

Distributed Control of Coded Networks

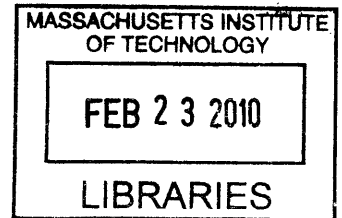
ARCHIVES

by

Fang Zhao

B.Eng., National University of Singapore (2000)

M.Eng., National University of Singapore (2001)



Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

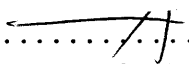
Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2010

© Massachusetts Institute of Technology 2010. All rights reserved.

Author 
Department of Electrical Engineering and Computer Science

November 12, 2009

Certified by 

Muriel Médard

Professor, Department of Electrical Engineering and Computer Science

Thesis Supervisor

Accepted by 

Terry P. Orlando

Chairman, Department Committee on Graduate Theses

Distributed Control of Coded Networks

by

Fang Zhao

Submitted to the Department of Electrical Engineering and Computer Science
on November 12, 2009, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

The introduction of network coding has the potential to revolutionize the way people operate networks. For the benefits of network coding to be realized, distributed solutions are needed for various network problems. In this work, we look at three aspects of distributed control of coded networks.

The first one is distributed algorithms for establishing minimum-cost multicast connections in coded networks. The subgraph optimization problem can be viewed as an linear optimization problem, and we look at algorithms that solve this problem for both static and dynamic multicasts. For static multicast, we present decentralized dual subgradient algorithms to find the min-cost subgraph. Due to the special structure of the network coding problem, we can recover a feasible primal solution after each iteration, and also derive theoretical bounds on the convergence rate in both the dual and the primal spaces. In addition, we propose heuristics to further improve our algorithm, and demonstrate through simulations that the distributed algorithm converges to the optimal subgraph quickly and is robust against network topology changes. For dynamic multicast, we introduce two types of rearrangements, link rearrangement and code rearrangement, to characterize disturbances to users. We present algorithms to solve the online network coding problem, and demonstrate through simulations that the algorithms can adapt to changing demands of the multicast group while minimizing disturbances to existing users.

The second part of our work focuses on analysis of COPE, a distributed opportunistic network coding system for wireless mesh networks. Experiments have shown that COPE can improve network throughput significantly, but current theoretical analysis fails to fully explain this performance. We argue that the key factor that shapes COPE's performance curve is the interaction between COPE and the MAC protocol. We also propose a simple modification to COPE that can further increase the network throughput.

Finally, we study network coding for content distribution in peer-to-peer networks. Such systems can improve the speed of downloads and the robustness of the systems. However, they are very vulnerable to Byzantine attacks, and we need to have a signature scheme that allows nodes to check the validity of a packet without decoding.

In this work, we propose such a signature scheme for network coding. Our scheme makes use of the linearity property of the packets in a coded system, and allows nodes to check the integrity of the packets received easily. We show that the proposed scheme is secure, and its overhead is negligible for large files.

Thesis Supervisor: Muriel Médard

Title: Professor, Department of Electrical Engineering and Computer Science

Acknowledgments

I am heartily thankful to my advisor, Professor Muriel Médard, for her invaluable guidance, support, encouragement, and care through the years. Besides teaching me how to be a researcher, she has been a constant source of inspiration. I cannot imagine having a better or friendlier advisor for my Ph.D study.

I thank my thesis committee members/collaborators, Professors Asuman Ozdaglar and Desmond Lun, for their guidance and insightful comments. It was a great pleasure working with them.

Many thanks also go to my friends for their friendship and help.

Last but not least, I owe my deepest gratitude to my family, for their unwavering love and support.

Fang Zhao
Cambridge, MA
November 2009

This thesis is based upon work supported by:

- BAE Systems National Security Solutions Inc., Defense Advanced Research Projects Agency (DARPA), and the Space and Naval Warfare System Center (SPAWARSYSCEN), San Diego, under subcontracts 6899963, 6917551, 060786, and 069145;
- National Science Foundation (NSF) under grants CCR-0325496, CNS-0627021, CNS-0627021;
- The Air Force Office of Scientific Research (AFOSR) under grant FA9550-06-1-0155.

Contents

1	Introduction	13
1.1	Distributed control of coded networks	14
1.2	Main contributions	16
1.3	Thesis outline	17
2	Minimum-cost Subgraph Algorithms for Static and Dynamic Multicasts with Network Coding	19
2.1	Background	19
2.1.1	Min-cost subgraph for static multicasts	20
2.1.2	Min-cost subgraph for dynamic multicasts	21
2.2	Problem formulation	23
2.2.1	Wireline network	23
2.2.2	Wireless network	25
2.3	Decentralized min-cost subgraph algorithm for static multicast	28
2.3.1	Subgradient method for decentralized subgraph optimization	29
2.3.2	Convergence rate analysis	33
2.3.3	Initialization and primal solution recovery	40
2.3.4	Simulation results	43
2.4	Min-cost subgraph algorithms for dynamic multicasts	48
2.4.1	Nonrearrangeable Algorithm	48
2.4.2	Rearrangeable algorithms	52
2.4.3	Simulation results	56

3	Analysis and Improvement to COPE	63
3.1	Background	63
3.1.1	The COPE system	63
3.1.2	Existing analysis on COPE	67
3.2	COPE performance analysis	68
3.3	Improvements on COPE	75
4	Signatures for Content Distribution with Network Coding	79
4.1	Background	79
4.1.1	Network coding in P2P networks	79
4.1.2	Byzantine detection scheme for network coded systems	82
4.2	Problem Setup	84
4.3	Signature scheme for network coding	85
4.4	Discussion	88
5	Conclusion and Future Work	93
5.1	Future work	95

List of Figures

2-1	(a) Example of an online step that causes link rearrangements to existing users; (b) Example of an online step that causes code rearrangements to existing users. The multicast rate from source node s to terminal nodes $\{t_1, t_2, t_3\}$ is 1. The thick lines indicate links used in the multicast, and the numbers against them indicate the rate of flow on them.	22
2-2	The “wireless multicast advantage” associated with omnidirectional antennas. The three destinations $j_1, j_2,$ and j_3 can all be reached at the same time with cost a_{ij_3} , and this is equivalent to having three unit capacity links from i to $j_1, j_2,$ and j_3 . Here, we also included a virtual unit-capacity link (i, i') to impose the constraint that information transmitted on the three links must be the same.	26
2-3	Average cost of random 4/8-terminal multicasts in 30/50-node wireless networks, using the decentralized subgraph optimization algorithms and centralized MIP algorithm. For modified primal recovery method, $N_a = 30$	45
2-4	Cost of a random 4-terminal multicast in a 30-node mobile wireless network, with $N_s = 50$, under various algorithms. For the modified primal recovery method, we used $N_a = 20$ and, for the look-back primal recovery method, we used $N_a = 50$	47
2-5	Extra energy required for multicasts in mobile wireless networks using decentralized subgraph optimization scheme in terms of percentage of the optimal value.	49

2-6	Nonrearrangeable algorithm.	51
2-7	MLR algorithm.	52
2-8	LMC algorithm.	54
2-9	α -scaled algorithm.	54
2-10	Extra cost of the multicast subgraph generated by the MLR algorithm in terms of percentage of C_{opt} for the Exodus network. We are showing both the individual data points for each trial and the average curve.	57
2-11	Extra cost of the multicast subgraph generated by the MLR algorithm for the EBONE network in terms of percentage of C_{opt}	58
2-12	Extra cost of the multicast subgraph generated by the α -scaled algo- rithm with $\alpha = 0.75$ and the MLR algorithm in terms of percentage of C_{opt} , on the Exodus network. We are also showing the individual data points for each trial for the α -scaled algorithm.	60
2-13	Extra cost of the multicast subgraph generated by the α -scaled algo- rithm with various α values, on the Exodus network.	61
3-1	Example of how COPE increases the throughput in the Alice-and-Bob wireless network.	64
3-2	Cross network.	65
3-3	COPE can provide a several-fold (3-4x) increase in the throughput of wireless ad hoc networks with UDP flows. This figure is taken from [29].	66
3-4	Throughput for COPE and non-COPE systems in an Alice-and-Bob network with cross traffic only.	71
3-5	Throughput for COPE and non-COPE systems in a cross network with cross traffic only.	73
3-6	Throughput for COPE and non-COPE systems in a cross network with cross traffic and traffic generated at the center node.	74
3-7	An example of queue status and packets sent in a cross network with COPE.	75

3-8	An example of queue status and packets sent in a cross network with modified COPE.	76
3-9	Throughput for COPE, modified COPE, and non-COPE systems in a cross network with cross traffic and traffic generated at the center node.	77
4-1	Content distribution with network coding. Assume the file being distributed is broken into three blocks, P_1 , P_2 , and P_3 . Any packet being transmitted is a random linear combination of all the blocks the sender has. For example, the packet sent from the source to peer A is a combination of P_1 , P_2 , and P_3 , whereas the packet sent from peer A to D is a combination of blocks A_1 and A_2 . A peer is able to decode the whole file when it receives 3 linearly independent blocks.	81

Chapter 1

Introduction

The concept of network coding, introduced by Ahlswede *et al.* in their pioneering work [4], is to allow and encourage mixing of data at intermediate network nodes. This is in contrast to conventional routing networks, where intermediate nodes only store and forward packets. A receiver in a coded network sees these mixed data packets and deduces from them the messages that were originally intended for the data sink. This shift in paradigm has a deep impact on a wide range of areas such as reliable delivery, resource sharing, efficient flow control, and security [16]. Due to this deep impact, network coding has generated a lot of research interest in recent years, and numerous subsequent papers, e.g., [38, 33, 27, 19, 44], have built upon this concept.

