual unconstrained optimization problem

$$\underset{\boldsymbol{p}}{\text{maximize}} \;\; q(\boldsymbol{p})$$

is equivalent to the solution of the primal problem as under our assumptions there is no duality gap. We suggest solving the dual by a block-coordinate ascent method. To that end, consider the $|\mathcal{N}|$ variable blocks $\boldsymbol{p_i}$. At the k-th iteration, we select a block $\boldsymbol{p_i}$ and update it in an ascent direction. We defer the discussion of how to select blocks in order to achieve convergence to the end of the section. We take as an ascent direction the gradient $\boldsymbol{\nabla_i}$ with respect to $\boldsymbol{p_i}$. For an appropriately chosen step size θ_k, the update takes the form

$$\boldsymbol{p_i}[k+1] := \boldsymbol{p_i}[k] + \theta_k \boldsymbol{\nabla_i}[k]. \tag{4.13}$$

4.2.2 Computing the Gradient

We can compute the gradient with the following Lemma

Lemma 3 *[55, Proposition 6.1.1] Let $\boldsymbol{x}(\boldsymbol{p})$ be the unique minimizer of the Lagrangian at a price vector $\boldsymbol{p}$, i.e.*

$$\boldsymbol{x}(\boldsymbol{p}) = \underset{\boldsymbol{x}}{\operatorname{argmin}}\, L(\boldsymbol{x}, \boldsymbol{p}). \tag{4.14}$$

Then, the dual function $q(\boldsymbol{p})$ is everywhere continuously differentiable, and its derivative with respect to $p_i^{(t)}$ is given by the constraint function evaluated at $\boldsymbol{x}(\boldsymbol{p})$

$$\frac{\partial q}{\partial p_i^{(t)}} = \sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(t)}(\boldsymbol{p}) - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{(t)}(\boldsymbol{p}) - \sigma_i^{(t)}. \tag{4.15}$$

Remark 1 *In other words, the derivative is given by the flow divergence out of node i for each session t.*

Remark 2 *Linear constraints and a* strictly *convex objective function (as we have assumed throughout) imply the uniqueness of the Lagrangian minimizer and therefore the validity of the lemma. Mere convexity is not sufficient, in general.*

An update at node i takes on the following form. With every edge adjacent to i we associate a processor that solves problem (4.12) and computes the minimizer. For an edge (i,j) this becomes

$$\boldsymbol{x_{ij}}(\boldsymbol{p_i} - \boldsymbol{p_j}) = \underset{0\le \boldsymbol{x_{ij}} \le c_{ij}}{\operatorname{argmin}} \left\{ F_{ij}(\boldsymbol{x_{ij}}) + \sum_{t\in\mathcal{T}} \left(x_{ij}^{(t)} \left(p_i^{(t)} - p_j^{(t)} \right) \right) \right\}, \tag{4.16}$$

and requires the price vectors of the neighboring nodes only. As this optimization is convex, it can be solved with standard algorithms like SQP [59] or a Projected Gradient method [55]. Owing to the simple constraints on $x_{ij}^{(t)}$, the orthogonal projection can be implemented easily. Node i gathers the $x_{ij}^{(t)}(\boldsymbol{p})$ from adjacent edges to compute the gradient with respect to $\boldsymbol{p_i}$ in the following way

$$\boldsymbol{\nabla_i} = \begin{pmatrix} \sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(1)}(\boldsymbol{p}) - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{(1)}(\boldsymbol{p}) - \sigma_i^{(1)} \\ \vdots \\ \sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(T)}(\boldsymbol{p}) - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{(T)}(\boldsymbol{p}) - \sigma_i^{(T)} \end{pmatrix}. \tag{4.17}$$

Note that calculating the gradient given the corresponding $x_{ij}^{(t)}$ variables involves very little computational effort. But the $x_{ij}^{(t)}$ variables are automatically computed as a byproduct when solving problem (4.12).

4.2.3 Convergence

Having seen how to compute a gradient at each step, we return to prove convergence of the described algorithm. We need to specify the sequence in which the variable blocks are updated. Consider the following definition [54]:

Definition 7 *We speak of a partially asynchronous order if there exists a positive constant K for which every coordinate is chosen at least once for relaxation between iterations r and $r + K,\ r = 0, 1, ...$. Furthermore, at any iteration the variables used to compute the update are at most K steps old.*

Remark 3 *For the choice of $K = |\mathcal{N}|$ this leads to a cyclical update rule, while for K large it comes close to a random choice of the current block of variables.*

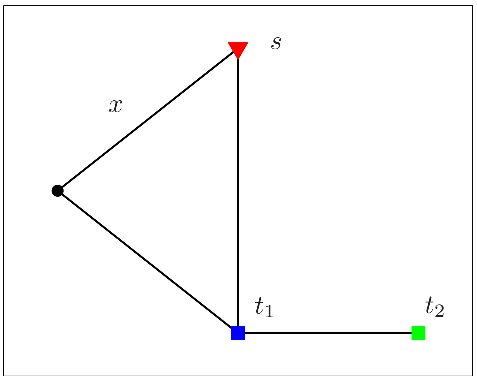
Our algorithm converges by a result from [54] which proves

Proposition 2 *[54, Proposition 5.2] Assume that the function $q(\boldsymbol{p})$ is continuously differentiable, the gradient satisfies a Lipschitz-condition, and a partially asynchronous update order is adopted. Then, block-coordinate ascent using the update rule (4.13) converges for a stepsize sufficiently small.*

Remark 4 *The algorithm converges under some more technical conditions even when an arbitrary ascent direction is used in place of the gradient. When using the gradient, these technical conditions are satisfied automatically [54, Section 7.5.3].*

The stepsize θ_k can be determined by standard line search algorithms like Backtracking [56] or the Golden Section Method [55] . Note, that (4.16) is a very low dimensional problem - its

Figure 4.1 A two sink multicast from s to t_1 and t_2.



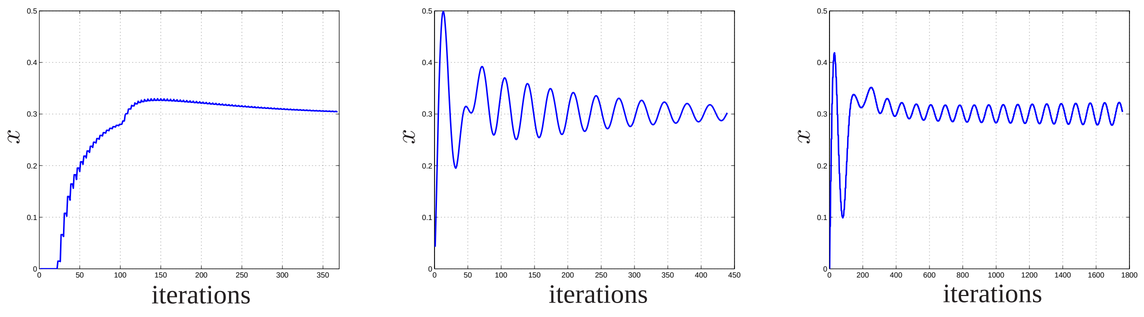
dimension is the number of multicast sinks T - and can be solved readily a number of times. The complexity of a node update scales proportionally to the complexity of (4.16), the number of adjacent edges of i and the number of steepest ascent iterations carried out.

From the algorithm description, we see that all steps, i.e. the computation of the resulting flow for a fixed price vector (4.16), the computation of the gradient (4.17) and the price update (4.13) require solely information that can be gathered in a *one-hop* neighborhood of the updating node. This gives rise to the decentralized operation of the algorithm. Moreover, if combined with an essentially cyclic update order, assuming a large constant K, we also conclude asynchronous convergence.

4.3 Performance Evaluation

To assess empirically the performance of our approach, we conduct a series of experiments. The link cost function is taken to be throughout $a_{ij}\exp(z'_{ij})$, where a_{ij} is a positive coefficient. Firstly, we illustrate a case where, owing to asynchronous updates, the primal-dual algorithm of [1] fails to converge. In contrast, our algorithm converges as promised by the theory. Consider Fig. 4.1, and the convergence curves in Fig. 4.2. The primal-dual algorithm converges correctly if applied synchronously but fails to converge if updated asynchronously (the updates are carried

Figure 4.2 Progress of the asynchronous algorithm (left), synchronous primal-dual algorithm [1] (center) and asynchronous primal-dual algorithm [1] (right). We measure convergence of the flow x, of which the optimal value is 0.3.



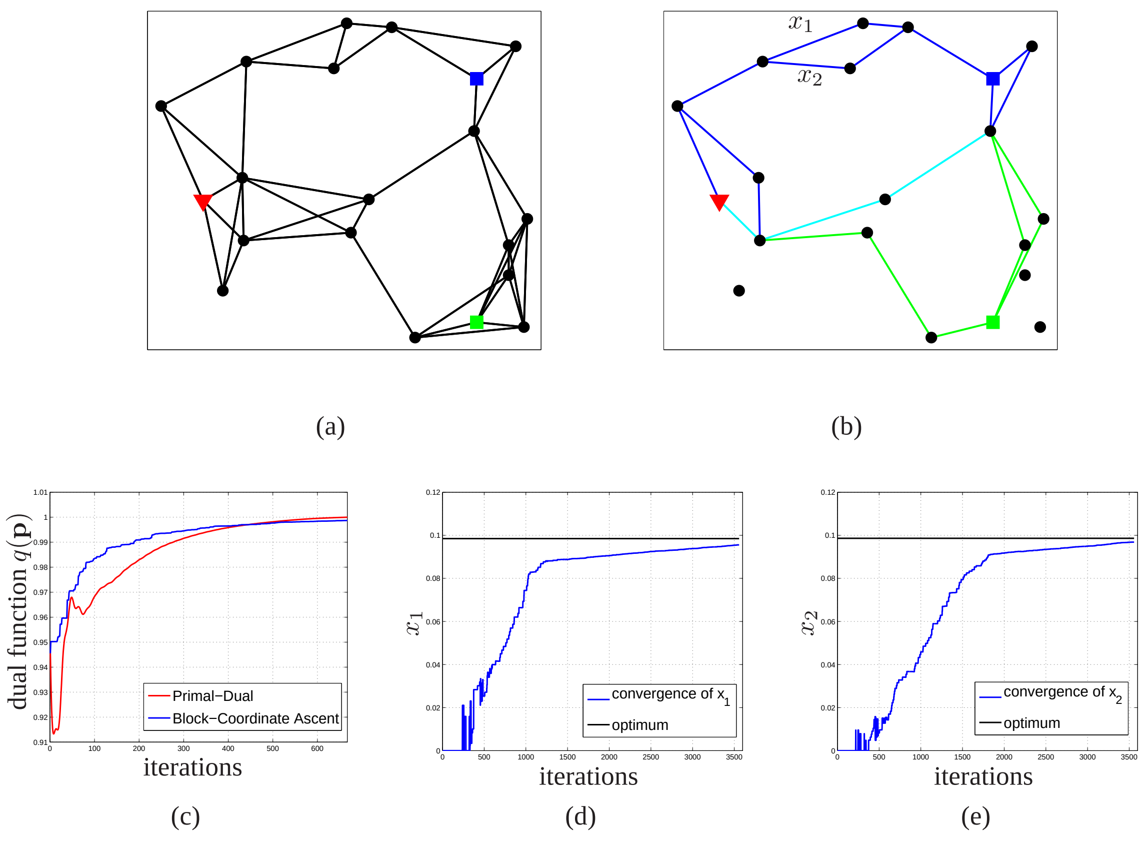
(a) (b) (c)

out in cyclical order, or $K = 4$). Another interesting observation is that coordinate ascent and synchronous primal-dual converge after a similar number of iterations; in the primal-dual, however, during an iteration *every* node in the network performs an update, as compared to just one node in the network for the coordinate ascent algorithm. This has two implications: Firstly, coordinate ascent needs less communication overhead, reducing control traffic in the network. Secondly, if for larger networks some of the updates can be carried out in parallel, this would lead to a significant speed-up.

Figures 4.3(a) - 4.3(e) illustrate the convergence of the block-coordinate ascent algorithm in a larger randomly generated unit disc graph. We see the topology in Fig. 4.3(a) and the flows after convergence in Fig. 4.3(b). In Fig. 4.3(c), we plot the value of the dual function $q(\boldsymbol{p})$ (normalized to 1) for the coordinate ascent and a random selection of updating nodes. In comparison, we plot the value of the Lagrangian for the primal-dual algorithm of [1] when updated in synchronous rounds. In Fig.s 4.3(d) and 4.3(e), we show the primal convergence for two selected flows. Note that the dual optimum is approximated much faster (convergence after about 500 iterations) than the primal (convergence after 2000-3000 iterations), a result consistent with [1, 51].

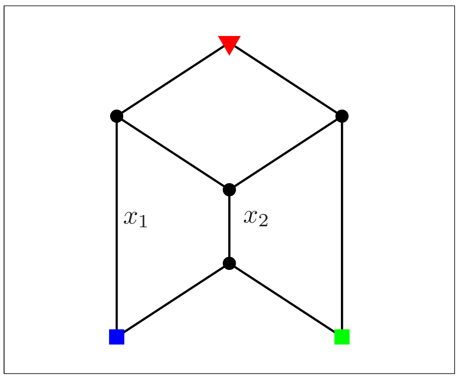
Finally, we investigate the behavior under dynamic conditions, when a link in the network

Figure 4.3 (a) Network topology, and (b) corresponding optimal flows. The source is red, the sinks are blue and green. The common flow to both sinks is cyan. Missing links specify a flow of zero. (c) Convergence of the dual function $q(\boldsymbol{p})$ of coordinate ascent (asynchronously) and primal-dual (synchronous rounds); the dual function is normalized to 1. The convergence of the link flows (d) x_1 and (e) x_2.

