Survivability and performance optimization in communication networks using network coding

by

Mirzad Mohandespour

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:

Ahmed E. Kamal, Co-Major Professor

Manimaran Govindarasu, Co-Major Professor

Aditya Ramamoorthy

Arun K. Somani

Mani Mina

Zhengdao Wang

Iowa State University

Ames, Iowa

2015

DEDICATION

To

Akram & Mostafa

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to extend my sincere gratitude to my parents Akram Shiravi and Mostafa Mohandespour, for all the moments they could have pursued their own personal dreams, instead they chose to make sure I had the opportunity to pursue my own; my wife, Mina Farahbakhsh, for her continued love and support during this long course of study; Dr. Ahmed Kamal, who made it possible for me to come to the US in the first place and patiently guided me through years of this program; Dr. Manimaran Govindarasu, who supported me when I needed help and allowed me to pursue new research directions; Dr. Zhengdao Wang, for his deep and elaborate feedback and comments that helped improve my research; Dr. Mani Mina, who has been like a brother and a mentor for me and has always inspired me; Dr. Arun K. Somani, for his feedback on my research and for what he taught me as a teaching assistant; Dr. Aditya Ramamoorthy, for being a great teacher and for the network coding class that built the foundation of my research.

ABSTRACT

The benefits of network coding are investigated in two types of communication networks: optical backbone networks and wireless networks. In backbone networks, network coding is used to improve survivability of the network against failures. In particular, network coding-based protection schemes are presented for unicast and multicast traffic models. In the unicast case, network coding was previously shown to offer near-instantaneous failure recovery at the bandwidth cost of shared backup path protection. Here, cost-effective polynomial-time heuristic algorithms are proposed for online provisioning and protection of unicast traffic. In the multicast case, network coding is used to extend the traditional live backup (1+1) unicast protection to multicast protection; hence called multicast 1+1 protection. It provides instantaneous recovery for single failures in any bi-connected network with the minimum bandwidth cost. Optimal formulation and efficient heuristic algorithms are proposed and experimentally evaluated. In wireless networks, performance benefits of network coding in multicast transmission are studied. Joint scheduling and performance optimization formulations are presented for rate, energy, and delay under routing and network coding assumptions. The scheduling component of the problem is simplified by timesharing over randomly-selected sets of non-interfering wireless links. Selecting only a linear number of such sets is shown to be rate and energy effective. While routing performs very close to network coding in terms of rate, the solution convergence time is around 1000-fold compared to network coding. It is shown that energy benefit of network coding increases as the multicast rate demand is increased. Investigation of energy-rate and delay-rate relationships shows both parameters increase non-linearly as the multicast rate is increased.

CHAPTER 1. GENERAL INTRODUCTION

Information technology has been changing the ways humans interact and live since its birth in the last decades of 20th century. The very visible example is the way we use smartphones for an increasing number of applications and services: listening to music, reading the news, staying connected via social networks, sharing media, shopping, navigation, health monitoring, and finally making phone calls. From a computer engineering perspective, what enables a smartphone (as an example) to offer such range of services is not merely the integration of computing power (hardware and software) in a small handheld device; It also depends on the telecommunication network that provides connectivity and data transmission. Today, telecommunication networks may be rather *invisible* from a user point of view as they mainly rely on fibers under the ground or waves in the air. Optical fiber networks provide the backbone of what we know as the Internet: connecting residential and business buildings, cities, countries, and continents. At the same time, wireless networks have emerged as crucial in the access section of network where mobile users demand connectivity.

Telecommunication networks are, of course, not limited to the Internet. However, as we have moved into 21st century, there seems to be a trend in using the Internet for other traditionally non-IP communications [10]. Services like VoIP and IPTV use the ubiquity offered by the Internet platform to deliver content to users regardless of their location [91]. It is on the same platform that Internet of Things is envisioned to connect countless number of new smart objects to the existing Internet [7][58][76].

The Internet, therefore, is set to provide even higher bandwidths with more reliability and connectivity. On the other hand, such services would still need to be economically affordable and sustainable to succeed in the market. The increasing demand for data would require higher bandwidth offered by service providers and better utilization of available bandwidth. As the

bandwidth of network links increases, specially in backbone networks, so does the cost of any disruption or failure. Therefore, networks need to also be resilient: to provide an acceptable level of service even in the event of failures. Backbone networks may even demand the same level of service in the event of failures, i.e., they have to be survivable. In this case, network traffic should be protected against failures, e.g., diverse redundant routes may be used to guarantee continued service in the case of a single link failure.

Another important performance parameter is energy. A study estimates that on a worldwide average, the Internet is going to amount for 7% of a country's electrical power consumption [22] with the access networks taking up to 70% of total Internet energy consumption. This is important both in terms of the cost of energy and the carbon footprint of communication networks. This has led to a new domain of research focused on green networking [12]. In the wireless domain, energy consumption is more critical due to the limited battery resource on mobile devices [16].

As a result, researchers in both academia and industry are challenged to find new ways to improve computer networks in terms of various performance parameters including bandwidth, resilience, and energy. One of the promising ideas that fundamentally challenges the status quo in communication networks is Network Coding [6]. In simple words, network coding generalizes the routing logic in communication networks. In addition to traditional routing functions of store, forward, and duplicate, network coding lets outgoing data at a network node to be a *function* of incoming data. The *coding* in network coding refers to computing such functions.

Since network coding is a generalization of routing, it logically follows that a network coding-capable network would perform at least as good as the corresponding routing-capable network. Moreover, researchers have been able to show scenarios where network coding increases the achievable transmission rate, decreases energy consumption for a given demand, or improves other quality of service parameters. One of the main research challenges, on the other hand, is to investigate the tradeoffs of network coding and routing: determining the benefits of network coding and weighting them against the cost of adding network coding capability to traditional networks. Therefore, evaluating the benefits of network coding in different networks (e.g., optical and wireless) and traffic models (e.g., unicast and multicast) has been subject of

research. This dissertation contributes to network coding research by further investigation of network coding benefits in survivability and performance optimization in communication networks.

1.1 Dissertation Organization

The dissertation contains the outcome of network coding research performed by the author under supervision and with the support provided by Dr. Ahmed E. Kamal and Dr. Manimaran Govindarasu. In the first component, we investigated the application of network coding in improving the survivability of optical and backbone networks. The second component discusses benefits of network coding in wireless mesh networks in terms of rate, energy, and delay. The next chapters are organized as follows:

- Chapter 2 is dedicated to an introduction to network coding. We discuss simple examples showing the advantage of network coding in terms of rate, energy, and delay. We also explain the idea of robust network coding using an example. This chapter serves merely as an introduction and the interested reader may refer to various network coding tutorials (e.g., [32]) for further study.

- Chapter 3 represents our work on application of network coding in protection of unicast connections in optical networks. The work is based on the idea of 1+N protection [41]. In its simplest form, 1+N is a network coding generalization of the routing-based 1:N protection. The complexity of finding minimum cost 1+N solution is discussed. Due to NP-hard nature of the problem, optimal results may only be found in offline manner. We contribute by proposing heuristic, online algorithms for provisioning and protection of multiple unicast connections using 1+N protection. The work includes time analysis of algorithms, experimental evaluation of the solution cost of our algorithms, and comparison with routing-based 1+1 protection and optimal offline results. An analytical evaluation of the cost of the network coding solution is also provided.

- Chapter 4 is an original work on extending the idea of 1+1 protection to protect optical multicast connections using simple network coding. If 1+1 protection is directly applied

to multicast protection, the solution may include nodes that receive two or more identical incoming flows via diverse incoming links. In a routing solution, such nodes can only select and forward one incoming flow. Upon an upstream failure, the node must reconfigure its switch to forward another working flow. Network coding is shown to eliminate this problem by combining the incoming flows via logical *OR* operation. Since minimum cost multicast 1+1 solution is NP-hard to find, we present online heuristic algorithms in addition to the offline optimal formulation. The work includes study of related work including related connectivity problems, performance evaluation of heuristic algorithms, and comparison with offline optimal solution and a well-known routing-based algorithm.

- Chapter 5 presents the second component of our research, i.e., network coding benefits in wireless mesh networks. We investigate the advantages of using network coding in a wireless mesh network with limited bandwidth and energy resources. In particular, our work relies on the well-established theory of randomized network coding for multicast communication. We present joint formulation of scheduling interference-free wireless links and optimizing different performance parameters (namely rate, energy, and delay) in network coding and routing paradigms. We use these formulations to investigate rate-energy and rate-delay relationships in optimal settings. By investigating such dependencies under routing and network coding assumptions, we are able to isolate and show scenarios where network coding provides energy benefits. On the other hand, we show how both energy and delay change as non-linear functions of multicast rate demand; higher multicast rates result in increasingly more energy consumption and delay. We explain how optimal scheduling of wireless links is a hard sub-problem in the discussed performance optimization problems. In order to reduce the complexity of scheduling component, heuristic scheduling algorithms are proposed. We further show the effectiveness of scheduling heuristics experimentally and support the experimental results by theoretical insights.

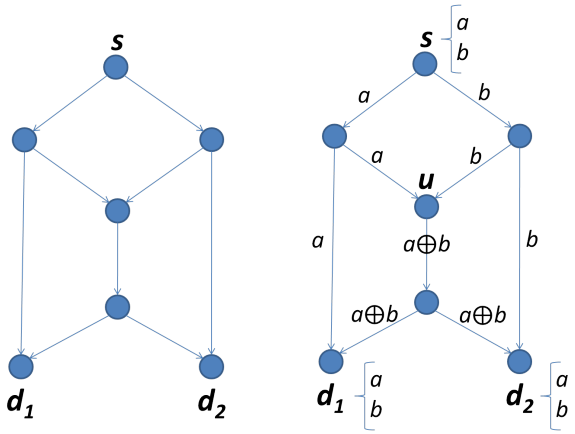
- Chapter 6 concludes the dissertation. A summary of contributions, achieved benefits through network coding, and future directions is presented.

CHAPTER 2. NETWORK CODING

The idea of network coding was presented in the fundamental work of Ahlswede et al. [6]. Network coding allows the outgoing flow of a node to be a function of its incoming flows. Network coding, therefore, extends the store-forward routing logic by enabling nodes to compute and transmit functions of incoming data. The main contribution can be summarized as max-flow min-cut theorem for network information flow (also referred to as main theorem of network coding). Simply put, the theorem states that given a multigraph $G(V, E)$ and a multicast connection with source s, and a set of k destination nodes $\{d_1, d_2, ..., d_k\}$, multicast rate $r = \min_i(maxflow(s, d_i))$ is achievable under network coding. Here, V denotes the set of vertices, E is the set of edges, and maximum flow from s to each destination d_i is denoted by $maxflow(s, d_i)$. Multicast rate cannot be higher than maximum flow from source to each destination. Therefore, it is upper-bounded by smallest maximum flow. The theorem proves that the upper-bound is, in fact, achievable.

In the next step Li *et al.* [59] showed that the multicast rate r can be achieved under linear network coding, i.e., when nodes generate linear functions of their incoming flows. In [53], an algebraic approach to network coding was introduced. The idea is to extract global coding matrices based on local coding vectors. Destination nodes can retrieve the original data once a full rank matrix of coefficients is received. In [35], authors propose algorithms to construct deterministic network codes in polynomial time. [64] shows that random linear network coding is capacity achieving in single unicast and single multicast packet networks. The paper addresses packet-level network coding as compared to symbol-level network coding. Packets experience erasure instead of error since network layer drops packets received in error, packet transmissions are not synchronized the way symbols are, i.e., time is not slotted, and packets carry side-information or header which could be used to store network coding coefficients.

Butterfly network The simplest example for the demonstration of network coding is the butterfly network. As figure 2.1(a) shows, butterfly network is used to model a multicast connection with source s and destinations d_1 and d_2. Suppose each directed edge has unit capacity. Under routing, the multicast rate would be 1.5. However, if intermediate node u is capable of generating a linear combination of data received on incoming edges, then the multicast rate of 2 units per second is achievable. Figure 2.1(b) shows the network coding example. Source s sends two data units a and b. Intermediate node u generates $a \oplus b$. Therefore, each destination would be able to decode both data units. For instance, d_1 receives a and $a \oplus b$. By performing an XOR operation, d_1 can retrieve b. It is not difficult to verify the main theorem of network coding here. Maximum flow from source to each destination is 2. Therefore, a multicast rate of 2 is achievable using network coding.



(a) (b)

Figure 2.1 Butterfly network

Relay network In a wireless network, broadcast links are used to transmit the same data to more than one receiver. In routing model, broadcast transmission can replace multiple single-cast transmissions. In network coding, the same broadcast functionality may be used to further transmit coded data. Figure 2.2 shows how routing and network coding compare in a simple relay network. Node u wants to send data unit α to node v via relay node r. Symmetrically, node v has data unit β to send to u via r. There are some assumptions:

transmission range of u and v does not allow direct communication but only through r, there is a single lossless channel, and each node has a single transceiver. Figure 2.2(a) shows how under these assumptions and routing, both data units can be delivered. In each time step, only one data unit (either α or β) can be sent over a single hop. As a result, it takes four time steps and four transmissions to complete the communication under routing.

If relay node r is capable of coding α and β, the number of transmissions can be reduced to three. As Figure 2.2(b) shows, after two time steps, r has both data units. Relay r then simply XOR codes the two data units into $\alpha \oplus \beta$. A single broadcast transmission is then enough to make sure both u and v have enough equations to solve for their intended data. In the case of u, for instance, it already has data unit α (its own data). Having received $\alpha \oplus \beta$, node u can solve for v. In this example, network coding can be seen as offering higher symmetric end-to-end rate. Instead of rate maximization, the benefit of network coding can be interpreted in the context of energy minimization. Simply put, network coding gets the job done with one less required transmission, which in this case is 25% less. Yet another way to look at the same benefit is in terms of delay minimization. This is straightforward as network coding requires one less transmission.

It is important to note that benefits of network coding come at a cost, i.e., computation cost. In other words, network coding trades transmissions with coding/decoding operations. In the relay example, one less transmission comes at the cost of one coding and two decoding operations. However, energy-wise such simple computations are considered less costly when compared to a wireless transmission.

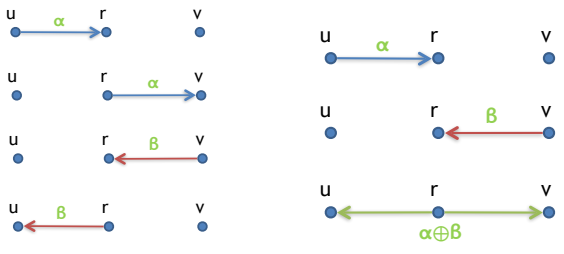


Figure 2.2 Relay network

Robust network coding Robust network coding, introduced by Koetter and Médard [53], is conceptually an extension of main network coding theorem to the case where edges are subject to failure. Such failures are modeled by removal of edges from the multigraph. Let us define a failure pattern f as a set of links failing together. One can apply the main theorem of network coding to $G_f(V, E\backslash f)$ to find the achievable rate r_f. Robust network coding extends this observation to a collection of failure patterns F. In particular, not only multicast rate of $r_F = \min_{f \in F}(r_f)$ is achievable under F but also there is a linear static network code that achieves this rate.

In other words, authors propose an algorithm for designing static (i.e. fixed) coding functions at intermediate nodes such that after occurrence of any failure pattern $f \in F$, all destination nodes would still be able decode all transmitted data symbols. The only requirement is that symbols are chosen from a finite field of size at least $r_F.k.|F|$. Other researchers have since contributed to robust network coding by improving the bound on the field size and proposing better polynomial algorithms (e.g. [35]).

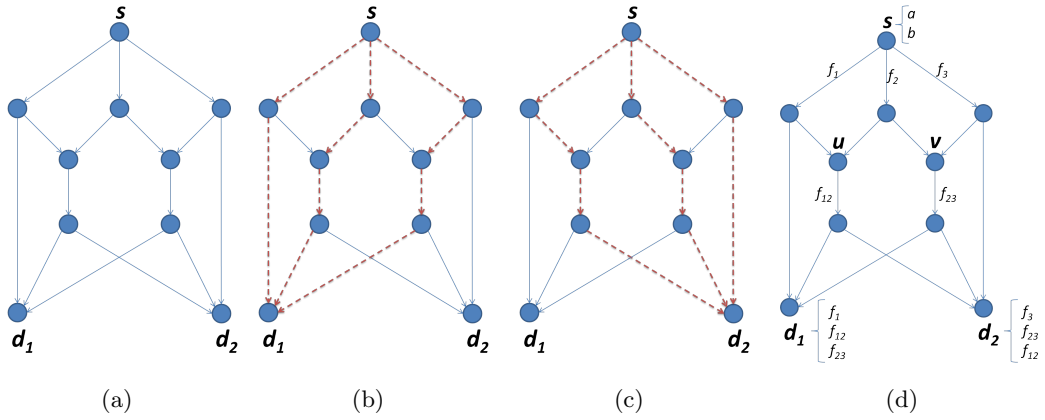


Figure 2.3 An example of robust network coding

Let us demonstrate the idea of robust network coding with a simple example. Figure 2.3(a) shows a multicast network with source s and two destination nodes d_1 and d_2. Assuming edges are unit capacity, maximum flow form s to each destination is 3 (Figures 2.3(b) and 2.3(c)). Moreover each maximum flow is carried by 3 edge-disjoint paths. Now consider the problem of single edge failure protection. A single edge failure would reduce the maximum flow to

at least one of destinations to 2. Therefore the multicast rate is dropped to 2 under single edge failure. A robust network coding algorithm would assign static linear network coding functions to network nodes, such that multicast rate of 2 is always achieved, no matter which edge fails. Figure 2.3(d) shows linear network coding functions at each coding node. Besides the source, there are only two other coding nodes i.e. only two nodes have more than one incoming flow. Other intermediate nodes would just forward the incoming flow (not shown in the figure). Multicast rate of 2 requires two units of data to be transmitted from s to both d_1 and d_2. We denote such two units of data by a and b. Three linear functions are generated at s: $f_1(a,b)$, $f_2(a,b)$, and $f_3(a,b)$. Coding nodes u and v generate functions f_{12} (a function of f_1 and f_2) and f_{23} (a function of f_2 and f_3). Ultimately each destination would receive 3 linear functions in two variables, a and b. For example d_1 receives f_1, f_{12}, and f_{23}. Static robust code design guarantees that given any single edge failure, at least two out three functions are linearly independent such that each destination would be able to decode both data units a and b. This neither involves any change is the coding functions nor rerouting of the flows, hence achieving instantaneous failure recovery. Note that in a traditional routing model, multicast rate of 2 is not always achievable under single edge failure. Moreover in the cases where it is achievable, it involves rerouting [1].

Network coding as technology Given the potential applications of network coding, one would expect research efforts to gradually result in technology that can improve real-world communication networks. Here, we look at some of the recent developments.

Since 2013, Internet Research Task Force (IRTF) has started a research group (RG) on network coding [34]. The charter indicates interest in areas where Internet can benefit from network coding and current status of practical implementations of network coding. One of the current ongoing publications is a network coding taxonomy [28].

In the industry section, Code On Technologies is a company founded by some of the influential researchers in network coding [3]. Since 2011, Cone On has been working of producing technology based on network coding. Given various potential applications of network coding,

[1] Based on [45].

the company has developed industrial partnerships to offer an ecosystem of network coding-based services. Partnering companies offer a range of services: software libraries for network coding such as Kodo [73] by Steinwurf [5], cloud services by Danish-based start-up Chocolate Cloud [2], faster WiFi services for venues and events by APSI WiFi [1], and more. Microsoft corporation also launched a research project on using network coding for internet-scalable file sharing and distribution (see project Avalanche [31][4]).

CHAPTER 3. 1+N PROTECTION IN POLYNOMIAL TIME: A HEURISTIC APPROACH

Modified from a paper published in the proceedings of GLOBECOM 2010 [69].

3.1 Abstract

The generalized 1+N protection [42], protects N unicast connections by a single Steiner tree connecting all end points of the connections. By sending network coded packets on the protection Steiner tree in parallel with the working traffic, 1+N is able to recover from any single link failure without enduring the delay from switching to the backup path. Optimal cost provisioning and 1+N protection of a given set of connections is an NP-hard problem comprising of three NP-hard subproblems: partitioning of the connections, finding edge disjoint primary paths and Steiner tree protection circuit for the subset of connections in each partition. In this paper a polynomial time heuristic algorithm for 1+N protection is proposed which combines heuristic steps to address the three NP-hard components of the problem. Our simulations show that the heuristic algorithm provides average cost reduction of 29.2% and 18.5% compared to 1+1 protection in COST239 and NSFNET networks. An asymptotic bound is also derived for the case of complete graph networks which shows that 1+N can achieve maximum of 66.6% cost improvement compared to 1+1. When compared to the optimal 1+N solution from ILP formulation, the heuristic algorithm increases the cost no more than 13%.

3.2 Introduction and Related Work

The 1+N protection method ([40][41]) protects multiple unicast connections against single link failures by performing network coding ([6]) over a single p-cycle ([79]) as the backup circuit.

Compared to the traditional 1:N protection, the application of network coding allows 1+N to provide lower failure recovery time while using the same backup capacity. The connections have to be provisioned using link disjoint paths. The p-cycle which protects these connections passes through all end points of the connections and has to also be link disjoint from all the connections. The connections and p-cycle itself are bidirectional and are assumed to be of the same capacity. Each end point receives and transmits coded backup data in two opposite directions on the p-cycle (called half p-cycles). Upon the failure of a connection (in result of a single link failure), 1+N scheme makes sure that end points of the corresponding connection can recover their intended data (for a specific round of communication) simply by *xoring* the coded data received on the two half p-cycles and their own data (of the same round).

In [43], the author extends 1+N scheme to protect against multiple link failures. In order to protect a group of connections against M simultaneous link failures, M link disjoint p-cycles are used. The idea is to have enough linearly independent equations received at each end point of each failed connection so that the end points can recover their intended data by solving the system of equations.