Much work in network coding has concentrated on a particular form of coding, *random linear network coding* (RLNC). RLNC was first introduced by Ho *et al.* in [19]. It is a coding method that lets nodes randomly pick their own coding coefficients from a given field without centralized coordination. It has been shown that as long as the field size is large enough, the probability that the receiver nodes can decode successfully is close to 1. This coding method is particularly attractive because it is completely distributed, and is robust to changes.

Network coding provides benefits along many diverse dimensions of communication networks, such as throughput, robustness, security, complexity, and wireless resources [16, 22]:

- **Throughput** – Ahlswede *et al.* showed in [4] that network coding can achieve the maximum multicast rate, which is not achievable by routing alone. In addition, Li *et al.* [38] and Koetter and Médard [33] showed that linear network codes suffice to achieve the capacity of a multicast connection in an error-free network. Beside increasing throughput for a multicast in a wireline network, network coding has also been shown to improve throughput in unicast connections [39] and in wireless networks [28].
- **Robustness** – Network coding provides robustness to both packet losses and link failures. In a lossy network, we can code packets across time, i.e., mix packets from the same flow together, to combat packet losses. This method can achieve the same sending rate as that when using link-by-link erasure codes. However, unlike the link-by-link erasure codes, the network coding approach does not require decoding at the intermediate nodes, thus avoid the delay problem. As for link failures, since network coding allows sharing of network resources among different flows, it can provide path protection and improve resource usage.
- **Security** – From a security point of view, network coding provides both benefits and drawbacks. Sending linear combinations of packets instead of the uncoded packets is a natural way to take advantage of multipath diversity for security against wiretapping attacks. However, network coding also introduces new security problems as it is particularly susceptible to Byzantine attacks. We will discuss this in more detail in Chapter 4.

1.1 Distributed control of coded networks

Distributed solutions are desirable for many network problems, as they are scalable, robust to network changes, and can take advantage of the distributed resources in the network, such as CPU, storage, etc.. For example, in conventional networks, routing is done in a distributed manner using algorithms such as the distributed

Bellman-Ford algorithm and the Dijkstra's shortest path algorithm [5]. Distributed flow management is available for congestion control. Distributed storage and peer-to-peer file sharing are also examples of distributed network solutions. When network coding is introduced, we need to develop new distributed methods or modify existing ones to adapt to the new paradigm.

In fact, distributed methods appear to have a natural place in network coding. The generation of network codes can be done in a distributed manner through the use of RLNC as mentioned previously. Another example is the problem of subgraph selection for multicasts. In routing networks, the subgraph selection problem is the Steiner tree problem, which is NP-complete, and hard to solve even in a centralized manner. However, with network coding, we can formulate the subgraph selection problem into a linear programming problem, and distributed methods are available to find the optimal subgraph [44].

To transfer the theoretical benefits of network coding into practical systems, we still need to develop distributed solutions to a variety of problems in coded networks. In this work, we focus on three aspects of distributed control of coded networks.

The first one is the minimum-cost subgraph selection for multicasts in coded networks. Similar to routing for unicast connections, when we need to establish a multicast connection, we have to first find the subgraph on which the multicast can be performed. As mentioned previously, this problem is NP-complete in a conventional routing network. However, in a coded network, distributed algorithms are available to find the minimum-cost subgraph. We study the algorithm proposed in [44], derive bounds on its convergence rates, and introduce methods to improve its convergence performance. In addition, we also study the dynamic multicast problem, where the members in the multicast group is not constant. We propose distributed algorithms to cater for the dynamic needs of the users, and at the same time, strive to keep the cost of the multicast low.

The second aspect we look at is distributed network coding for unicasts in wireless networks. Specifically, we study the COPE system proposed in [28, 29]. COPE is a packet-level network coding technique that exploits wireless broadcast to improve

throughput in congested networks. The nodes in COPE performs opportunistic coding, and decoding is done at the next hop. The coding and decoding are performed locally, and does not require any centralized control. Experiments demonstrate that COPE can significantly improve the network throughput with UDP traffic. We analyze the coding structures in COPE to understand the reasons behind these gains, and also, propose modifications to COPE that can further improve its throughput.

Finally, we study the usage of network coding in content distribution in peer-to-peer (P2P) networks. Network coding has been shown to improve the speed of content distribution, however, this system is very susceptible to Byzantine attacks. This is a major obstacle against practical implementation of network coded content distribution. We propose a distributed signature system that can efficiently detect the presence of corrupted data in received packets.

1.2 Main contributions

The main contributions of this work are summarized below:

- For static multicast, we present distributed subgradient algorithms to find the min-cost subgraph, and derive their convergence rate in both the primal and the dual domains. We also propose various heuristics for dual variable initialization and primal solution recovery to further improve the convergence rate, and use simulations to verify their performance. We also show through simulations that the algorithm is robust to changes in the network and can converge to new optimal solutions quickly as long as the rate of change in the network is slow as compared to the speed of computation and transmission.
- For dynamic multicasts, we propose both nonrearrangeable and rearrangeable algorithms for the subgraph selection problem, and use simulation results to show that one of our proposed algorithms, the α -scaled algorithm, can effectively bound the growth of the multicast cost without causing too many disturbances to existing users.

- We analyze the performance of COPE and argue that the main reason that shapes COPE’s performance curve is the interaction between COPE and the MAC protocol used in the wireless network. The local fairness imposed by the MAC protocol among competing nodes plays an important role here. In addition, we propose a simple modification to the COPE system that can further improve the network throughput.
- For network coded content distribution in P2P networks, we propose a new efficient, packet-based signature scheme, designed specifically for RLNC systems, to detect Byzantine attacks by checking the membership of a received packet in the valid vector space. This scheme allows an one-hop containment of the contamination, and its overhead is very small.

Various parts of the work in this thesis appear in various published and as yet unpublished papers [58, 59, 30, 60, 57, 61].

1.3 Thesis outline

In Chapter 2, we first introduce the background to the minimum-cost subgraph problem in Section 2.1. This problem is then formulated in Section 2.2. In Section 2.3, we present the decentralized algorithm for subgraph optimization, analyze its convergence rate, propose heuristics to improve the algorithm, and simulate its convergence performance. In Section 2.4, we propose algorithms to cater for the changing need of a dynamic multicast group, and demonstrate the effectiveness of our algorithms through simulations.

In Chapter 3, Section 3.1 provides the background on the COPE system and existing work on its analysis. We present our new analysis of the COPE performance in Section 3.2, and propose a modification to COPE to improve its throughput performance in Section 3.3.

In Chapter 4, we give the background on network coding in P2P networks and its security problems in Section 4.1. We present the network model used in Section 4.2,

and the proposed digital signature scheme in Section 4.3. Section 4.4 discusses the overhead of the signature scheme.

Finally, the thesis is concluded in Section 5.

Chapter 2

Minimum-cost Subgraph Algorithms for Static and Dynamic Multicasts with Network Coding

In this chapter, we study the subgraph optimization problem for both static and dynamic multicasts in coded networks.

2.1 Background

One of the main advantages of network coding over traditional routed networks is in the area of multicast, where common information is transmitted from a source node to a set of terminal nodes. When coding is used to perform multicast, the problem of establishing a minimum-cost multicast connection is equivalent to two effectively decoupled problems: one of determining the subgraph to code over, and the other of determining the code to use over that subgraph. The latter problem has been studied extensively in [19, 23, 20, 13], and a variety of methods have been proposed, which include employing simple random linear coding at every node. As mentioned in Chapter 1, such random linear coding schemes are completely decentralized, requiring no coordination between nodes, and can operate under dynamic conditions [21]. These papers, however, all assume the availability of dedicated network resources.