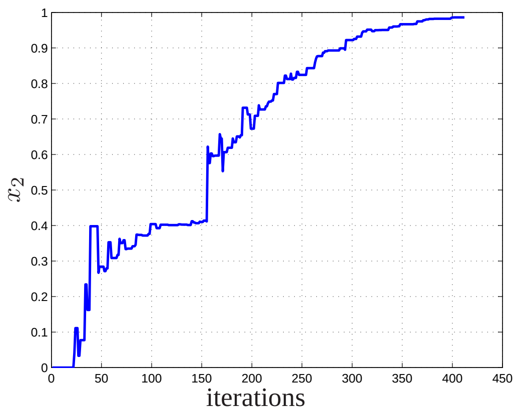


suddenly fails. Fig. 4.4 shows the reaction of the algorithm to the failure of one edge. Since the price variables $\boldsymbol{p}$ are unconstrained, every value can be used as a feasible starting point of the algorithm. Consequently, the coordinate ascent algorithm can run continuously and without needing a restart in the event of network changes. If the network topology does not change for a sufficiently long time it will converge to the new optimal value.

Figure 4.4 In the network (a), the edge carrying flow x_1 fails after 150 iterations. In the graph (b), the flow x_2 first converges to the optimal value of 0.4. After the link failure, it converges to the new optimal value of 1. Note that since the links have different convex cost functions, the link flows are not zero or one as in [2].



(a)



(b)

4.4 Discussion

We have proposed a fully decentralized and asynchronous algorithm for the minimum-cost multicast problem. The main contribution lies in relaxing the need for synchronization in the network. Moreover, simulations show fast convergence as compared to previously known approaches like the primal-dual algorithm [1] for convex problems. Since the algorithm requires no assumption on network synchronization, it adjusts well to topology changes such as link or node failures. However, in order to achieve this, we had to adopt a more restrictive network model - assuming a strictly convex problem and the schedule fixed. Many network optimization problems are inherently convex (see for instance [32] for a number of examples). If this is not the case, an option is to consider a formulation that approximates the original problem, but is strictly convex. This is very similar to what we already did - replacing the $\max$ function with the l_p norm. Of course, since we are now solving a similar but not exactly the same problem a penalty is incurred. It is the discretion of the network designer to choose whether he prefers to pay this penalty or, alternatively, to ensure network synchronization. In particular, if the cost of synchronizing the network is high an asynchronous mode may offer advantages.

5

Joint Scheduling and Instantaneously Decodable Network Coding

So far, the focus has been on coding within a multicast session, i.e. *intra*-session network coding. In this chapter, we allow data of different users to mix, leading to *inter*-session network coding. This is, in its full generality, a difficult problem [5], and in fact may even require complicated non-linear processing [11]. On the other hand, approaches to inter-session network coding that are not necessarily optimal yet are practical from an engineering point of view, have demonstrated large performance gains [7]. In a wireless network, due to broadcasting, nodes frequently overhear packets that are not intended for them. This additional "evidence" can be used to combine several packets in one transmission.

Following prior work like [7], we investigate instantaneous binary XOR coding. Each broadcast transmission from a given node is the binary sum of some number of incoming packets. A collection of L packets can be combined only if it is *instantaneously decodable* for all neighbors. This is achieved when each neighbor knows all but one of the L packets; either because it overheard them from prior transmissions or because it previously received those packets directly. Each receiver can then cancel out all but the one packet that is new to him. As we show, the instantaneous decodability condition can be formulated as a *conflict graph* model, where valid packet combinations correspond to *stable sets*. This is, in general, an NP-hard combinatorial problem, which is inherent in the instantaneous decodability condition. However, our simulations indicate that for moderate size networks the optimal solution can be within reach. Note that although the problem we address is different from the one in Chapter 3, we again use a conflict graph model for describing valid configurations. There, we have used conflict graphs do describe conflicts on simultaneous transmissions. Here they encapsulate the constraints on the network code.

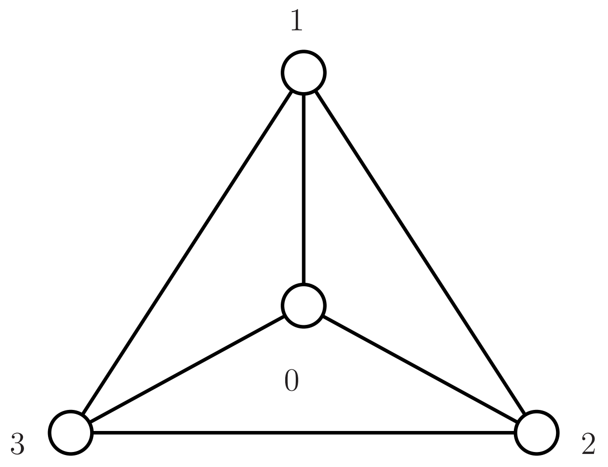
We formulate a linear program that optimizes over both the schedule and network coding decision. With this problem formulation, we are able to compute the achievable rate region of our technique and to quantify the gains over routing. However, in most practical mobile networks a low-complexity, decentralized and online algorithm is preferable. We formulate such an algorithm based on ideas from [60], where the authors derive a widely applicable class of online scheduling algorithms achieving optimal throughput. To include network coding, we introduce a system of *virtual queues* that can be served jointly subject to the constraints arising from the conflict graph model.

There are two lines of work that are related to our approach. In [7], the authors introduce COPE, an 802.11-based protocol that uses network coding to enhance the performance of the MAC-layer. There, the idea of combining packets locally, opportunistically and heuristically was developed and shown to yield significant performance gains. The decision regarding which packets to combine is made by means of a sequential (essentially greedy) search heuristic. Our goal is to optimize over the set of network coding decisions and over the schedule. In [41], the authors analyze theoretically the performance of COPE-type network coding by formulating a linear program that captures the network coding, routing and scheduling constraints. In contrast to their work, our approach optimizes over a larger set of network coding decisions and furthermore we present an online algorithm that stabilizes every point within the rate region.

The other line of work starts with [12] (see also [13]), where the authors consider a fixed network and relax the instantaneous decodability assumption to allow the mixing of packets only subject to being decodable eventually. The benefit of this approach is a larger family of allowed operations and therefore the potential for improvement in the network throughput. The price is an increased complexity, which is handled by allowing at most two packets to mix. As a result, the throughput implications are unclear. The achievable region of this technique was later shown in [14] to be stabilizable with an online backpressure algorithm.

In the previous chapters, we dealt with average rates and throughputs. Our algorithms have not operated at the level of stochastic packet arrivals, except in the assumption that the stochastic processes describing the arrivals are ergodic and, therefore, that their average rates summarize most of their meaningful properties. This was also a consequence of random linear network coding, where all nodes always perform the same operation - taking a linear combination of all the packets in their memory - and therefore a flow formulation with side constraints captures the problem. The inter-session coding scheme that we propose in this chapter is quite different from random linear network coding, in that the coding decisions of a node depend not only on which packets that node has in its memory, but also on the memory of the neighbors. We therefore adopt a model that addresses stochastic packet arrivals, the corresponding queue lengths evolution, and dynamic online scheduling and network coding decisions.

Figure 5.1 The topology graph $\mathcal{G}_t$ of a network with $n = 3$ standard nodes and a relay node 0. Each link represents two directed links, going in opposite directions.



5.1 Network Model

5.1.1 General Model and Assumptions

Consider a wireless network, the *topology* of which is represented as a directed graph $\mathcal{G}_t = (\mathcal{N}_t, \mathcal{A}_t)$ with node set $\mathcal{N}_t = \{0, 1, \ldots, n\}$ and arc set $\mathcal{A}_t = \{(i, j) : 0 \leq i, j \leq n, i \neq j\}$. The case where $n = 3$ is depicted in Fig. 5.1. From the definition, the network is fully connected and therefore symmetric. However, we assume that node 0 is a special *relay* node with extended capabilities. This model is appropriate, for example, when the network consists of a number of ground nodes $1, \ldots, n$ and one unmanned aerial vehicle (UAV), node 0, with extended range and power and a larger set of coding and modulation schemes. The network operates with constant-length packets and in slotted time, where the slot index t is an integer corresponding to the time interval $[t, t + 1)$.

We assume, for the sake of simplicity, that the relay serves solely the purpose of enhancing communication between the other nodes and does not inject individual packets. Exogenous packet arrivals at node i with destination j (resulting from processes at the application layer of node i) occur according to *admissible* stochastic processes $\mathbf{A}(t) = (A_i^j(t))$, for $1 \leq i, j \leq n, i \neq j$, with average rates $\lambda_i^j = E[A_i^j]$. We use the same definition of admissible as the authors in [61, Definition 3.4]

Definition 8 *A process $A(t)$ is admissible with rate λ if*

- *The time average expected arrival rate satisfies:*

$$\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} E\left\{A(\tau)\right\} = \lambda. \tag{5.1}$$

- *For all time slots t, we have $E\{A(t)^2|\mathbf{H}(t)\} \leq A^2_{max}$, where $A_{\max}$ is a positive constant and $\mathbf{H}(t)$ represents the history up to time t, i.e. all events in slots $\tau \in \{0, \ldots, t-1\}$.*

- *For any $\delta > 0$, there exists an interval size T such that for any initial time t_0 the following condition holds:*

$$E\left\{\frac{1}{T}\sum_{k=0}^{T-1} A(t_0+k)|\mathbf{H}(t_0)\right\} \leq \lambda + \delta. \tag{5.2}$$

Since the network is fully connected, owing to interference at most one node in the network can transmit per slot. Assume that transmissions from and to the relay are always successful. Any other link can be either ON, in which case it can support the transmission of one packet per slot or OFF, in which case no packet can be transmitted over this link. The topology state at time t is thus given by a binary vector $S(t) = (S_{ij}(t))$, for $i, j \in \{1, \ldots, n\}, i \neq j$, with $S_{ij}(t) = 1$ indicating that the corresponding link is ON. Assume that the state $S(t)$ is known at all nodes and that it evolves according to a finite state, irreducible Markov chain with state space $\mathcal{S}$. Let π_s denote the average fraction of time that the process spends in state $S(t) = s$. For such chains the time averages π_s are well defined and with probability 1 we have

$$\pi_s = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} 1_{[S(\tau)=s]}, \text{ for all } s \in \mathcal{S}, \tag{5.3}$$

where $1_{[\cdot]}$ is the indicator function.

If node i transmits a packet designated for node j, and j receives it successfully, it is removed from the system. Otherwise, the following sequence of actions is carried out

- node i removes it from its queue,
- the relay (which by assumption receives every packet successfully) assumes responsibility for the packet and stores it for further transmission,
- all nodes that have overheard the packet store it until it has reached its destination for the purpose of possibly using it at a later stage for network coding.

This scheme requires a certain amount of perfect feedback in the following form: After any packet transmission from a non-relay node, every other node has to acknowledge (or negatively acknowledge) the reception to the relay. Periodically, but not necessarily in every slot, the non-relay nodes also need feedback from the relay indicating that they may discard the overheard packets that were delivered in the interim and are no longer needed. Note that feedback between non-relay nodes is not required, which is consistent with our assumption that these nodes have more limited capabilities than the relay. For our analysis we will use three different graphs, each of them describing a different aspect of the system. In addition to the topology graph $\mathcal{G}_t$, we will introduce the queuing network (directed) graph $\mathcal{G}_q$ and the network coding conflict (undirected) graph $\mathcal{G}_c$, both to be precisely defined later.

5.1.2 Queuing Model

In our model, each node $i \in \{1, \ldots, n\}$ has queues R_i^j, $j \in \{1, \ldots, n\} \setminus i$, one for each possible packet destination. The relay, on the other hand, has a system of *virtual queues* in which it stores received packets (that failed to reach their designated destination) for the purpose of performing network coding. More precisely, the relay partitions all overheard packets in $n \cdot (2^{n-1} - 1)$

equivalence classes, according to their next-hop j and the set of nodes $\mathcal{Q} \subset \{1, ..., n\} \setminus j$, $\mathcal{Q} \neq \emptyset$ that have knowledge of them. The set $\mathcal{Q}$ is never empty as there is always one node, the original sender i, that has the packet. The relay keeps track of a virtual queue for each such class of packets. Let $\mathbf{X}(t) = (X^j_{\mathcal{Q}}(t))$ denote the queue length vector of all packet classes at time t.

Consider the two-stage *queuing network* $\mathcal{G}_q$ in Fig. 5.2 consisting of the queues (R^j_i) and the virtual queues at the relay (X^j_Q). Here we explicitly model the dynamic behavior of the queues used to accommodate scheduling and network coding. Packets that leave the system (the arrows with solid tips) are directed to an artificial node E, the *system exit*. This queuing network, in particular its stability region and an online stabilizing algorithm, is the focus of our analysis.