In [44] and [42], the single failure protection version of 1+N is extended to a more general protection circuit which might not necessarily be a cycle. Hence, the constraint of having a p-cycle as the backup circuit is relaxed. While [44] considers only unidirectional connections and gives a general description of the protection circuit, in [46], the idea of 1+N protection was extended to overlay protection. The paper addresses multiple link failures in addition to single link failure, and offers simpler protection circuit. In [60], the idea is further extended to protect again adversarial errors in addition to link failures. In [42], the authors show that the optimal 1+N protection circuit for a given set of bidirectional unicast connections is a tree. More specifically it is a Steiner Minimal Tree (SMT) that connects all the end points of the connections and has the same bidirectional bandwidth as the connections. To guarantee single link failure protection, the connections have to be provisioned using link disjoint paths and the Steiner tree must also be link disjoint from all the connections. The authors further show how 1+N can actually be implemented on top of the Steiner tree by rooting the tree at one specific

node, referred to as node X, and defining two flow directions with respect to node X: from the leaf nodes toward X (upstream) and from X toward leaf node (downstream).

In the upstream direction, each end point of each bidirectional connection locally *xors* the transmitted and received data packets corresponding to each communication round. These locally coded packets are sent toward the node X on the Steiner tree. Each non-leaf node simply *xors* all incoming coded packets (and its locally coded packet if it is in fact an end point) into one packet and sends it up toward the node X. Ultimately the node X *xors* all the coded packets it receives. Under normal failure-free conditions, node X will simply get a zero packet; since data packets coming from two end points of the each connection will cancel each other. In the case of a single failure, end points of the failed connection will receive nothing (a zero packet) from the other end point and their locally coded packets would simply be their own data packet for that communication round. Therefore once all coded data packets reach the node X and added together, the node X will be left with one final packet which is the *xor* of two data packets sent from end points of the failed connection.

While the upstream information flow involves collection and coding of data packets, in the reverse direction node X simply sends the final packet toward leaf nodes and each intermediate node just forwards the received packet in that direction; no coding occurs. End points of the failed connection can then recover their intended data for each communication round by adding the received coded data on the Steiner tree in the downstream direction to their own data packet of the same round.

Figure 3.1 gives an example 1+N protection scenario in which three connections (S_1, D_1), (S_2, D_2), and (S_3, D_3) are protected using a Steiner tree subgraph. For simplicity, the Steiner tree is shown to be symmetric around the node X. As the figure shows, there is a failure on working path of connection (S_2, D_2); D_2 receives nothing from S_2. In other words, a zero is received at D_2 instead of data packet b. The sum expressions on each link represent the coded packets in the upstream direction. The node X adds two received upstream packets, $a \oplus b \oplus c$ and $a \oplus c$, to get b. It then sends b back in the downstream direction to all destinations. Clearly, D_2 is the only one in need of b (for simplicity, only path to D_2 is shown). The same example can be extended for protection of bidirectional connections.

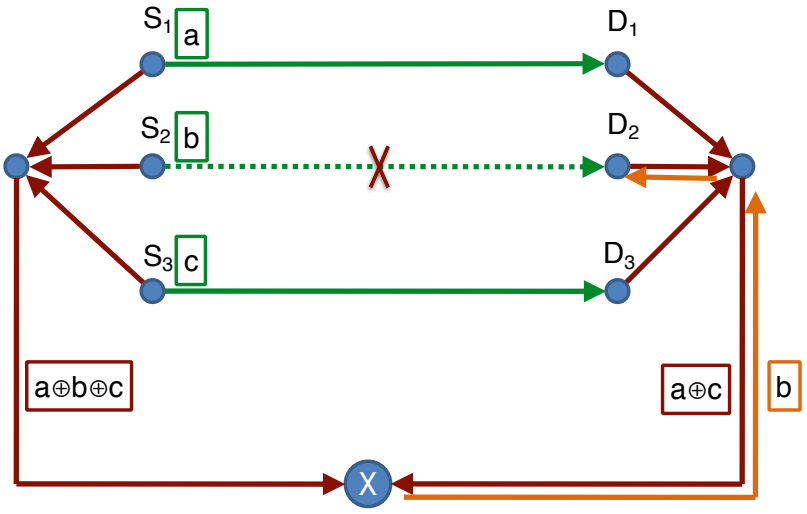


Figure 3.1 An example of 1+N protection

The preceding 1+N scheme provides 100% protection against any single link failure. Compared to traditional 1:N protection technique which acheives the same protection level at the same cost, it offers the advantage of having much lower recovery time. Compared to 1+1 protection technique which offers instantaneous recovery from single failures, 1+N presents near instantaneous recovery at lower cost: N connections are protected by single protection circuit in 1+N while each connection requires a dedicated disjoint protection path in 1+1.

Optimal cost 1+N solution, however, is not easy to find. Simply trying to protect all connections together will not necessarily give the optimal cost (it could even be infeasible). The first step, therefore, is to partition the set of connections. The subset of connections in each partition are then protected together. To find the optimal partitioning is NP-hard [8]. Even after partitioning is done, provisioning link disjoint paths and Steiner tree protection of the subset of connections in each partition are still NP-hard problems [30][84].

Therefore we revert to polynomial time heuristic algorithms to solve the three NP-hard components of the problem. The suboptimal cost of 1+N protection is then compared to optimal cost of 1+1 for real world networks. To have an idea of how well those heuristic algorithms perform compared to the optimal 1+N protection, an analytical-experimental study is presented for the case of complete graphs.

Section 3.3 introduces basic models and assumptions. In Section 3.4 problem statement and algorithm design are presented. Simulation results and experimental comparison between 1+N and 1+1 are shown in Section 3.5. Our analytical bound on the performance of 1+N compared to 1+1 is given in Section 3.6. Section 3.7 concludes the paper.

3.3 Models and Assumptions

An optical network is modeled as an undirected graph $G(V, E)$ with V as the set of nodes and E as the set of undirected edges. Each edge represents a fiber link. Edge capacity represents the number of wavelength channels per fiber link. All edge capacities are assumed to be equal.

A set of κ bidirectional connections C is defined as

$$C = \{(s_i, t_i) | s_i, t_i \in V, s_i \neq t_i, 1 \leq i \leq \kappa\} \ . \tag{3.1}$$

All connections are assumed to demand unit capacity equal to one wavelength channel. This means that a single unit of capacity (equal to the bandwidth of one wavelength channel) is enough to carry the traffic of one and only one connection, i.e. no traffic grooming is allowed.

We further assume that the edge capacity is not a limiting constraint in provisioning connections or protection circuit. This assumption reflects the fact that each fiber link has huge amount of bandwidth.

We also define one unit of cost as one unit of capacity used on one edge. Therefore the cost of provisioning a connection is equal to reserving one unit of capacity on a simple path connecting end points of the connection, i.e. is equal to length of the path (since each connection demands one unit of capacity per each edge).

3.4 Problem Statement and Algorithms

Given the network graph G and the set of connections C, the main problem is to provision and protect all connections against any single link failure using the technique of 1+N at minimum cost. As stated before, the optimal solution involves the following two steps:

1. Optimal partitioning of the set of connections: The partitioning determines which connections should be protected together, i.e., the subset of connections in each partition are

protected using the same Steiner tree. Different partitions are provisioned and protected independently, therefore, the total cost associated with a partitioning is equal to the sum of individual partitions costs. Minimum cost partitioning is an instance of famous Set Partitioning Problem (SPP) which is NP-hard [8].

2. Minimum cost provisioning and protection of each partition: This problem is comprised of two NP-hard subproblems namely, minimum cost edge disjoint paths [30] and Steiner Minimal Tree [84]. Since the optimal solution to the problem requires solving the two subproblems jointly, it is at least as hard as the hardest of the two subproblems, i.e., NP-hard.

Due to the exponential time nature of the problem, the ILP formulation of the problem as an optimization problem can only find the optimal solution for small networks and a few number of connections in a reasonable amount of time [42]. The way to solve real world instances of the problem is to revert to efficient heuristic algorithms. We start by designing a heuristic algorithm for the partitioning step.

Since there are exponentially many ways to partition a set of given connections, a polynomial time algorithm should not try to check all possible partitions. Two extreme cases are: I) Single partition which includes all the connections. In this case all connections are provisioned using edge disjoint paths and a single edge disjoint Steiner tree is used to protect all connections. II) Each connection is a separate partition and protected separately; Steiner tree in this case is simply a secondary path edge disjoint from connection's primary path. This is in fact equivalent to 1+1 protection; 1+1 protection is included as a special case of 1+N protection in the solution space. It is worth noting that the number of connections in a partition may be limited by the network graph connectivity since to provision and protect a larger partition would require more "disjointness".

Algorithm 1 shows our greedy partitioning algorithm. The COST function returns the cost to provision and protect a partition. The algorithm starts by a new empty partition p as the current partition. The first connection to be added to a new partition is the one whose COST is minimum among all remaining connections (lines 3 to 6). The cost returned by COST function

for such a single-connection partition is equal to the cost of 1+1 provision and protection (which is found using Bhandari's algorithm [11] and is optimal).

The algorithm then greedily chooses the next connection c to be added to the current partition p in such a way that the cost of new partition is locally minimized (line 8). A connection c is considered a candidate only if the cost of new partition formed by adding c to the current partition ($COST(p \cup \{c\})$) is less than the total cost of considering c as single-connection partition ($COST(\{c\})$) plus the cost of current partition $COST(p)$. If no such candidate connection exists (line 12) the current partition p is considered as complete and is included in the final output partitioning P (line 13). The algorithm stops when all connections are covered. P is the partitioning of the connections.

Algorithm 1 Greedy algorithm to find a partitioning of connections. The COST function returns the cost of provisioning and 1+N protection of a partition.

Input: G(V,E): network graph, C: set of connections
Output: P: partitioning of connections

1: $P \leftarrow \emptyset, p \leftarrow \emptyset$
2: **while** $C \neq \emptyset$ **do**
3: **if** $p = \emptyset$ **then**
4: $cmin = argmin_{c \in C} \{COST(\{c\})\}$
5: $p \leftarrow \{cmin\}$
6: $C \leftarrow C \backslash cmin$
7: **else**
8: $C_p = \{c \in C | COST(p \cup \{c\}) < COST(p) + COST(\{c\})\}$
9: $cmin = argmin_{c \in C_p} \{COST(p \cup \{c\})\}$
10: **if** $cmin \neq 0$ **then**
11: $p \leftarrow p \cup \{cmin\}$
12: $C \leftarrow C \backslash cmin$
13: **else**
14: $P \leftarrow P \cup p$
15: $p \leftarrow \emptyset$
16: **end if**
17: **end if**
18: **end while**
19: $P \leftarrow P \cup p$

The underlying component of above algorithm is minimum cost provisioning and protecting of a partition (COST function) which is an NP-hard problem (consisting of two NP-hard subproblems). We use the following heuristic steps to solve this problem:

1. The problem is split into two separate subproblems: Provisioning minimum cost edge disjoint paths and finding minimum cost Steiner tree for subset of connections in the partition.
2. Two heuristic algorithms are used to solve the subproblems: Greedy Shortest Paths algorithm [54] and Greedy Steiner Tree algorithm by Takahashi [81].

Algorithm 2 shows the Greedy Shortest Paths algorithm [54]. The algorithm tries to find a set of minimum cost edge disjoint paths for the subset of connections in a partition p. In each round it finds the connection with the minimum length shortest path among all remaining connections, routes the connection, and removes the route from the graph to guarantee edge disjointness. The Greedy Shortest Paths algorithm does not guarantee that edge disjoint paths will be found for all connections in the partition (even if they are actually feasible to find). When the next shortest path does not exist (line 4) the algorithm returns an empty set of routes. In other words it only returns successfully if edge disjoint paths could be found for all connections in the partition.

Algorithm 2 Greedy shortest paths algorithm.

Input: G: network graph, p: a partition
Output: R: set of edge disjoint routes for p

1: $R \leftarrow \emptyset$
2: **while** $p \neq \emptyset$ **do**
3: $min = argmin_{c_i \in p} \{|r_i|\}$
$\{r_i$ is the shortest path route of connection c_i in $G\}$
4: **if** $|r_{min}| = \infty$ **then**
5: return $\emptyset$
6: **else**
7: $R \leftarrow R \cup r_{min}$
8: $G \leftarrow G \backslash r_{min}$
9: **end if**
10: **end while**
11: return R

The Greedy Steiner Tree algorithm by Takahashi [81] (Algorithm 3) starts by a terminal node (end point node of a connection) as the current subtree (line 2) and continuously finds the next closest terminal node to the current subtree (line 5) and connects it to the subtree by a shortest path (line 9). In the case that the Steiner cannot be found, at some point the

distance of the next closest terminal would become infinity (line 6) and an empty tree would be returned.

Algorithm 3 Greedy Steiner tree algorithm.

Input: G: network graph, V_p: set of end points of connections in partition p
Output: T: Steiner tree

```
1: pick an arbitrary v ∈ V_p
2: T ← v
3: V_p ← V_p\v
4: while V_p ≠ ∅ do
5:    w = argmin_{u∈V_p} {|r_u|}
      {r_u is the shortest path route between u and T}
6:    if |r_w| = ∞ then
7:       return ∅
8:    else
9:       T ← T ∪ r_w
10:      V_p ← V_p\w
11:   end if
12: end while
13: return T
```

In our partitioning algorithm (Algorithm 1), for each partition the COST function runs Greedy Shortest Paths algorithm to find a set of edge disjoint paths. Upon success, it runs Greedy Steiner Tree algorithm on the residual graph after removing all paths. This is to guarantee that the Steiner tree is disjoint from connections paths. Only if both steps are successful, a finite cost value will be returned by the COST function.

While the time complexity of the COST function depends on the specific implementation of each of the heuristic algorithms, the worst case time complexity of Algorithm 1 is $O(|C|^2.T_{COST})$ where T_{COST} represents time complexity of the COST function. In our implementation T_{COST} is $O(|V|^2.|C|^2)$ therefore the total worst case time complexity is $O(|V|^2.|C|^4)$.

3.5 Simulation Results

Two real world networks 14-node NSFNET and 11-node COST239 and one artificial 14-node complete graph network are used in the simulations. The total cost of our heuristic algorithm for provisioning and 1+N protection of a given set of connections is compared to the same cost when 1+1 technique is used.

Connections are randomly generated for each network. To observe the effect of the number of connections in each network and for each scheme, the number of randomly generated connections is varied from 1 to a number close to maximum number of connections in each network, i.e., $|C|$ is close to $\binom{|V|}{2}$. For example in the case of 14-node NSFNET, $|C|$ takes values from 1 to 90 ($\binom{14}{2} = 91$). A set of connections of specific size is randomly generated 100 times, then the cost of each scheme is averaged over all rounds and reported. One unit of cost is defined as one unit of capacity on an edge.

Figures 3.2 to 3.7 present the cost and percentage of the cost reduction for NSFNET, COST239, and complete graph network. Percentage of the cost reduction represents relative improvement in total cost when 1+N is compared to 1+1. In all figures, the horizontal axis represents the number of randomly generated connections.

Figures 3.2 and 3.3 show the results for the NSFNET where maximum of 90 connections are generated. While 1+N always performs at least as well as 1+1, the maximum cost reduction (21.5%) is achieved when the maximum number of connections is considered. In Figures 3.4 and 3.5 the performance of 1+N on the COST239 network is shown. Here again the maximum cost reduction (34.5%) is achieved when maximum number connections (55) are generated. The reason that 1+N performs better in COST239 compared to NSFNET has to do with the edge-to-node ratio (edge density) of the networks: 19/14 and 26/11 for NSFNET and COST239 respectively. Intuitively a network with higher edge to node ratio would have more 1+N potential, i.e., it is more likely to have larger feasible partitions (more connections provisioned and protected together).

The expected trend continues when we look at the simulation results of complete graph network (Figures 3.6 and 3.7). To show the effect of edge density on 1+N performance better, a complete graph with the same number of nodes as NSFNET (14 nodes) is simulated. The maximum cost reduction increases from 21.5% in NSFNET (with 19 edges) to 60.2% in the complete graph (with 91 edges). The same edge density effect is observable by comparing the cost reduction corresponding to a given number of connections in the Figures 3.3, 3.5, and 3.7.

Figure 3.8 summarizes and compares the percentage of the cost reduction in NSFNET, COST239, and complete graph network. The horizontal axis represents the number of randomly

generated connections (between 1 and 65). The diagram shows up to 55 connections for 11-node COST239 because that is the maximum number of possible connections given 11 nodes. Each point in the digram is the averaged value over 100 rounds of simulation.

The following observations are made from figures:

- Both costs (1+1 and 1+N) seems to be linear in terms of number of connections.
- 1+N performs better as the number of connections increases. Intuitively this increases the potential to protect more connections together and reduce the total cost.
- 1+N performs better in networks with higher edge density. The graph densities are 19/14 in NSFNET, 26/11 in COST239 and 91/14 in complete graph; more "disjointness" potential in the network makes larger partitions possible.

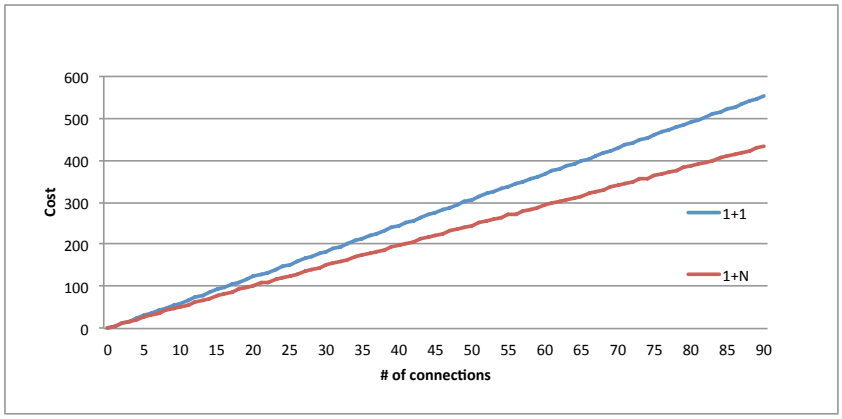


Figure 3.2 Total cost of 1+N and 1+1 in 14-node NSFNET.

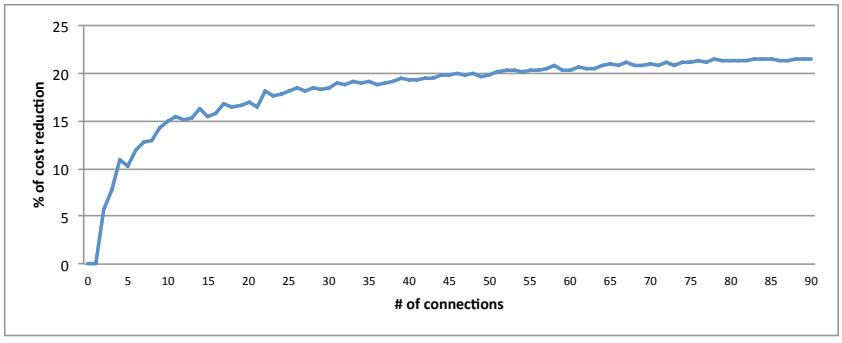


Figure 3.3 1+N cost reduction compared to 1+1 in 14-node NSFNET.

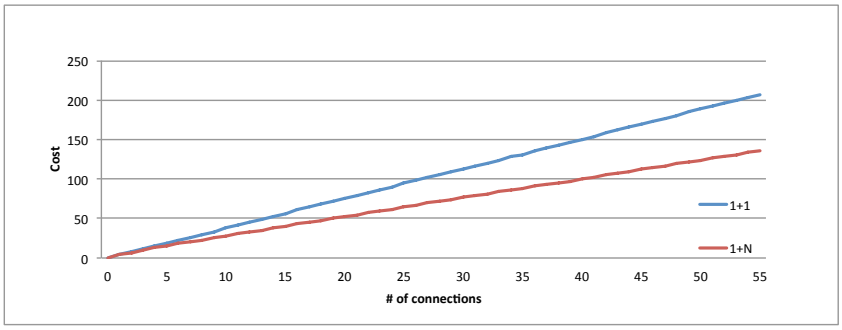


Figure 3.4 Total cost of 1+N and 1+1 in 11-node COST239.

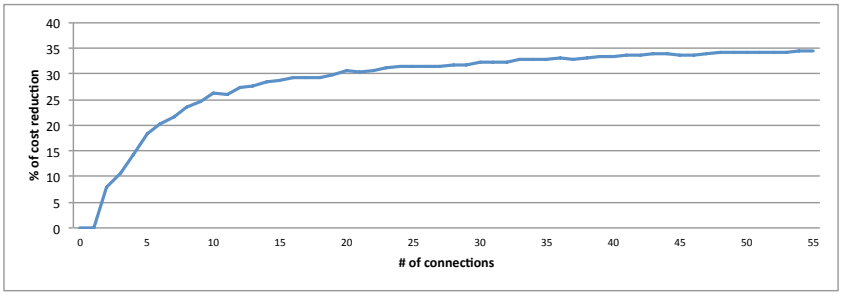


Figure 3.5 1+N cost reduction compared to 1+1 in 11-node COST239.

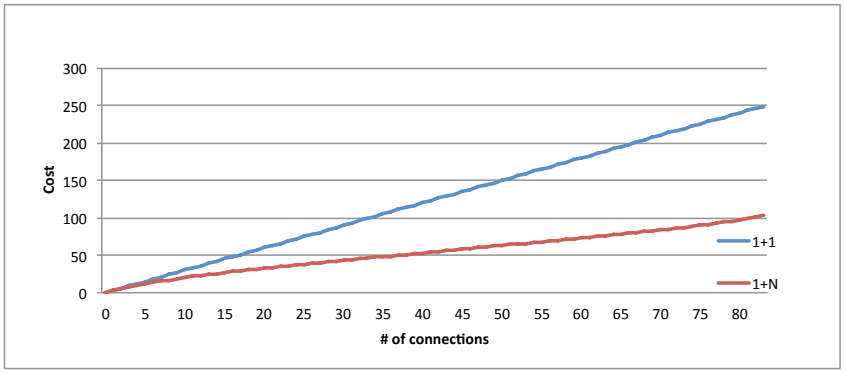


Figure 3.6 Total cost of 1+N and 1+1 in 14-node complete graph.