In this chapter, we focus on the former problem, which is to find the min-cost subgraph that allows the given multicast connection to be established (with appropriate coding) over coded packet networks. This problem has been studied in [54], [43]. The analogous problem for routed network is the Steiner tree problem, which is known to be NP-complete [7, 52]. When coding is allowed, the min-cost subgraph problem can be formulated as a linear programming (LP) problem, and in this chapter, we examine algorithms to solve it for both static and dynamic multicasts.

2.1.1 Min-cost subgraph for static multicasts

By static multicast, we refer to the case where a connection is setup for the user of a multicast group whose membership stays constant throughout the connection duration. The network topology, on the other hand, is not necessarily static. Lun *et al.* showed in [43] that the min-cost subgraph problem can be solved in a decentralized manner by using the dual subgradient method. In Section 2.3, we give an overview of this method, and study its convergence performance both theoretically and numerically.

There has been much work on using subgradient method to solve the Lagrangian dual of a convex constraint optimization problem. The convergence behavior of the subgradient method used on the dual problem is well understood under various step size rules. However, in practice, the main interest is in solving the primal problem, and recovering, from the dual iterations, feasible or near-feasible primal solutions that converge to the optimal solution. There are special cases where the primal solutions computed as a by-product of the dual iterations are feasible, such as in [41], but this is not the case in general. There are only a few papers studying the recovery of primal solutions from the dual iterations, for example [50], [36], and [31]. Recently, Nedić and Ozdaglar also looked at the convergence rate of the primal solutions in [46].

In Section 2.3, we present two slightly different formulations of the min-cost subgraph problem, both of which have the same optimal solutions. These two formulations give rise to two different distributed algorithms. One of them gives us a theoretical bound on the convergence rate of the primal solution, however, its in-

intermediate primal solutions are not always feasible. The second one, on the other hand, produces a feasible subgraph after each iteration, which allows us to start the multicast with minimum delay. We would like to point out that this is possible due to the special structure of the network coding problem, and it is not true in general for the dual subgradient method. More details on this are presented in Section 2.3.1.

We also introduce heuristics to improve the convergence performance of our algorithm, and through simulations, we show that the algorithm produces significant reduction in multicast energy as compared to the centralized routing algorithm just after a few iterations, and it converges to the optimal solution quickly.

One of the challenges of wireless networks, such as ad hoc networks and sensor networks, is variability of the network topology. Topology changes can be caused by mobility of users, sleeping or waking up of nodes, or the shadowing effect due to moving obstacles. Our algorithm is also put to test in a dynamic wireless network model, and we show that the subgradient method is robust to topology changes, and nodes are able to adjust their transmission power levels to move smoothly and quickly to a new optimal subgraph in a distributed manner.

2.1.2 Min-cost subgraph for dynamic multicasts

In applications such as real-time video distribution and teleconferencing, users can join or leave the multicast group at any time during the session. In such cases, we need to adjust the multicast subgraph to cater for the needs of this dynamic group. Lun *et. al.* gave a dynamic programming formulation of this problem in [42], which aims to deliver continuous service to the users. However, link and code rearrangement, which are defined later, can still occur under their formulation.

In the context of traditional routing networks, this problem corresponds to the the dynamic Steiner tree (DST) problem [25]. In DST, it is important to limit the number of rearrangements as a connection evolves, because rearranging a large multi-point connection may be time consuming and may require significant use of network resources in the form of CPU time. In addition, rearrangement of a connection may result in the blocking of some parts of the connection as rearrangement proceeds.

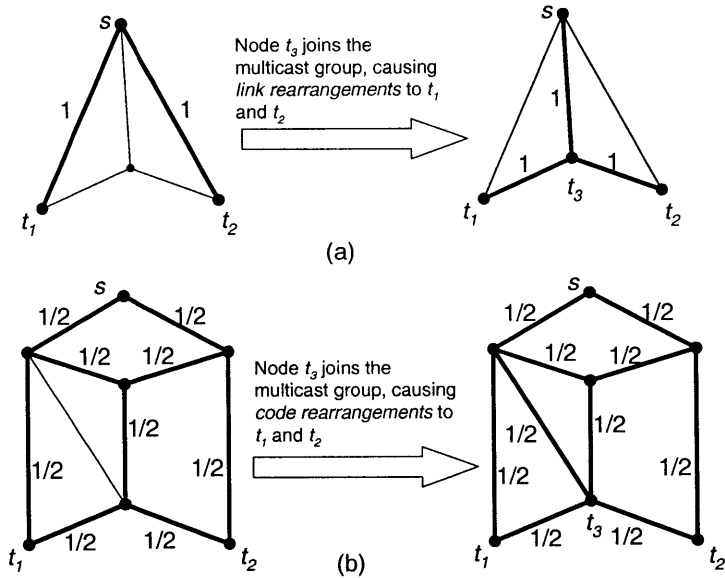


Figure 2-1: (a) Example of an online step that causes link rearrangements to existing users; (b) Example of an online step that causes code rearrangements to existing users. The multicast rate from source node s to terminal nodes $\{t_1, t_2, t_3\}$ is 1. The thick lines indicate links used in the multicast, and the numbers against them indicate the rate of flow on them.

Therefore, the DST problem comes in two flavors [25, 48]. One is the nonrearrangeable version, in which rearrangement of existing routes is not allowed. In the other version, rearrangement is allowed, but the cost of rearrangements is taken into consideration.

The situation is similar in networks with coding. When the membership of the multicast group changes, we want to minimize the disturbance to existing users in the group by limiting both *link rearrangements* and *code rearrangements*. A *link rearrangement* occurs when some links in the current multicast subgraph is removed causing alternate paths to be used to serve existing users (see Fig. 2-1(a) for an example). Like in the routing networks, owing to the change in the physical connection, this kind of rearrangement causes disruptions to the continuous service to the multicast group. The second kind of rearrangement, which we call *code rearrangement*, is more subtle. Code rearrangement occurs when new incoming links are added to existing nodes in the multicast subgraph. Fig. 2-1(b) shows an example

for code rearrangements. Since random coding is used by the intermediate nodes in the subgraph to perform network coding, when a node has an additional incoming link, it will have to generate a new set of random parameters to mix the incoming streams. All receivers downstream, therefore, have to use these new parameters and recompute the inversion matrix to decode the data streams. This scenario does not involve any physical switching of paths for the existing terminals, but it still causes a minor disruption to the continuous service due to this reprocessing of network coding parameters. Note that the disruptions caused by code rearrangements are generally smaller than that caused by link rearrangements.

In Section 2.4 we present algorithms that adapt to the changing demand of the multicast group, and at the same time, minimize disturbances to existing users. We also compare their performances through simulation.

2.2 Problem formulation

In this section, we present the LP formulation of the min-cost subgraph problem in both wireline and wireless networks. We also derive the Lagrangian dual of these LP problems, which will be used in the distributed algorithms presented in Section 2.3.

2.2.1 Wireline network

We look at the problem of single multicast in wireline networks, and model the network with a directed graph $G = (N, A)$, where N is the set of nodes and A is the set of links in the network. Each link $(i, j) \in A$ is associated with a non-negative number a_{ij} , which is the cost per unit flow on this link. We assume that the total cost of using a link is proportional to the flow, z_{ij} , on it. For the multicast, suppose we have a source node $s \in N$ producing data at a positive rate R that it wishes to transmit to a non-empty set of terminal nodes T in N .

It is shown in [4] that a subgraph z is capable of supporting a multicast connection of rate R from source s to T if and only if the min-cut from s to any $t \in T$ is greater than or equal to R . Hence, the problem of finding the min-cost subgraph can be

formulated into the following LP problem [44]:

$$\begin{aligned}
& \text{minimize} && f(z) = \sum_{(i,j) \in A} a_{ij} z_{ij} \\
& \text{subject to} && z_{ij} \geq x_{ij}^{(t)}, && \forall (i,j) \in A, t \in T, \\
& && \sum_{\{j|(i,j) \in A\}} x_{ij}^{(t)} - \sum_{\{j|(j,i) \in A\}} x_{ji}^{(t)} = \delta_i^{(t)}, && \forall i \in N, t \in T, \\
& && x_{ij}^{(t)} \geq 0, && \forall (i,j) \in A, t \in T,
\end{aligned} \tag{2.1}$$

where $x_{ij}^{(t)}$ corresponds to the virtual flow on link (i,j) for terminal t , z_{ij} is the actual flow on link (i,j) in the multicast subgraph, and

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

Although the decision variables, z_{ij} , are unbounded in the above formulation, it is easy to see that any optimal solution of (2.1), z^* , is bounded, *i.e.*,

$$0 \leq z_{ij}^* \leq b_{ij}, \quad \forall (i,j) \in A, \tag{2.2}$$

for any $b_{ij} > R$. Thus, including the additional constraint (2.2) in (2.1) would not change the optimal solution set. However, it will affect its Lagrangian dual, and consequently, the algorithm for solving this problem. We will see later in Section 2.3.2 that this additional constraint can help us derive a theoretical bound on the convergence rate of our distributed algorithm.