In the original network of Fig. 5.1, a packet broadcasted from node i can, depending on the state s, either reach its destination or be overheard by the relay and possibly a subset of its neighbors. In the queuing model, correspondingly, it is either transferred to the system exit E or to one of the virtual queues at the relay. As a result, for a given topology state s, each queue R^j_i will have exactly one state-dependent outgoing link denoted by $(R^j_i, d^j_i(s))$, where we define

$$d^j_i(s) = \begin{cases} E & \text{if } s_{ij} = 1, \\ X^j_{Q'} & \text{if } s_{ij} = 0, \end{cases} \tag{5.4}$$

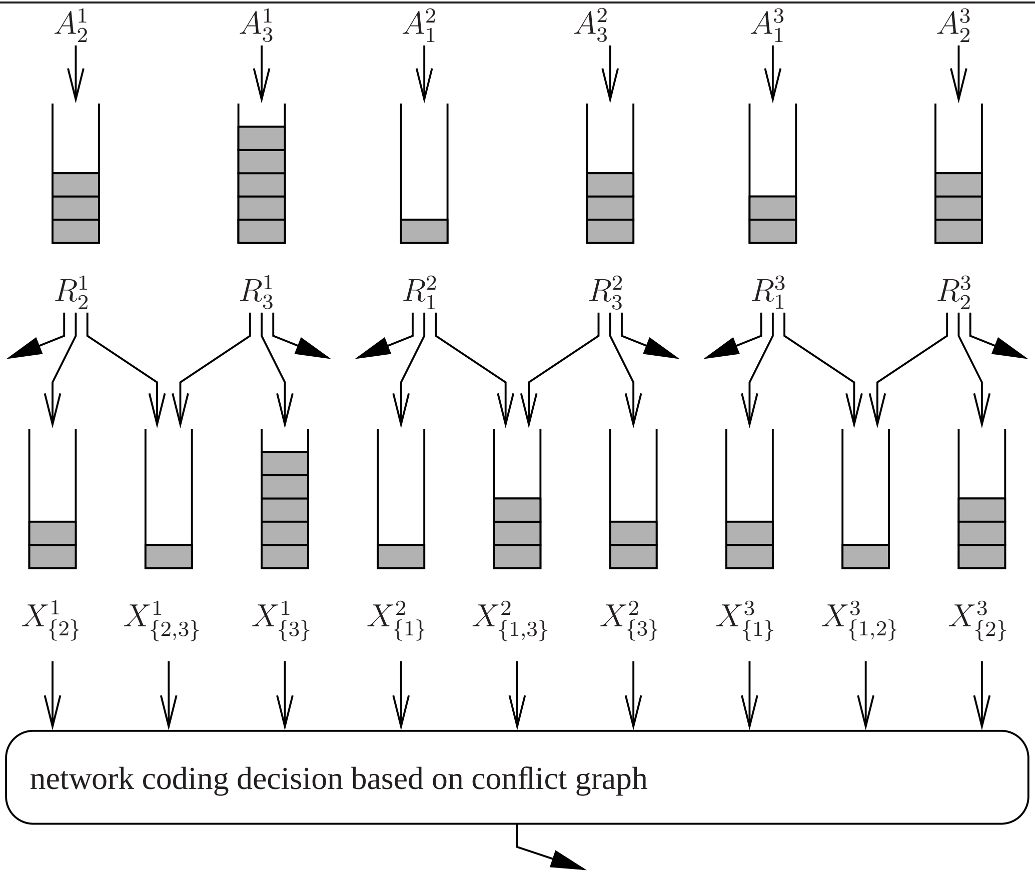
where $Q^{'} = \{k | k \neq j, s_{ik} = 1\} \cup i$.

A queue with backlog $X(t)$ evolves according to the discrete-time dynamics $X(t+1) = \max(X(t) - \mu(t), 0) + A(t)$, where $A(t)$ is the arrival process, and $\mu(t)$ the service process. For queue stability, we use the following definition [61, Definition 3.1]

Definition 9 *A queue is called (strongly) stable if*

$$\lim_{t\to\infty} \sup \frac{1}{t} \sum_{\tau=0}^{t-1} E\{X(\tau)\} < \infty. \tag{5.5}$$

Figure 5.2 The corresponding queuing network graph $\mathcal{G}_q$ for the network in Fig. 5.1. Directed links indicate possible packet transitions; packets that leave the system (the arrows with solid tips) are directed to an artificial node E, the *system exit*. A subset of the virtual queues $X_{\mathcal{Q}}^j$ can be served jointly in one time slot if they correspond to a stable set in the conflict graph $\mathcal{G}_c$



A network of queues is strongly stable if all queues comprising the network are strongly stable.

5.1.3 Network Coding

We can represent valid network coding combinations resulting from the instantaneous decoding condition by a graphic model. In this *conflict graph* approach, we construct an undirected graph with vertices corresponding to the queues. Two queues are connected with a link if they *cannot* be served jointly, i.e. packets from the two queues cannot be XORed together, because they violate the instantaneous decodability condition. This is made precise in the following definition.

Definition 10 *For the system of queues $(X^j_{\mathcal{Q}})$, the conflict graph $\mathcal{G}_c = (\mathcal{V}, \mathcal{E})$ is an undirected graph with a one-to-one correspondence between vertices $\mathcal{V}$ and queues. Two vertices $X^i_{\mathcal{Q}_1}$ and $X^j_{\mathcal{Q}_2}$ are* **not** *connected if*

- $i \neq j$,
- $i \in \mathcal{Q}_2$, *and* $j \in \mathcal{Q}_1$,

otherwise they are connected with an undirected link.

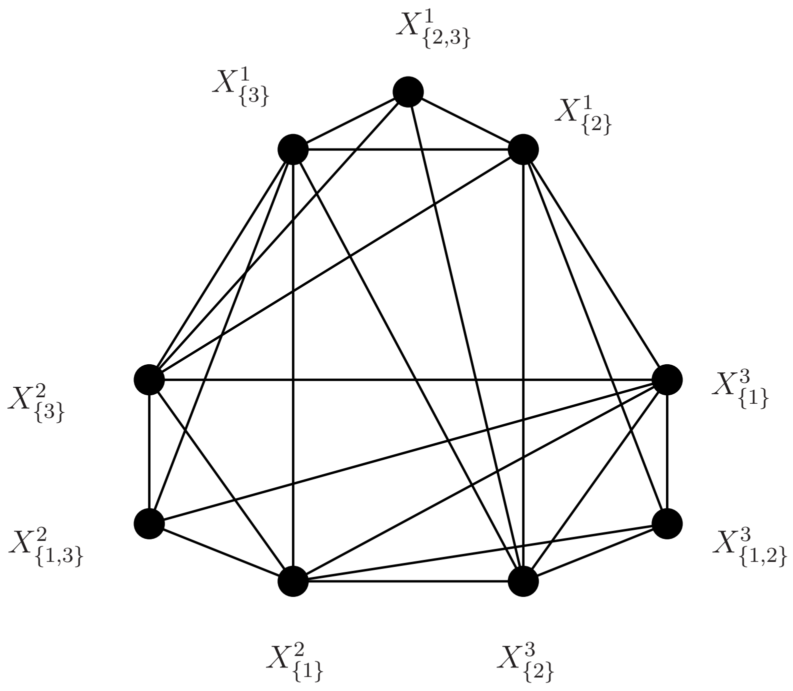
The first condition guarantees that the packets in the two queues have different destinations and the second condition means that each destination has overheard the packet meant for the other destination node. We define a valid configuration of queues as a set of nodes in the conflict graph without any conflicting pair, i.e. a valid configuration is a stable set.

For the network in Fig. 5.1, the corresponding conflict graph is depicted in Fig. 5.3.

5.1.4 Joint Scheduling and Network Coding

We return to the queuing model (see Fig. 5.2) and give a precise definition

Figure 5.3 The conflict graph $\mathcal{G}_c$ corresponding to the virtual queues at the relay in Fig. 5.2.



Definition 11 *The queuing network $\mathcal{G}_q = (\mathcal{N}_q, \mathcal{A}_q)$ is a directed graph, with node set*

$$\mathcal{N}_q = \left\{(R_i^j) \cup (X_{\mathcal{Q}}^j) \cup E\right\} \tag{5.6}$$

and arc set

$$\begin{aligned} \mathcal{A}_q &= \left\{(R_i^j, E)\right\} \quad \forall R_i^j \in \mathcal{N}_q & (5.7)\\ &\cup \left\{(R_i^j, X_{\mathcal{Q}}^j)\right\} \quad \text{if } i \in \mathcal{Q} & (5.8)\\ &\cup \left\{(X_{\mathcal{Q}}^j, E)\right\} \quad \forall X_{\mathcal{Q}}^j \in \mathcal{N}_q. & (5.9)\end{aligned}$$

Owing to the interference constraints, the control action in each time slot is to serve either one of the links $(R_i^j, d(s))$ or a valid subset of the (X_Q^j, E) links subject to the network coding constraints. A *control input* $\mathbf{I}(t) = (I_{ab}(t))$ for the queuing network is a binary vector with $I_{ab}(t) = 1$ if link $(a, b) \in \mathcal{A}_q$ is activated in slot t.

The control space $\mathcal{I}_s$ for a state s thus consists of

$$\begin{aligned}\mathcal{I}_s &= \mathcal{I}'_s \cup \mathcal{I} \qquad (5.10)\\ &= \left\{\left(R_i^j, d(s)\right) : \quad i,j \in \{1,\dots,n\}, i \neq j\right\}\\ &\quad \cup \left\{\left(X_{Q_l}^{j_l}, E\right) : \left(X_{Q_l}^{j_l}\right) \text{ is a stable set in } \mathcal{G}_c\right\},\end{aligned}$$

where $\mathcal{I}'_s$ denotes the state-dependent part, and $\mathcal{I}$ the state-independent part of the control.

Let $\mathbf{c}(I(t), S(t)) = (c_{ab}(I(t), S(t)))$ denote the link capacity vector under control $I(t) \in \mathcal{I}_{S(t)}$ and state $S(t) \in \mathcal{S}$. Based on the previous discussion, the capacity of link (a, b), measured in packets/slot is

$$c_{ab}(I(t), S(t)) = \begin{cases} 1 & \text{if } I_{ab}(t) = 1, \\ 0 & \text{otherwise}. \end{cases} \qquad (5.11)$$

Consider the region defined by

$$\Gamma = \sum_{s \in \mathcal{S}} \pi_s \mathrm{CH}\left\{\mathbf{c}(I, s) : I \in \mathcal{I}_s\right\}, \qquad (5.12)$$

where CH$(\cdot)$ denotes the convex hull and the different convex hulls are added using the usual set summation. Using the decomposition from Eqn. (5.10), we can rewrite the region Γ as follows, isolating the contribution of the stable set polytope of the conflict graph

$$\Gamma = \left\{\sum_{s \in \mathcal{S}} \pi_s \mu_s \mathrm{CH}\left\{\mathbf{c}(I, s) : I \in \mathcal{I}'_s\right\} + \left[\sum_{s \in \mathcal{S}} \pi_s (1 - \mu_s)\right] P_{STAB}(\mathcal{G}_c) : \mu_s \in [0, 1], \forall s \in \mathcal{S}\right\} (5.13)$$

The significance of this region is that every vector (g_{ab}) of long-term link transmission rates that can be supported by the network has to lie in Γ [61]. For the introduced constrained queuing system, two questions naturally arise and we will address them next: The optimal service policy and its associated stability region.

5.2 Stability Region

We begin by studying the stability region (or network layer capacity region, as opposed to the information theoretic notion of capacity) which is defined as follows [61]

Definition 12 *The stability region Λ is the closure of the set of all arrival rate matrices $\left(\lambda_i^j\right)$ that can be stably supported by the network considering all possible policies for routing, scheduling and restricted network coding (i.e. instantaneous decodability and network coding only at the relay).*

The characterization of the stability region is given in the following theorem.

Theorem 1 *The stability region for the constrained queuing system in Fig. 5.2 is the set of all arrival rate vectors $\left(\lambda_i^j\right)$ such that for all links $(a,b) \in \mathcal{A}_q$ there exists a non-negative flow vector $(f(a,b))$ and a transmission rate vector $(g(a,b)) \in Cl(\Gamma)$ satisfying*[1] *the flow conservation constraints*

$$\lambda_i^j \leq f(R_i^j, E) + \sum_Q f(R_i^j, X_Q^j), \quad \forall \lambda_i^j, \tag{5.14}$$

$$\sum_i f(R_i^j, X_Q^j) \leq f(X_Q^j, E), \quad \forall X_Q^j, \tag{5.15}$$

and the capacity constraints

$$f(a,b) \leq g(a,b), \quad \forall (a,b) \in \mathcal{A}_q. \tag{5.16}$$

[1] $Cl(\cdot)$ denotes the closure of a set.

Proof This is a straightforward application of [61, Theorem 3.8] to the queuing network $\mathcal{G}_q$.

5.3 Online Algorithm

The stability region tells us that if the average arrival rates were fixed and known a priori, there exists a policy that stabilizes the network. However, it might not be causal, i.e. the decision at time t might depend on events occurring after time t. An *online* algorithm on the other hand, decides at time t solely based on the history up to this time and the current state of the network. As the authors have shown in [60, 61], there exists a class of online algorithms, so called *differential backpressure* algorithms that stabilize every point in the interior of the stability region.

Consider the following three-step algorithm.

1. Computation of backpressure weights: In each time slot t, first observe the topology state variable $S(t)$. Then compute for all links $(R_i^j, d(s))$ the differential backlogs $w_i^j(t)$ as follows

$$w_i^j(t) = \begin{cases} R_i^j(t) & \text{if } d(s) = E, \\ R_i^j(t) - X_{Q'}^j(t) & \text{if } d(s) = X_{Q'}^j. \end{cases}$$

Compute the maximum weighted stable set of $\mathcal{G}_c$ with weights $X_Q^j(t)$

$$\mathbf{c}^* = \arg \max_{\mathbf{c} \in \mathrm{STAB}(\mathcal{G}_c)} \left\{ \mathbf{X}^T(t)\mathbf{c} \right\},$$

and denote the corresponding weight $w^*(t) = \mathbf{X}^T(t)\mathbf{c}^*$.