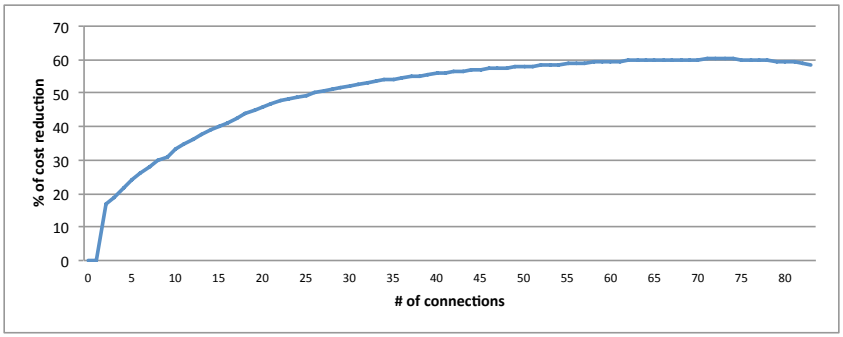


Figure 3.7 1+N cost reduction compared to 1+1 in 14-node complete graph.

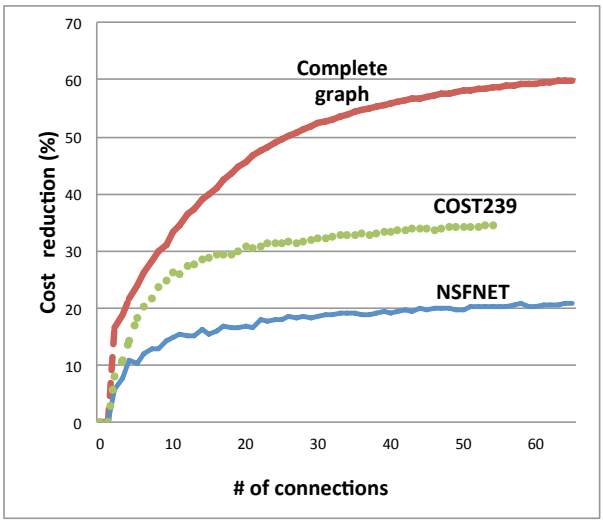


Figure 3.8 1+N cost reduction compared to 1+1 in 3 different networks.

Table 3.1 summarizes the simulation results regarding the cost efficiency of 1+N with respect to 1+1 in the three simulated networks. The numbers are averaged over 100 rounds of simulation. For each network maximum and average percentage of cost reduction is given.

Table 3.1 1+N cost reduction over 1+1 in 3 different networks.

Cost reduction (%)	NSFNET	COST239	Complete
Max	21.5	34.5	60.2
Average	18.5	29.2	50.7

The performance of our heuristic algorithm for 1+N is also compared to the optimal 1+N results obtained from an ILP formulation of the problem (we use a revised version of the ILP in [42] which runs faster). Since the optimal solution requires exponential time in terms of number of connections, the comparison can only be made for few cases with limited number of connections. Table 3.2 presents the results for two cases of 5 and 10 randomly generated connections in NSFNET and COST239 as two practical networks. The cost value reported for 5 connections is averaged over 10 instances while in the case of 10 connections we could only run one instance. N is the number of connections. Degree of suboptimality is the percentage of cost increase when heuristic algorithm is compared to optimal solution.

Table 3.2 Cost of 1+N : Heuristic vs. ILP.

Network	N	Heuristic	ILP	Degree of suboptimality (%)
NSFNET	5	27	26	3.8
	10	52	46	13
COST239	5	14.8	14	5.7
	10	26	25	4

3.6 Asymptotic Analysis

Based on two observations made earlier on 1+N performance, we consider an asymptotic analysis. The best scenario, which we expect to give the best 1+N cost efficiency compared to 1+1, would then be to consider a complete graph (densest graph) with maximum number of connections possible. If we let the number of nodes go to infinity, asymptotic cost efficiency of 1+N versus 1+1 would be achieved.

A complete graph $G(V, E)$ with $|V| = n$ nodes has $\binom{n}{2} = \frac{n(n-1)}{2}$ edges which is equally the maximum number of possible distinct connections. Provisioning each connection takes only one unit of cost. Total cost of provisioning all connections, therefore, is $\frac{n(n-1)}{2}$. We consider this cost as the fixed minimum provisioning cost independent of the protection scheme used.

Protection of each connection using 1+1 scheme requires two units of cost because the protection path has to be edge disjoint from the primary path. Given that primary path is provisioned using the single edge connecting the end points of connection, shortest protection path should traverse two edges to be edge disjoint. Hence protection cost of 1+1 is $n(n-1)$ and total cost of provisioning and protection using 1+1 scheme is $\frac{3}{2}n(n-1)$.

Now we look at the cost associated with 1+N. As mentioned above, we assume that the cost of provisioning connections is fixed ($\frac{n(n-1)}{2}$). Again our approach is to predict shape of optimal 1+N solution through finding optimal partitioning of connections.

We are considering a complete graph where every edge represents a connection's primary path and the set of connections end points is equal to V. A Steiner minimal tree that connects all end points in this case is a simple path of length $n-1$. Using such a path all the remaining connections (edges) can be protected using 1+N technique. In other words the cost of protecting the first partition which consists of $\frac{n(n-1)}{2} - (n-1)$ connections is just $n-1$. While we may try to figure out what is the minimum cost partitioning to protect the remaining $n-1$ connections, there is a more important observation to make: even if 1+1 is used to protect the remaining connections (as a special case of 1+N), the total 1+N protection cost would still be linear in terms of n. In fact the total cost in this case is $n - 1 + 2(n-1) = 3n - 3$.

Therefore in a complete with maximum number of connections possible, the protection cost of 1+1 is in the order of number of edges (n^2) while protection cost using 1+N is in the order of number of nodes (n). Asymptotic total cost (provisioning and protection) ratio of 1+N to 1+1 is as follows:

$$\lim_{n\to\infty} \frac{\frac{n(n-1)}{2} + (3n-3)}{\frac{3}{2}n(n-1)} = \frac{1}{3} \tag{3.2}$$

In terms of percentage of cost reduction, this means that 1+N asymptotically needs 66.6% less resources compared to 1+1. This result is in compliance with the simulation results on the complete graph which showed maximum 60.2% of cost reduction. It also proves the efficiency

of our heuristic algorithm on complete graphs which is capable of achieving a performance very close to the asymptotic bound.

3.7 Conclusion

A heuristic algorithm for minimum cost provisioning and 1+N protecting of a given set of connections is presented. The core idea is to greedily partition the given set of connections such that total cost is minimized. The subset of connections in each partition are independently provisioned and protected using Greedy Shortest Paths and Greedy Steiner Tree heuristic algorithms. Performance of the algorithm is evaluated both experimentally by simulating different network scenarios and analytically by finding an asymptotic bound. The simulation results show that cost efficiency of our heuristic 1+N algorithm with respect to 1+1 increases when the number of connections or graph density is increased. Given the fact that the comparison was made between a suboptimal algorithm for 1+N scheme and an optimal algorithm for 1+1 scheme, our results show maximum cost savings of 21.5%, 34.5%, and 60.2% in 14-node NSFNET, 11-node COST239, and 14-node complete graph networks. Moreover the suboptimal cost found by the heuristic algorithm shows at most 5.7% and 13% increase of cost in the case of 5 and 10 connections (respectivley) compared to the optimal 1+N results from ILP formulation of the problem. The final contribution of this paper is an asymptotic bound which shows 1+N can achieve 66.6% cost reduction compared to 1+1 in complete graphs.

CHAPTER 4. MULTICAST 1+1 PROTECTION: THE CASE FOR SIMPLE NETWORK CODING

Extended and modified from a paper accepted for publication in the proceedings of ICNC 2015 [70].

4.1 Abstract

We discuss how the idea of unicast *1+1* protection can be efficiently extended to protect multicast connections in optical backbone networks. Particularly, we show how to achieve instantaneous failure recovery and cost efficiency by allowing intermediate nodes to merge their incoming flows by a simple network code, i.e., logical OR operation. Under simple network coding, the problem of minimum cost multicast *1+1* protection is formulated as a 2-connectivity problem. In order to solve this problem, an optimal ILP and three efficient heuristic algorithms are proposed. Simulation results on real-world networks show that the average cost of our best heuristic algorithm is only 2.6% higher compared to the optimal ILP solution.

4.2 Introduction

Multicast is a one-to-many traffic model in which a source node transmits the same information to a set of destination nodes. Such traffic model is used in backbone networks for provisioning high data rate applications such as Internet TV (IPTV) [14][91], distribution of financial information [67], and data dissemination in cloud and grid computing [15]. Many such applications demand highly available always-on connections. Underlying backbone networks, on the other hand, are subject to service disruption because of link and component failures. Moreover, even a single link failure can disrupt the connection to multiple destination nodes

in the multicast traffic model. Therefore efficient multicast protection techniques that satisfy availability requirements are needed.

While multicast protection has been the subject of extensive studies, most of proposed techniques are tuned for non-instantaneous recovery, i.e., when a certain amount of delay in recovery is acceptable. In this paper we focus on the problem of multicast protection with instantaneous failure recovery.

Dedicated *1+1* protection has been commonly used in optical networks to provide instantaneous failure recovery against single failures for unicast connections. A pair of disjoint primary and backup routes are used to deliver two copies of each data unit from source to destination simultaneously. Failure of one route therefore causes no service disruption. If a multicast connection was provisioned as a set of independent unicast connections, the same technique could be directly applied to each connection. However, clearly that does not provide a cost efficient solution. When a tree is used to provision a multicast connection, a natural generalization of the idea of *1+1* protection is to have a pair of disjoint primary and backup trees connecting source to all destinations. While the cost efficiency is improved here, the required connectivity could be higher. If the network is 2-connected, one can always find a pair of disjoint paths between the source and each destination node but not necessarily a pair of disjoint trees.

The idea of *1+1* protection has been used in [78] to design a cost-efficient multicast protection technique against single link failures. A minimum cost disjoint path pair, Optimal Path Pair (*OPP*), is found from source to each destination. In order to reduce the cost, path pairs to different destinations are allowed to share bandwidth on common links. Figure 4.1 shows an example of how *OPP* works. A multicast connection is given with source node s and destination nodes d_1 and d_2 on a Butterfly network. Assuming that one unit of capacity is reserved on each link of each path to each destination, the total reserved capacity without sharing is $6 + 6 = 12$ in unicast *1+1* protection. In the case of *OPP*, total reserved capacity would be reduced to $12 - 3 = 9$ because of sharing on links (s, u), (s, v), and (w, x). However the capacity improvement by *OPP* has a negative effect on the recovery delay.

Even though *OPP* finds two disjoint paths per destination node, because of the link sharing it cannot send two disjoint flows to all destination nodes. In Figure 4.1, node w can only forward

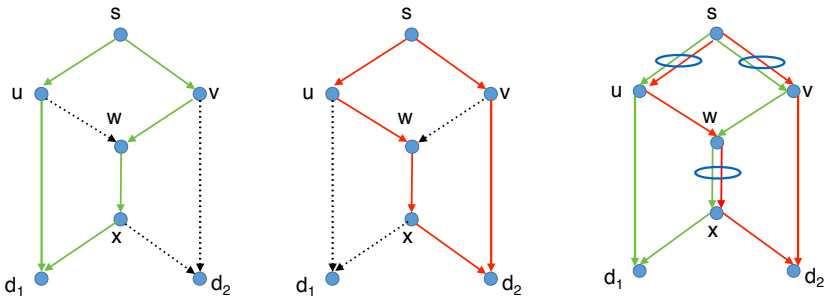


Figure 4.1 Unicast *1+1* protection vs. *OPP*

one of the incoming flows from nodes u and v because only one unit of capacity is reserved on link (w, x). Figure 4.2 shows this situation. Suppose node w chooses to forward flow from v to the downstream node x, then destination node d_1 will receive two disjoint flows but that is not the case for node d_2.

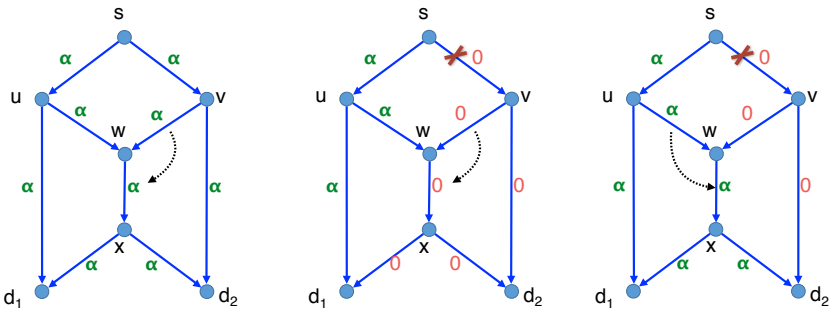


Figure 4.2 Required switching at intermediate node w

Given a failure on link (s, v), node d_1 will still receive data through path $(s-u-d_1)$ however node d_2 will be totally disrupted. It is only after node w realizes that incoming flow from node v has failed and switches to incoming from node u that the flow to node d_2 could be restored through path $(s-u-w-x-d_2)$ (Figure 4.2). The recovery delay in this case is due to *OPP* not delivering two disjoint copies of each data unit to each destination. The same situation could happen for multiple intermediate nodes in a general multicast network.

In this paper, we propose a solution to the multicast *1+1* protection problem that offers both cost-efficiency of sharing and instantaneous recovery of unicast *1+1* protection. In order to eliminate the recovery delay due to the sharing, our solution introduces the idea of merging

flows at intermediate nodes which can be implemented using simple logical OR operation. The problem of minimum cost multicast *1+1* protection is then formulated as an ILP. The optimal results obtained by solving the ILP model are compared with two efficient heuristic algorithms for multicast *1+1* protection.

The rest of the paper is organized as follows: Section 4.3 explains the main idea behind multicast *1+1* protection by an example. Section 4.4 reviews the related work in the fields of multicast protection in networks and connectivity problems in graph theory. Section 4.5 presents the assumptions and problem statement. In Section 4.6, we present optimal formulation of the problem. Section 4.7 describes the heuristic algorithms. Simulation results for optimal and heuristic algorithms are presented in Section 4.8. Finally Section 4.9 concludes the paper.

4.3 The Idea

As discussed earlier, the recovery delay in *OPP* is due to the fact that the failure will have to be detected, intermediate nodes (such as w in Figure 4.2) are signaled, and switches will have to be reconfigured. All of this can take tens of milliseconds. This delay can be avoided if the intermediate node merges the incoming flows into one outgoing flow by a simple logical OR operation. As Figure 4.3 shows, under normal operation node w would OR equal data units from u and v ($\alpha + \alpha = \alpha$). This results in a single data unit which is forwarded to node x. In case of a single failure on an upstream link, e.g., (s, u) or (s, v), node w receives an empty packet (equally a zero data unit) on one link and α on the other. The OR operation would still produce α. Therefore the outgoing flow on (w, x) will not be affected by any upstream single link failure. Both destinations, as a result, will at least receive one copy of α under any single link failure. This example shows how merging flows at intermediate node w, makes it possible to have the benefits of sharing (cost efficiency) and dedicated protection (instantaneous recovery) at the same time. In general, merging happens whenever an intermediate node has multiple incoming flows (belonging to the same multicast connection) that share an outgoing link.

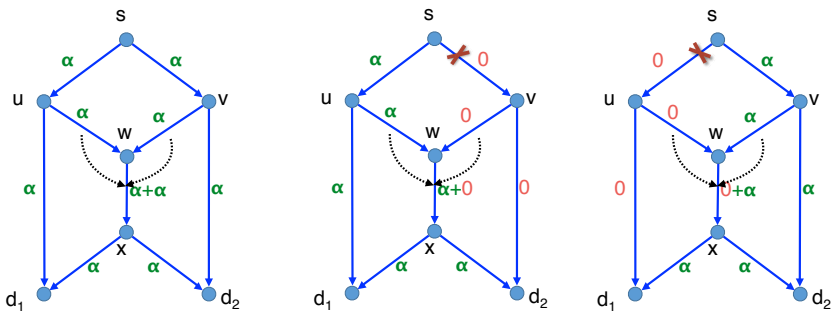


Figure 4.3 Multicast *1+1* protection using merging flows

4.4 Related Work

Due of the nature of our problem, we consider related work in two domains of research: 1- network protection and survivability and 2- connectivity problems in graphs. While the former focuses on more practical aspects of network and traffic, and offers offline optimal solutions and fast online heuristic algorithms, the latter takes a more theoretic approach and focuses on complexity analysis and approximation bounds.

4.4.1 Multicast protection with instantaneous recovery

In [78] and [93] two different classifications of general multicast protection techniques are given. Here we focus on multicast protection techniques that offer instantaneous failure recovery. Due to the recovery requirement, such techniques are mainly categorized as dedicated protection techniques. Therefore in the case of multiple sessions, there is no inter-session backup sharing and each session is protected independently.

In [78] and [93] two different classifications of multicast protection techniques are given. Here we give a classification based on whether an approach includes a primary tree or not.

With Primary Tree Normally a Steiner Minimal Tree heuristic such as Prim-based heuristic [74] or MPH (Minimum Path Heuristic) [81] is used to find the primary tree. Protection of the primary tree can take different forms:

- Tree-based: The idea is to protect the primary tree with an edge, link or node disjoint secondary tree. A dual-tree approach was proposed in [26]. A primary tree is protected

by secondary (node or link) disjoint tree that connects all the leaf nodes. More recently in [20] authors propose a novel Steiner tree heuristic called Steiner Node Heuristic (SNH). SNH incrementally adds non-destination nodes to the set of destinations and finds a new Steiner tree using MPH. The algorithm proceeds to the next step only if adding a new terminal reduces the cost. Therefore, by construction SNH is proved to be at least as good as MPH. SNH is then used to find a pair of disjoint trees. Given n nodes and k terminals, the time complexity is $O(k^2n^3)$ for SNH compared to $O(kn^2)$ for MPH. The new algorithm is directly used to find to disjoint (arc or node) trees for the purpose of multicast protection. Performance of SNH, MPH and Pruned Prim's Heuristic (PPH)-based tree protection techniques are then compared. Although the authors give a clear example of when SNH performs better than MPH, their simulation does not show a noticeable difference between the two. The reason could do with the simulation network; only one network (similar to USNet) is used. The blocking probability, instead, shows some improvement when the number of used links is chosen as the measure of cost. In [21] authors present a slightly modified MPH algorithm to provide better blocking probability in arc-disjoint tree protection. In [63] authors devise tree-based protection to protect each segment on the primary tree.

- Cycle-based: Most of the work in this category involve use of p-cycles to protect the primary tree. P-cycles are particularly efficient in protecting dynamic multicast sessions [27][92].
- Path or Segment-based: Disjoint paths or segments can be used to protect paths, segments or simply individual links on the primary tree [61] [72]. In [66] the technique of Robust Network Coding is applied to optical multicast protection. In order to protect against k link failures, a k-link-connected Steiner subraph is found. The algorithm starts by a 1-link-connected Steiner subgraph and augments it by adding source-destination paths.

Without Primary Tree Instead of breaking the problem into provision of a multicast tree and protection of multicast tree, a subgraph which satisfies the required connectivity between source and terminals is proposed.

- Path-based: [78] presents *OPP* (described before). In [13] path-based protection is used to protect against source failure as well as link failures.
- Ring-based: The idea of collapsed-ring (a ring used in both directions) for multicast protection is presented in [75]. The authors actually describe their method as *1+1* protection for multicast. In [85] a Hamiltonian cycle is used to protect multiple multicast trees (different sessions) at the same time.

Network coding In the context of network coding, *Robust Network Coding* [53] provides instantaneous failure recovery for multicast. Static linear codes are designed such that a feasible multicast rate can be protected against any failure pattern for which the rate remains feasible. Moreover it allows for simple formulation of optimal cost problem, i.e., minimum cost subgraph supporting the rate under given failure pattern.

In [45] a review of optical multicast protection using network coding is presented. Implementing robust network coding in optical backbone networks is challenging. The main problem is implementation of linear network coding functions at the optical layer. In [52] optical-electrical-optical (OEO) conversion is assumed at network nodes in order to implement linear network coding functions. Robust network coding is then used to protect against single link failures. A minimum link cost ILP formulation of network coding subgraph is presented [51]. The authors further add the cost associated with OEO converter ports to the optimization problem. The resulting multi-objective problem is solved by an evolutionary approach. The simulations show that in most cases the network coding solution can actually be converted to a routing solution.

An all-optical implementation of robust network coding is presented in [66]. Instead of OEO converters which require terminating of optical signal, all-optical implementation of linear network coding is discussed including optical switching, buffering, and logical operations. The problem of unit rate multicast protection against k link failures is then addressed using robust network coding. A heuristic algorithm, Robust Coded Multicast (RCM), is proposed to find a subgraph with $k+1$-connectivity for every source-destination pair. The algorithm starts by a Steiner tree as a 1-connected subgraph and in each round augments the connectivity

by one. Authors also present an ILP formulation for minimum cost 2-connected subgraph which provides single failure protection under robust network coding. For a review of multicast protection using network coding.

We make the observation that for a unit rate multicast connection to be protected, the network nodes only need to support one simple network code, i.e., OR operation. This can also be viewed as merging flows at intermediate nodes using logical OR. Therefore our approach simplifies the network coding operation and its implementation at optical layer, while still offering instantaneous failure protection at minimum cost.

4.4.2 Related connectivity problems

Survivable network design problem (*SNDP*) is defined as a connectivity problem [56]. The input graph $G(V, E)$ is an undirected graph with edge weights. There is an integral connectivity requirement r_{uv} for every pair of vertices u and v. The problem has two versions of edge connectivity (EC-SNDP) and vertex connectivity (VC-SNDP). If set X represents, the set of all connectivity requirements then edge connectivity problem can be shown as X-EC-SNDP. For example spanning tree problem can be referred to as 1-EC-SNDP. The 2-EC-SNDP would represent a 2-edge connected spanning subgraph. In [36] a 2-approximation algorithm for general EC-SNDP is given and in [39] a 5/4-approximation for 2-EC-SNDP is proposed.