The Lagrangian dual problem for (2.1) is given by:

$$\begin{aligned}
& \text{maximize} && q(p) = \sum_{t \in T} q^{(t)}(p^{(t)}) \\
& \text{subject to} && \sum_{t \in T} p_{ij}^{(t)} = a_{ij}, && \forall (i,j) \in A, \\
& && p_{ij}^{(t)} \geq 0, && \forall (i,j) \in A, t \in T,
\end{aligned} \tag{2.3}$$

where

$$q^{(t)}(p^{(t)}) = \min_{x^{(t)} \in F_x^{(t)}} \sum_{(i,j) \in A} p_{ij}^{(t)} x_{ij}^{(t)}, \quad \forall t \in T, \quad (2.4)$$

and $F_x^{(t)}$ is the bounded polyhedron of points $x^{(t)}$ satisfying the conservation of flow constraints

$$\begin{aligned} \sum_{\{j|(i,j) \in A\}} x_{ij}^{(t)} - \sum_{\{j|(j,i) \in A\}} x_{ji}^{(t)} &= \delta_i^{(t)}, \quad \forall i \in N, \\ x_{ij}^{(t)} &\geq 0, \quad \forall (i,j) \in A. \end{aligned}$$

Note that subproblem (2.4) is a standard shortest path problem with link costs $p_{ij}^{(t)}$, which can be solved using a multitude of distributed algorithms (e.g., distributed Bellman-Ford).

When constraint (2.2) is included in the primal problem, the dual problem becomes

$$\begin{aligned} \text{maximize} \quad q(p) &= \sum_{t \in T} q^{(t)}(p^{(t)}) + \sum_{(i,j) \in A} r_{ij}(p_{ij}) \\ \text{subject to} \quad p_{ij}^{(t)} &\geq 0, \quad \forall (i,j) \in A, t \in T, \end{aligned} \quad (2.5)$$

where

$$r_{ij}(p_{ij}) = \min_{z \in F_z} (a_{ij} - \sum_{t \in T} p_{ij}^{(t)}) z_{ij}, \quad \forall (i,j) \in A,$$

and F_z is the bounded region of z given by

$$0 \leq z_{ij} \leq b_{ij}, \quad \forall (i,j) \in A.$$

2.2.2 Wireless network

Under this model, we consider wireless networks where nodes are placed randomly within a 10×10 square with a radius of connectivity r . The energy required to transmit at unit rate to a distance d is taken to be d^2 . Let f_{ij} be the cost function of link (i,j) , and in our model, $f_{ij}(z_{ij}) = a_{ij} z_{ij}$ where $a_{ij} = d_{ij}^2$ is the energy required to send at unit rate over this link and z_{ij} is the rate of flow on this link. We justify this assumption on the basis that we take energy as the most significant constraint, so there are, for example, sufficient time or frequency slots to guarantee that no two

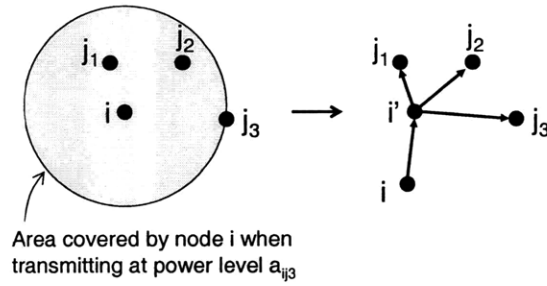


Figure 2-2: The “wireless multicast advantage” associated with omnidirectional antennas. The three destinations j_1 , j_2 , and j_3 can all be reached at the same time with cost a_{ij_3} , and this is equivalent to having three unit capacity links from i to j_1 , j_2 , and j_3 . Here, we also included a virtual unit-capacity link (i, i') to impose the constraint that information transmitted on the three links must be the same.

transmissions ever interfere. This model is discussed in more depth in [44].

We consider wireless networks in which antennas are omnidirectional. When we transmit from node i to node j , we get transmission to all nodes whose distance from i is less than that from i to j “for free” — a phenomenon referred to as the “wireless multicast advantage” in [53]. If we impose an ordering \preceq on the set of outgoing links from i , such that $(i, k) \preceq (i, j)$ if and only if $a_{ik} \leq a_{ij}$, we can then assume that we obtain a lossless broadcast link of unit rate from node i to all nodes k such that $(i, k) \preceq (i, j)$ for cost a_{ij} . Consider the example shown in Fig. 2-2, where there are three nodes within distance r from node i . If node i transmits with power a_{ij_3} to node j_3 , the two nearer nodes, j_1 and j_2 also receive this information without additional cost. Thus, the situation here is quite different from the wireline case. Instead of picking links to transmit on in the wireline networks, the nodes in wireless networks pick power levels to transmit with, and this in turn determines their radius of coverage.

Similar to the wireline case, in a wireless network, the min-cost subgraph that can be used to perform multicast with network coding is given by the following linear

optimization problem:

$$\begin{aligned}
& \text{minimize} && f(z) = \sum_{(i,j) \in A} a_{ij} z_{ij} \\
& \text{subject to} && \sum_{\{k | (i,k) \in A, (i,k) \succeq (i,j)\}} (z_{ik} - x_{ik}^{(t)}) \geq 0, \quad \forall (i,j) \in A', t \in T, \\
& && \sum_{\{j | (i,j) \in A\}} x_{ij}^{(t)} - \sum_{\{j | (j,i) \in A\}} x_{ji}^{(t)} = \sigma_i^{(t)}, \quad \forall i \in N, t \in T, \\
& && x_{ij}^{(t)} \geq 0, \quad \forall (i,j) \in A, t \in T,
\end{aligned} \tag{2.6}$$

Here, A' is a subset of A with the property that the constraint

$$\sum_{\{k | (i,k) \in A, (i,k) \succeq (i,j)\}} (z_{ik} - x_{ik}^{(t)}) \geq 0$$

is unique for all $(i,j) \in A'$.

The subgraph optimization scheme also uses the Lagrangian dual of (2.6) given below.

$$\begin{aligned}
& \text{maximize} && q(p) = \sum_{t \in T} q^{(t)}(p^{(t)}) \\
& \text{subject to} && \sum_{\{k | (i,k) \in A', (i,k) \preceq (i,j)\}} \sum_{t \in T} p_{ik}^{(t)} = a_{ij}, \quad \forall (i,j) \in A, \\
& && p_{ij}^{(t)} \geq 0, \quad \forall (i,j) \in A', t \in T,
\end{aligned} \tag{2.7}$$

where

$$q^{(t)}(p^{(t)}) = \min_{x^{(t)} \in F_x^{(t)}} \sum_{(i,j) \in A} \left(\sum_{\{k | (i,k) \in A', (i,k) \preceq (i,j)\}} p_{ik}^{(t)} \right) x_{ij}^{(t)}, \tag{2.8}$$

and $F^{(t)}$ is the bounded polyhedron of points $x^{(t)}$ satisfying the conservation of flow constraints.