2. Scheduling: Select the maximum weight among $\left\{w^*(t), w_i^j(t)\right\}$, for $i, j = 1, \ldots, n$. The queue scheduled for service is the relay if the maximum is $w^*(t)$, or otherwise the queue R_i^j corresponding to the maximum backpressure weight $w_i^j(t)$.

3. Network coding: If the relay is scheduled for transmission, identify the queues which are

members of the stable set $\mathbf{c}^*$ computed in the previous step, and serve them jointly. To that end, take the packets at the head of each queue, combine them with binary XOR and transmit the resulting combination.

The described algorithm stabilizes every arrival rate vector within the stability region. This is established by the following result, originally due to [60].

Theorem 2 *[61, Theorem 4.5] The backpressure algorithm stabilizes the network for an arrival rate vector λ if there exist a scalar $\epsilon > 0$ such that $\lambda + \epsilon\mathbf{1} \in \Lambda$, where $\mathbf{1}$ denotes the vector with all entries equal to 1.*

A remarkable consequence is that the algorithm stabilizes the system for all points in the interior of the stability region without even requiring knowledge of the stability region.

5.4 Performance Evaluation

We illustrate the performance of our scheme in three ways. Firstly, we illustrate the network coding gains by computing the volume of the stable set polytope $P_{STAB}(\mathcal{G}_c)$ and comparing with the volume of the constraint polytope when no network coding is allowed. This approach has been pursued in [37] in the context of network coding for switches with multicast capabilities. Secondly, we compute the stability region for network coding and for routing, and thirdly, we simulate the online scheduling and network coding algorithm.

5.4.1 Polytope Volume Computation

Consider the case $n = 3$ nodes and the 9 virtual queues which can be scheduled for joint service according to the conflict graph in Fig. 5.3. By inspection, the conflict graph contains one maximum stable set of cardinality 3, namely $\{X^2_{1,3}, X^1_{2,3}, X^3_{1,2}\}$, similarly nine maximal stable sets of cardinality 2 and nine stable sets corresponding to the individual vertices, so it can be written as the convex hull of these 19 points and the origin. Using the Multi-Parametric Toolbox for MATLAB [62], we have used this representation to compute its volume, which turns out to be $2.8660 \cdot 10^{-4}$. Without network coding, only one virtual queue can be served at a time, so the "conflict graph" when only routing is allowed is the complete graph K_9. The volume of the resulting stable set polytope, which is a 9-dimensional standard simplex, is $(9!)^{-1} = 2.7557 \cdot 10^{-6}$. The ratio of the two volumes is $Vol(P_{STAB}(\mathcal{G}_c))/(9!)^{-1} = 104$.

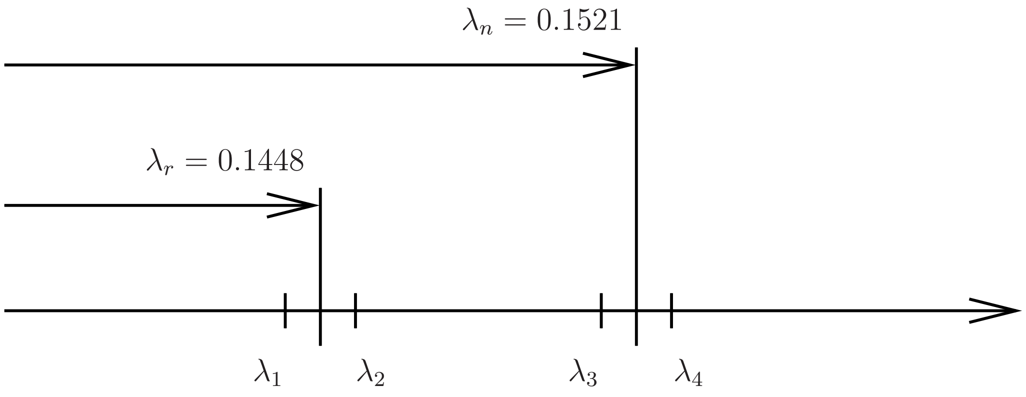
5.4.2 Stability Region

We compute the stability region as characterized in Theorem 1 for the special case when all injection rates are equal. Though this computation is not easier than the general case, it has the useful property that the network throughput is parameterized by a scalar $\lambda = \lambda_i^j$. We consider $n = 3$ and the state process is assumed to be i.i.d. across time and across links with each link being ON with probability 0.2 and OFF with probability 0.8. Routing, i.e. serving one virtual queue at a time, leads to a maximum symmetric rate λ_r and network coding to a rate λ_n which, due to the fact that network coding includes routing as a special case, is at least as large as λ_r. The maximum symmetric rates, $\lambda_r = 0.1448$ for routing and $\lambda_n = 0.1521$ for network coding, are shown in Fig. 5.4.

5.4.3 Online Algorithm

To illustrate the performance of the online algorithm, we simulate its behavior for symmetric input rates which are close to the breaking points for routing and network coding, respectively. Consider $\lambda_1, \ldots, \lambda_4$ as indicated in Fig. 5.4 and the corresponding sample paths in Fig. 5.5. For λ_1, which is in the stability region of both policies, we see that routing leads on average to significantly more packets in the system. When we slightly increase the rate to λ_2 routing breaks down, while network coding is largely unaffected. Going further to λ_3 network coding is still stable, though at a higher average backlog. Finally, at λ_4 both systems operate beyond

Figure 5.4 Routing stabilizes the network for all rates smaller than $\lambda_r = 0.1448$, network coding extends the stability region to $\lambda_n = 0.1521$, which is an increase of about 5%. In Fig. 5.5 we simulate sample paths of the online backpressure algorithm for the points $\{\lambda_1, \lambda_2\} = \lambda_r \pm 0.0003$, and $\{\lambda_3, \lambda_4\} = \lambda_n \pm 0.0003$.



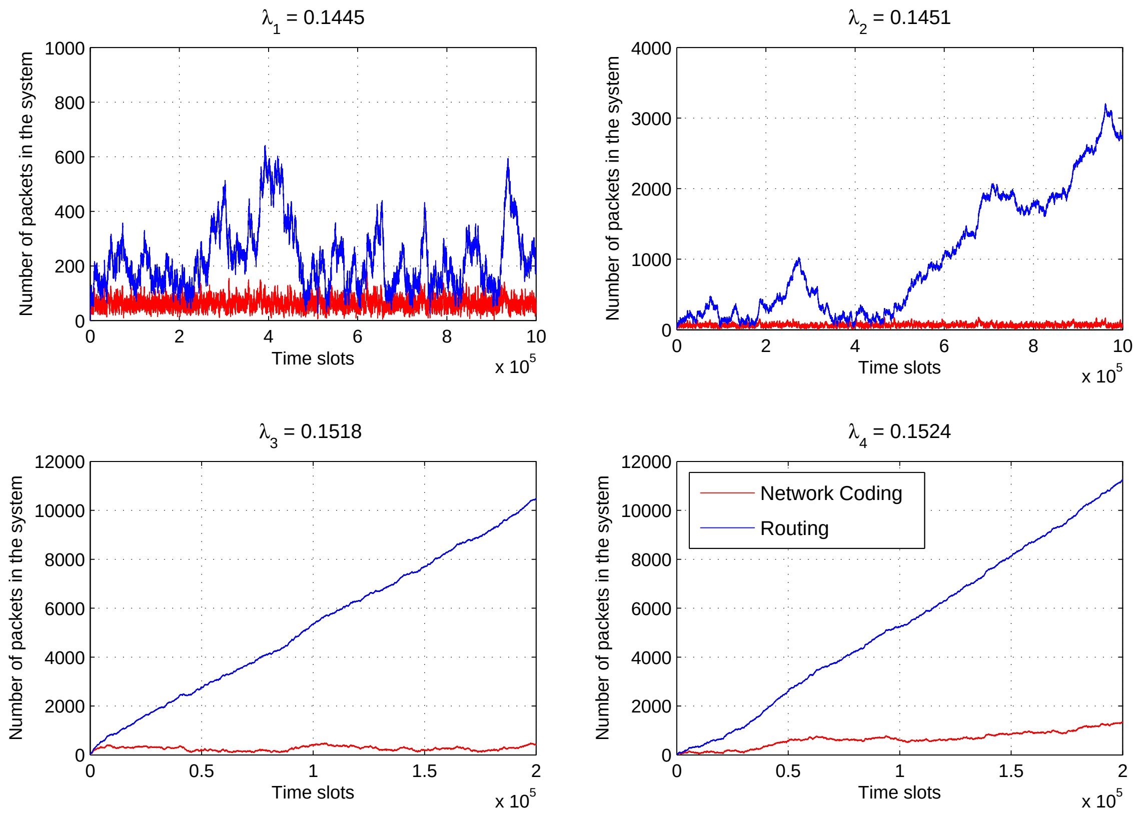
stability but network coding "degrades" more gracefully.

5.5 Discussion

We investigated the stability region as well as online stabilizing algorithms for instantaneously decodable network coding. In contrast to the previous chapters, we adopted a model that deals with the queue length evolution of the system and captures the system dynamics. We showed that network coding can extend the stable operation regime of the network and, on average, reduce the backlog in the system. The networks we considered had, in contrast to the previous chapters, certain structural constraints; we assumed that just one node in the network is capable of performing network coding operations and can reach all its neighbors in one hop.

As an extension, it is promising to look at allowing more nodes to do network coding. Nevertheless, the results are interesting in their own right as in many cases, owing to structural constraints or when networks are highly heterogenous, one may choose to use local network coding methods. Moreover, there is evidence that in many cases only a small subset of the network nodes need to do network coding in order to get the throughput benefits. For example, in [63] the authors consider a genetic algorithm to minimize the number of coding nodes in

Figure 5.5 The total number of queued packets in the system for different injection rates. In particular, for λ_1 both routing and network coding stabilize the system, for λ_2 and λ_3 only network coding stabilizes the system, while for λ_4 both policies result in an unstable system.



the network. It turns out that in most cases only a very small fraction of the nodes need to do network coding.

6

Delay Control in Network Coded Broadcasting

In this chapter[1], we explore the issue of delay in network coding, when applied to broadcast erasure channels. Random linear network coding generally improves the throughput but can lead to higher delay, as receivers need to collect several coded packets before being able to decode them and thus recover the original information. Thus, there is a tension between increased throughput and decoding delay in the network [3, 64] and this is the focus of this chapter.

Depending on the system and application, and particularly depending on the transport protocol, different notions of decoding delay may be used. In [65], delay is defined as the time between the (stochastic) arrival of a packet at the source and its decoding by a receiver. In contrast, we use the notion of delay as suggested by the authors in [3]. There, a receiver experiences a unit of delay every time it successfully receives a packet, that is either a redundant linear combination

[1]This chapter is joint work with P. Sadeghi and R. Shams.

of previously received (coded) packets or that is not instantaneously decodable for all receivers. Note that under this definition the order in which packets are transmitted and arrive does not matter.

Out-of-order packet delivery may be a reasonable assumption if the underlying transport protocol is unreliable. In such a situation, if we wish to transmit, e.g., video, we can use a *multiple description* code [66], where each packet brings new information to the receiver regardless of the order. On the other hand, if the transport protocol ensures highly reliable in-order transmission, it is reasonable to use a *successive refinement* source code, where each subsequent packet improves the quality, but only if all previous packets are received.

Typically, *online* network coding algorithms for broadcast erasure channels [3, 65, 67, 68] use feedback from the receivers to the source to optimize the selection of packets to be combined and transmitted. The goal is to minimize decoding delay, possibly subject to constraints on the throughput. In particular, in [3], a number of such algorithms were proposed and compared in terms of performance.

In our approach, we allow for network coding subject to instantaneous decoding, thus we adopt the same strategy as in Chapter 5. The difference is that now we wish to broadcast a number of packets to all receivers, whereas in Chapter 5 each packet had a designated next-hop. We next present a systematic framework for the minimization of decoding delay based on combinatorial optimization. We show that this problem can be cast into an integer linear programming (ILP) framework, where an instantaneously decodable packet transmission corresponds to a *set packing* problem [69] on an appropriately defined set structure. Furthermore, we provide a customized and efficient method for finding the optimal solution to the set packing problem, which is in general NP-hard. Our numerical results show that for a moderate number of receivers, the optimal solution can be computed efficiently. Finally, we illustrate how our optimal algorithm can be converted to a heuristic with very small computational complexity. The performance of the heuristic is evaluated by means of simulations and shown to perform well compared to the optimal solution.

6.1 Network Model and Definitions

Consider a single source that broadcasts to N receivers, denoted by R_i for $i = 1, \dots, N$. The data is divided into K packets, denoted by m_j for $j = 1, \dots, K$. Each receiver is interested in all of the packets. Time is slotted, and the source can transmit one (possibly coded) packet per slot. A link L_i connects the source to receiver R_i. Link L_i experiences an erasure with probability of $p_{e,i}$ in each slot. We assume that the erasure random processes are independent and stationary. Before transmission of the next packet, the source collects error-free and delay-free 1-bit feedback from each destination indicating if the packet was successfully received or not.

Definition 13 *At the end of transmission round ℓ, the knowledge of receiver R_i is the set consisting of all packets that the receiver has decoded so far.*

Definition 14 *A coded packet is instantaneously decodable for receiver R_i if it is a linear combination containing exactly one packet not in the knowledge space of R_i.*

A coded packet is called non-innovative for receiver R_i if it only contains source packets that the receiver has decoded so far. Otherwise, the packet is innovative.