In the rooted SNDP problem, the connectivity is only required between a root $s \in V$ and a set $T \subset V$ of terminals. In [19] an approximation ratio of $O(klog|n|)$ is found for Rooted VC-SNDP where all terminals have same demand of k disjoint paths to source.

Some researchers use Steiner terminology to define the problem. [77] considers 2-vertex connected Steiner Minimal Network (SMN) problem on undirected graphs. Authors propose a factor 2 approximation algorithm with running time of $O(|V|^2|S|^3)$. The same algorithm runs in $O(|V|^2|S|^2)$ for the edge connectivity case.

The directed versions (directed graph and directed connectivity) of the same problems have received less attention from research community ([49]). In [29] authors discuss *rooted k-connectivity* spanning subgraph in directed graphs: a minimum cost spanning subgraph $G' = (V, E')$ of a directed graph $G = (V, E)$ so that G' contains k (edge or vertex) disjoint paths from

a specified root $r \in V$ to every other node in V (also called (k, r)-arborescence). Authors show that this version of the problem is polynomially solvable. The work in [25] addresses rooted connectivity problems for a subset of terminal nodes. Authors call it directed Steiner problem with connectivity constraints (DSCC) in which each terminal may have different connectivity requirement from the root.

The recent work of [17] addresses *directed rooted connectivity* (directed version of *Rooted SNDP*) and *directed rooted k-connectivity*. Authors mention that directed edge-disjoint and vertex-disjoint problems are essentially equivalent. The main contributions are:

- Directed rooted connectivity problem in acyclic directed graphs with total connectivity of $O(1)$ is polynomially solvable via dynamic programming.
- Directed rooted (general) connectivity problem with 2 terminals is APX-hard, even in acyclic digraphs with uniform costs.

For relations between *undirected SNDP* and *directed SNDP*, reader may refer to [57]. Reference paper [55] provides a survey of approximation algorithms for connectivity problems.

4.5 Assumptions and Problem Statement

The backbone network is modeled as directed graph $G(V, E)$ where V is the set of nodes and E is the set of directed edges. A physical bidirectional link between a pair of nodes u and v is modeled as two edges: (u, v) and (v, u). One wavelength channel is defined as the unit capacity. We assume that each link carries W wavelength channels in each direction. Failure of a link causes all those channels to fail. This is modeled by removal of both directed edges. The cost of reserving one unit of capacity on edge (u, v) is defined as c_{uv} which could be different for each edges depending on physical link properties such as physical length. A multicast request is represented as $M(s, D = \{d_1, ..., d_k\})$ where $s \in V$ is the source node and there are k distinct destination nodes such that $\forall i, d_i \in V$ and $s \notin D$. We assume unit multicast rate, i.e., multicast demand can be delivered over a single wavelength-channel in an optical network. This assumption is justified by the high bandwidth offered by a single optical channel. Moreover we assume static traffic model and no traffic grooming. We further assume that network nodes

are capable of merging incoming flows by simple logical OR operation. Finally, the single link failure multicast *1+1* protection problem is defined as follows:

Problem. *Given $G(V, E)$ where nodes are capable of merging incoming flows and a unit rate multicast connection $M(s, D = \{d_1, ..., d_k\})$, find the minimum cost (link cost) subgraph $H \subseteq G$ that provides instantaneous single link failure recovery.*

In the following we describe necessary and sufficient conditions for subgraph H.

Lemma 1. *Subgraph $H \subseteq G$ provides instantaneous single link failure recovery iff it includes 2 link-disjoint paths from s to each d_i.*

Proof. (Sufficient condition) Assume H includes 2 link-disjoint paths from s to every d_i which are denoted by p^1_{s,d_i} and p^2_{s,d_i}. A single link failure could at most hit one of the two paths for each destination. For a specific destination d_i, suppose p^1_{s,d_i} is failed and p^2_{s,d_i} is not. The data unit traveling on p^2_{s,d_i} would possibly be merged with other flows at intermediate nodes. Since any such merging operation would be a logical OR operation whose other operands are either zero or the same data unit (from other intact flows), the data unit traveling on p^2_{s,d_i} will not be affected by the failure and will be delivered to d_i. In the same way any destination node will receive at least one copy of each data unit in the event of any single link failure.

(Necessary condition) Assume subgraph H provides unit rate multicast connection $M(s, D = \{d_1, ..., d_k\})$ with instantaneous recovery for any single link failure. If H does not include (at least) two link disjoint paths from s to (at least) one destination d_i, then min-cut between s and d_i is at most 1. This means there is a single link whose failure disconnects s from d_i which contradicts the assumption of single failure protection. □

It is also worth noting that the necessary condition applies to any approach that provides single link failure protection even if it does not support instantaneous recovery. Finding minimum cost subgraph H that provides bi-connectivity between source and each destination node is known to be NP-hard [66].

4.6 Optimal Formulation

Based on the necessary and sufficient conditions presented in Section 4.5, the problem of minimum cost multicast *1+1* protection is equivalent to finding a minimum cost subgraph that provides bi-connectivity between the source and each destination. Such subgraph can be optimally found using the following ILP formulation.

Binary variable x_{uv} is equal to one if edge (u, v) is used in the solution. Binary variable $f_{uv}^{d_i}$ represents the flow from s to d_i on edge (u, v). Equation 4.1 presents the total cost of the solution to be minimized. Equation 4.2 is flow conservation at the source, destination nodes, and other intermediate nodes. Equation 4.2 makes sure that $x_{uv} = 1$ if edge (u, v) is used by any flow. Equation 4.3 defines the binary variables. The ILP basically sends 2 units of flow from s to each destination d_i. Since flow variables are defined as binary, 2 units of flow would be carried by 2 edge-disjoint paths.

$$Min \quad \sum_{(u,v)\in E} c_{uv} \cdot x_{uv} \tag{4.1}$$

$$\sum_{(u,v)\in E} f_{uv}^{d_i} - \sum_{(w,u)\in E} f_{wu}^{d_i} = \begin{cases} +2 & u = s \\ -2 & u = d_i \\ 0 & o.w. \end{cases} \quad \forall u \in V\,, \quad \forall d_i \in D\,.$$

$$x_{uv} \geq f_{uv}^{d_i} \quad \forall (u, v) \in E\,, \quad \forall d_i \in D\,. \tag{4.2}$$

$$f_{uv}^{d_i}\,, x_{uv} \in \{0, 1\} \quad \forall (u, v) \in E\,, \quad \forall d_i \in D\,. \tag{4.3}$$

4.6.1 Single link failure protection

The minimum cost objective ensures that the edge-disjoint path pair from s to a single destination d_i will not include any directed cycles since any such cycle could be removed to find a lower cost solution [94]. This includes the case for any two oppositely directed edges (u, v) and (v, u). In other words the edge-disjoint path pair from s to any destination d_i will not include oppositely directed edges. Therefore they are link-disjoint (note that a link was

modeled by two oppositely directed edges). Hence the solution provides full single link failure protection.

4.6.2 Single node failure protection

Multicast *1+1* protection can also provide single node failure protection. Here we use the node splitting method proposed by [80] to build a modified graph $G'(E', V')$. Every node $u \in V$ is substituted by two nodes $u^1, u^2 \in V'$ and one directed edge $(u^1, u^2) \in E'$ of cost zero. For every directed edge $(v, w) \in E$, a directed edge (v^2, w^1) is added to E' with same cost. Multicast connection is also modified to $M'(s^2, D' = \{d_1^1, ..., d_k^1\})$.

Finding two edge-disjoint paths in the modified graph $G'(E', V')$ corresponds to finding two node-disjoint paths in the original graph $G(V, E)$. Solving the same ILP for G' and M' generates two edge-disjoint paths from s^2 to each d_i^1. This is equal to two node-disjoint paths for G and M. Moreover since the added (u_1, u_2) edges are zero cost, the cost of the edge-disjoint solution in G' is the same as its corresponding node-disjoint solution in G.

4.7 Heuristic Algorithms

The problem of minimum cost *1+1* protection for multicast is NP-hard. Obtaining the minimum cost by solving ILP formulation may not be practical for the real scenarios of dynamically changing multi-session multicast traffic. Therefore it is necessary to propose heuristic algorithms capable of providing fast yet efficient online solutions.

The core problem is 2-connectivity from source to all destinations. The 1-connectivity problem is the famous Steiner tree problem for which there is a well-known heuristic, i.e., Minimum Path Heuristic (*MPH*) [81]. The idea is to find the closest destination to the source, connect it by shortest path, set the cost of edges on the path to zero, then find next closest destination, and continue until all destinations are covered. The very same idea can be extended to build a 2-connected subgraph: substituting the notion of shortest path with shortest disjoint path-pair which can be found by Suurballe's algorithm [80] (Algrithm 4). We call this Minimum Path-Pair Heuristic (MPPH).

While *MPPH* maintains 2-connectivity at each step, another method is to start from a 1-connected subgraph (Steiner tree) and augment it to a 2-connected one. In [66] authors have proposed an algorithm based on a similar idea. First a Steiner tree is found, then the connectivity to destinations is augmented one at a time. This is done by removing the edges of the path from source to each destination on the Steiner tree and then finding a second shortest path to that destination. We propose an algorithm that augments a Steiner tree found by *MPH* to a 2-connected subgraph using *MPPH*. Hence called *MPH+MPPH* (Algorithm 5). The cost of Steiner edges returned by *MPH* are set to zero so that *MPPH* has incentive to use Steiner tree edges. In the simulation resutls we also use a more involved version of this algorithm called *MPH+MPPH(all)* which basically runs $|D|$ instances of *MPPH*. In each instance one destination node is fixed as the first destination in *MPPH* algorithm.

Algorithm 4 Minimum Path-Pair Heuristic: *MPPH*

Input: $G(V, E)$, $M(s, D)$
Output: Subgraph H

$H \leftarrow s$
while $D \neq \emptyset$ **do**
  $j \leftarrow argmin_{d_i \in D}(|p_{d_i}|)$
  $\{p_{d_i}$ is the shortest path-pair from s to d_i in $G\}$
  $\forall (u, v) \in p_{d_j} : c_{uv} \leftarrow 0$
  {updates the cost of edges of path-pair p_{d_i} in $G\}$
  $H \leftarrow H \cup p_{d_j}$
  $D \leftarrow D \backslash d_j$
end while
return H

Algorithm 5 *MPH+MPPH*

Input: $G(V, E)$, $M(s, D)$
Output: Subgraph H

$T \leftarrow MPH\,(G(V, E), M(s, D))$
$\forall (u, v) \in T : c_{uv} \leftarrow 0$
{updates the cost of edges of T in $G\}$
$H \leftarrow MPPH\,(G(V, E), M(s, D))$
return H

Time complexity of *MPH* [81] is $|D|.O(S)$ where $|D|$ is number of destinations and $O(S)$ is the time complexity of shortest path algorithm. Time complexity of *MPPH* depends on finding

a shortest path-pair (step 3) which has the same time complexity as shortest path algorithm. In each iteration of *MPPH* we need to find the destination with minimum shortest path-pair among the remaining destinations. Therefore the time complexity of *MPPH* is $|D|^2.O(S)$. Time compexlity of *MPH+MPPH* hence is the same as *MPPH*. Finally *MPH+MPPH(all)* would have time complexity of $|D|^3.O(S)$. In our implementation $O(S) = O(|V|^2)$ however a more efficient implementation can achieve $O(|E| + |V|log|V|)$ for Dijkstra's shortest path algorithm.

4.8 Simulation Results

The original Pan-European network COST239 (11 nodes, 26 links) [9] and a modified version are used in the simulations. In the modified version, evey link (u, v) in original network is replaced by a new node w_{uv} and two new links (u, w_{uv}) and (w_{uv}, v). Therefore it has 11+26=37 nodes and 2*26=52 links. We refer to it as COST239+. For COST239 two cost functions are used: 1- unit link cost where links have equal unit cost, 2- physical distance cost where distance between cities is used as the link cost (in km). In the case of COST239+, we only consider physical distance and the new nodes are assumed to be halfway between original nodes.

In each case the costs reported by ILP and heuristics (*OPP*, *MPPH*, *MPH+MPPH*, and *MPH+MPPH(all)*) are compared. Our results cover the complete range of session size which is 2 to 11 for COST239 and 2 to 37 for COST239+. In each case 100 random multicast sessions are generated and the average cost for each session size is calculated. The same random sessions are applied as input to ILP and heuristic algorithms.

Figure 4.4 shows the results for COST239 with unit distance cost. While all heuristics perform well compared to ILP, *MPH+MPPH* and *MPH+MPPH(all)* are almost the same as optimal. Figure 4.5 shows the results for COST239 with physical distance as the link cost. Here again *MPH+MPPH* and *MPH+MPPH(all)* peform better than *OPP* and *MPPH*. In Figure 4.6 the results on COST239+ network are presented. Again we observe that *MPH+MPPH* and *MPH+MPPH(all)* perform very close to optimal. For the sake of readability, in each figure we have only shown a subset of session sizes for which the optimal vs. heuristic difference is more

visible. In Table 4.1 we summarize the average and worst case performance (over all session sizes) of our best heuristics and *OPP* compared to optimal.

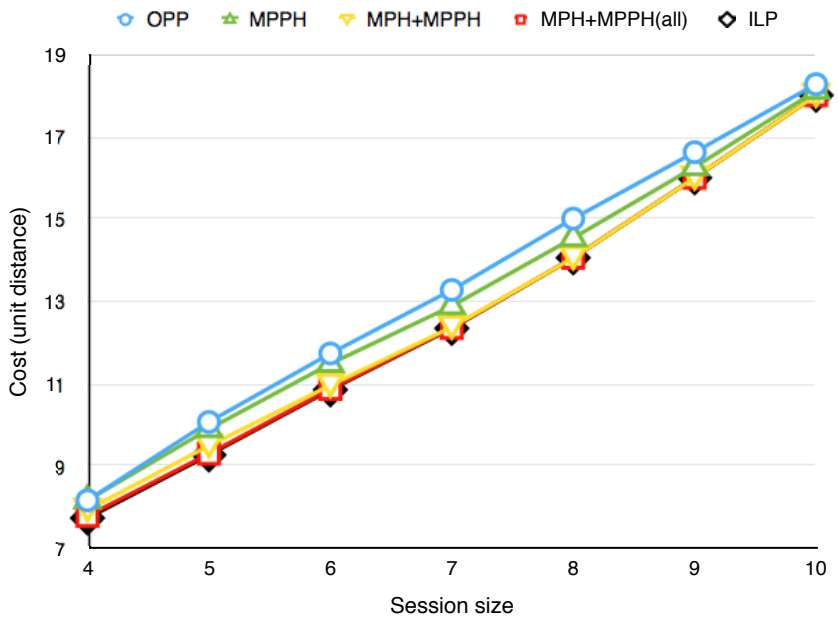


Figure 4.4 COST239 network: unit link cost

Table 4.1 Average/worst case percentage of extra cost (vs. optimal)

Network	OPP	MPH+MPPH	MPH+MPPH(all)
COST239(unit)	4.4/8.6	0.6/2.6	0.1/0.7
COST239(phys)	8.3/11.3	5.0/7.4	2.6/4.7
COST239+(phys)	5.5/7.7	2.6/4.4	1.6/2.4

4.9 Conclusion

The idea of *1+1* protection is extended to multicast protection using simple network coding. The *1+1* protection sends, simultaneously, two copies of each data unit to every destination, and simple network coding (OR operation) guarantees that upon any single link failure, at least one data copy is received by all destinations. No rerouting or switch reconfiguration is required and destination nodes would not experience any service disruption under any single link/node failure. The necessary and sufficient condition for the existence of *1+1* protection solution

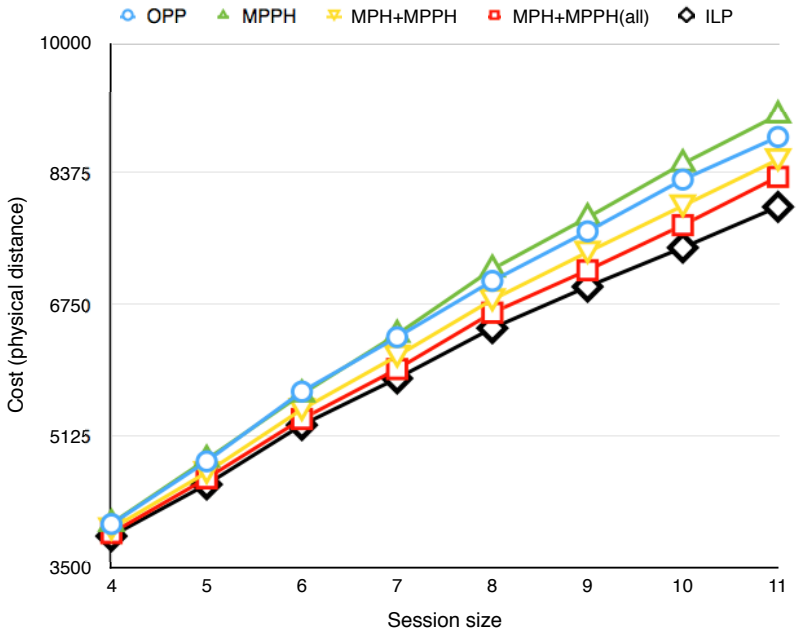


Figure 4.5 COST239 network: physical distance as link cost

under merging flows is simply 2-connectivity from source to each destination. This allows us to easily formulate the problem as a network flow problem which can be solved to find minimum cost subgraph supporting *1+1* protection. An optimal ILP formulation of the problem and three heuristic algorithms are proposed as offline and online solutions. The simulation results on two sample networks show impressive performance by our top two heuristics as compared to the optimal: on average, our best heuristic increases the cost by no more than 2.6% and in the worst case, the gap between our best heuristic and optimal is only 4.7%. Future work would include implementation of OR operation, other failure models and traffic models, e.g., dynamic multi-session multicast.

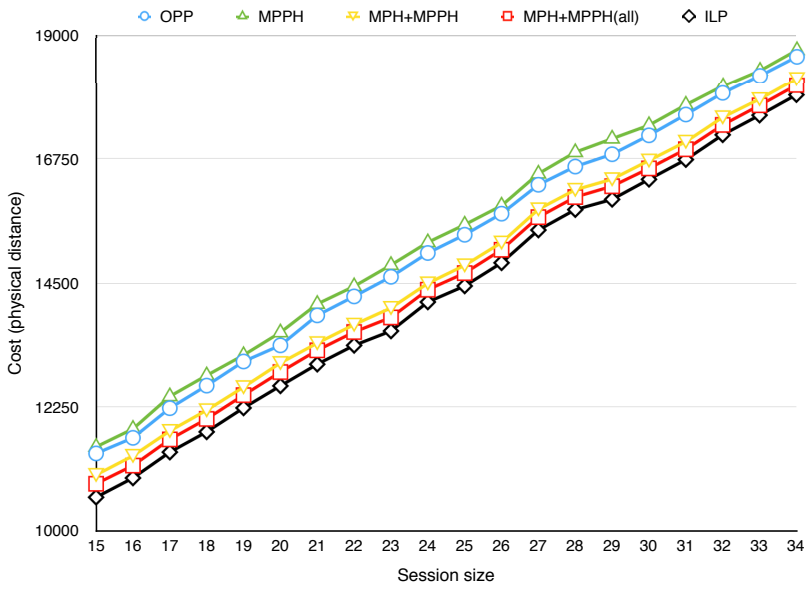


Figure 4.6 COST239+ network: physical distance as link cost

CHAPTER 5. RATE, ENERGY, AND DELAY TRADEOFFS IN WIRELESS MULTICAST: NETWORK CODING VS. ROUTING

Modified from a paper submitted for 2nd review to IEEE Transactions on Mobile Computing [68].

5.1 Abstract

We build on the framework of joint scheduling and network coding optimization and extend it to include rate, energy, and delay in network coding and routing paradigms. We then study energy-rate and delay-rate relationships to see how minimum energy and delay change as functions of multicast rate demand. The main observation is that as the rate demand approaches maximum achievable rate, the solution tends to increasingly use more diverse, longer paths. This translates into non-linearly higher energy and delay for higher input rates. In the case of energy, we are also able to show that network coding provides more benefits (when compared to routing) at higher rates. Another observation is related to the scheduling over maximal independent sets (MISs). We present results on comparing the performance of scheduling over all, exponentially growing, MISs and small randomly selected subsets of MISs. Our results point to the effectiveness of latter in achieving near-optimal rate and energy while reducing the complexity of the problem.

5.2 Introduction

Based on the theory of network coding, the maximum achievable multicast rate is equal to the minimum of all source-terminal max-flows [6]. In a typical wired network, we can simply find the max-flow between source and each terminal. The minimum max-flow value

would then give us the maximum achievable rate. Network coding provides a way of breaking the main problem (maximum rate in multicast) into independent subproblems (max-flow for each individual source-terminal unicast) that in fact achieves the optimal solution. In a routing paradigm, however, the solution could take the form of multiple multicast trees each delivering a portion of rate. That is where the problem becomes harder to solve. Unlike the network coding case, it is not easy to optimally break the problem into independent polynomially solvable subproblems. Offering a polynomial time solution to the problem of maximum multicast rate is, in fact, one of the main benefits of network coding. Furthermore, there is the question of whether or not routing can achieve the maximum rate offered by network coding in a given network. The butterfly network ([6]) is, for instance, a case where routing cannot achieve the same rate.

In a wireless network, interference and wireless constraints such as single transceivers bring a new level of complexity to this problem in both network coding and routing paradigms. Given a placement of wireless nodes, the primary question is which wireless links can be active at the same time while respecting interference and other wireless constrains. This is in contrast to a typical wired network in which all links can be active at the same time. With wireless constraints, even the basic problem of max-flow between two given nodes is not easy to solve anymore.