To simplify the constraints in the dual problem (2.7), we can sort the outgoing links from node i in A' according to their costs. Note that in A' , no two outgoing links from a node i are of the same cost. Consider the example in Fig. 2-2 where there are three outgoing links from i with $(i, j_1) \prec (i, j_2) \prec (i, j_3)$, the equality constraints

in (2.7) with respect to these links become

$$\begin{aligned}\sum_{t \in T} p_{ij_1}^{(t)} &= a_{ij_1}, \\ \sum_{t \in T} p_{ij_1}^{(t)} + \sum_{t \in T} p_{ij_2}^{(t)} &= a_{ij_2}, \\ \sum_{t \in T} p_{ij_1}^{(t)} + \sum_{t \in T} p_{ij_2}^{(t)} + \sum_{t \in T} p_{ij_3}^{(t)} &= a_{ij_3}.\end{aligned}$$

These are equivalent to

$$\begin{aligned}\sum_{t \in T} p_{ij_1}^{(t)} &= a_{ij_1}, \\ \sum_{t \in T} p_{ij_2}^{(t)} &= a_{ij_2} - a_{ij_1}, \\ \sum_{t \in T} p_{ij_3}^{(t)} &= a_{ij_3} - a_{ij_2}.\end{aligned}$$

Therefore, if we define

$$s_{ij} = a_{ij} - \max_{\{k | (i,k) \in A', (i,k) \prec (i,j)\}} a_{ik}, \quad (2.9)$$

the dual problem (2.7) can be simplified to

$$\begin{aligned}\text{maximize} \quad & \sum_{t \in T} q^{(t)}(p^{(t)}) \\ \text{subject to} \quad & \sum_{t \in T} p_{ij}^{(t)} = s_{ij}, \quad \forall (i,j) \in A, \\ & p_{ij}^{(t)} \geq 0, \quad \forall (i,j) \in A', t \in T.\end{aligned} \quad (2.10)$$

2.3 Decentralized min-cost subgraph algorithm for static multicast

We focus on static multicasts in this section. Section 3.1 gives an overview of the dual subgradient method for decentralized subgraph optimization. The convergence rate of this method is analyzed in Section 2.3.2. Various heuristics to improve the convergence performance of the canonical algorithm in both static and dynamic wireless networks are presented in Section 2.3.3, and Section 2.3.4 gives some numerical

results.

2.3.1 Subgradient method for decentralized subgraph optimization

The subgraph optimization scheme in [44] tries to converge to the optimal primal solution by using subgradient method on the dual problem. This algorithm is completely decentralized and each node only has to know the cost of its incoming and outgoing links, and exchange information with neighboring nodes. We first give an overview of the algorithm under the wireline network model in Section 2.3.1. Section 2.3.1 describes the extension of this algorithm to the wireless case.

Subgradient method in wireline networks

In the following, we describe the distributed algorithms for solving (2.1) with and without constraint (2.2). We refer to the algorithm that solves problem (2.1) and its dual (2.3) as *Algorithm A*, and the algorithm for solving the primal with constraint (2.2) and dual (2.5) as *Algorithm B*. Most of the discussions and simulations in Section 2.3 are based on Algorithm A, since it has better convergence performance in practical settings. However, in Section 2.3.2, we use Algorithm B to derive a theoretical bound on the convergence rate of the primal solutions, which is not available for Algorithm A.

Algorithm A

1. **Initialize $p[0]$** — Before the first iteration, each node initializes $p[0]$.
2. **Compute $x[n]$** — In the n th iteration, use $p[n]$ as link costs, and run a distributed shortest path algorithm to determine $x[n]$.
3. **Update $p[n + 1]$** — Update $p[n + 1]$ using subgradient obtained through $x[n]$ values.

$$g_{ij}^{(t)}[n] = x_{ij}^{(t)}[n],$$

$$p[n + 1] := [p[n] + \theta[n]g^{(t)}[n]]_P^+,$$

where $g[n]$ is the subgradient for $p[n]$, $\theta[n]$ is the step size for the n th iteration, and $[\cdot]_P^+$ denotes the projection onto the constraint set P in (2.3). This projection can be done in a distributed manner, and specifically, $p_{ij}^{(t)}[n+1]$ is given by

$$p_{ij}^{(t)}[n+1] = \max \left(0, p_{ij}^{(t)}[n] + \theta[n]x_{ij}^{(t)}[n] + d_{ij}[n] \right), \quad (2.11)$$

where $d_{ij}[n] \leq 0$ is a number computed based on the $p[n]$, $x[n]$, and $\theta[n]$ values [44].

4. **Recover $\tilde{x}[n]$** — At the end of each iteration, nodes recover a primal solution, $\tilde{x}[n]$, based on the dual computations. Let $\{\mu_l[n]\}_{l=1,\dots,n}$ be a sequence of convex combination weights for each non-negative integer n , i.e., $\sum_{l=1}^n \mu_l[n] = 1$ and $\mu_l[n] \geq 0$ for all $l = 1, \dots, n$. Further, let us define

$$\gamma_{ln} = \frac{\mu_l[n]}{\theta[l]}, \quad l = 1, \dots, n, n = 0, 1, \dots,$$

and

$$\Delta\gamma_n^{\max} = \max_{l=2,\dots,n} \{\gamma_{ln} - \gamma_{(l-1)n}\}.$$

According to [50], if the step sizes $\{\theta[n]\}$ and the convex combination weights $\{\mu_l[n]\}$ are chosen such that

- (a) $\gamma_{ln} \geq \gamma_{(l-1)n}$ for all $l = 2, \dots, n$, and $n = 0, 1, \dots$,
- (b) $\Delta\gamma_n^{\max} \rightarrow 0$ as $n \rightarrow \infty$, and
- (c) $\gamma_{1n} \rightarrow 0$ as $n \rightarrow \infty$ and $\gamma_{nn} \leq \delta$ for all $n = 0, 1, \dots$, for some $\delta > 0$,

then we obtain an optimal solution to the primal problem (2.1) from any accumulation point of the sequence of primal iterates $\{\tilde{x}[n]\}$ given by

$$\tilde{x}_{ij}^{(t)}[n] = \sum_{l=1}^n \mu_l[n]x_{ij}^{(t)}[l], \quad n = 0, 1, \dots$$

An example of a set of parameters that satisfy the above conditions are $\theta[n] = n^{-\alpha}$ for $n = 0, 1, 2, \dots$ where $0 < \alpha < 1$, and $\mu_l[n] = 1/n$ for $n = 1, 2, 3, \dots$ and

$l = 1, \dots, n.$

5. **Determine** $\tilde{z}[n]$ — Each node computes the $\tilde{z}_{ij}[n]$ values from the $\tilde{x}_{ij}^{(t)}[n]$ values. In order to minimize the cost, $\tilde{z}_{ij}[n] = \max_{t \in T} \tilde{x}_{ij}^{(t)}[n]$.
6. **Repeat** — Steps 2 to 5 are repeated until the primal solution has converged.

For details of this algorithm and related proofs, please refer to [44].

Since the intermediate $\{\tilde{z}[n], \tilde{x}[n]\}$ values after each iteration are always feasible solutions to the primal problem, we do not have to wait till the primal solution converges to start the multicast. Instead, the multicast can be started after the first iteration, and we can shift the flows gradually through the iterations to operate on a more cost effective subgraph. Note that in general, this is not true for dual subgradient methods, and it works out here due to the unique structure of the network coding problem. Specifically, the flow variables, z_{ij} , are not involved in the flow conservation constraints, and they do not appear in the dual iterations. This allows us to pick feasible z values after each dual iteration based on a set of feasible virtual flows x .

If the boundedness constraint (2.2) is included in the primal problem, we have Algorithm B for solving this new problem and its dual (2.5).

Algorithm B

1. **Initialize** $p[0]$.
2. **Compute** $x[n]$ and $z[n]$ — Computation of $x[n]$ is the same as that in Algorithm A. For $z[n]$, we have

$$z_{ij}[n] = \begin{cases} 0, & \text{if } \sum_{t \in T} p_{ij}^{(t)} \leq a_{ij}, \\ b_{ij}, & \text{if } \sum_{t \in T} p_{ij}^{(t)} \geq a_{ij}. \end{cases}$$

3. **Update** $p[n + 1]$ — Update $p[n + 1]$ using subgradient obtained through $x[n]$ and $z[n]$ values.

$$g_{ij}^{(t)}[n] = x_{ij}^{(t)}[n] - z_{ij}[n],$$

$$p_{ij}^{(t)}[n+1] = \max\left(0, p[n] + \theta[n]g_{ij}^{(t)}[n]\right).$$

4. **Recover $\tilde{z}[n]$** — Recovery of the primal solution $\tilde{z}[n]$ is done by taking a convex combination of all past $z[n]$ values, similar to the recovery of $\tilde{x}[n]$ in Algorithm A.

$$\tilde{z}_{ij}^{(t)}[n] = \sum_{l=1}^n \mu_l[n] z_{ij}^{(t)}[l], \quad n = 0, 1, \dots$$

5. **Repeat** — Steps 2 to 4 are repeated until the primal solution has converged.

As we will see in Section 2.3.2, Algorithm B gives us a nice theoretical bound on the primal convergence rate of the min-cost subgraph problem, which is not available for Algorithm A. However, a major drawback of Algorithm B as compared to Algorithm A is that the $\tilde{z}[n]$ values are not always feasible, therefore, we cannot start the multicast right away as in the case of Algorithm A. This is very undesirable in practice, and is one of the main reasons why we only focus on Algorithm A in our simulations.