Definition 15 *A scheme is called throughput optimal if all transmissions are innovative for the entire set of receivers.*

Definition 16 *In time slot ℓ, receiver R_i experiences one unit of delay if it successfully receives a packet that is either non-innovative or not instantaneously decodable.*

The source only applies coding when all receivers will be able to decode immediately, then a delay at R_i can only occur if the received packet at R_i is not innovative. Note that in the last definition, we do not count channel inflicted delays due to erasures. The delay only counts ‘algorithmic’ delay, i.e when we are not able to provide innovative and instantaneously decodable packets to a receiver [3]. This definition captures the part of the delay that is due to algorithm design, as opposed to the part of the delay that is due to unfavorable erasure patterns.

A zero-delay scheme would require all packets to be both innovative and instantaneously decodable to all receivers. Thus zero-delay implies throughput optimality, but not vice versa. Achieving zero delay is difficult since which packets are innovative and instantaneously decodable depends on the random packet erasures experienced by each receiver. An offline algorithm is one that knows all future realizations of erasures; it is non-causal. In contrast, an *online* algorithm decides on what to send in any given slot solely based on the information received in past slots. The authors in [3, Theorem 1] show that for the case of $N = 2$ and $N = 3$ receivers, there exists an *offline* algorithm that has zero-delay. The authors then prove that not even an offline zero-delay algorithm exists for $N \geq 4$.

In this chapter, we focus on designing online algorithms.

6.2 Optimization Framework

We assume that packets are transmitted in two phases. In the first phase, lasting K slots, each packet is transmitted uncoded. After this first phase, in every slot that follows, we form instantaneously decodable packets according to the algorithm we describe next.

Assume, we are in slot ℓ, $\ell \geq K$. Recall that the source knows all prior packet losses at all receivers due to the feedback. This information can be summarized in an $N \times K$ binary *receiver-packet incidence* matrix A with elements

$$a_{ij} = \begin{cases} 1 & \text{if } R_i \text{ needs } m_j, \\ 0 & \text{otherwise}. \end{cases} \tag{6.1}$$

The columns of matrix A are denoted by $\mathbf{a}_1, \ldots, \mathbf{a}_K$. We assume that packets received by all receivers are removed from the receiver-packet incidence matrix. Hence, A does not contain any all-zero columns.

Example Consider $N = 2$ receivers and $K = 3$ packets. Before the transmission begins, the receiver-packet incidence matrix A is a 2×3 matrix of ones. If we send packet m_1 in slot $\ell = 1$ and only receiver R_2 successfully receives it, A becomes

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

If we send packet m_2 in slot $\ell = 2$ and only receiver R_1 successfully receives it, A will then be

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

As we do not know the future realizations of the link erasures, we seek to maximize myopically for the next slot the number of receivers that experience no delay. Let $\mathbf{x}$ denote a binary decision vector of length K that describes which packets are being coded together. Under instantaneous decoding, going to higher field sizes does not lead to further gains. The transmitted packet consists of the binary XOR of the source packets for which $x_j = 1$. Consider sets $M_1, \ldots, M_K \subset \{R_1, \ldots, R_N\}$, where M_j is the set of receivers that still *need* source packet m_j. Let $\mathbf{w}^T = (|M_1|, \ldots, |M_K|)$, and let $\mathbf{1}_N$ be the all-one column vector of dimension N.

Then, maximizing the number of receivers for which a transmission is innovative subject to the constraint of instantaneous decodability can be posed as a zero-one integer linear program (ILP). The column vector $\mathbf{x}$ of length K contains the decision variables; they are integers with values zero or one.

$$\begin{aligned} & \max \mathbf{w}^T \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{1}_N \\ & \mathbf{x} \in \{0,1\}^K \end{aligned} \tag{6.2}$$

This is a standard problem in combinatorial optimization, usually called *set packing* [69]. Here the universe is the set of all receivers and we need to find disjoint (due to instantaneous decodability condition) subsets M_j with the largest total size. In the (most desirable) case when equality holds in every row of $A\mathbf{x} \leq \mathbf{1}_N$ (that is, the transmitted packet is innovative for every receiver) this becomes a *set partition* problem. This is equivalent to a zero-delay transmission.

6.3 Algorithms for Solving (6.2)

Unfortunately, the set packing problem is NP-hard [69]. In this section, we present an efficient algorithm designed to take advantage of the specific problem structure. Since the underlying combinatorial problem is NP-hard, its worst case execution time is exponential in the size of the problem instance. However, for many practical situations of interest, our method performs well empirically.

Consider the following definitions.

Definition 17 *Two binary-valued variables are said to be constrained if they cannot be simultaneously* 1 *in a solution. Formally,* x_i *and* x_j *are constrained if for any* $\mathbf{x}$ *satisfying* $A\mathbf{x} \leq \mathbf{1}_N$, $x_i + x_j \leq 1$. *We also say that* x_j *is constrained to* x_i *and vice versa.*

Note that x_i and x_j are constrained if and only if there exists at least one row index p in A for which $a_{pi} = a_{pj} = 1$.

Definition 18 *The set of all variables constrained to x_i is called the constrained set of x_i and is denoted by $\mathcal{C}_i$. That is,*

$$\mathcal{C}_i = \{x_j | j \neq i, A\mathbf{x} \leq \mathbf{1}_N \Rightarrow x_i + x_j \leq 1\}. \tag{6.3}$$

If x_i and x_j are not constrained to each other ($x_i \notin \mathcal{C}_j$ and $x_j \notin \mathcal{C}_i$), then columns $\mathbf{a}_i$ and $\mathbf{a}_j$ in A cannot have non-zero elements in the same row position. That is, for each row index p, $a_{pi} = 1 \Rightarrow a_{pj} = 0$ and $a_{pj} = 1 \Rightarrow a_{pi} = 0$.

Definition 19 *A variable x_i is said to be unconstrained if $\mathcal{C}_i = \emptyset$. The set of all unconstrained variables is denoted by $\mathcal{U}$ and is referred to as the unconstrained set.*

If x_i is an unconstrained variable, then for each row index p, $a_{pi} = 1 \Rightarrow a_{pj} = 0$ for all $j \neq i$ (otherwise, x_i and x_j would become constrained).

Example Consider the following receiver-packet incidence matrix A

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Variables x_1 and x_3 are constrained because for $p = 1$, $a_{p1} = a_{p3} = 1$. Variables x_1 and x_4 are not constrained to each other because columns $\mathbf{a}_1$ and $\mathbf{a}_4$ do not have a non-zero element in the same row position. Variable x_6 is unconstrained because its non-zero elements are in rows 6 and 7 and no other column has a non-zero element in rows 6 or 7. In summary, $\mathcal{C}_1 = \{x_2, x_3\}$, $\mathcal{C}_2 = \{x_1\}$, $\mathcal{C}_3 = \{x_1, x_4\}$, $\mathcal{C}_4 = \{x_3\}$ and $\mathcal{C}_5 = \mathcal{C}_6 = \emptyset$.

6.3.1 Exhaustive Search

The algorithm that we propose is essentially a divide-and-conquer type of algorithm that takes advantage of the problem structure to efficiently prune the search space. We make the following observations for pruning the parameter space:

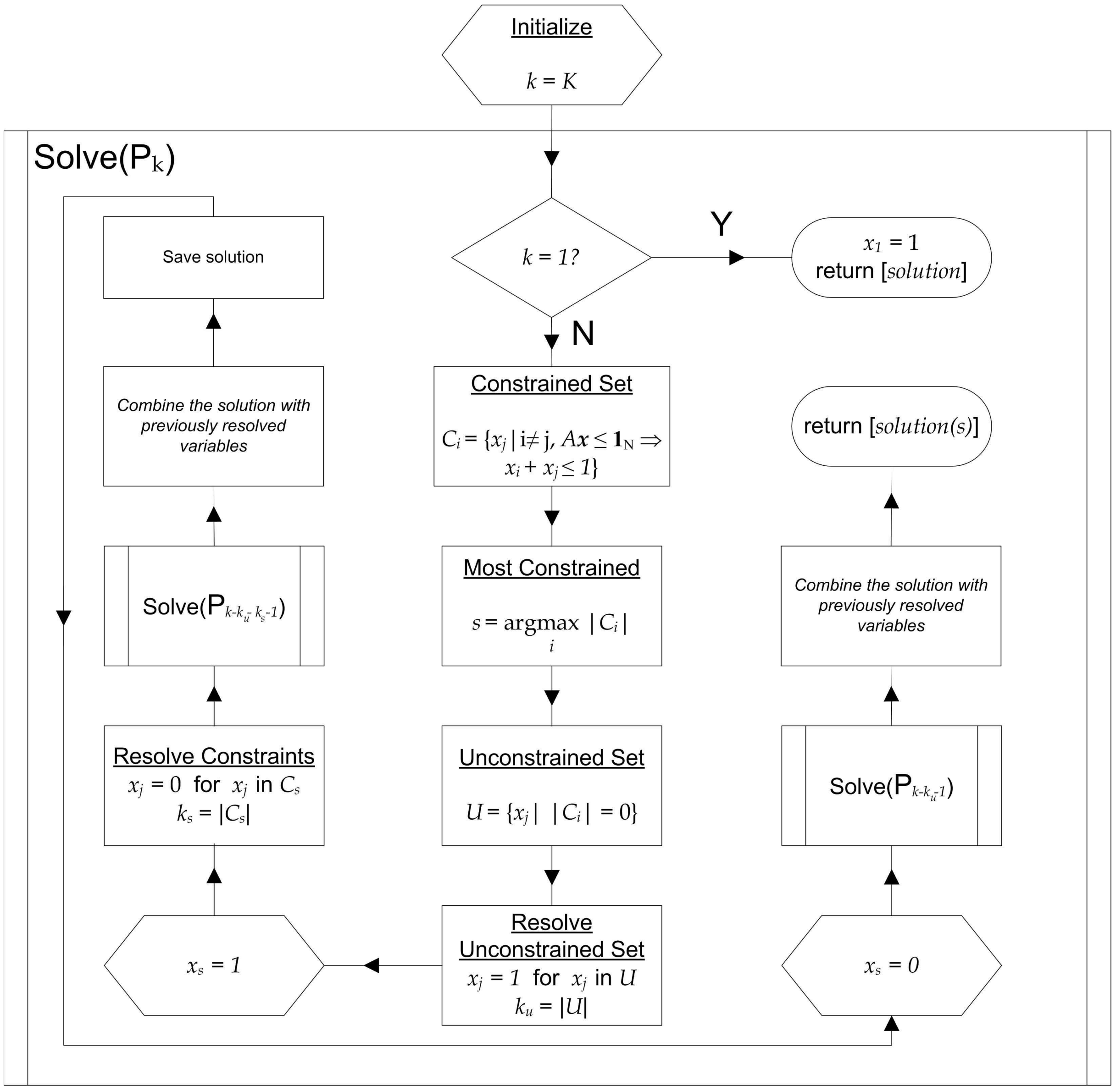
1. Unconstrained variables must be set to 1. If $\mathcal{C}_i = \emptyset$, then $x_i = 1$ since setting $x_i = 1$ results in a strictly higher value of $\mathbf{w}^T\mathbf{x}$ than setting $x_i = 0$ and in no way constrains the values of x_j for $j \neq i$. In the above example, $x_5 = x_6 = 1$ because no other variable is constrained to them.

2. If a constrained variable is set to 1, then all members of its constrained set must be set to 0. In the above example, setting $x_1 = 1$ forces x_2 and x_3 to zero.

With these observations, we can proceed to discuss the suggested algorithm. Let $\mathcal{P}_k$ denote a problem instance of size k whose input is an $N \times k$ receiver-packet incidence matrix A_k and whose output is a set of solutions vectors $\mathbf{x}$ of length k which satisfy the instantaneous decodability condition $A_k\mathbf{x} \leq \mathbf{1}_N$. Consider Algorithm 1 and its corresponding flow chart in Fig. 6.1.

Remark 5 *The algorithm is recursive. In line 12, k_u unconstrained variables are set to one, $x_s = 1$ and therefore k_s variables constrained by x_s are set to zero, hence a total of $k_s + k_u + 1$ variables are resolved. Similarly, in line 16, k_u unconstrained variables are set to one and $x_s = 0$, hence a total of $k_u + 1$ variables are resolved.*

Figure 6.1 A schematic of Algorithm 1 for finding the optimal network coding solution of (6.2).


Initialize
$k = K$
Solve(P_k)
$k = 1?$
Y
$x_1 = 1$
return [*solution*]
N
Save solution
Constrained Set
$C_i = \{x_j \mid i \neq j, A\mathbf{x} \leq \mathbf{1}_N \Rightarrow x_i + x_j \leq 1\}$
return [*solution(s)*]
Combine the solution with previously resolved variables
Solve($P_{k-k_u-k_s-1}$)
Most Constrained
$s = \operatorname{argmax}_i |C_i|$
Combine the solution with previously resolved variables
Resolve Constraints
$x_j = 0$ for x_j in C_s
$k_s = |C_s|$
Unconstrained Set
$U = \{x_j \mid |C_i| = 0\}$
Solve(P_{k-k_u-1})
$x_s = 1$
Resolve Unconstrained Set
$x_j = 1$ for x_j in U
$k_u = |U|$
$x_s = 0$

Algorithm 1 Exhaustive Recursive Search for the Optimal Solution(s) of (6.2)

```
1: Start with the original problem of size $k = K$.
2: Solve($\mathcal{P}_k$):
3: if k=1 then
4:     Return $x_1 = 1$ (since the variable is not constrained).
5: else
6:     Determine the constrained set for all variables $x_1$ to $x_k$.
7:     Denote the index of the variable with the largest constrained set by $s$ and the cardinality
       of its constrained set by $k_s$.
8:     Denote the cardinality of the unconstrained set $\mathcal{U}$ by $k_u$.
9:     Set all the unconstrained variables to 1.
10:    Set $x_s = 1$ and the variables in its corresponding constrained set $\mathcal{C}_s$ to 0.
11:    Reduce the problem by removing resolved variables. Reduce $A_k$ accordingly.
12:    Solve($\mathcal{P}_{k-k_u-k_s-1}$).
13:    Combine the solution with previously resolved variables. Save solution.
14:    Set $x_s = 0$.
15:    Reduce the problem by removing resolved variables. Reduce $A_k$ accordingly.
16:    Solve($\mathcal{P}_{k-k_u-1}$).
17:    Combine the solution with previously resolved variables. Return solution(s).
18: end if
```

Remark 6 *In line 7 of Algorithm 1, we have chosen to resolve the variable with the largest constraint set first. As the search is exhaustive, the order in which variables are resolved does not matter, in principle. Our choice is motivated by empirical observations, after trying many different rules.*

It is straightforward to see that Algorithm 1 corresponds to an exhaustive search and therefore is guaranteed to return all optimal solutions of (6.2). A formal proof of that can be found in [23]. However, we note that not every solution returned by Algorithm 1 is optimal. The non-optimal solutions can be easily discarded by testing against the objective function (6.2) at the end of the algorithm. We also note that in Algorithm 1, we can simply remove those packets received by every receiver from the problem. If there are K_0 such variables, we can start step 1) above from $k = K - K_0$ instead of K.