One way around this problem is timesharing. We may assign different timeshares to links that are not allowed to be active at the same time. In general, instead of assigning timeshares to individual links, non-conflicting links are grouped into what is called independent sets. All links in each independent set are "independent", i.e., mutually non-conflicting. Timesharing is then applied to these independent sets. Given a valid assignment of timeshares, we have what is called a realizable network. In a realizable network, each link has its scheduled capacity which represents the percentage of time it is active. Therefore a realizable network can be seen as a wired network abstraction of a wireless network. Timesharing eliminates the interference and wireless constraints by basically not allowing conflicting links to be active at the same time.

As a result, given a scheduling of wireless links, the complexity of maximum multicast rate problem is the same as wired case. However, in order to find the true maximum, one

has to consider solving the two components jointly; scheduling component and network coding (or routing) component need to be jointly solved. As it turns out, solving the joint problem is NP-hard [38]; no matter which paradigm, routing or network coding, is chosen. In fact scheduling becomes an integral hard component of any performance optimization that depends on it. This is, of course, in addition to the inherent complexities due to the routing paradigm or the performance parameter (e.g., delay) optimization.

We know from previous work that network coding offers benefits in terms of rate, energy, and delay. Network coding is shown to achieve higher rates, and for a given a rate provide a lower energy/delay solution. Our study continues the same line of research by focusing on optimal or near-optimal scheduling of wireless links and is motivated by following questions:

1. How can rate-energy and rate-delay relations be described under optimal scheduling? For example, how does minimum required energy change as the multicast rate demand is increased?

2. What performance benefits does network coding provide in wireless multicast under optimal scheduling? For example, given a feasible multicast rate, is it always possible to find a lower energy solution with network coding? Does the energy benefit of network coding depend on the input multicast rate?

3. Are there simple yet effective ways to reduce the complexity of scheduling component?

In particular our work follows the work of Jain et al. [38] in that we too consider optimal scheduling of interference-free wireless links using conflict graph modeling. However unlike [38], we consider network coding in addition to routing. In the network coding aspect, our work follows the recent work of Traskov et al. [82] on joint scheduling and network coding for multicast. We too focus on multicast mode of communication to benefit from the availability of well-established theory of network coding for multicast in our analysis. We also use the same hypergraph modeling technique. Hypergraph modeling enables us to model point-to-multipoint links as hyperarcs that capture broadcast nature of wireless transmission. Broadcast links are specially beneficial in multicast transmission. In contrast, we extend the performance measures

to include rate, energy, and delay, investigate their relations, and provide a inter-paradigm routing versus network coding comparison.

We will start by a review of the literature in Section 5.3. Our focus is on previous work addressing performance optimization problems that involve scheduling. Both routing and network coding, and different modes of communication are considered. In Section 5.4 network and interference models are presented. Section 5.5 discusses scheduling using conflict graph. In sections 5.6 and 5.7, multicast rate region, energy, and delay formulation in network coding and routing paradigms are presented. In section 5.8, we present some illustrative examples to show how joint scheduling and network coding/routing works and make few observations based on the shape of optimal solutions. We present the experimental evaluation and analytical discussion in Section 5.9. Section 5.10 concludes the paper.

5.3 Related Work

5.3.1 Routing

The work of Jain et al. [38] is a fundamental work in joint scheduling and routing. The technique of modeling interference using conflict graph introduced here has since been used by other authors. Given a placement of wireless nodes and a unicast traffic matrix (source and sink pairs), the objective is to find optimal scheduling and routing that maximizes cumulative rate for all source-sink pairs. Two models of interference are considered:

- Protocol model: all interfering transmissions are lost.
- Physical model: interference is tolerated as long as it respects signal-to-noise ratio (SNR) at the receiver node.

A weighted conflict graph is used in physical interference model. It is proved that a set of transmissions is schedulable if and only if it falls into independent set (stable set) polytope of the conflict graph. The maximum total rate is formulated as the intersection of two polytopes: a linear multi-commodity flow formulation and the independent set polytope. This problem in its most general form is shown to be NP-hard and hard to approximate due to independent

set sub-problem. Alternatively subsets of independent sets and clique constraints are used to find lower/upper bounds on the objective. The trade-off between connectivity and rate is investigated. It is shown that in a non-greedy traffic model, more nodes participating in wireless mesh network can, in fact, result in better rate. The paper also includes a comparison of heuristic scheduling and heuristic routing algorithms with optimal joint scheduling and routing.

In [62] a minimum cost Integer Linear Program (ILP) formulation is given for a single multicast tree in multi-radio and multi-channel wireless networks. The interference is only discussed between multiple multicast sessions. In [87] the problem of finding a single minimum energy broadcast or multicast tree is addressed.

5.3.2 Network coding

In [88], simulation results are reported on maximum achievable multicast rate by a greedy tree packing routing algorithm and a heuristic network coding algorithm (distributed practical network coding [18]) in a wired setting. Both heuristics are reported to have close to optimal performance. However network coding provides other advantages in terms of lower resource usage and robustness. Furthermore, it is pointed out that for a linear cost function, minimum cost network coding problem has a linear formulation while minimum cost multicast tree problem is NP-hard. In [86], network coding is used to reduce the number of required transmissions, thereby saving energy. The traffic model is many to many where each node in the network wants to transmit to all other nodes. The main contribution is the design of two optimal network coding algorithms for special case circular and grid networks. The algorithms are optimal in the sense that they minimize the number of required transmissions assuming XOR network coding capability at all nodes. Application of network coding (even though limited to XOR operation) is shown to outperform routing-based flooding algorithms in general networks via simulation.

In [90], minimum energy wireless multicast under joint scheduling and network coding is considered. The notion of elementary capacity graph (ECG) is used to model a set of concurrently active wireless links. A broadcast link $u \rightarrow Y_u$ (u is the transmitter and Y_u is the

set of receivers) is modeled by adding a virtual node u' and edges (u, u') and (u', v) for every v in Y_u. A feasible ECG is the one in which no receiver Signal to Interference plus Noise Ratio (SINR) is violated. Timesharing at MAC layer is represented by the convex combination of ECGs and network layer is represented by end-to-end flows. Instead of considering the set of all ECGs (exponentially many), K feasible ECGs are heuristically and randomly found. An iterative algorithm is proposed that starts from a basis set of ECGs, optimizes the energy, and updates the set of ECGs. While the solution is still suboptimal, it is shown to outperform equal timeshares and a fixed random ECG scheduling. Bit-per-Joule measure is used to present the energy-rate tradeoff. The results show a non-linear energy-rate relation.

The work in [89] addresses the problem of minimum-energy wireless multicast using network coding. Random linear network coding (RLNC) is assumed at network nodes. Wireless network model is multi-hop and static with multiple power levels available at each node. A sub-optimal notion of elementary graphs is used instead of independent sets on a conflict graph. Each elementary graph involves a single transmitter. Therefore, instead of independent set polytope, polytope of single links is considered, as if each independent set was reduced to a single link. However unlike [38], the single link could be a broadcast link. Multicast mode of transmission can, therefore, benefit from broadcast links. Similar to [90], every broadcast link is converted to a set of point-to-point links by adding a virtual node. The difference from the flow formulation in [38] is the assumption of multicast network coding; flows to different terminal nodes are coded together on each common link and, therefore, share the same capacity. Minimum energy problem for a given multicast rate is then solved under network coding and routing assumptions.

In [65], the minimum cost coded multicast is discussed. However scheduling is not addressed here. The work discusses wire-line and wireless, static and dynamic multicast. Comparison with routing is made only in the case of single multicast routing tree for which heuristic algorithms are used. In the case of wireless multicast, cost is defined as energy consumption. Minimum energy network coded multicast is compared to a heuristic routing algorithm called Multicast Incremental Power (MIP).

In [24], energy minimization for unicast connections based on XOR coding (similar to COPE [48]) is addressed. An offline optimization is presented and the scheduling problem

is addressed. The main idea is that every pair of unicast connections is decomposed into two unicast connections with no intersession coding and one multicast (2-source) butterfly-like connection that models the intersession coding.

In Traskov et al. [82], optimal wireless multicast using network coding is addressed. General formulation for optimization of different objectives (such as rate, cost, or energy) in the presence of wireless interference is provided. In comparison with [38], every possible broadcast transmission from any node to any subset of neighbors is modeled by a separate *hyperarc*. The resulting *hypergraph*, therefore, contains a hyperarc for every possible transmission. Another difference is that flow conservation relies on network coding assumptions (similar to [89]). However, the basic formulation is similar to [38]: the intersection of independent set polytope and flow conservation polytope. The paper uses Lagrangian relaxation to decompose the problem into scheduling and network coding sub-problems. A online solution is then proposed to solve the problem in distributed fashion.

Niati et al. [71] also address the same problem for rate and energy optimization. In comparison with [83], a subset of independent sets are used for simplicity. Inside each independent set, broadcast links are converted to multiple single links by a technique similar to [89], eliminating the need for hypergraph modeling. The same polytope intersection method is used. However flow variables are defined per edge, terminal, and timeshare. As a result flows should respect specific timeshare capacity on the corresponding edge. It is argued that this method provides a more accurate schedule for each transmission. The energy consumption is defined as a function of timeshares instead of flows. This results in non-linearity of the objective which is addressed by an iterative method.

In [50], fundamental bounds on the benefit of network coding are discussed. In the particular case of a single wireless multicast, it is shown that both rate and energy gains of network coding are bounded by a constant factor.

5.4 Network Model

Our basic network model includes a static multi-hop single-channel wireless network. All nodes are assumed to have single transceivers and transmit at a fixed power level. The set

of nodes is represented by $\mathcal{N}$. For each node $i \in \mathcal{N}$ the set of reachable neighbors is given by $N(i) \subseteq \mathcal{N}$. We assume that in the absence of interfering transmissions, all nodes in $N(i)$ can hear a transmission from i without error, i.e., lossless wireless channels. We use protocol interference model: all interfering transmissions are lost. In network coding analysis, nodes are assumed to have linear network coding capability. Similar to [82], hyperarcs are used to model the wireless broadcast: a transmission from node i to any subset $J \subseteq N(i)$ is modeled by a hyperarc (i, J). Let $\mathcal{A}$ be the set of all hyperarcs, then the wireless network can be represented by hypergraph $\mathcal{H}(\mathcal{N}, \mathcal{A})$. Without loss of generality we assume all hyperarcs have unit capacity.

Next we define conflicting hyperarcs in the same way they are defined in [82]. The meaning of *conflict* here is that two hyperarcs cannot be active at the same time. Two hyperarcs (i_1, J_1) and (i_2, J_2) conflict if and only if at least one of the following conditions is true:

1. $J_1 \cap (\{i_2\} \cup N(i_2)) \neq \emptyset$.
2. $J_2 \cap (\{i_1\} \cup N(i_1)) \neq \emptyset$.

The above conditions cover half-duplex constraint, i.e., a node cannot receive and transmit at the same time, and secondary interference model, i.e., a receiver may not be in the range of two transmissions.

Again following the framework of [82], all the conflict relations in the hypergraph $\mathcal{H}(\mathcal{N}, \mathcal{A})$ are captured by an undirected conflict graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Vertexes in $\mathcal{V}$ represent hyperarcs in $\mathcal{A}$. For every pair of conflicting hyperarcs in $\mathcal{A}$, there is an undirected edge in $\mathcal{E}$ between their corresponding vertexes in $\mathcal{V}$.

5.5 Scheduling

An independent set in $\mathcal{G}$ represents a set of hyperarcs all of which can be active at the same time without interference or violation of single transceiver constraint.

Following the notation in [38], we can define a usage vector U of length $|\mathcal{A}|$ to represent the fraction of time each hyperarc is active. Based on [38, Theorem 2], we have:

The necessary and sufficient conditions for a usage vector to be schedulable is that the usage vector lies within independent set polytope of the conflict graph.

Such a schedulable usage vector is referred to as a *realizable network* in [89]. The independent set polytope is the convex hull of incidence vectors of all maximal independent sets. An independent set is maximal if it is not a subset of any other independent set. Similar to [82], we define a length-$|\mathcal{A}|$ binary incidence vector corresponding to each maximal independent set k:

$$I_k^{iJ} = \begin{cases} 1 & \text{if } (i,J) \in \text{maximal independent set } k \\ 0 & \text{o.w.} \end{cases} \qquad \forall (i,J) \in \mathcal{A}\,. \tag{5.1}$$

Let $\mathcal{I} = \{I_k\}$. The independent set polytope can then be formulated as:

$$POLY(\mathcal{I}) = \left\{ \sum_k \tau_k \cdot I_k \;\middle|\; \sum_k \tau_k = 1 \quad \wedge \quad \tau_k \geq 0 \right\}. \tag{5.2}$$

Here $\{\tau_k\}$ represents timeshares. Any assignment $\{\tau_k\}$ produces a *realization* of the wireless network with a scheduled capacity for each hyperarc. We denote this scheduled capacity by $\bar{c}_{iJ}$:

$$\bar{c}_{iJ} = \sum_k \tau_k \cdot I_k^{iJ}\,, \quad \forall (i,J) \in \mathcal{A}\,. \tag{5.3}$$

$$\sum_k \tau_k = 1\,, \quad \tau_k \geq 0\,, \quad \forall k\,. \tag{5.4}$$

Therefore corresponding to each set of schedulable timeshares $\{\tau_k\}$ we have a set of scheduled capacities $\{\bar{c}_{iJ}\}$.

While equations (5.3) and (5.4) give a linear formulation of schedulable capacities, the main difficulty remains in finding maximal independent sets. In general, the number of maximal independent sets grows exponentially with the size of input graph (conflict graph in our case). We will discuss this problem in later sections. In what follows we would refer to equations (5.3) and (5.4) as the scheduling constraints.

5.6 Network Coding

5.6.1 Multicast rate region

We model a multicast session by $\mathcal{M}(s, \mathcal{T})$ with the source $s \in \mathcal{N}$ and the set of terminals $\mathcal{T} \subset \mathcal{N}$. Multicast rate region is the intersection of two polytopes: independent sets polytope (set of scheduled capacities) and the flow polytope. Equations (5.5) and (5.6) show the flow polytope according to network coding assumptions. $\mathcal{R}$ is the multicast rate.

$$\sum_{j \in J} f^t_{iJj} \leq \bar{c}_{iJ}\,, \quad \forall t \in \mathcal{T}\,, \quad \forall (i, J) \in \mathcal{A}\,. \tag{5.5}$$

$$\sum_{(i,J)} \sum_{j \in J} f^t_{iJj} \; - \sum_{\{(j,J) | i \in J\}} f^t_{jJi} \quad = \quad \begin{cases} \mathcal{R} & \text{if } i = s \\ -\mathcal{R} & \text{if } i = t \\ 0 & \text{o.w.} \end{cases}, \quad \forall i \in \mathcal{N}\,, \quad \forall t \in \mathcal{T}\,. \tag{5.6}$$

In this notation, variable f^t_{iJj} represents the value of flow from node i to node j that is sent over hyperarc (i, J). Equation (5.5) states that sum of flows to a specific terminal t on each hyperarc is bounded by the scheduled capacity of the hyperarc. However, there is no cross-constraint for flows to different terminals. This is the fundamental network coding assumption: on any given hyperarc, flows to different terminals are coded together and treated as if they are sharing the same capacity. Equation (5.6) is the hypergraph version of standard flow conservation constraint. This is an advantage of network coding formulation that flow constraints are written independently per terminal.

The multicast rate region under network coding, therefore, is formulated as the intersection of linear equations (5.3), (5.4), (5.5) and (5.6). Any feasible multicast rate $\mathcal{R}$ must lie within this intersection. Although the rate formulation is linear, it still involves exponential number of maximal independent sets to account for in (5.3) and (5.4). Figure 5.1 shows the overview of our modeling and problem solving methodology. The formulations for scheduling and different parameter optimizations are jointly solved. Assuming that scheduling is done optimally by considering all MISs, the scheduling component becomes independent of input multicast session.

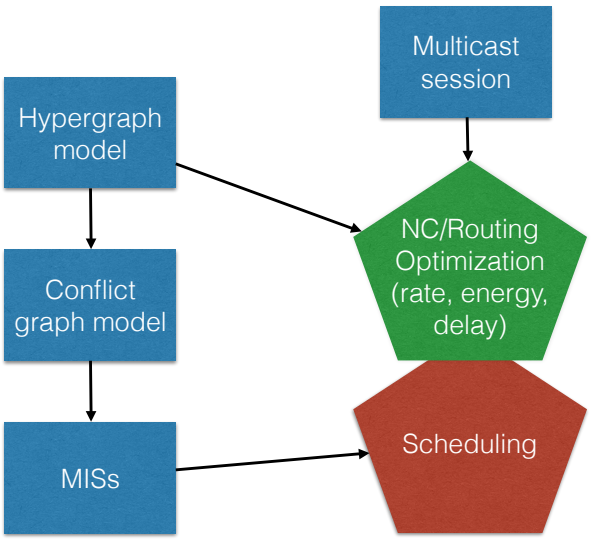


Figure 5.1 Joint scheduling and parameter optimization model.

5.6.2 Energy

For simplicity and without loss of generality, in the single transmission power model we define one unit of energy being equal to one unit of flow sent over one hyperarc. Although simple, the energy model fits our objective of evaluating fundamental energy-rate relations under optimal scheduling where all transmissions are scheduled and interference-free. It also suits the grid network models used in the simulations where all hops are equal distance.

Let e_{iJ} be the energy used by hyperarc (i, J) in each time unit. Equation (5.8) defines e_{iJ} as the maximum of all terminal flows on (i, J).

$$\mathcal{E} = \sum_{(i,J)\in\mathcal{A}} e_{iJ} \tag{5.7}$$

$$\sum_{j\in J} f^t_{iJj} \leq e_{iJ}\,, \quad \forall t \in \mathcal{T}\,, \quad \forall (i, J) \in \mathcal{A}\,. \tag{5.8}$$

It is important to note the difference between e_{iJ} in equation (5.8) and $\bar{c}_{iJ}$ in equation (5.5). $\bar{c}_{iJ}$ serves as the capacity limit for each hyperarc which may or may not be fully used. e_{iJ}, on the other hand, is used for energy minimization and would be exactly equal to the maximum

of terminal flows on (i, J). In our analysis we wish to minimize the total energy consumption: $\sum_{(i,J)\in\mathcal{A}} e_{iJ}$.

5.6.3 Delay

The delay minimization is a hard sub-problem. A realistic model should capture variable link transmission delays, buffering delays, and coding delays. Our focus in this paper is to compare network coding and routing under optimal scheduling. Given the complexity of scheduling and routing sub-problems, we choose a simpler delay model that serves as a lower-bound. We define the delay from source to each terminal as the length, hop-count, of longest flow path to that terminal. It is worth noting that even with the lower-bound delay model, delay minimization remains a hard sub-problem. This model suits the grid network models (with equal hop distances) used in our simulations.

Equation (5.9) makes sure that binary variable b_{ij}^t is equal to 1 if there is a non-zero terminal t flow from node i to node j. L is a large constant. Variable D_i^t represents the maximum distance of node i from source s on all the terminal t flows. When b_{ij}^t is 1, the constraint in (5.10) makes sure D_j^t at least equals $D_i^t + 1$. Therefore, D_t^t is the maximum hop distance of terminal t from source s on any of its own flow paths. Parameter $\mathcal{D}$ limits the maximum delay (flow path length) from source to all terminals. This can be an input parameter, e.g., when minimizing rate subject to a given delay, or an optimization variable when minimizing the delay.

$$\sum_{\{(i,J)\in\mathcal{A}|j\in J\}} f_{iJj}^t \leq L \cdot b_{ij}^t \,, \tag{5.9}$$

$$D_j^t - D_i^t \geq (L+1) \cdot b_{ij}^t - L \,, \tag{5.10}$$

$$D_s^t = 0, \quad D_t^t \leq \mathcal{D} \,, \tag{5.11}$$

$$b_{ij}^t \in \{0,1\}, \quad D_i^t \in Z^+ \,, \tag{5.12}$$

$$\forall t \in \mathcal{T} \,, \quad \forall i \in \mathcal{N} \,, \quad \forall j \in N(i) \,.$$

5.7 Routing

5.7.1 Multicast rate region

In routing paradigm, formulating multicast rate region becomes harder. In fact it depends on the number of disjoint multicast trees used. For example, a routing formulation may be written to maximize the multicast rate over a single multicast tree. A single minimum cost multicast tree is an example of minimum cost Steiner tree which is NP-hard [47].

Instead of limiting the solution to a single multicast tree, we could allow the rate to be divided between multiple capacity-disjoint multicast trees. In this case, each tree delivers its own share of rate. While trees may use the same hyperarcs, they may not share any capacity on any common hyperarc. This allows us to add the sub-rates on each tree to find the total delivered multicast rate. In this sense, the trees are called disjoint. The problem of increasing the multicast rate by adding multicast trees is similar to the problem of packing Steiner trees [37]. As we increase number of trees in the formulation the result may become better or remain the same but it is never worse.

In the network coding formulation, flows to different terminals could share capacity on any hyperarc. However in routing, flows to different terminals can only share capacity on the same tree, i.e., there is no inter-tree sharing. Moreover, in the network coding formulation, flows are defined per terminal but in routing, flows are defined per tree and per terminal. Other than that, the basic hypergraph modeling and formulation of the set of scheduled capacities, (5.3) and (5.4), are the same.

We now give the multicast rate region formulation. Equations (5.13) to (5.20) maximize multicast rate that can be sent over $|\mathcal{Z}|$ capacity-disjoint multicast trees. In equation (5.13), $\mathcal{R}^z$ is the rate delivered by tree $z \in \mathcal{Z}$, and $\mathcal{R}$ is the total rate. Variable f_{iJ}^z is the flow of tree z on hyperarc (i, J). Variable $f_{iJ}^{z,t}$ is the flow of terminal t, on tree z, on hyperarc (i, J). Equation (5.14) states the fact that trees do not share capacity on any hyperarc. Equation (5.15) defines f_{iJ}^z as the maximum terminal flow on each hyperarc for each tree. This equation also implies that flows to different terminals on the same tree may share capacity. Ultimately non-zero terminal flows will have the same value on each hyperarc for each tree.