Subgradient method in wireless networks

The main steps in the distributed min-cost subgraph algorithm for wireless networks are the same as that in Algorithm A of the wireline case, except for steps 3 and 5, in which some modifications are required. The details of the changes are highlighted below.

In step 3, when updating $p[n+1]$, the subgradient for $p_{ij}^{(t)}[n]$ in the wireless case is given by

$$g_{ij}^{(t)}[n] = \sum_{\{k|(i,k) \in A, (i,k) \succeq (i,j)\}} x_{ik}^{(t)}[n],$$

and again, $p_{ij}[n+1]$ is the Euclidean projection of $p_{ij}[n] + \theta[n]g_{ij}[n]$ onto the feasible set P_{ij} .

In step 5, we compute $\tilde{z}[n]$ based on the recovered primal solution $\tilde{x}[n]$. Recall that in the primal problem (2.6) we have the constraints

$$\sum_{\{k|(i,k) \in A, (i,k) \succeq (i,j)\}} (\tilde{z}_{ik} - x_{ik}^{(t)}) \geq 0, \quad \forall (i,j) \in A', t \in T. \quad (2.12)$$

Assume that the sorted list of outgoing links from node i in A' based on their costs is $\{(i, j_1), \dots, (i, j_k)\}$, and start from the most expensive links (i, j_k) , the above constraint becomes

$$\tilde{z}_{ij_k} - \tilde{x}_{ij_k}^{(t)} \geq 0, \forall t \in T.$$

To minimize total cost, the optimal \tilde{z}_{ij_k} value should be $\max_{t \in T} \tilde{x}_{ij_k}^{(t)}$. In cases where more than one outgoing links are of the same cost, we just need to make sure that the sum of the flows on these links satisfy constraint (2.12). The distribution of the total flow among these links can be done randomly without affecting the total cost. Once \tilde{z}_{ij_k} value is determined, we can move on to the second most expensive link (i, j_{k-1}) , whose constraint now becomes

$$(\tilde{z}_{ij_{k-1}} - \tilde{x}_{ij_{k-1}}^{(t)}) + (\tilde{z}_{ij_k} - \tilde{x}_{ij_k}^{(t)}) \geq 0, \forall t \in T,$$

and we have $\tilde{z}_{ij_{k-1}} = \max_{t \in T} (\tilde{x}_{ij_{k-1}}^{(t)} + \tilde{x}_{ij_k}^{(t)}) - z_{ij_k}$. By repeating the above process, we can obtain the optimal primal solution \tilde{z} from \tilde{x} .

2.3.2 Convergence rate analysis

In this section, we study the convergence rates of our dual subgradient method presented in Section 2.3.1 in both the primal and the dual spaces. For clarity of presentation, we use the wireline model in this section, as its notations are much simpler than the wireless one. All results here can be easily extended to the wireless case.

Convergence rate for the dual problem

The analysis and results in this subsection apply to both Algorithm A and Algorithm B. Here, we will just present the analysis for Algorithm A, as the extension to Algorithm B is fairly straight-forward.

With properly chosen stepsizes, the standard subgradient method proposed in the Section 2.3.1 converges to dual optimal solutions eventually [6], but it is hard to analyze the convergence rate of the standard method. To this end, we consider

the incremental subgradient method studied in [45]. The incremental subgradient method can be used here because the objective function in (2.3) is the sum of $|T|$ convex component functions, and the constraint set is non-empty, closed and convex (see Chapter 2 of [45]). At each iteration, p is changed incrementally through a sequence of $|T|$ steps. Each step is a subgradient iteration for a single component function $q^{(t)}$. Thus, an iteration can be viewed as a cycle of $|T|$ subiterations. Denote the terminal nodes by $\{1, 2, \dots, N_T\}$, where $N_T = |T|$. The vector $p[n+1]$ is obtained from $p[n]$ as follows.

$$\begin{aligned}\psi_0[n] &:= p[n], \\ \psi_i[n] &:= [\psi_{i-1}[n] + \theta[n]g^{(i)}[n]]_P^+, \\ p[n+1] &:= \psi_{N_T}[n].\end{aligned}$$

We first prove two propositions that are useful for the convergence rate analysis.

Proposition 1. *Problem (2.3) satisfies the subgradient boundedness property, which means there exists a positive scalar C such that*

$$\begin{aligned}\|g\| &\leq C, \quad \forall g \in \partial q^{(t)}(p[n]) \cup \partial q^{(t)}(\psi_{i-1,n}), \\ &\forall i = 1, \dots, N_T, \quad \forall n.\end{aligned}$$

Proof. This is true because $q^{(t)}$ is the pointwise minimum of a finite number of affine functions, and in this case, for every p , the set of subgradients $\partial q^{(t)}(p)$ is the convex hull of a finite number of vectors. Thus, the subgradients are bounded. \square

Proposition 2. *Let the optimal solution set be P^* , there exists a positive scalar μ such that*

$$q^* - q(p) \geq \mu(\text{dist}(p, P^*))^2, \quad \forall p \in P.$$

Proof. Problem (2.3) can be reformulated into a linear programming problem as fol-

lows.

$$\begin{aligned}
& \text{maximize} && q'(v) = \sum_{t \in T} \sum_{i \in N} r_i^{(t)} \delta_i^{(t)} = R \sum_{t \in T} (r_s^{(t)} - r_t^{(t)}) && (2.13) \\
& \text{subject to} && r_i^{(t)} - r_j^{(t)} \leq p_{ij}^{(t)}, \quad \forall (i, j) \in A, t \in T, \\
& && \sum_{t \in T} p_{ij}^{(t)} = a_{ij}, \quad \forall (i, j) \in A, \\
& && p_{ij}^{(t)} \geq 0, \quad \forall (i, j) \in A, t \in T.
\end{aligned}$$

The decision vector, v , is a concatenation of vectors p and r , and we denote the feasible set by V . For any feasible $p \in P$ from (2.3), there is a corresponding v in (2.13) with the same p -component and $q'(v) = q(p)$. Furthermore, for any feasible $v \in V$, we can extract a p vector from it that gives the same total cost in (2.3). Therefore, the two formulations (2.3) and (2.13) have the same optimal values, i.e., $q^* = q'^*$.

Since the set of solutions for a linear programming problem is a set of weak sharp minima [9], there exists a positive α such that

$$q'^* - q'(v) \geq \alpha(\text{dist}(v, V^*)), \quad \forall v \in V.$$

So for any $p \in P$ in (2.3), we have

$$q^* - q(p) = q'^* - q'(v) \geq \alpha(\text{dist}(v, V^*)) \geq \alpha(\text{dist}(p, P^*)).$$

The last inequality comes from the fact that p/P^* is the projection of v/V^* on P , and the projection operation is non-expansive. Since P is a bounded polyhedron, the distance between any two points in P is bounded, i.e., $\text{dist}(p, p') \leq B$ for all $p, p' \in P$ for some positive B . Therefore,

$$q^* - q(p) \geq \frac{\alpha}{B}(\text{dist}(p, P^*))^2.$$

Let $\mu = \alpha/B$, and the proposition is proved. □

With these propositions, we have the following result for constant step size.

Proposition 3. *For the sequence $\{p[n]\}$ generated by the incremental subgradient method with the step size $\theta[n]$ fixed to some positive constant θ , where $\theta \leq \frac{1}{2\mu}$, we have*

$$(\text{dist}(p[n+1], P^*))^2 \leq (1 - 2\theta\mu)^{n+1}(\text{dist}(p[0], P^*))^2 + \frac{\theta|T|^2C^2}{2\mu}, \quad \forall n. \quad (2.14)$$

Proof. The proof for this proposition follows from Proposition 1.2 and the proof of Proposition 2.3 in [45]. Since the dual problem satisfies Proposition 1 (bounded subgradient), from Lemma 2.1 in [45], we have

$$\|p[n+1] - r\|^2 \leq \|p[n] - r\|^2 - 2\theta(q(r) - q(p[n])) + \theta^2|T|^2C^2, \quad \forall r \in P, \forall n.$$

Using this relation with $r = p^*$ for any optimal $p^* \in P^*$, we see that

$$\|p[n+1] - p^*\|^2 \leq \|p[n] - p^*\|^2 - 2\theta(q^* - q(p[n])) + \theta^2|T|^2C^2, \quad \forall r \in P, \forall n, \quad (2.15)$$

and by taking the minimum over all $p^* \in P^*$, we have

$$\begin{aligned} (\text{dist}(p[n+1], P^*))^2 &\leq (\text{dist}(p[n], P^*))^2 - 2\theta(q^* - q(p[n])) + \theta^2|T|^2C^2 \\ &\leq (1 - 2\theta\mu)(\text{dist}(p[n], P^*))^2 + \theta^2|T|^2C^2, \quad \forall n, \end{aligned} \quad (2.16)$$

where the last inequality comes from Proposition 2. From this relation, by induction, we can see that

$$(\text{dist}(p[n+1], P^*))^2 \leq (1 - 2\theta\mu)^{(n+1)}(\text{dist}(p[0], P^*))^2 + \theta^2|T|^2C^2 \sum_{i=0}^n (1 - 2\theta\mu)^i, \quad \forall n,$$

which combined with

$$\sum_{i=0}^n (1 - 2\theta\mu)^i \leq \frac{1}{2\theta\mu},$$

yields the desired relation (2.14). \square

In summary, we have shown that the convergence rate for the incremental subgradient method on (2.3) is linear for a sufficiently small stepsize. However, only

convergence to a neighborhood of the optimal solution set can be guaranteed, which is typical for constant step size rules. Moreover, our result also highlights the trade-off between the error and the convergence rate constant. The smaller the θ value, the smaller the size of the neighborhood, but on the other hand, we get slower convergence.