6.3.2 Fast Search

There are situations where one would like to obtain a (possibly suboptimal) solution quickly. This may be the case, for example, when the total number of packets to be transmitted is very large. Therefore, consider the following heuristic.

Weight Sorted Heuristic Algorithm - As in Algorithm 1, we start with the original problem of size $k = K$. We rearrange the columns of the matrix A in descending order of w_j. We set the head variable $x_1 = 1$ and given its corresponding constrained set $\mathcal{C}_1$ resolve $k_1 = |\mathcal{C}_1|$ variables that are to be set to zero. We then solve the smaller problem of size $\mathcal{P}_{k-k_1}$ and continue until the problem cannot be further reduced. One main difference between this heuristic and Algorithm 1 is that at each recursion, the head variable is only set to one; the other possibility of $x_1 = 0$ is not explored. In a sense, this heuristic algorithm finds *greedy* solutions to the problem at each recursion by serving the highest priority packet. In this heuristic algorithm, all k_u unconstrained variables are automatically set to 1 in the course of the algorithm. The computational complexity of this method is at worst proportional to K, which can happen when there is no constraint between packets.

6.4 Performance Evaluation

We compare our optimal algorithm and the proposed heuristic with the random opportunistic algorithm proposed in [3]. In Fig. 6.2, we have plotted the total delay (the sum of the delays experienced by the different receivers) for the transmission of $K = 100$ packets. Both the optimal algorithm as well as the heuristic outperform the random opportunistic algorithm of [3] - in certain regimes substantially. Also note that, for the scenario in this experiment, the performance of our heuristic is much closer to the optimal algorithm than to the algorithm in [3]. This illustrates that precisely defining the optimal setting and then seeking approximations within this framework is a promising approach.

Figure 6.2 Median of decoding delay for the transmission of $K = 100$ packets to $N = 3$ to $N = 100$ receivers. Channel erasures are memoryless and occur with a probability of $p = 0.5$ independently in every link. We compare Algorithm 1, our heuristic, and the random opportunistic algorithm in [3].

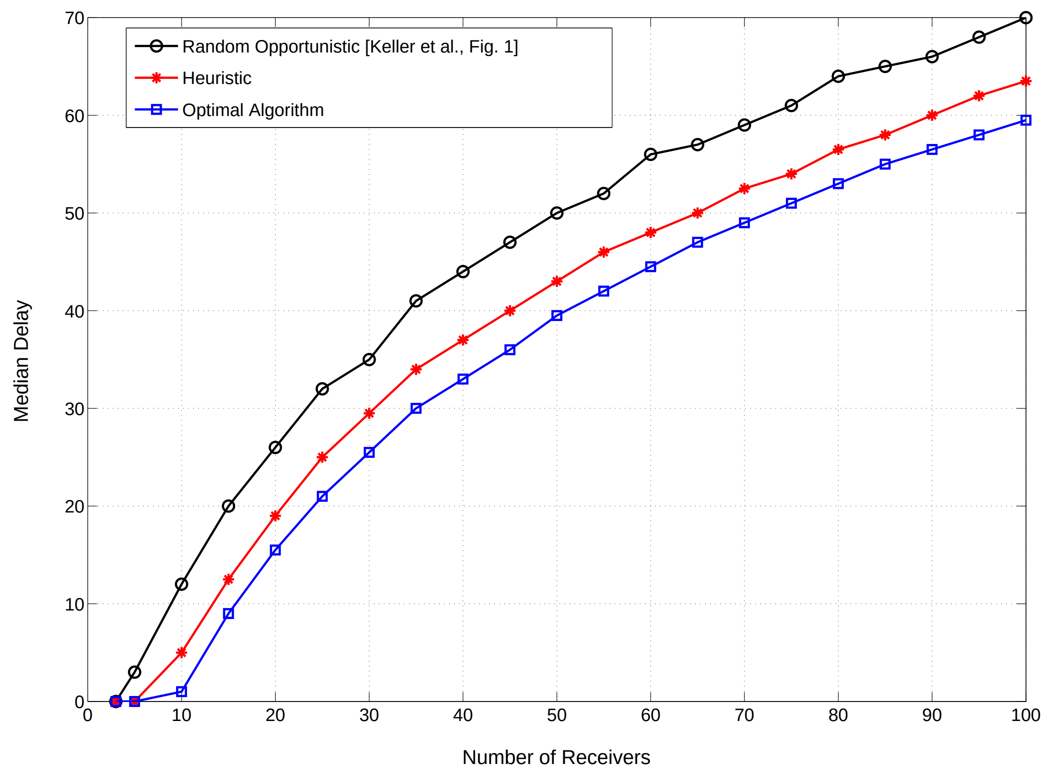
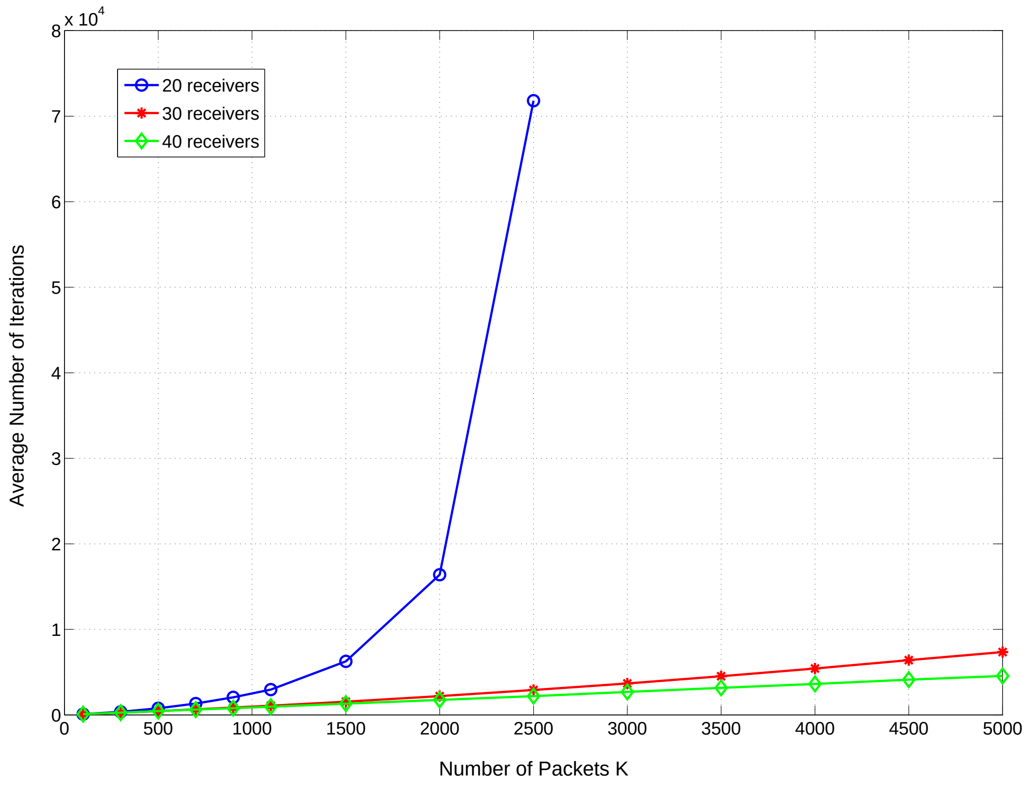


Figure 6.3 The effect of increasing the number of packets on the computational complexity of Algorithm 1. Our measure for computational complexity is the number of recursions.



In Fig. 6.3, we show the computational complexity of the optimal algorithm as a function of the number of packets K and number of receivers N. For a wide range of meaningful problem sizes the algorithm is very efficient. This is somewhat surprising as the underlying problem is NP-hard. Another interesting observation is that in this figure more receivers require fewer iterations. This is consistent with the theory and practice of integer programming. It is well-known that if in an integer program the number of variables is kept constant but more constraints (in our case each receiver corresponds to a new constraint) are added, the computational efficiency improves [69, Section 1.2]. This is in sharp contrast with linear programming, where the computational complexity increases with the number of variables *and* the number of constraints.

6.5 Discussion

We have defined delay in an “order-oblivious” fashion and derived an optimal algorithm to minimize the delay for one transmission at a time. Applying this algorithm successively, leads to reductions in delay as compared to the approach in [3]. Furthermore, although the underlying combinatorial problem is NP-hard, our algorithm can handle problems of reasonable size efficiently.

7 Conclusion

We have looked at possibilities to apply network coding to wireless multi-hop networks in a way that is well-matched with the other layers of the network, especially the multiple access layer. We addressed optimal transmission scheduling for network coded multicast traffic, asynchronous algorithms for computing multicast subgraphs, and inter-session network coding with instantaneous decoding.

Many of the problems we addressed, such as optimal scheduling or network coding subject to instantaneous decoding, are inherently hard. At the core, they require solving NP-hard stable set problems. This means that, in practice, it is likely that one would have to resort to approximations and heuristics. How practical, then, is our approach, and why did we not choose heuristics to begin with? Our contribution is a better understanding of the problem structure and an encapsulation of the inherent combinatorial difficulty. Once the combinatorial problem is clearly formulated, we can capitalize on the rich set of approximation algorithms for NP-hard problems. This is at the same time a safeguard from applying heuristics, where unnecessary.

The mathematical tools we used in our work - multi commodity flow problems with side constraints, graphical models for conflicts, and techniques from optimization decomposition, to name a few - date back a long time. In the context of network performance optimization, their application has been well-known for at least two decades, as documented in the seminal work of Bertsekas and Gallager [32]. When network coding enters the picture, however, these techniques have to be adapted and modified to take into account the special properties of network coding. For instance, we have seen that when scheduling network coded transmissions, we want to activate hyperarcs, instead of arcs. We have also shown that, we can use conflict graph models to express conflicts on the schedule, as well as on the code. Sometimes major changes are called for, sometimes just subtle twists are required. In any case, it is imperative to take the peculiarities that network coding introduces into account when designing coded networks. Otherwise performance will certainly fall short of the possible.

The goal of our work has been to analyze the performance of network coding in wireless networks and provide algorithms that can guide the more practical issue of protocol design. Our models are in the language of mathematical programming, whereas network protocols are hardware and software solutions. Thus, there is still a gap to bridge. When designing theoretical abstractions for practical problems, one can not include all practically relevant constraints into the formulation. Without simplification, it is impossible to get to the crux of a problem and to get useful insight. There is a great deal of work to be done on the implementation of network coding in wireless networks, and it is likely that it will be necessary to incorporate more constraints and practical considerations into our models. Particularly, in the rapidly growing field of heterogeneous networking it is likely that new modeling challenges will arise.

In this work, we have addressed delay in Chapter 6. However, most of our focus was on throughput. If network traffic has to satisfy streaming or realtime guarantees, then delay becomes at least as important as throughput. Network coding requiring in order packet delivery - and therefore meeting more stringent delay constraints - has been addressed, for instance in [65]. However, the throughput-delay tradeoff in coded wireless networks remains a largely open problem. One promising approach is to adapt the formulation in Chapter 6 to accommodate packets with service deadlines, thus relaxing the instantaneous decoding condition. A possible approach could be to combine in order delivery with the restriction that packets are decodable after a fixed (small) number of slots, rather than instantaneously.

Another open problem is to understand the practical limits of inter-session network coding. We have seen that by imposing instantaneous decoding, the problem becomes tractable. When instantaneous decoding is relaxed, the problem becomes the so called *index coding with side information* problem [70], which is a very difficult combinatorial problem, even when restricting attention to linear codes. The setup in [70] is static. That is, the goal is to minimize the number of transmissions for a fixed batch of packets. In practice, however, new packets arrive and receive service continuously. Therefore, even if we can compute the “optimal” solution for a fixed batch of packets, it becomes outdated as soon as new packets arrive. The challenge is thus to design algorithms that generalize instantaneous decoding, assume dynamic arrivals and departures, and are useful from an engineering perspective.

To illustrate how the framework proposed in this thesis can guide network coding implementation in practice, we discuss three projects addressing the implementation of network coding. They were student research projects carried out under my supervision at the Institute for Communications Engineering at Technical University Munich.

7.1 Implementation of Network Coding: Case Studies

7.1.1 Case 1: Wireless Video Transmission using Network Coding

In this project, the goal was to implement a wireless transmission protocol for streaming network coded video. The implementation is in the framework of the discrete event simulator NS-2 [71], a free network simulation platform that is widely used in academia. For a survey on network coding for video applications see [72].