Equation (5.16) bounds the flow between each pair of neighboring nodes in terms of all hyperarcs originating from one and covering the other. In the optimal solution, there could be a non-zero flow value $f_{ij}^{z,t}$ for at most one $j \in N(i)$; flow to a certain terminal follows a simple path and does not split. The inequality allows for other neighbors to receive zero value flow. Equality would eliminate all hyperarcs and limit the solution to simple arcs. Equation (5.17) is the flow conservation constraint. Note that it is written per terminal and per tree for each node.

The following equations are to make sure that flows belonging to each tree would in fact form a tree. Equation (5.18) states that each node, if belongs to a tree, has one parent node on that tree. The binary varaible g_{ij}^{z} is set to 1, if node i is the parent of node j on tree z. The source node is the root and has no parent. Equation (5.19) indicates that any terminal flow must come from the parent node on each tree. Such flow could be carried by any (and many) hyperarcs originating from the parent node. Therefore, in equation (5.17), sum of incoming flows would be equal to incoming flows from the parent node. Finally, equation (5.20) defines binary variables g_{ij}^{z}.

In summary, any multicast rate $\mathcal{R}$ that satisfies constraints (5.13) to (5.20) together with constraints (5.3) and (5.4) is achievable using $|\mathcal{Z}|$ multicast trees.

$$\mathcal{R} = \sum_{z \in \mathcal{Z}} \mathcal{R}^z \,. \tag{5.13}$$

$$\sum_{z \in \mathcal{Z}} f_{iJ}^{z} \leq \bar{c}_{iJ} \,, \quad \forall (i,J) \in \mathcal{A} \,. \tag{5.14}$$

$$f_{iJ}^{z,t} \leq f_{iJ}^{z} \,, \quad \forall (i,J) \in \mathcal{A} \,, \quad \forall t \in \mathcal{T} \,, \quad \forall z \in \mathcal{Z} \,. \tag{5.15}$$

$$f_{ij}^{z,t} \leq \sum_{\{(i,J) \in \mathcal{A} | j \in J\}} f_{iJ}^{z,t} \,, \quad \forall i \in \mathcal{N} \,, \quad \forall j \in N(i) \,, \quad \forall t \in \mathcal{T} \,, \quad \forall z \in \mathcal{Z} \,. \tag{5.16}$$

$$\sum_{\{j \in N(i)\}} f_{ij}^{z,t} - \sum_{\{k | i \in N(k)\}} f_{ki}^{z,t} = \begin{cases} \mathcal{R}^z & i = s \\ -\mathcal{R}^z & i = t \\ 0 & o.w. \end{cases} \,, \quad \forall i \in \mathcal{N} \,, \quad \forall t \in \mathcal{T} \,, \quad \forall z \in \mathcal{Z} \,. \tag{5.17}$$

$$\sum_{\{i|j\in N(i)\}} g_{ij}^{z} \quad \begin{cases} \leq 1 & \forall j \in \mathcal{N} - \{s\} \\ = 0 & j \in \{s\} \end{cases}, \quad \forall z \in \mathcal{Z}\,. \tag{5.18}$$

$$\sum_{t\in\mathcal{T}} f_{ij}^{z,t} \leq L \cdot g_{ij}^{z}\,, \quad \forall i \in \mathcal{N}\,, \quad \forall j \in N(i)\,, \quad \forall z \in \mathcal{Z}\,. \tag{5.19}$$

$$g_{ij}^{z} \in \{0,1\}\,, \quad \forall i \in \mathcal{N}\,, \quad \forall j \in N(i)\,, \quad \forall z \in \mathcal{Z}\,. \tag{5.20}$$

5.7.2 Energy

Following the same energy assumptions for network coding, energy per hyperarc under routing would be:

$$\sum_{z\in\mathcal{Z}} f_{iJ}^{z} = e_{iJ}\,, \quad \forall (i,J) \in \mathcal{A}\,. \tag{5.21}$$

5.7.3 Delay

Since g_{ij}^{z} is used to make sure that each node on each tree has exactly one parent, the same variable may be used to find the distance of each node from source on each tree.

$$D_j^z - D_i^z \geq (L+1).g_{ij}^{z} - L\,, \tag{5.22}$$

$$D_s^z = 0, \quad D_t^z \leq \mathcal{D}\,, \tag{5.23}$$

$$D_i^z \in Z^+\,, \tag{5.24}$$

$$\forall i \in \mathcal{N}\,, \quad \forall j \in N(i)\,, \quad \forall t \in \mathcal{T}\,, \quad \forall z \in \mathcal{Z}\,.$$

5.8 Illustrative Examples and Observations

In this section, we present examples of maximum rate under network coding and routing on a 3-3 grid network for the purpose of illustration. Figure 5.2 shows a unit distance 3-3 grid network. Each node has equal radio and interference range of unit distance, i.e., each node

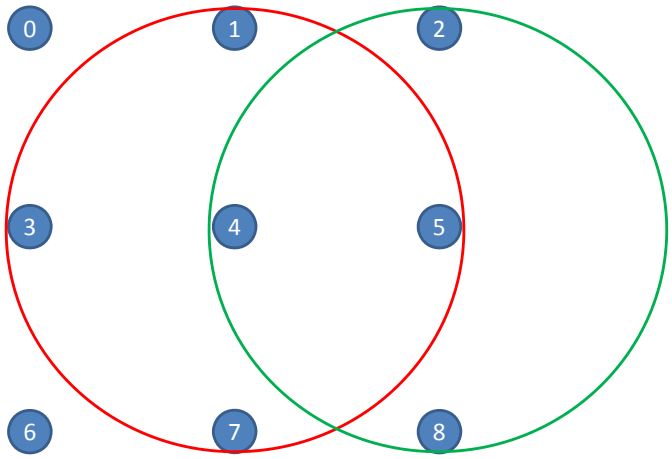


Figure 5.2 The example 3-3 grid network. All nodes have unit transmission and interference ranges.

can transmit to and interfere with all its lateral neighbors but not the diagonal ones. The interference model is protocol model with secondary interference, i.e., a receiver may not be in the range of two transmissions. All wireless links are assumed to have unit capacity.

Next we look at the shape of optimal solution for an example multicast session of $(0, (5, 7))$. Under network coding, the maximum achievable rate is found by timesharing over 5 maximal independent sets (MISs) as shown in Figure 5.3. In addition to point-to-point links, in two cases, MISs include point-to-multipoint links, i.e., links $(4, \{5, 7\})$ and $(8, \{5, 7\})$. All such links are, of course, modeled as hyperarcs in the hypergraph modeling. For each MIS, its corresponding optimal timeshare is also given. Note that sum of all timeshares is equal to 1.

In order to find the scheduled capacity for each link, we need to sum timeshares of all MISs covering that link. For example link $(0, 1)$ appears in two MISs with timeshares of $1/8$ and $1/4$. Therefore the scheduled capacity of link $(0, 1)$ is $3/8$. Figure 5.4(a) shows the calculated scheduled capacities for all links. Only links with non-zero capacity are shown. Figures 5.4(b) and 5.4(c) show the flow from source (node 0) to each terminal. Note that flows respect the scheduled capacity at each link and flow conservation at each intermediate node. The sum of flows to each terminal is equal to $3/4$ which gives the maximum achievable rate of 0.75.

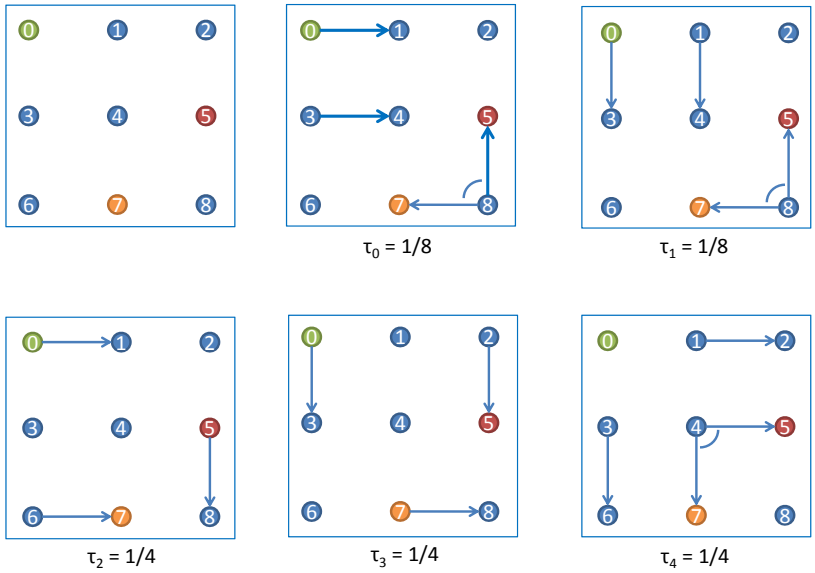


Figure 5.3 Maximum rate with network coding: optimal scheduling.

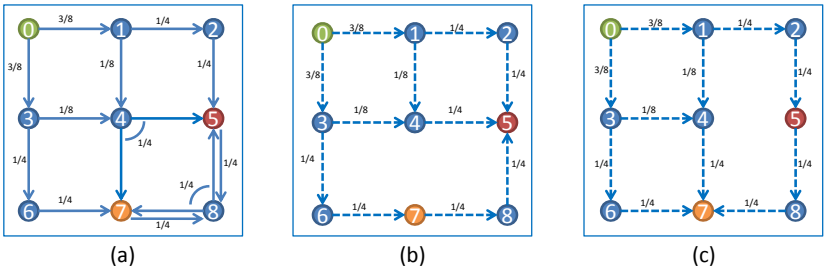


Figure 5.4 Maximum rate with network coding: (a) scheduled capacities, (b) flows to the first terminal (node 5), (c) flows to the second terminal (node 7). Multicast rate = 3/4.

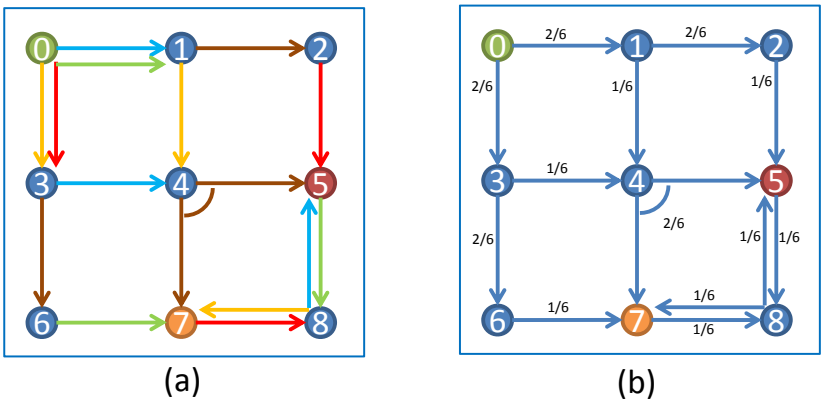


Figure 5.5 Maximum rate with routing: (a) 5 MISs shown in different colors, (b) scheduled capacities.

Close observation of Figure 5.4(a) shows that network coding is used at node 8. The incoming links $(5, 8)$ and $(7, 8)$ each have scheduled capacity of $1/4$. There is one outgoing point-to-multipoint link $(8, \{5, 7\})$ with scheduled capacity of $1/4$. Under routing assumptions, the outgoing capacity of $1/4$ would become the bottleneck as compared to the sum of incoming capacities which is $1/2$. However, under network coding assumptions, if the incoming flows belong to different terminals (as shown in Figures 5.4(b) and 5.4(c)), they can be coded together. Therefore the outgoing capacity of $1/4$ would be enough.

We also investigate maximum rate under routing for the same example. This time, the solution includes 5 different MISs. For brevity, we have summarized all MISs in one figure. Figure 5.5(a) shows 5 MISs in different colors. In the solution, each MIS gets a timeshare of $1/6$ except for MIS $\{(1, 2), (3, 6), (4, \{5, 7\})\}$ (in dark brown) whose assigned timeshare is $2/6$. Therefore, sum of timeshares is 1. We have calculated the scheduled capacity for each link in Figure 5.5(b). For example, link $(0, 1)$ appears in two MISs (two colors) with timeshares of $1/6$ assigned to each. Therefore, the scheduled capacity of link $(0, 1)$ is $2/6$.

As shown in Figure 5.6, the routing ILP packs 4 multicast trees on this network each delivering a flow of $1/6$. As a result, the multicast rate under this routing solution is $4/6$ or approximately 0.66. We also examined ILPs for packing 5 and 6 trees and maximum rate under routing did not increase; this could suggest that rate of 0.66 is in fact maximum rate under routing.

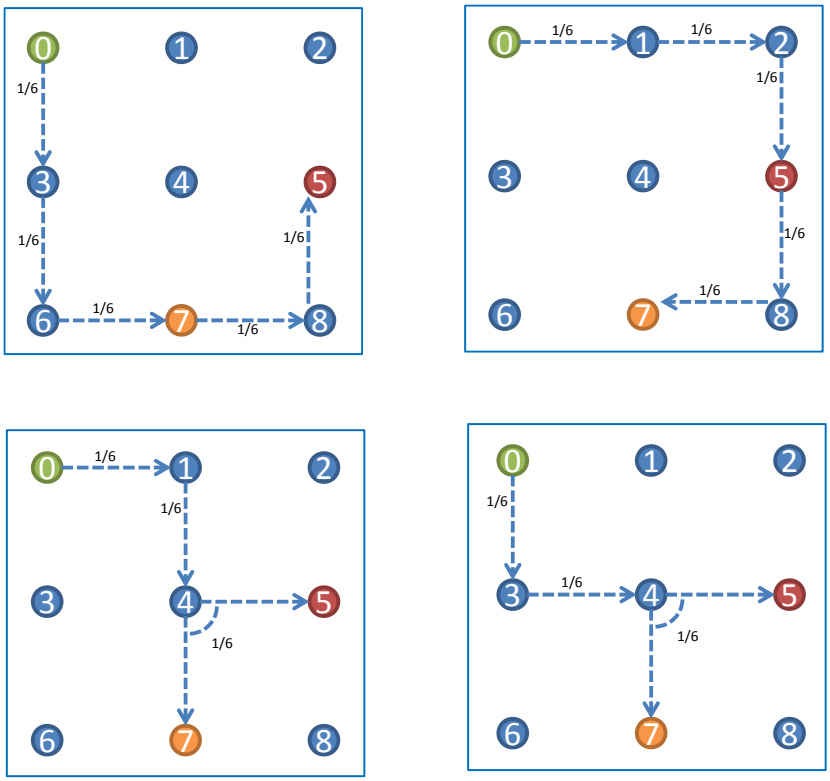


Figure 5.6 Maximum rate with routing: 4 multicast trees each sending a rate of 1/8. Multicast rate = 4/8.

5.8.1 Observation: network coding theorem in joint optimization framework

An interesting observation is related to scheduling and network coding being jointly solved. When network coding is solved independent of scheduling, network coding theorem states that maximum multicast rate is equal to the minimum of max-flows from source to all terminals. *However, the same theorem may not be applied to the joint scheduling and network coding framework.* To show this, one can use the same joint scheduling and network coding formulation to find the max-flow between source-terminal pairs. In our case, the max-flow in either case of $(0, 5)$ or $(0, 7)$ turns out to be higher than maximum multicast rate of 0.75. The max-flows values are indeed equal to $7/9$ or approximately 0.77 (not shown in the figures).

The reason is that optimizing for a single pair such as $(0, 5)$, would result in a different scheduling and a different set of scheduled capacities than optimizing for the multicast session $(0, (5, 7))$. In other words the max-flow and multicast problems are solved on different sets of *realizable* networks[1].

5.9 Experimental Evaluation and Discussion

We have formulated rate, energy, and delay under both network coding and routing assumptions in Section 5.6 and Section 5.7. Each of the three performance parameters can be used as an objective or a constraints in an optimization problem. Here, we wish to consider the following optimization problems:

- Maximizing rate
- Minimizing energy subject to the rate demand
- Minimizing delay subject to the rate demand

We are going to solve these problem in both network coding and routing paradigms.

We start with the 4-5 grid network with unit lateral distance in Figure 5.7. Each node has equal radio and interference range of unit distance. In other words a node can transmit to all its lateral neighbors but not the diagonal ones. The interference model is protocol model with secondary interference, i.e., a receiver may not be in the range of two transmissions.

[1]To the best of our knowledge, this observation has not been made before.

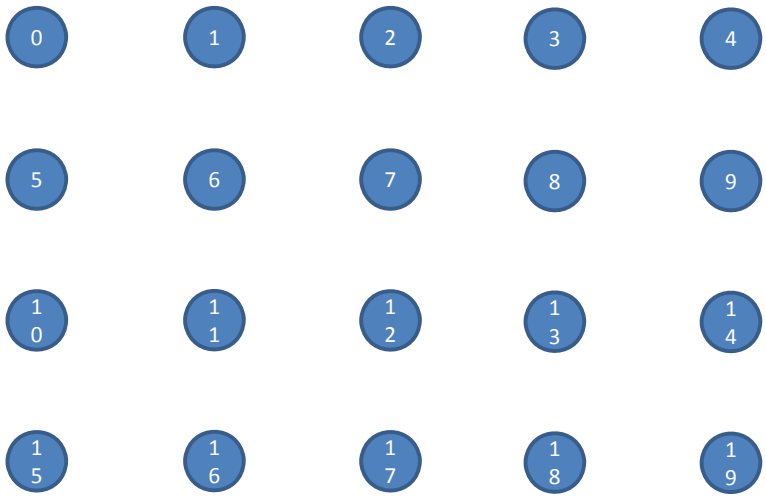


Figure 5.7 4-5 grid network.

Table 5.1 gives the general information about some grid networks. Each row represents a grid network of a different size. While number of hyperarcs increases linearly with number of nodes, the total number of MISs increases exponentially. We note that MISs are found on a conflict graph whose vertexes represent hyperarcs in the network graph. After modeling each grid network and its corresponding conflict graph, we used *igraph* library [23] to find all MISs.

For the 4-5 grid, we have also included a second column (marked by an asterisk) in which, instead of modeling hyperarcs, only simple arcs are modeled. Modeling just simple arcs is equivalent to not having any broadcast link transmission in the network, which is a disadvantage. We have included this column to show its effect on the size of conflict graph and number of MISs. Independence number is the size of largest independent set, i.e., maximum independent set.

As the last row in Table 5.1 shows, even 5-5 grid has over 13 million MISs. Therefore, to consider all MISs, the formulation would require the same number of timesharing variables. For our primary simulation results, we work with the 4-5 grid network (Figure 5.7) whose set of MISs is still manageable. We generate 100 random multicast sessions of size 3. Source and two terminal nodes are selected randomly out of 20 nodes in the grid for each sessions. We use IBM ILOG CPLEX 12.5 [33] as the optimization solver.

5.9.1 Complexity

The first and most important difficulty with solving any of the above optimization problems is dealing with exponential number of MISs. Formulating the set of all scheduled capacities requires collecting exponentially many MISs which is hard. The second hard component has to do with packing Steiner trees for the routing formulation which is NP-hard [37]. In other words, even given the set of scheduled capacities, solving routing optimization problems is NP-hard. In network coding, in contrast, solving maximum rate and minimum energy problems for a given set of scheduled capacities takes the form of linear program which is solvable in polynomial time. Finally, the third hard component is the delay. Our formulation of delay uses integer variables which even in the case of network coding results in an ILP formulation and is hard to solve.

In what follows, we propose heuristic scheduling methods to reduce the complexity of scheduling sub-problem and yet arrive at near optimal results.

Table 5.1 Properties of sample grid networks. In 4-5 grid*, simple arcs are used instead of hyperarcs.

	2-2 grid	3-3 grid	4-4 grid	4-5 grid	5-5 grid	4-5 grid*
# of nodes	4	9	16	20	25	20
# of hyperarcs	12	55	128	172	231	62
Independence #	2	4	8	10	11	10
# of MISs	8	293	30,908	460,821	13,608,626	24,132

5.9.2 Maximizing the rate

We consider this problem as the core component in both network coding and routing paradigms. Since considering all MISs results in long convergence times even for network coding, we examine 4 sub-optimal cases that involve collecting only a subset of all MISs, in addition to complete set of MISs. Random MISs are found by randomly selecting (without replacement) a given number MISs from the pool of all MISs.

- All MISs (MIS-all)
- All MISs when a simple-arc model is used (MIS-arc)

- 500 randomly selected MISs (MIS-500)
- 1000 randomly selected MISs (MIS-1000)
- 2000 randomly selected MISs (MIS-2000)

5.9.2.1 Network coding

Table 5.2 shows the average rate reported by each algorithm over the same input of 100 random sessions for network coding.

Table 5.2 Average rate of different algorithms under network coding.

	MIS-arc	MIS-500	MIS-1000	MIS-2000	MIS-all
Average rate	0.719	0.793	0.830	0.836	0.861
Optimality gap	16%	8%	4%	3%	0%
Time (s)	0.241	0.020	0.028	0.043	6.171

Given that optimal solution was found over nearly half a million MISs (Table 5.1), we see a remarkable performance by randomly selecting a much smaller number of MISs. For instance, with only 1000 randomly selected MISs an optimality gap of 4% is achieved. The figure also shows the importance of hyperarc modeling. The simple-arc modeling removes the benefit of broadcast transmission in the wireless grid. Under simple-arc modeling, even considering all MISs results in much lower performance (16%) compared to selecting 1000 random MISs under hyperarc modeling (4%).

Another advantage for randomly selecting a subset of MISs is the convergence time in CPLEX. The reported time is the average time for a single multicast session in seconds. While MIS-arc reduces the time at the cost of 16% optimality gap, MIS-1000 reduces the gap to 4% and further reduces the time by one tenth compared to MIS-arc. We use MIS-1000 is our heuristic of choice to deal with the first hard component of the problem namely exponential number of maximal independent sets.