Convergence analysis for the primal problem

As mentioned in Section 2.3.1, it is advantageous to use Algorithm A in practice, as its primal solution is always feasible through the iterations. Unfortunately, due to the unboundedness of z_{ij} in formulation (2.1), it is very hard to derive its primal convergence rate. Therefore, in this section, we turn our focus to Algorithm B, for which we derive a bound on its convergence rate.

We first prove that our primal problem (2.1) with constraint (2.2) satisfies the Slater condition in Proposition 4, then present the main convergence rate result in Proposition 5.

Proposition 4. The Slater condition *There exists a vector $\{\bar{z}, \bar{x}\} \in F$ such that*

$$\bar{z}_{ij} > \bar{x}_{ij}^{(t)} \quad \forall (i, j) \in A, t \in T., \quad (2.17)$$

where $F = \{F_z, F_x\}$ is the feasible set for the boundedness constraints for z and the conservation of flow constraints for x .

Proof. For the virtual flows, $x_{ij}^{(t)}$, based on the conservation of flow constraints, there exists feasible solutions where $x_{ij}^{(t)} \leq R$ for all $(i, j) \in A$ and $t \in T$. Since the upper bound on z_{ij} is $b_{ij} > R$, we can always find a set of z that is strictly greater than the corresponding x . Therefore, our primal problem satisfies the Slater condition. \square

Proposition 5. *Let $\{\bar{z}, \bar{x}\}$ be a Slater vector satisfying (2.17), and C be the subgradient norm bound in Proposition 1, define*

$$B^* = \frac{2}{\gamma}(f(\bar{z}) - q^*) + \max \left\{ \|p[0]\|, \frac{1}{\gamma}(f(\bar{z}) - q^*) + \frac{\theta C^2}{2\gamma} + \theta C \right\},$$

where $\gamma = \min_{\{i,j\} \in A, t \in T} (\bar{z}_{ij}^{(t)} - \bar{x}_{ij}^{(t)})$. If constant stepsize θ is used in the dual iterations, and simple averaging is used in the primal recovery, i.e., $\mu_l[n] = 1/n$ for $l = 1, 2, \dots, n$, then the primal cost after the n th iteration is bounded by

$$f^* - \frac{1}{\gamma}[f(\bar{z}) - q^*] \frac{B^*}{n\theta} \leq f(\tilde{z}[n]) \leq f^* + \frac{\|p[0]\|^2}{2n\theta} + \frac{\theta C^2}{2}. \quad (2.18)$$

Proof. We first derive the lower bound on $f(\tilde{z}[n])$ (the left side of (2.18)). Recall that in Algorithm B, $g_{ij}^{(t)}[n] = g(z_{ij}[n], x_{ij}^{(t)}[n]) = x_{ij}^{(t)}[n] - z_{ij}[n]$, and

$$p[n+1] = \max(0, p[n] + \theta g[n]) \geq p[n] + \theta g[n],$$

we have

$$\theta g(z[n], x[n]) \leq p[n+1] - p[n] \quad \forall n \geq 0.$$

Therefore, $\sum_{i=0}^{n-1} \theta g(z[i], x[i]) \leq p[n] - p[0] \leq p[n]$, where the last inequality follows from $p[0] \geq 0$. By the convexity of the function g , it follows that

$$g(\tilde{z}[n], \tilde{x}[n]) \leq \frac{1}{n} \sum_{i=0}^{n-1} g(z[i], x[i]) = \frac{1}{n\theta} \sum_{i=0}^{n-1} \theta g(z[i], x[i]) \leq \frac{p[n]}{n\theta}.$$

Because $p[n] \geq 0$, the positive elements in $g(\tilde{z}[n], \tilde{x}[n])$ satisfy $g(\tilde{z}[n], \tilde{x}[n])^+ \leq p[n]/n\theta$ for all $n \geq 0$. Let the amount of constraint violation of $(\tilde{z}[n], \tilde{x}[n])$ be $\|g(\tilde{z}[n], \tilde{x}[n])^+\|$, we have

$$\|g(\tilde{z}[n], \tilde{x}[n])^+\| \leq \frac{\|p[n]\|}{n\theta} \quad \forall n \geq 1. \quad (2.19)$$

Given a dual optimal solution p^* , we have

$$q(p^*) = q^* \leq f(z) + (p^*)'g(z, x)$$

for any $x \in F_x$ and $z \in F_z$. Thus,

$$\begin{aligned} f(\tilde{z}[n]) &= f(\tilde{z}[n]) + (p^*)'g(\tilde{z}[n], \tilde{x}[n]) - (p^*)'g(\tilde{z}[n], \tilde{x}[n]) \\ &\geq q^* - (p^*)'g(\tilde{z}[n], \tilde{x}[n]). \end{aligned} \quad (2.20)$$

Because $p^* \geq 0$ and $g(\tilde{x}[n], \tilde{x}[n])^+ \geq g(\tilde{x}[n], \tilde{x}[n])$, we further have

$$-(p^*)'g(\tilde{z}[n], \tilde{x}[n]) \geq -(p^*)'g(\tilde{z}[n], \tilde{x}[n])^+ \geq -\|p^*\| \|g(\tilde{z}[n], \tilde{x}[n])^+\|. \quad (2.21)$$

From (2.19)(2.20) and (2.21), it follows that

$$f(\tilde{z}[n]) \geq q^* - \|p^*\| \frac{\|p[n]\|}{n\theta}. \quad (2.22)$$

Since our primal problem satisfies the Slater condition and the dual iterates have bounded subgradients, from Lemma 1 and 3 in [46], we have

$$\|p^*\| \leq \frac{1}{\gamma}(f(\bar{z}) - q^*), \quad \text{and} \quad \|p[n]\| \leq B^* \quad \forall n \geq 1,$$

where γ and B^* are defined in the above proposition. Substitute these bounds into (2.22), and we have the lower bound on $f(\tilde{z}[n])$

$$f(\tilde{z}[n]) \geq q^* - \frac{1}{\gamma}[f(\bar{z}) - q^*] \frac{B^*}{n\theta}.$$

Next, we derive the upper bound on $f(\tilde{z}[n])$ (right side of (2.18)). By the convexity of $f(z)$ and the definition of $(z[n], x[n])$ as a minimizer of the Lagrangian function $f(z) + p'g(z, x)$ over $x \in F_x$ and $z \in F_z$, we have

$$\begin{aligned} f(\tilde{z}[n]) &\leq \frac{1}{n} \sum_{i=0}^{n-1} f(z[i]) \\ &= \frac{1}{n} \sum_{i=0}^{n-1} (f(z[i]) + p[i]'g(z[i], x[i])) - \frac{1}{n} \sum_{i=0}^{n-1} p[i]'g(z[i], x[i]) \\ &= \frac{1}{n} \sum_{i=0}^{n-1} q(p[n]) - \frac{1}{n} \sum_{i=0}^{n-1} p[i]'g(z[i], x[i]) \\ &\leq q^* - \frac{1}{n} \sum_{i=0}^{n-1} p[i]'g(z[i], x[i]). \end{aligned} \quad (2.23)$$

Since $p[n+1] = [p[n] + \theta g[n]]^+$, by using the non-expansive property of projection

and the fact that 0 is in the feasible region of the dual problem (2.5), we have

$$\|p[i+1]\|^2 \leq \|p[i]\|^2 + 2\theta p[i]'g[i] + \theta^2 \|g[i]\|^2.$$

Since $g[i] = g(z[i], x[i])$, we further obtain

$$-p[i]'g(z[i], x[i]) \leq \frac{\|p[i]\|^2 - \|p[i+1]\|^2 + 2\theta \|g(z[i], x[i])\|^2}{2\theta}, \quad 0 \leq i \leq n-1.$$