The underlying transport protocol is the connectionless User Datagram Protocol (UDP) [73]. UDP does not feature an acknowledgment mechanism, as for example TCP. The packets of the MPEG-4 video stream are partitioned in blocks (generations) and the generations are transmitted using standard random network coding over $GF(2^8)$. Network coding requires acknowledgments only after each generation; this small additional feedback can be easily implemented without having to redesign the transport protocol.

The evaluation of the video quality takes into account three components: average packet loss rate, average packet delay, and the perceived video quality. An interesting result was that, in practice, short generation sizes (6 to 8 packets) with some overlap performed best [74]. Of course, when the focus is solely on increasing the throughput large generation sizes are required. However, in video streaming delay is often more critical than throughput and a moderate packet loss rate can be tolerated. The results are summarized in [74].

7.1.2 Case 2: Network Coding for Heterogeneous Networks

In this research project, we look at implementing network coding across different networks. Modern user equipments can connect to both wireless local area networks (WiFi) via the 802.11 protocol and at the same time to 3G or 4G wireless systems such as UMTS, LTE, or LTE-Advanced. The cellular network usually provides reliable coverage but incurs a high cost per packet, whereas LANs provide cheaper access but are not always reliable. To ensure a satisfactory quality of experience for an application like streaming video, the best engineering solution may be to use both networks simultaneously. In such a scenario, network coding can help improve reliability, reduce the cost for achieving a certain quality of experience, and reduce the coordination requirements between the different network access points.

Concretely, we investigate how we can use network coding to combine information sent over the two different connections and simulate and compare different schemes with the OPNET modeler [75]. OPNET is, as opposed to NS-2, a commercial tool that is widely used in industry. Many protocols can be specified by means of state transition diagrams and additional C or C++ code, e.g. for the finite field arithmetics, can be integrated.

From a theretical perspective, the work [76] looks at streaming media and the trade-off between the probability of interruption and the buffering time. As a concrete application of the results in [76], we consider a file download using network coded TCP [77]. The novelty over [77] is that we consider network coded TCP over heterogeneous networks and investigate various association policies and their quality of experience as well as their cost. A detailed project description can be found in [78].

7.1.3 Case 3: Network Coding for Relaying in LTE-A

Relaying is a promising way to increase coverage in cellular wireless networks and to improve connections to cell-edge users. Consequently, relaying has been incorporated into recent wireless communication standards like LTE-Advanced [79].

The full information theoretic characterization of the relay channel is still an open problem and physical layer techniques (see e.g. [80] or [81] for an actual implementation) are not only difficult to implement but also may be inadequate if the network and the channel conditions change quickly. However, as we have seen in the example of Chapter 2, relaying can be easily implemented on higher layers using network coding. Network coding allows us to cooperate at higher layers, thus giving us much more flexibility to exploit the time variations of the network.

In particular, in this project we consider a TCP connection from the base station to a mobile user equipment via a relay. Network coding for TCP has been studied in [77], where the authors use a sliding window for selecting the packets to be encoded. The relay can be either used to extend the coverage of the base station or to enhance the communication if both relay and end user are covered by the base station. Questions that we address are how we can use network coding to optimize the throughput and delay characteristics of the TCP session, and if we assume a streaming media application to what extent network coding can reduce the completion time of the download. A detailed project description can be found in [82].

A

Appendix

A.1 Analytical Solution of the Linear Program (2.7)-(2.14)

Consider the abbreviations:

$$A = \left(p_{1\{23\}2} + p_{1\{23\}3} + p_{1\{23\}\{23\}}\right) \tag{A.1}$$

$$B = \left(p_{1\{23\}3} + p_{1\{23\}\{23\}}\right) \tag{A.2}$$

$$C = \left(p_{1\{23\}2} + p_{1\{23\}\{23\}}\right) \tag{A.3}$$

$$D = p_{233}. \tag{A.4}$$

Note that the variable x_{23} is redundant, we can simply replace it with x_{12}. We proceed to eliminate x_{12}, according to the Fourier-Motzkin procedure [31, Section 2.8]. To that end, we rewrite the constraints involving x_{12} in the following way (and keep the ones not involving x_{12}):

$$x_{12} \geq 0 \tag{A.5}$$

$$x_{12} \geq R - x_{13} \tag{A.6}$$

$$\alpha C \geq x_{12} \tag{A.7}$$

$$\alpha A - x_{13} \geq x_{12} \tag{A.8}$$

$$(1-\alpha)D \geq x_{12} \tag{A.9}$$

$$0 \leq x_{13} \leq \alpha B \tag{A.10}$$

$$0 \leq \alpha \leq 1. \tag{A.11}$$

After eliminating x_{12}, removing redundant constraints, and arranging for the elimination of x_{13}, we are left with:

$$x_{13} \geq R - \alpha C \tag{A.12}$$

$$x_{13} \geq R - (1-\alpha)D \tag{A.13}$$

$$x_{13} \geq 0 \tag{A.14}$$

$$\alpha B \geq x_{13} \tag{A.15}$$

$$\alpha A \geq R \tag{A.16}$$

$$0 \leq \alpha \leq 1. \tag{A.17}$$

After eliminating x_{13}, removing redundant constraints, and arranging for the elimination of α, we are left with:

$$\alpha \geq R/A \tag{A.18}$$

$$\alpha(B-D) \geq R - D \tag{A.19}$$

$$1 \geq \alpha. \tag{A.20}$$

Now, if $B > D$, Eqn. (A.19) becomes

$$\alpha \geq (R - D)/(B - D), \tag{A.21}$$

and after eliminating α, we ultimately get

$$R \leq B, \tag{A.22}$$

and the maximal rate is $R^* = B$; this corresponds to not using the relay.

On the other hand, if $B < D$, Eqn. (A.19) becomes

$$(R - D)/(B - D) \geq \alpha, \tag{A.23}$$

and after eliminating α, we ultimately get

$$R \leq AD/(A - B + D), \tag{A.24}$$

and the maximal rate is $R^* = AD/(A - B + D)$.

To determine the coefficient α^*, we plug in R^* in constraints (A.18) and (A.19), and get

$$\alpha^* = \frac{D}{A - B + D}. \tag{A.25}$$

A.2 Distributed Maximal Stable Set Algorithm

Consider the following algorithm, due to [49]. The input is the weighted conflict graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with weights as defined in Eqn. (3.37). Let w_v denote the weight of vertex $v \in \mathcal{V}$. The algorithm is executed at each node $v \in \mathcal{V}$ as described in the pseudo-code for Algorithm 2 below. Algorithm 2 calls two subroutines, also given below. We use $N_{\mathcal{G}}(v)$ to denote the neighborhood of node v in the conflict graph $\mathcal{G}$. Boolean variables $ss(u)$ and $join(u,t)$, $u \in \{v\} \cup N_{\mathcal{G}}(v)$ and $t \in \mathcal{V}$ are initialized to $False$. Setting $ss(u)$ to $True$ means that u belongs to the stable set. Setting $join(u,t)$ to $True$ means that node u is not a stable set member but a neighbor of stable set member t. Setting either one of these variables to $True$ is communicated to the neighbors by means of messages $\mathtt{SS}(u)$, and $\mathtt{Join}(u,t)$, respectively. After the initialization phase, node v performs updates upon receiving messages from its neighbors. After it has permanently decided whether to be a stable set member or not, it exits the algorithm (this happens in one of the two subroutines).

Algorithm 2 Decentralized weighted maximal stable set [49]

```
1: initialize:
2: $ss(u) \leftarrow False$
3: $join(u,t) \leftarrow False$
4: if for each $u \in N_{\mathcal{G}}(v)$ we have $w_v > w_u$ then
5:     $ss(v) \leftarrow True$
6:     send $\mathtt{SS}(v)$
7:     exit
8: end if
9: loop
10:     on receiving $\mathtt{SS}(u)$ :
11:         execute subroutine1
12:     on receiving $\mathtt{Join}(u,t)$ :
13:         execute subroutine2
14: end loop
```

The following proposition shows the correctness of the algorithm and bounds the number of iterations needed for convergence,

Algorithm 3 Subroutine 1.

```
1: $ss(u) \leftarrow True$
2: if for each $z \in N_{\mathcal{G}}(v) : w_z > w_u$ there exists a $x \in V$ such that $join(z, x)$ is true then
3:     send Join$(v, u)$
4:     exit
5: end if
```

Algorithm 4 Subroutine 2.

```
1: $join(u, t) \leftarrow True$
2: if for each $z \in N_{\mathcal{G}}(v) : w_z > w_v$ is $join(z, x)$ for some $x \in V$ then
3:     send SS$(v)$
4:     $ss(v) \leftarrow True$
5:     exit
6: else if for at least a $z \in N_{\mathcal{G}}(v)$ is $ss(z)$ and for each $u \in N_{\mathcal{G}}(v) : w_u > w_z$ is $join(u, x)$ for
   some $x \in V$ then
7:     send Join$(v, \max_{w_z}\{z : ss(z)\})$
8:     exit
9: end if
```

Proposition 3 *[49, Theorem 1] All nodes in the network exit the algorithm being assigned either membership or non-membership to a stable set. The stable set computed is maximal. Furthermore, the number of steps needed for a node to terminate the algorithm is upper bounded by $2\alpha(\mathcal{G})$, i.e. twice the stability number of the conflict graph $\mathcal{G}$.*

A.3 The Asynchronous Algorithm with Polymatroidal Constraints

With polymatroidal constraints, and assuming one hyperarc (i, J) for each node i, the problem is

$$\text{minimize} \sum_{(i,J)\in\mathcal{A}} f_{iJ}(z_{iJ})$$

subject to:

$$\sum_{j\in K} x_{ij}^{(t)} \le z_{iJ} b_{iJK}, \quad \forall\,(i,J)\in\mathcal{A}, K\subset J, t\in T, \tag{A.26}$$

$$\sum_{j:(i,j)\in\mathcal{A}'} x_{ij}^{(t)} - \sum_{j:(j,i)\in\mathcal{A}'} x_{ji}^{(t)} = \sigma_i^{(t)}, \qquad \forall\, i\in\mathcal{N}, t\in T, \tag{A.27}$$

$$x_{ij}^{(t)} \ge 0, \quad \forall\,(i,j)\in\mathcal{A}', t\in T. \tag{A.28}$$

Here, as usual, $\mathcal{A}$ denotes the set of hyperarcs, whereas $\mathcal{A}'$ is the set of induced arcs, i.e. $\mathcal{A}' = \{(i,j) : i\in\mathcal{N}, j\in J\}$.

Since we assume that the f_{iJ} are monotonically increasing, constraint (A.26) implies that

$$z_{iJ} = \max_{K\subset J, t\in T} \left\{ \frac{\sum_{j\in K} x_{ij}^{(t)}}{b_{iJK}} \right\}. \tag{A.29}$$

This can be the replaced with the soft-maximum

$$z'_{iJ} = L\log\left(\sum_{K\subset J, t\in T} \exp\left(\frac{1}{L} \frac{\sum_{j\in K} x_{ij}^{(t)}}{b_{iJK}} \right)\right). \tag{A.30}$$

Thus, the optimization problem becomes

$$\text{minimize} \sum_{(i,J)\in\mathcal{A}} f_{iJ}\left(L\log\left(\sum_{K\subset J, t\in T}\exp\left(\frac{1}{L}\frac{\sum_{j\in K} x_{ij}^{(t)}}{b_{iJK}}\right)\right)\right)$$

subject to

$$\sum_{j:(i,j)\in\mathcal{A}'} x_{ij}^{(t)} - \sum_{j:(j,i)\in\mathcal{A}'} x_{ji}^{(t)} = \sigma_i^{(t)}, \qquad \forall i \in \mathcal{N}, t \in \mathcal{T}, \tag{A.31}$$

$$x_{ij}^{(t)} \geq 0, \qquad \forall (i,j) \in \mathcal{A}', t \in \mathcal{T}, \tag{A.32}$$

which is again a multicommodity flow problem, and moreover only locally coupled through the objective function. Therefore, the analysis in Chapter 4 applies.

B

Abbreviations

List of Abbreviations

CDMA	code division multiple access
FEC	forward error correction
GF	Galois field
ILP	integer linear program
LAN	local area network
LP	linear program
LTE	long term evolution
LTE-A	long term evolution - advanced
MAC	multiple access

MPEG	moving picture experts group
MWSS	maximum weighted stable set
SNR	signal-to-noise ratio
SQP	sequential quadratic programming
TCP	transmission control protocol
UAV	unmanned aerial vehicle
UDP	user datagram protocol
UMTS	universal mobile telecommunication system
XOR	exclusive or

Bibliography

[1] D. Lun, N. Ratnakar, M. Médard, R. Koetter, D. Karger, T. Ho, and E. Ahmed, "Minimum-cost multicast over coded packet networks," *IEEE Trans. Inform. Theory*, vol. 52(6), pp. 2608–2623, June 2006.

[2] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[3] L. Keller, E. Drinea, and C. Fragouli, "Online broadcasting with network coding," in *Net-Cod 08*, Hong Kong, Jan. 2008.

[4] S.-Y. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory*, vol. 49, no. 2, pp. 371 –381, Feb. 2003.

[5] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[6] T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. Karger, "A random linear network coding approach to multicast," *IEEE Trans. Inform. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[7] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," *IEEE/ACM Trans. Networking*, vol. 16, pp. 497–510, June 2008.

[8] F. Zhao and M. Médard, "On analyzing and improving COPE performance," in *Information Theory and Applications Workshop (ITA)*, San Diego, CA, 2010.