5.9.2.2 Routing

We use MIS-1000 scheduling to find maximum rate for the same set of 100 random sessions under routing. As we discussed before, maximum achievable rate depends on the number of

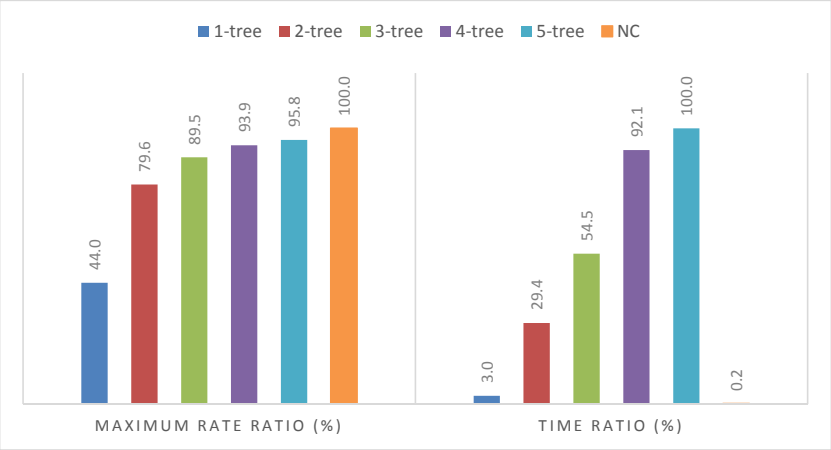


Figure 5.8 Maximum rate: routing vs. network coding.

disjoint multicast trees we pack. Figure 5.8 shows the maximum rate and the convergence time for packing 1 to 5 disjoint trees alongside the network coding result for MIS-1000. The figure shows the rate percentage of each routing method with respect to network coding. The convergence time ratio is also shown as a percentage with respect to largest time: packing 5 trees.

While convergence was completed for 1 and 2 trees, we had to a set time limit of 120 seconds per session for 3, 4, and 5-tree cases. It could virtually take an unbounded time for the solver to converge for all sessions. Therefore, the figure is rather a representation of time-effort versus achievable rate when it comes to routing.

The figure suggests that routing rate can get very close to network coding rate however at a much larger time. For example, the 5-tree result has a gap of mere 4.2% with network coding but its time-to-converge is not comparable to network coding: average time of 111 seconds for 5-tree routing versus 0.028 seconds for network coding. Using a single tree provides a reasonable convergence time but the the rate is dropped by 56%.

5.9.3 Minimizing energy

We now look at the problem of minimizing energy for a given rate demand. This problem represents relation between energy and rate in our model of wireless multicast and solving it

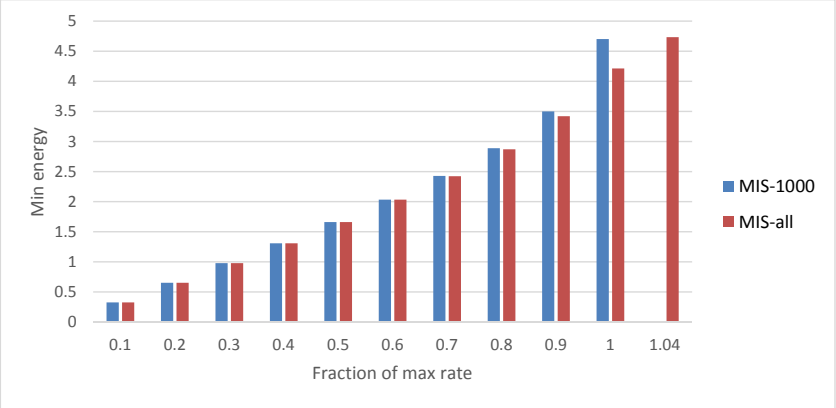


Figure 5.9 Energy as a function of rate demand.

could provide insight into the trade-off between the two performance measures. We also use energy as another measure to compare MIS-1000 with MIS-all here.

Figure 5.9 shows a column diagram for each of the scheduling methods. The horizontal axis is scaled according to the maximum rate of MIS-1000. The maximum rate of MIS-all is about 4% more at 1.04. Lower points are fractions in increments of 0.1 of MIS-1000 maximum rate. Some interesting observations:

- MIS-1000 only lags behind MIS-all when the demanded rate reaches the maximum value. For all lower fractions the energy results are nearly equal.
- In both methods energy is a non-linear and increasing function of rate, i.e., higher rates require increasingly higher energy.
- Energy reported by MIS-all at its maximum rate (point 1.04) is nearly equal to energy reported by MIS-1000 at its maximum rate (point 1).

Next we investigate how routing and network coding compare in terms of energy it takes to deliver certain fractions of maximum rate. For the scheduling we use MIS-1000. Network coding is compared with 3-tree routing. In Figure 5.10, the maximum rate of 3-tree routing is normalized as 1.0[2]. Horizontal axis shows fractions of maximum 3-tree rate in increments of 0.1 from 0.1 to 1.0.

[2]Note that maximum rate under 3-tree routing was, on average, about 10% lower than that of network coding (Figure 5.8).

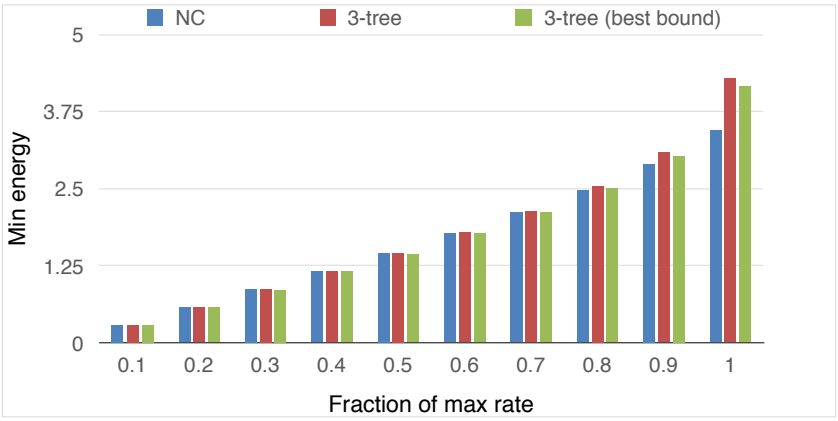


Figure 5.10 Minimum energy for different fractions of maximum rate: network coding vs. routing.

Minimum energy of network coding for each fraction and each session can be found quickly by solving corresponding LPs. However in the case of routing, in order to collect results for 100 sessions on 10 fractions each, we had to introduce time limits for each ILP. The shown results are based on a 120-second time limit. For each ILP, in the given time limit, we collect both best integer solution and best integer bound. The difference between the two values represents the optimality gap. However, since the integer solution and best bound only show a 3% gap in the worst case (point 1.0), 3-tree results are optimal for the most part and close to optimal in higher rates. We make the following observations:

- 3-tree routing has a very close energy performance compared to network coding for fractions 0.8 and lower.
- Network coding performs better in higher rates. Specifically at the maximum rate case (point 1.0), 3-tree routing requires over 20% more energy to deliver the same rate.
- Routing, too, shows non-linear energy-rate relation: increasingly more energy at higher rates.

5.9.4 Minimizing delay

Next we look at the relation between rate demand and delay. This is done through minimizing delay for a range of rate demands. We work with the same set of 100 random sessions. The

delay is measured in the number of hops for the longest flow path for each session (Equations (5.9) to (5.12)). Including delay constraints results in an ILP even under network coding. Due to convergence time problem, we only present network coding results here.

In Figure 5.11, on the horizontal access, the maximum rate under network coding without any delay constraint is normalized as 1. Lower fractions of maximum rate are also listed in increments on 0.1, starting from 0.1. For each of the 100 random sessions, we have found the minimum delay for all fractions of maximum rate (from 0.1 to 1.0). Because of the convergence time problem, we introduced a 120-second time limited for each ILP. Also we limited the results to MIS-1000 scheduling.

For each ILP, in the given time limit, we collect both best integer solution and best integer bound. The difference between the two values represents the optimality gap. Each point on the horizontal axis of Figure 5.11 shows the best integer solution and best integer bound, averaged over 100 sessions. As Figure 5.11 shows, we in fact have the optimal results for maximum rate (fraction 1.0). The other non-optimal results are lower-bounded by the best bounds.

To put the results in perspective, we also calculated the actual source-terminal shortest path lengths on the 4-5 grid. The average shortest path length (hop-distance) between source and farthest terminal node for all sessions is 3.88 hops. We make the following observations:

- With increasing rate demand, minimum delay can only grow. On one end of the diagram, point 0.1, the integer solution is very close to the best bound and on the other end, point 1.0, we do have the optimal integer solution. For the other middle points, for instance point 0.4, the integer solution of 0.1 in fact provides a better lower-bound. Therefore, our reported integer solutions are much closer to their optimal value than what their best bounds might imply.
- Delay, too, demonstrates a non-linear behavior with respect to increasing rate demand. As Figure 5.11 shows, approaching the maximum rate limit, the number hops source-terminal flows have to travel increases sharply. What happens here is that paths tend to go through boundaries of the network to reduce interference and deliver more rate.

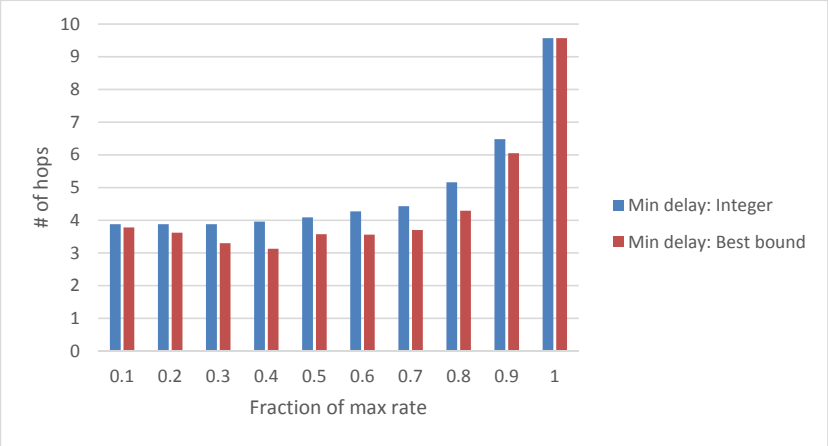


Figure 5.11 Delay as a function of rate demand.

- The average number of hops for the maximum rate is 9.57. When compared to the previously reported average hop-distance of 3.88 on the grid, it shows how longer and more diverse paths are chosen to avoid interference.
- The largest delay was reported for session $(2, \{3, 9\})$ which is 13 hops. This is equal to the longest path on the boundary of the grid in Figure 5.7: starting from node 2, moving counter-clockwise on the boundary, and ending at node 3. Again this example shows how longer and more diverse paths are used to avoid interference.

5.9.4.1 Effect of delay on energy

We also examined the difference delay constraint makes in minimum energy. Here, we find minimum energy for maximum rate with and without a tight delay constraint. Each session has a maximum rate under MIS-1000. We have also found minimum energy for the maximum rate under MIS-1000 previously. We also have the minimum delay (number of hops) required for maximum rate for each session. We find minimum energy for maximum rate subject to delay being less than or equal minimum delay. This is compared to minimum energy for maximum rate when there is no delay constraint. Our result averaged over 100 random sessions shows adding minimum delay constraint increases the minimum required energy by 12% from average of 4.21 to 4.71.

5.9.5 Maximum rate on larger networks

There is an interesting observation when looking at the maximum rate results on the 4-5 grid: sessions whose source and terminals are closer to center of the network and farther from boundaries show higher rates, particularly 1 or close to 1. It is worth noting that in our network model with unit capacity wireless links, maximum rate for any connection (unicast or multicast) is upper-bounded by 1. This can be seen by looking at the outgoing flow from source or incoming flow to a terminals. For example no two outgoing links of source node can be active at the same time. Given that all individual links have unit capacity, by any assignment of timeshares the scheduled outgoing capacity of the source would be at most 1.

It would, therefore, seem that given a sufficiently large grid network, the maximum rate for a multicast session whose source and terminals are sufficiently far from boundaries should be 1. We examined this idea experimentally by extending the 4-5 grid to a 10-10 grid. The 100 random sessions are modified such that they all fall into a central 4-5 subset of the extended grid. In other words with respect to the 4-5 subset, we have the same set of random sessions.

Table 5.3 shows the average maximum rate for the same 100 random sessions mapped to the center of a larger 10-10 grid. It was not possible to find all maximal independent sets for a 10-10 grid as they are in the order of 2^{100}. Therefore, we found results for 1000, 2000, 5000, and 10000 randomly selected sets of MISs. In this case since the pool of all MISs is not available, we would build the given number (e.g., 1000) of random MISs one by one. To build each random MIS, we start by a new random ordering of all hyperarcs and an empty independent set. We then process the list from beginning to end adding hyperarcs if and only if they are mutually non-conflicting with all hyperarcs currently in the set. The resulting set is maximal in the sense that no new hyperarcs may be added.

Based on the results in Table 5.3, we make two conclusions:

- The average maximum rate being very close to 1, confirms our conjecture about achieving upper-bound of 1 given a sufficiently large network. To better see the effect of a larger network, one can compare the averages in Table 5.3 with the average maximum rate reported for the same set of sessions on the 4-5 grid in Table 5.2: 0.861.

- Achieving the upper-bound also shows the effectiveness of scheduling heuristic: randomly selecting a relatively small subset of all MISs.

Table 5.3 Maximum rate for 100 session mapped to the center of a larger 10-10 grid.

Scheduling	MIS-1000	MIS-2000	MIS-5000	MIS-10000
Average max rate	0.968	0.988	0.998	0.999

5.9.6 Analytical insight on MISs

We seek an explanation as for why such relatively small subsets of random MISs are capable of achieving near optimal results. We draw insight from the core component of multicast network coding problem, i.e., the maximum flow problem. We claim that maximum flow between any pair of nodes on an infinite gird can be found using just 3 independent sets. We show this by constructing the solution.

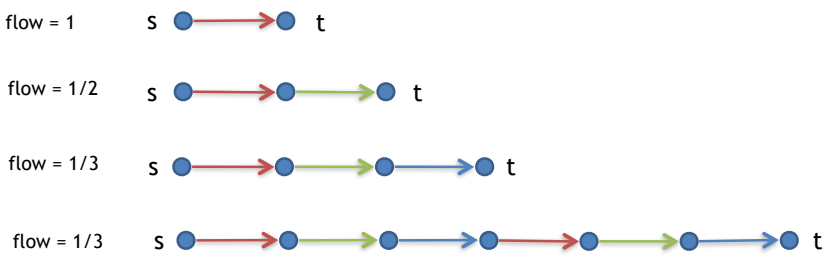


Figure 5.12 Flow on a single path.

In Figure 5.12 colors red, green and blue represent 3 different timeshares. Under our network and interference model:

- A single hop path can carry a unit flow.
- Two timeshares are required in a 2-hop path resulting in a flow of 1/2.
- In a 3-hop path, no two links may be active at the same time. This results in 3 timeshares and a flow of 1/3.
- In a path longer than 3 hops, the same timeshares may be repeated (red, green and blue) without violating interference or wireless constraints.

Therefore, a flow of at least 1/3 may always be sent from s to t by a single path. Given a large enough grid, we can find 3 paths diverse enough such that the inter-path interference is limited to 3 outgoing links of s and 3 incoming links of t. Paths then may be adjusted so that all paths start in different colors (outgoing links of s) and end in different colors (incoming links of t) while the sequence of red, green and blue is repeated on each path.

Figure 5.13(a) shows a scenario where two paths end in red color (same timeshare) which violates the single transceiver constraint at t. In Figure 5.13(b), top and bottom paths are extended by 2 hops to resolve this problem. In general, a path that ends in red, for instance, would end in blue when extend by 2 hops. With another 2-hop extension, the path would end in green (since the color sequence repeats). Therefore, we can always make sure that all flow paths start and end in different colors. As a result a maximum flow of 1 is sent via 3 paths using even scheduling on 3 independent sets (colored red, green and blue). It is important to note that under our network and interference model, rate of 1 is the upper-bound. The main

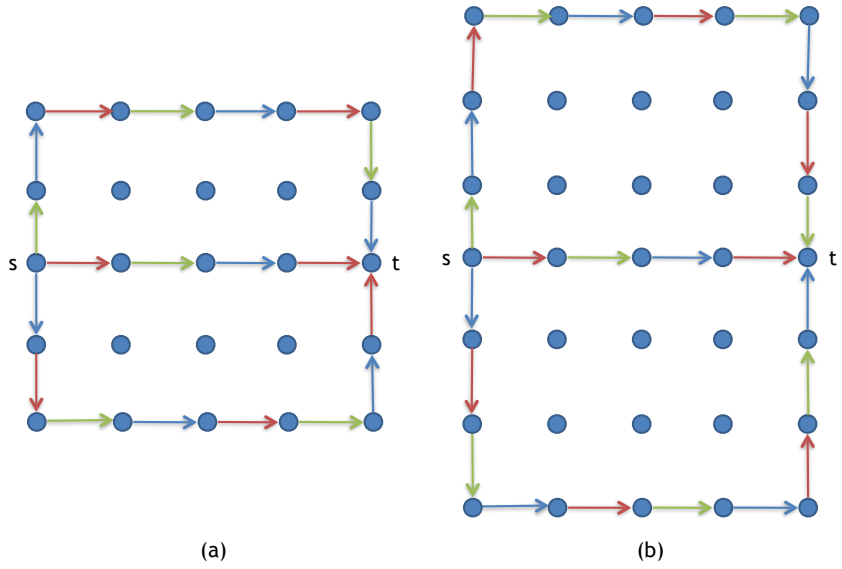


Figure 5.13 Scheduling 3 flow paths. (a) Two paths end in the same color (red). (b) Top and bottom paths are extended by 2 hops; All paths end in different colors. Maximum flow of 1 is achieved by 3 independent sets (red, green, and blue).

observation is that while the formulation of optimal scheduling involves exponential number

of MISs, the solution itself may not. This is one explanation for why small random subsets of MISs were shown to be efficient in our experimental evaluations.

5.10 Conclusion

We built on the formulation of multicast rate region under joint scheduling and network coding for rate maximization and extended it to include energy and delay minimization subject to rate demand. In a similar way, rate maximization, energy and delay minimization are also formulated under joint scheduling and routing. The primary hard component of both formulations, i.e., scheduling MISs, is addressed by randomly selecting a relatively small subset of MISs. The rate maximization results show the effectiveness of this simple scheduling method as compared to considering all of the exponentially growing MISs. Joint scheduling and routing was shown to be able to perform close to joint scheduling and network coding in terms of maximum rate. However the solution suffered from huge increase in running time (average time of 111 seconds for 5-tree routing versus 0.028 seconds for network coding). The relation between energy and rate was investigated in joint scheduling and network coding/routing. Minimum energy was shown to grow non-linearly as a function of rate demand in both scenarios. It was also shown that only when approaching the maximum rate, network coding provides energy benefit over a time-limited routing solution. The same analysis was also performed for the relation between delay and rate. The results show non-linear delay growth as a function of rate; the delay increases sharply when rate approaches its maximum. Finally, the effectiveness random selection of MISs was verified again by achieving the upper-bound multicast rate on a larger network. We have also presented analytical insights on the effectiveness of selecting small MIS subsets.

One direction for future work involves examining the energy-rate and delay-rate non-linearity in more general or realistic network models such as lossy channel models, physical interference model, and multiple transmission powers. Another is to extend our theoretical analysis supporting the experimental results on effectiveness of random MISs. The whole study can also be extended for multiple multicast sessions with varied session sizes.

CHAPTER 6. CONCLUSION

This dissertation has been an effort in investigating and utilizing the potential of network coding in improving performance of communication networks. The diversity of applications discussed are but an indicator of such potential: from survivability enhancement for the Internet backbones to energy saving in wireless networks.

6.1 Network Coding for Survivability in Backbone Networks

In the first component, application of network coding in protection of unicast and multicast optical connections was investigated. Due to the challenges in implementing full linear network coding capability at optical level, the proposed approaches are designed based on simple network coding operations, namely logical OR and XOR.

In the case of unicast connections, our work was based on the idea of 1+N protection [41]. Since the optimal cost 1+N solution is NP-hard to find, one may only compute such optimal solutions offline. This is not suitable for protection of dynamic traffic in real-world scenarios. Therefore, the challenge is to design heuristic algorithms that can deliver solutions with near-optimal cost and have online computable polynomial running time. In order to design such an algorithm, the problem was first converted to a partitioning problem. The main questions are: which unicast connections should be protected together and which connections should be protected independently. To protect connections against single link failures, a group of jointly protected unicast connections would be provisioned by link-disjoint working paths and protected by a link-disjoint protection subgraph that takes the form of a Steiner tree.

A greedy partitioning algorithm is designed that adds a new unicast connection to a group of jointly protected connections only if a local cost function is reduced. Once the partitioning

is complete, Greedy Shortest Paths and Greedy Steiner Tree heuristic algorithms are used to find link-disjoint working paths and link-disjoint Steiner protecting subgraph for each partition. Different partitions are provisioned and protected independently. The proposed heuristic approach has a worst case time complexity of $O(|V|^2.|C|^4)$ where $|V|$ is the number of vertexes (nodes) in the network graph and $|C|$ is number of unicast connections to be protected.

Next, the cost of the algorithm is evaluated experimentally by simulating different network scenarios. In comparison to the traditional 1+1 protection, the algorithm shows cost savings of 21.5%, 34.5%, and 60.2% in 14-node NSFNET, 11-node COST239, and 14-node complete graph networks. The main observation is that the cost efficiency of our heuristic 1+N algorithm increases when the number of connections or graph density is increased.

Moreover, the performance of our online heuristic is compared to offline optimal solution. Due to long convergence times, comparison is done for two cases of 5 and 10 connections in NSFNET and COST239 networks. In worst case scenario, our algorithm has 5.7% and 13% increase in cost respectively. Finally, the performance of 1+N protection in asymptotic case is analytically investigated which shows that, compared to traditional 1+1 protection, 1+N can achieve 66.6% cost reduction in complete graphs.

In the case of multicast protection, the idea of multicast *1+1* protection is proposed. It is based on the traditional routing-based unicast *1+1* protection: two copies of each data unit are sent from source to the destination to guarantee the data delivery in the event of a single link failure. The same approach can be applied to multicast protection; two copies of each data unit are sent to each terminal. However, unless flows to different terminals are allowed to share link capacity on common links, this method would be inefficient in terms of bandwidth usage.

The bandwidth efficiency comes at the cost of increased recovery delay. An intermediate node might have to select to forward only one of the incoming flows. In the event of an upstream link failure, the intermediate node must reconfigure its switch to forward another working flow and optical switch reconfiguration is time-consuming. Fortunately network coding has a solution for this problem; it is via a simple logical operation (OR) at the intermediate node. Instead of forwarding only one of the incoming flows, the intermediate node would code (or merge) all incoming flows into one outgoing flow by OR operation. As long as at least one

incoming flow is intact, outgoing flow does not change and the failure is masked. No rerouting or switch reconfiguration is required and destination nodes would not experience any service disruption under any single link/node failure. The necessary and sufficient conditions for the existence of *1+1* protection solution under merging flows is simply 2-connectivity from source to each destination. This allowes us to formulate the problem as a network flow problem which can be solved to find minimum cost subgraph supporting multicast *1+1* protection. In addition to optimal ILP formulation, three heuristic algorithms are also proposed as online solutions.

The simulation results on different versions of COST239 network show remarkable performance by our best heuristic algorithm; compared to offline optimal solution, the cost is increased by no more than 2.6% in the average case and by no more than 4.7% in the worst case. Future work in multicast 1+1 protection would involve implementation of logical coding operation at optical level, addressing the buffering and synchronization issues at the optical level, and considering more realistic traffic models, bandwidth granularity, and traffic grooming.

6.2 Network Coding for Performance Optimization in Wireless Networks

Performance optimization in wireless networks is more challenging compared to wired network due to inherent wireless constraints. For example, one such constraint is the number of radios available at a wireless node. A single transceiver radio, for instance, would not allow simultaneous transmission and reception at a wireless node. Another important challenge is wireless interference. In comparison to a typical wired network, where all links can be independently active, wireless links may interfere with or disrupt each other if simultaneously use the same channel.

Given the mobility and reliance on battery power, energy consumption becomes an even more important performance parameter in wireless networks. The limited bandwidth in single channel scenarios would also require better utilization of available bandwidth.

Similar to the wired networks, network coding generalizes the routing *logic* when applied to wireless networks. Such generalization, in particular, applies to broadcast links. Point-to-multipoint or broadcast links are one of inherent benefits of using wireless medium in the

routing paradigm. Network coding, too, takes advantage of this mode of communication, to broadcast coded data to multiple receivers.

Network coding has been shown in the past to provide advantages in terms of rate, energy, and delay in wireless networks. We seek to address network-layer performance optimization and MAC-layer interference management is a joint manner. We address interference constraints in wireless by scheduling wireless links. Given a valid schedule of wireless links, we have what is referred to as a *realizable* network. Since solving performance optimization problems depends on underlying *realizable* network, these two components are jointly solved.

We built on an existing formulation of multicast rate region under joint scheduling and network coding due to [82]. To formulate the optimal solution, in the first step, a hypergraph model is used to capture every possible point-to-multipoint transmission at every network node. Next, all interference and single transceiver constraints are incorporated into a conflict graph. Vertexes in the conflict graph represent wireless links or hyperarcs. Two vertexes are adjacent if and only if they conflict with one another; they cannot be active at the same time. Scheduling interference-free wireless links, therefore, takes the form of timesharing over MISs in the conflict graph. The main problem here is that there are exponential number of MISs to account for. On the positive side, such timesharing formulation is linear.

We extended the performance optimization to include rate, energy, and delay. Moreover, routing-based formulation is presented in addition to network coding. This problem involves three levels of complexity: scheduling over exponential number of MISs, routing, and performance optimization (specially delay minimization). We observe that the scheduling is the common hard component in all of the intended performance optimizations. We propose to perform scheduling over small random subsets of MISs, thereby simplifying the common hard component.

The effectiveness of heuristic scheduling, rate and energy benefits of network coding over routing, and energy-rate and delay-rate relationships are experimentally evaluated on grid network topology.

- The heuristic scheduling is shown to be very effective in achieving near-optimal multicast rate and energy. It is also shown that a small random subset of MISs continues to achieve near optimal rate as the number of all MISs exponentially grows in a larger network.

- Energy-rate and delay-rate relationships is shown to be non-linear. The solution requires increasingly more energy and longer paths as multicast rate demand increases. We are able to intuitively explain this phenomenon by looking at the shape of solutions. In order to deliver higher rates, the flow paths tend to diverge to the boundaries of the network, thereby reducing inter-path interference. Longer paths in this case would result in more energy consumption and more delay; both have non-linear growth.

- With regard to comparison of network coding and routing, a 5-tree routing solution achieves nearly the same maximum multicast rate achieved by network coding. The routing solution, however, suffered from a dramatic increase in running time: average time of 111 seconds for 5-tree routing versus 0.028 seconds for network coding. Furthermore, by looking at the energy-rate relationship in both paradigms, it is shown that energy benefits of network coding are unevenly distributed. In particular, network coding benefit grows as multicast rate demand approaches maximum achievable rate. In the lower rates, the energy offered by routing and network coding solutions are mostly equal. Another advantage of network coding, again, is its much lower convergence time.

- To support the experimental evidence given for the effectiveness of the small random subsets of MISs, we also take an analytical approach to the problem. The maximum flow problem, as the core problem in network coding multicast, is considered. It is shown that given a large enough grid network, the upper-bound rate of 1 is always achievable by using only three independent sets.

In conclusion, the combination of polynomial-time and effective scheduling heuristic, and linear formulation of network coding, provides an online algorithm for rate and energy optimization is wireless multicast. The fact that scheduling and network coding both rely on random selection, of MISs and network codes respectively, makes both methods easy to implement.

One direction for future work involves examining the energy-rate and delay-rate non-linearity in more general or realistic network models, e.g., lossy channel models, physical interference model, and multiple transmission powers. Theoretical explanation of these relationships, although difficult due to the multi-level complexity of the problem, is yet another direction. The whole study can also be extended for multiple multicast sessions with varied session sizes.

BIBLIOGRAPHY

[1] Apsi wifi services llc. http://apsiwifi.com. Accessed: 12/1/2014.

[2] Chocolate cloud. http://www.chocolate-cloud.cc. Accessed: 12/1/2014.

[3] Code on technologies. http://www.codeontechnologies.com. Accessed: 12/1/2014.

[4] Project avalanche. http://research.microsoft.com/en-us/projects/avalanche/. Accessed: 12/1/2014.

[5] Steinwurf. http://steinwurf.com. Accessed: 12/1/2014.

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

[7] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[8] E. Balas and M.W. Padberg. Set partitioning: A survey. *Siam Review*, 18(4):710–760, 1976.

[9] Peter Batchelor. *Ultra high capacity optical transmission networks: Final report of action COST 239*. Faculty of Electrical Engineering and Computing, 1999.

[10] Emmanuel Bertin, Noel Crespi, and Thomas Magedanz. *Evolution of Telecommunication Services: The Convergence of Telecom and Internet: Technologies and Ecosystems*. Springer Berlin Heidelberg, 2013.

[11] R. Bhandari. *Survivable networks: algorithms for diverse routing*. Kluwer Academic Pub, 1999.

[12] Aruna Prem Bianzino, Claude Chaudet, Dario Rossi, and J Rougier. A survey of green networking research. *Communications Surveys & Tutorials, IEEE*, 14(1):3–20, 2012.

[13] M. Cha, W. Art Chaovalitwongse, J. Yates, A. Shaikh, and S. Moon. Efficient and scalable provisioning of always-on multicast streaming services. *Computer Networks*, 53(16):2825–2839, 2009.

[14] Meeyoung Cha, Gagan Choudhury, Jennifer Yates, Aman Shaikh, and Sue Moon. Case study: Resilient backbone design for iptv services. In *Workshop on IPTV Services over World Wide Web*, 2006.

[15] Kai Chen, Chengchen Hu, Xin Zhang, Kai Zheng, Yan Chen, and Athanasios V Vasilakos. Survey on routing in data centers: insights and future directions. *Network, IEEE*, 25(4):6–10, 2011.

[16] Yan Chen, Shunqing Zhang, Shugong Xu, and Geoffrey Ye Li. Fundamental trade-offs on green wireless networks. *Communications Magazine, IEEE*, 49(6):30–37, 2011.

[17] J. Cheriyan, B. Laekhanukit, G. Naves, and A. Vetta. Approximating rooted steiner networks. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1499–1511. SIAM, 2012.

[18] Philip A Chou, Yunnan Wu, and Kamal Jain. Practical network coding. In *Proceedings of the annual Allerton conference on communication control and computing*, volume 41, pages 40–49. The University; 1998, 2003.

[19] J. Chuzhoy and S. Khanna. Algorithms for single-source vertex connectivity. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 105–114. IEEE, 2008.

[20] C.K. Constantinou and G. Ellinas. A novel multicast routing algorithm and its application for protection against single-link and single-link/node failure scenarios in optical wdm mesh networks. *Optics Express*, 19(26):B471–B477, 2011.

[21] C.K. Constantinou and G. Ellinas. Survivable multicast routing in mixed-graph mesh optical wdm networks. In *Optical Network Design and Modeling (ONDM), 2012 16th International Conference on*, pages 1–4. IEEE, 2012.

[22] Marco Conti, Song Chong, Serge Fdida, Weijia Jia, Holger Karl, Ying-Dar Lin, Petri Mähönen, Martin Maier, Refik Molva, Steve Uhlig, et al. Research challenges towards the future internet. *Computer Communications*, 34(18):2115–2134, 2011.

[23] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695(5), 2006.

[24] T. Cui, L. Chen, and T. Ho. Energy efficient opportunistic network coding for wireless networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 361–365. IEEE, 2008.

[25] G. Dahl. Directed steiner problems with connectivity constraints. *Discrete applied mathematics*, 47(2):109–128, 1993.

[26] A. Fei, J. Cui, M. Gerla, and D. Cavendish. A "dual-tree" scheme for fault-tolerant multicast. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 3, pages 690–694. IEEE, 2001.

[27] T. Feng, L. Ruan, and W. Zhang. Intelligent p-cycle protection for dynamic multicast sessions in wdm networks. *Optical Communications and Networking, IEEE/OSA Journal of*, 2(7):389–399, 2010.

[28] Victor Firoiu, Brian Adamson, Vincent Roca, Cédric Adjih, Josu Bilbao, Frank Fitzek, Antonia Masucci, Marie-Jose Montpetit, et al. Network coding taxonomy. 2014.

[29] A. Frank. Rooted k-connections in digraphs. *Discrete Applied Mathematics*, 157(6):1242–1254, 2009.

[30] MR Garey, RL Graham, and DS Johnson. The complexity of computing Steiner minimal trees. *SIAM journal on applied mathematics*, 32(4):835–859, 1977.

[31] Christos Gkantsidis, Mitch Goldberg, and John Miller. Avalanche: File swarming with network coding, 2005.

[32] Tracey Ho and Desmond Sui Lun. *Network coding: an introduction*, volume 6. Cambridge University Press Cambridge, 2008.

[33] IBM ILOG. Inc. cplex 12.5 user manual, 2012.

[34] IRTF. Network coding research group. https://irtf.org/nwcrg, 2013.

[35] Sidharth Jaggi, Peter Sanders, Philip A Chou, Michelle Effros, Sebastian Egner, Kamal Jain, and Ludo MGM Tolhuizen. Polynomial time algorithms for multicast network code construction. *Information Theory, IEEE Transactions on*, 51(6):1973–1982, 2005.

[36] K. Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[37] Kamal Jain, Mohammad Mahdian, and Mohammad R Salavatipour. Packing steiner trees. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 266–274. Society for Industrial and Applied Mathematics, 2003.

[38] Kamal Jain, Jitendra Padhye, Venkata N Padmanabhan, and Lili Qiu. Impact of interference on multi-hop wireless network performance. *Wireless networks*, 11(4):471–487, 2005.

[39] R. Jothi, B. Raghavachari, and S. Varadarajan. A 5/4-approximation algorithm for minimum 2-edge-connectivity. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 725–734. Society for Industrial and Applied Mathematics, 2003.

[40] A.E. Kamal. 1+ N protection in optical mesh networks using network coding on p-cycles. In *Proc. of the IEEE Globecom*, 2006.

[41] A.E. Kamal. 1+N Network Protection for Mesh Networks: Network Coding-Based Protection using p-Cycles. *IEEE/ACM Transactions on Networking*, 18(1):67–80, Feb. 2010.

[42] A.E. Kamal and O. Al-Kofahi. Toward an optimal 1+ N protection strategy. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 162–169, 2008.

[43] Ahmed E. Kamal. 1+ N protection against multiple faults in mesh networks. In *Proc. of the IEEE International Conference on Communications (ICC)*, 2007.

[44] Ahmed E. Kamal. A generalized strategy for 1+ N protection. In *IEEE International Conference on Communications, 2008. ICC'08*, pages 5155–5159, 2008.

[45] Ahmed E. Kamal and Mirzad Mohandespour. Network coding-based protection. *Optical Switching and Networking*, 2013.

[46] Ahmed E. Kamal, Aditya Ramamoorthy, Long Long, and Shizheng Li. Overlay protection against link failures using network coding. *IEEE/ACM Transactions on Networking (TON)*, 19(4):1071–1084, 2011.

[47] RM Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 1972.

[48] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Médard, and Jon Crowcroft. Xors in the air: practical wireless network coding. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 243–254. ACM, 2006.

[49] H. Kerivin and A.R. Mahjoub. Design of survivable networks: A survey. *Networks*, 46(1):1–21, 2005.

[50] Alireza Keshavarz-Haddad and Rudolf H Riedi. Bounds on the benefit of network coding for wireless multicast and unicast. *Mobile Computing, IEEE Transactions on*, 13(1):102–115, 2014.

[51] Minkyu Kim. *Evolutionary approaches toward practical network coding*. PhD dissertation, Massachusetts Institute of Technology, 2008.

[52] Minkyu Kim, Muriel Médard, and Una-May O'Reilly. Network coding and its implications on optical networking. In *Optical Fiber Communication Conference*. Optical Society of America, 2009.

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

[54] S.G. Kolliopoulos and C. Stein. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming*, 99(1):63–87, 2004.

[55] G. Kortsarz and Z. Nutov. Approximating minimum cost connectivity problems. *TF Gonzalez, editor*, 2007.

[56] P. Krysta and V. Kumar. Approximation algorithms for minimum size 2-connectivity problems. *STACS 2001*, pages 431–442, 2001.

[57] Y. Lando and Z. Nutov. Inapproximability of survivable networks. *Theoretical Computer Science*, 410(21-23):2122–2125, 2009.

[58] Gyu Myoung Lee, Noel Crespi, Jun Kyun Choi, and Matthieu Boussard. Internet of things. In *Evolution of Telecommunication Services*, pages 257–282. Springer, 2013.

[59] S-YR Li, Raymond W. Yeung, and Ning Cai. Linear network coding. *Information Theory, IEEE Transactions on*, 49(2):371–381, 2003.

[60] Shizheng Li and Aditya Ramamoorthy. Protection against link errors and failures using network coding. *Communications, IEEE Transactions on*, 59(2):518–528, 2011.

[61] L. Liao, L. Li, and S. Wang. Multicast protection scheme in survivable wdm optical networks. *Journal of Network and Computer Applications*, 31(3):303–316, 2008.

[62] Tehuang Liu and Wanjiun Liao. Multicast routing in multi-radio multi-channel wireless mesh networks. *Wireless Communications, IEEE Transactions on*, 9(10):3031–3039, 2010.

[63] Long Long and Ahmed E. Kamal. Protecting multicast services in optical internet backbones. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 57(1):17–28, 2013.

[64] Desmond S Lun, Muriel Médard, Ralf Koetter, and Michelle Effros. On coding for reliable communication over packet networks. *Physical Communication*, 1(1):3–20, 2008.

[65] Desmond S Lun, Niranjan Ratnakar, Muriel Médard, Ralf Koetter, David R Karger, Tracey Ho, Ebad Ahmed, and Fang Zhao. Minimum-cost multicast over coded packet networks. *Information Theory, IEEE Transactions on*, 52(6):2608–2623, 2006.

[66] E.D. Manley, J.S. Deogun, L. Xu, and D.R. Alexander. All-optical network coding. *Optical Communications and Networking, IEEE/OSA Journal of*, 2(4):175–191, 2010.

[67] NF Maxemchuk, DH Shur, et al. An internet multicast system for the stock market. *ACM transactions on computer systems*, 19(3):384–412, 2001.

[68] Mirzad Mohandespour, Manimaran Govindarasu, and Zhengdao Wang. Rate, energy, and delay tradeoffs in wireless multicast: network coding vs. routing. *Mobile Computing, IEEE Transactions on (submitted)*, 2014.

[69] Mirzad Mohandespour and Ahmed E. Kamal. 1+ n protection in polynomial time: a heuristic approach. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5. IEEE, 2010.

[70] Mirzad Mohandespour and Ahmed E. Kamal. Multicast 1+1 protection: the case for simple network coding. In *ICNC 2015 (accepted)*. IEEE, 2015.

[71] Raheleh Niati, Amir H Banihashemi, and Thomas Kunz. Throughput and energy optimization in wireless networks: joint mac scheduling and network coding. *Vehicular Technology, IEEE Transactions on*, 61(3):1372–1382, 2012.

[72] T. Panayiotou, G. Ellinas, N. Antoniades, and A. Hadjiantonis. A novel segment-based protection algorithm for multicast sessions in transparent optical networks with mesh

topologies. In *Proc. IEEE/OSA Optical Fiber Communications Conference (OFC), Los Angeles, CA*, 2011.

[73] Morten V Pedersen, Janus Heide, and Frank HP Fitzek. Kodo: An open and research oriented network coding library. In *NETWORKING 2011 Workshops*, pages 145–152. Springer, 2011.

[74] R.C. Prim. Shortest connection networks and some generalizations. *Bell system technical journal*, 36(6):1389–1401, 1957.

[75] T. Rahman and G. Ellinas. Protection of multicast sessions in wdm mesh optical networks. In *Optical Fiber Communication Conference, 2005. Technical Digest. OFC/NFOEC*, volume 2, pages 3–pp. IEEE, 2005.

[76] ITU-T Recommendation. Overview of internet of things, June 2012.

[77] Hong Shen and Longkun Guo. Efficient 2-approximation algorithms for computing 2-connected steiner minimal networks. *Computers, IEEE Transactions on*, 61(7):954–968, 2012.

[78] N.K. Singhal, L.H. Sahasrabuddhe, and B. Mukherjee. Provisioning of survivable multicast sessions against single link failures in optical wdm mesh networks. *Journal of lightwave technology*, 21(11):2587, 2003.

[79] D. Stamatelakis and WD Grover. IP layer restoration and network planning based on virtual protection cycles. *IEEE Journal on selected areas in communications*, 18(10):1938–1949, 2000.

[80] John W Suurballe and Robert Endre Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2):325–336, 1984.

[81] H. Takahashi and A. Matsuyama. An approximate solution for the steiner problem in graphs. *Math. Japonica*, 24(6):573–577, 1980.

[82] Danail Traskov, Michael Heindlmaier, Muriel Médard, and Ralf Koetter. Scheduling for network-coded multicast. *IEEE/ACM Transactions on Networking (TON)*, 20(5):1479–1488, 2012.

[83] Danail Traskov, Michael Heindlmaier, Muriel Médard, Ralf Koetter, and Desmond S Lun. Scheduling for network coded multicast: A conflict graph formulation. In *GLOBECOM Workshops, 2008 IEEE*, pages 1–5. IEEE, 2008.

[84] Jens Vygen. Np-completeness of some edge-disjoint paths problems. *Discrete Appl. Math.*, 61(1):83–90, 1995.

[85] X. Wang, L. Guo, J. Cao, J. Wu, and W. Hou. Multicast protection scheme based on hamiltonian cycle in fault-tolerant optical mesh networks. *Optical Fiber Technology*, 16(5):292–298, 2010.

[86] Jörg Widmer, Christina Fragouli, and Jean-Yves Le Boudec. Low-complexity energy-efficient broadcasting in wireless ad-hoc networks using network coding. In *Proc. Workshop on Network Coding, Theory, and Applications*, 2005.

[87] Jeffrey E Wieselthier, Gam D Nguyen, and Anthony Ephremides. Energy-efficient broadcast and multicast trees in wireless networks. *Mobile Networks and Applications*, 7(6):481–492, 2002.

[88] Yunnan Wu, Philip A Chou, and Kamal Jain. A comparison of network coding and tree packing. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, page 143. IEEE, 2004.

[89] Yunnan Wu, Philip A Chou, and Sun-Yuan Kung. Minimum-energy multicast in mobile ad hoc networks using network coding. *Communications, IEEE Transactions on*, 53(11):1906–1918, 2005.

[90] Yunnan Wu, Philip A Chou, Qian Zhang, Kamal Jain, Wenwu Zhu, and Sun-Yuan Kung. Network planning in wireless ad hoc networks: a cross-layer approach. *Selected Areas in Communications, IEEE Journal on*, 23(1):136–150, 2005.

[91] Yang Xiao, Xiaojiang Du, Jingyuan Zhang, Fei Hu, and Sghaier Guizani. Internet protocol television (iptv): the killer application for the next-generation internet. *Communications Magazine, IEEE*, 45(11):126–134, 2007.

[92] F. Zhang, W.D. Zhong, and Y. Jin. Optimizations of *p*-cycle-based protection of optical multicast sessions. *Lightwave Technology, Journal of*, 26(19):3298–3306, 2008.

[93] W.D. Zhong and F. Zhang. An overview of p-cycle based optical multicast protection approaches in mesh wdm networks. *Optical Switching and Networking*, 8(4):259–274, 2011.

[94] Mateusz Zotkiewicz, Walid Ben-Ameur, and Michal Pioro. Finding failure-disjoint paths for path diversity protection in communication networks. *Communications Letters, IEEE*, 14(8):776–778, 2010.