[9] T. Ho and D. S. Lun, *Network Coding - An Introduction*. New York,NY: Cambridge University Press, 2008.

[10] S. Jaggi, P. Sanders, P. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Inform. Theory*, vol. 51, no. 6, pp. 1973 – 1982, June 2005.

[11] R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of linear coding in network information flow," *IEEE Trans. Inform. Theory*, vol. 51, no. 8, pp. 2745 – 2759, Aug. 2005.

[12] D. Traskov, N. Ratnakar, D. Lun, R. Koetter, and M. Médard, "Network coding for multiple unicasts: An approach based on linear optimization," in *Proc. 2006 IEEE International Symposium on Information Theory (ISIT 2006)*, Seattle, WA, July 2006.

[13] T. Ho, Y. Chang, and K. J. Han, "On constructive network coding for multiple unicasts," in *Proc. 44th Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL, Sept. 2006.

[14] A. Eryilmaz and D. S. Lun, "Control for inter-session network coding," in *Proc. Workshop on Network Coding (NetCod 2007)*, Jan. 2007.

[15] Y. Wu, P. A. Chou, Q. Zhang, K. Jain, and W. Zhu, "Network planning in wireless ad-hoc networks: A cross-layer approach," *IEEE J. Select. Areas Commun.*, vol. 23, no. 1, pp. 136–150, Jan. 2005.

[16] Y. Sagduyu and A. Ephremides, "On joint MAC and network coding in wireless ad-hoc networks," *IEEE Trans. Inform. Theory*, vol. 53, no. 10, pp. 3697–3713, Oct. 2007.

[17] D. Traskov, M. Heindlmaier, M. Médard, R. Koetter, and D. Lun, "Scheduling for network coded multicast: A conflict graph formulation," in *IEEE GLOBECOM Workshops 2008*, New Orleans, LA, Nov.-Dec. 2008.

[18] M. Heindlmaier, D. Traskov, R. Koetter, and M. Médard, "Scheduling for network coded multicast: A distributed approach," in *IEEE GLOBECOM Workshops 2009*, Honolulu, HI, Nov.-Dec. 2009.

[19] W. Chen, D. Traskov, M. Heindlmaier, M. Médard, S. Meyn, and A. Ozdaglar, "Coding and control for communication networks," *Queueing Systems*, vol. 63, pp. 195–216, Dec. 2009.

[20] D. Traskov, P. Médard, M. Sadeghi, and R. Koetter, "Joint scheduling and instantaneously decodable network coding," in *IEEE GLOBECOM Workshops 2009*, Honolulu, HI, Nov.-Dec. 2009.

[21] D. Traskov, J. Lenz, N. Ratnakar, and M. Médard, "Asynchronous network coded multicast," to appear in *Proc. 2010 IEEE International Conference on Communications (ICC)*, 2010.

[22] P. Sadeghi, D. Traskov, and R. Koetter, "Adaptive network coding for broadcast channels," in *Network Coding, Theory, and Applications, 2009. NetCod '09. Workshop on*, June 2009, pp. 80–85.

[23] P. Sadeghi, R. Shams, and D. Traskov, "An optimal adaptive network coding scheme for minimizing decoding delay in broadcast erasure channels," *EURASIP Journal of Wireless Communications and Networking, Special Issue on Network Coding for Wireless Communications*, 2010.

[24] D. Traskov, M. Heindlmaier, M. Médard, and R. Koetter, "Scheduling for network coded multicast," submitted to *IEEE/ACM Trans. Networking*.

[25] T. Ho, M. Médard, M. Effros, and D. Karger, "On randomized network coding," in *Proc. 41st Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL, Sept. 2003.

[26] T. P. A. Chou and K. Jain, "Practical network coding," in *Proc. 41st Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL, Sept. 2003.

[27] R. Koetter and F. R. Kschischang, "Coding for errors and erasures in random network coding," in *Proc. 2007 IEEE International Symposium on Information Theory (ISIT 2007)*, Nice, France, June 2007, pp. 791 –795.

[28] D. S. Lun, "Efficient operation of coded packet networks," Ph.D. dissertation, Massachusetts Institute of Technology1, 2006.

[29] E. C. van der Meulen, "Three-terminal communication channels," *Adv. in Appl. Probab.*, vol. 3, no. 1, pp. 120–154, 1971.

[30] C. Fragouli, D. Lun, M. Médard, and P. Pakzad, "On feedback for network coding," in *Proc. 2007 Conference on Information Sciences and Systems (CISS 2007)*, March 2007, pp. 248 –252.

[31] D. Bertsimas and J. N. Tsitsiklis, *Linear Optimization*. Belmont, MA: Athena Scientific, 1997.

[32] D. Bertsekas and R. Gallager, *Data networks*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, Inc., 1992.

[33] M. Riemensberger, M. Heindlmaier, A. Dotzler, D. Traskov, and W. Utschick, "Optimal slotted random access in coded wireless packet networks," to appear in *Proc. 6th Workshop on Resource Allocation in Wireless Networks (RAWNET)*, 2010.

[34] D. Traskov, D. S. Lun, R. Koetter, and M. Médard, “Network coding in wireless networks with random access,” in *Proc. 2007 IEEE International Symposium on Information Theory (ISIT 2007)*, Nice, France, June 2007, pp. 2726 –2730.

[35] L. Bao and J. Garcia-Luna-Aceves, “A new approach to channel access scheduling for ad-hoc networks,” in *Proc. of the 7th Annual International Conference on Mobile Computing and Networking*, Rome, Italy, 2001, pp. 210–221.

[36] L. Tassiulas and A. F. Ephremides, “Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks,” *IEEE Trans. Automat. Contr.*, vol. 37, pp. 1936–1948, Dec. 1992.

[37] J. Sundararajan, M. Médard, M. Kim, A. Eryilmaz, D. Shah, and R. Koetter, “Network coding in a multicast switch,” in *Proc. IEEE INFOCOM 2007*, Anchorage, AK, May 2007.

[38] J. Sundararajan, M. Médard, R. Koetter, and E. Erez, “A systematic approach to network coding problems using conflict graphs,” in *Information Theory and Applications Workshop (ITA)*, San Diego, CA, 2006, invited paper.

[39] C. Caramanis, M. Rosenblum, M. Goemans, and V. Tarokh, “Scheduling algorithms for providing flexible, rate-based, quality of service guarantees for packet-switching in Banyan networks,” in *Proc. 2004 Conference on Information Sciences and Systems (CISS 2004)*, Princeton, NJ, 2004.

[40] X. Yang and G. de Veciana, “Inducing multiscale clustering using multistage MAC contention in CDMA ad hoc networks,” *IEEE/ACM Trans. Networking*, vol. 15, no. 6, pp. 1387–1400, Dec. 2007.

[41] S. Sengupta, S. Rayanchu, and S. Banerjee, “An analysis of wireless network coding for unicast sessions: The case for coding-aware routing,” in *Proc. IEEE INFOCOM 2007*, Anchorage, AK, May 2007, pp. 1028–1036.

[42] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*. Berlin: Springer-Verlag, 2004.

[43] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric algorithms and combinatorial optimization*. Berlin-Heidelberg: Springer-Verlag, 1988.

[44] D. West, *Introduction to Graph Theory*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, Inc., 2001.

[45] S. Biswas and R. Morris, “Opportunistic routing in multi-hop wireless networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 69–74, 2004.

[46] T. Larsson, M. Patriksson, and A.-B. Stroemberg, “Ergodic, primal convergence in dual subgradient schemes for convex programming,” *Mathematical Programming*, vol. 86, no. 2, pp. 283–312, Nov. 1999.

[47] Y. Wu and S.-Y. Kung, “Distributed utility maximization for network coding based multicasting: a shortest path approach,” *IEEE J. Select. Areas Commun.*, vol. 24, no. 8, pp. 1475–1488, Aug. 2006.

[48] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1982.

[49] S. Basagni, “Finding a maximal weighted independent set in wireless networks,” *Telecommunication Systems*, vol. 18, no. 1-3, pp. 155–168, Sept. 2001.

[50] S. Sakai, M. Togasaki, and K. Yamazaki, “A note on greedy algorithms for the maximum weighted independent set problem,” *Discrete Applied Mathematics*, vol. 126, no. 2-3, pp. 313 – 322, 2003.

[51] F. Zhao, M. Médard, D. S. Lun, and A. Ozdaglar, “Convergence rates of min-cost subgraph algorithms for multicast in coded networks,” in *Proc. 45th Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL, Sept. 2007.

[52] T. Ho, B. Leong, R. Koetter, and M. Médard, “Distributed asynchronous algorithms for multicast network coding,” in *Proc. 1st Workshop on Network Coding (NetCod 2005)*, 2005.

[53] N. Freris and P. Kumar, “Fundamental limits on synchronization of affine clocks in networks,” in *46th IEEE Conference on Decision and Control (CDC 2007)*, Dec. 2007, pp. 921–926.

[54] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA: Athena Scientific, 1997.

[55] D. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1995.

[56] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge University Press, 2004.

[57] S. Deb and R. Srikant, “Congestion control for fair resource allocation in networks with multicast flows,” *IEEE/ACM Trans. Networking*, vol. 12, no. 2, pp. 274–285, April 2004.

[58] D. Bertsekas, *Network Optimization - continuous and discrete models*. Belmont, MA: Athena Scientific, 1998.

[59] P. Boggs and J. Tolle, “Sequential quadratic programming,” *Acta Numerica*, vol. 4, pp. 1–51, 1995.

[60] L. Tassiulas and A. F. Ephremides, “Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks,” *IEEE Trans. Automat. Contr.*, vol. 37, pp. 1936–1948, Dec. 1992.

[61] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource allocation and cross-layer control in wireless networks*. Hanover, MA: Foundations and Trends in Networking (NOW).

[62] M. Kvasnica, P. Grieder, and M. Baotic, “Multi-parametric toolbox (MPT),” [Online]. Available: http://control.ee.ethz.ch/mpt/, 2004.

[63] M. Kim, M. Medard, and U.-M. O'Reilly, "Integrating network coding into heterogeneous wireless networks," in *2008 IEEE Military Communications Conference (MILCOM 2008)*, 16-19 2008, pp. 1 –7.

[64] A. Eryilmaz, A. Ozdaglar, and M. Médard, "On delay performance gains from network coding," in *Proc. Annual Conference on Information Sciences and Systems*, Princeton, NJ, Mar. 2006, pp. 864–870.

[65] J. Sundararajan, D. Shah, and M. Médard, "Online network coding for optimal throughput and delay - the three-receiver case," in *Proc. International Symposium on Information Theory and Its Applications (ISITA)*, Dec. 2008, pp. 1 –6.

[66] V. K. Goyal, "Multiple description coding: compression meets the network," *IEEE Signal Processing Mag.*, vol. 18, pp. 74–93, Sept. 2001.

[67] J. K. Sundararajan, D. Shah, and M. Médard, "ARQ for network coding," in *Proc. IEEE Int. Symp. on Inform. Theory (ISIT)*, Toronto, Canada, July 2008, pp. 1651–1655.

[68] D. E. Lucani, M. Stojanovic, and M. Médard, "Random linear network coding for time division duplexing: When to stop talking and start listening," in *Proc. IEEE Conf. on Computer Commun. INFOCOM*, Apr. 2009, pp. 1800–1808.

[69] D. Bertsimas and R. Weissmantel, *Optimization Over Integers*. Belmont, MA: Dynamic Ideas, 2005.

[70] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol, "Index coding with side information," in *Proc. 47th Annual IEEE Symposium on Foundations of Computer Science*, Los Alamitos, CA, 2006, pp. 197–206.

[71] http://www.isi.edu/nsnam/ns/.

[72] M. J. Montpetit and M. Médard, “Video-centric network coding strategies for 4G wireless networks: An overview,” in *7th IEEE Consumer Communications and Networking Conference (CCNC) 2010*, 9-12 2010, pp. 1 –5.

[73] A. S. Tanenbaum, *Computer networks*, 4th ed. Upper Saddle River, NJ: Prentice-Hall, Inc., 2002.

[74] K. Soussi, “Evaluation of the performance of video transmission using network coding,” Master’s thesis, Technical University Munich, 2009.

[75] http://www.opnet.com/.

[76] A. ParandehGheibi, M. Médard, S. Shakkottai, and A. Ozdaglar, “Avoiding interruptions - QoE trade-offs in block-coded streaming media applications,” in *Proc. 2010 IEEE International Symposium on Information Theory (ISIT 2010)*, Austin, TX, June 2010.

[77] J. K. Sundararajan, D. Shah, M. Médard, M. Mitzenmacher, and J. Barros, “Network coding meets TCP,” in *Proc. IEEE INFOCOM 2009*, Rio de Janeiro, Brazil, April 2009, pp. 280 –288.

[78] A. Kulkarni, “Network coding for heterogeneous networks,” Master’s thesis, Technical University Munich, 2010.

[79] S. W. Peters, A. Y. Panah, K. T. Truong, and J. Robert W. Heath, “Relay architectures for 3GPP LTE-Advanced,” *EURASIP Journal on Wireless Communications and Networking*, 2009.

[80] S. Zhang, S.-C. Liew, and P. Lam, “On the synchronization of physical-layer network coding,” in *Information Theory Workshop (ITW)*, Punta del Este, Uruguay, Oct. 2006, pp. 404 –408.

[81] S. Katti, S. Gollakota, and D. Katabi, “Embracing wireless interference: analog network coding,” in *In Proc. of ACM SIGCOMM*, 2007, pp. 397–408.

[82] J. Cafarelli, “Network coding for relay networks,” Master’s thesis, Technical University Munich, 2010.