By summing over $i = 0, 1, \dots, n-1$, and combining with (2.23), we have

$$\begin{aligned} f(\tilde{z}[n]) &\leq q^* + \frac{\|p[0]\|^2 - \|p[n]\|^2}{2n\theta} + \frac{\theta}{2n} \sum_{i=0}^{n-1} \|g(z[i], x[i])\|^2 \\ &\leq q^* + \frac{\|p[0]\|^2}{2n\theta} + \frac{\theta C^2}{2} \quad \forall n \geq 1. \end{aligned} \tag{2.24}$$

By combining (2.22) and (2.24), we have the desired relation. \square

This proposition shows that when using constant stepsize, the primal solutions converge to a neighborhood of the optimal solution with rate $O(1/n)$. We are interested in the constant stepsize rule for dual subgradient algorithms, mainly because of its practical importance and simplicity for implementations. Note that there is a trade-off between the size of the neighborhood and the convergence rate. If we want the primal solution to be close to the optimal one, we need to choose a small stepsize, but this would make the convergence rate very slow. This is a typical problem with using constant stepsize, and one way to avoid this situation is to use diminishing stepsize.

2.3.3 Initialization and primal solution recovery

In order to improve the convergence performance of the subgradient algorithms, we introduce some heuristics for steps 1 and 4 in the algorithm presented in Section 2.3.1. Specifically, we propose several methods for initializing the dual vector $p[0]$, and for recovering primal solutions $\{\tilde{x}[n]\}$, for both static and dynamic wireless networks.

Static networks

We start with static networks, where the topology of the network is fixed throughout the multicast. We first introduce a naive way of initializing the dual variables.

- **Averaging method** — The simplest way to generate feasible initial values for the dual variables is to assign $p_{ij}^{(t)} = s_{ij}/N_T$ for all $t \in T$ and all $(i, j) \in A$. This method is useful in static networks since no prior information of the multicast problem is available at the nodes.

For the recovery of primal solution $\tilde{x}[n]$, we have the following two options.

- **Original primal recovery** — This is the recovery method presented in step 4 in Section 2.3.1 with simple averaging.
- **Modified primal recovery** — Using the original primal recovery method, we observed in simulations that the cost of the multicast starts at a high value, and then converges slowly to the optimal value through iterations. One reason for the slow convergence is that it is recovered by averaging $x[n]$ values from all the iterations. The effect of the first few high cost iterations takes a large number of iterations later to dilute. A heuristic way to improve the convergence rate is to discard these “bad” primal solutions after some time, and just average over the most recent N_a number of iterations in primal solution recovery.

Dynamic networks

As opposed to the static assumption, many wireless networks have topologies that are dynamic. Whenever a topology change occurs, we need to restart the distributed algorithm, as the subgraph used for multicast before the topology change might have become infeasible. In such cases, all the methods discussed in the previous subsection for dual variable initialization and primal solution recovery are still applicable. However, since the difference between the optimal solutions to the multicast problem before and after the changes are usually small, we should make use of the solutions \hat{x} and \hat{p} before the topology changes in the new iterations to improve the convergence

rate. We propose additional methods to initialize p and update \tilde{x} , that make use of this old information.

For dual variable initialization, we present two additional heuristics.

- **Scaling method** — In this method, each node i scans through its set of outgoing links in A' . If a link (i, j) is an existing link in \hat{A}' before the topology change, scale the $\{\hat{p}_{ij}^{(t)}\}$ values so that they satisfy the new dual constraints. Specifically, denoting $\sum_{t \in T} \hat{p}_{ij}^{(t)} = \hat{s}_{ij}$, we assign

$$p_{ij}^{(t)} = \hat{p}_{ij}^{(t)} \times \frac{s_{ij}}{\hat{s}_{ij}}.$$

On the other hand, if link (i, j) is a new link after the topology change, we simply use

$$p_{ij}^{(t)} = s_{ij}/|T|.$$

- **Projection method** — In this method, we use an intermediate \tilde{P} which is given by

$$\tilde{p}_{ij}^{(t)} = \begin{cases} \hat{p}_{ij}^{(t)} & \text{if } (i, j) \text{ is an old link,} \\ 0 & \text{if } (i, j) \text{ is a new link.} \end{cases}$$

We can then project this \tilde{P} onto the new feasible region of the dual problem using (2.11) to obtain an initial point P for the decentralized algorithm.

On the primal side, we observe that as long as no removed link was in the multicast subgraph used before the topology change, the old $\{\hat{x}[n]\}$ values from the previous iterations are still valid under the new topology. Thus, they can be used in the recovery of the current primal optimal solution. Based on this observation, we propose the following heuristics for primal solution recovery.

- **Look-back primal recovery** — When a topology change occurs, each node checks if any of its links used in the multicast is removed owing to topology change. If yes, it sends out a signal to all nodes, and $\{\tilde{x}[n]\}$ is computed based only on the new $\{x[n]\}$ values as in the original primal recovery method

above. On the other hand, if no link is removed, the averaging is done over N_a iterations before and after the topology change. The assumption that nodes can be informed of the removal of an active link within one iteration is reasonable, since, in each iteration, distributed Bellman-Ford is used to compute $x[n]$ and sending such a signal to all nodes should take less time than running distributed Bellman-Ford.

2.3.4 Simulation results

Static networks

We use the wireless network model presented in Section 2.2.2 for our simulations, because wireless networks are a primal application for network coding. The random wireless networks are setup in a 10×10 square with a rate of connectivity $r = 3$. We run the distributed algorithm to determine the minimum energy subgraphs on these networks for multicast connections with unit rate. Here, the energy required to transmit at unit rate to a distance d is taken to be d^2 . We assume that there is no collision or interference in the network. Simulations results showed that the standard subgradient method has a better convergence time as compared to the incremental subgradient method, thus, in this section, we only present results for the standard method for Algorithm A.

Fig. 2-3 shows the average convergence performance for the proposed algorithms for networks with 30/50 nodes and 4/8 terminals in the multicast. The step sizes used in the subgradient method are $\theta[n] = n^{-\alpha}$ with $\alpha = 0.8$ for $n = 0, 1, \dots$. For the modified primal recovery method, the parameter N_a is set to 30. As we can see, the two primal cost curves coincide for the first 30 iterations, and after that, the modified method converges to the optimal value faster than the original method.

To compare the performance of the proposed scheme to the cost of multicast when network coding is not used, we use the Multicast Incremental Power (MIP) algorithm described in [53], which is a centralized heuristic algorithm to perform minimum-energy multicast in wireless networks. For the same setting, the average cost values

for the multicast given by MIP algorithm are also shown in Fig. 2-3. As can be seen, in both cases, even the initial high cost values from our distributed algorithms are lower than that from the centralized MIP algorithm. Moreover, in fewer than 50 iterations, the cost of the multicast using modified primal recovery is within 5% higher than the optimal value. Therefore, in a small number of iterations, the decentralized subgraph optimization algorithms yield solutions to the multicast problem with energy significantly lower than that for multicast without network coding even if a centralized scheme is used.

To have a feel of how Algorithm B would have performed in the above scenario, we observe in Fig. 2(a) that after 200 iterations, the cost difference between Algorithm A and the optimal value is about 0.5. For algorithm B to arrive at a neighborhood of the optimal solutions of this size, the stepsize should be smaller than 0.016 even if we use a very small $C = 5$. With this stepsize, it would take Algorithm B thousands of iterations to arrive at where Algorithm B is in 200 steps. Therefore, for all our simulations, we will use Algorithm A only.

Dynamic networks

To illustrate the performance of our algorithms in dynamic networks, we use random networks with mobile nodes. The mobility model used in our simulations is the Random Direction Mobility Model [10], where each node selects a random direction between $[0, 2\pi]$ and a random speed between $[\text{minspeed}, \text{maxspeed}]$. A node travels to the border of the simulation area in that direction, then randomly chooses another valid direction and speed, and continues the process. Note that our algorithms are applicable to all mobility models, and we have chosen this specific one for its simplicity.

In our studies, we assume that the nodes are traveling at a speed that is slow relative to the node computation speed and link transmission rate. Under such assumptions, we consider the movement of the nodes in small discrete steps, and between each step, the set of links in the network and their costs are considered constant. We refer to the period between two discrete steps as a “static period”, and let the number

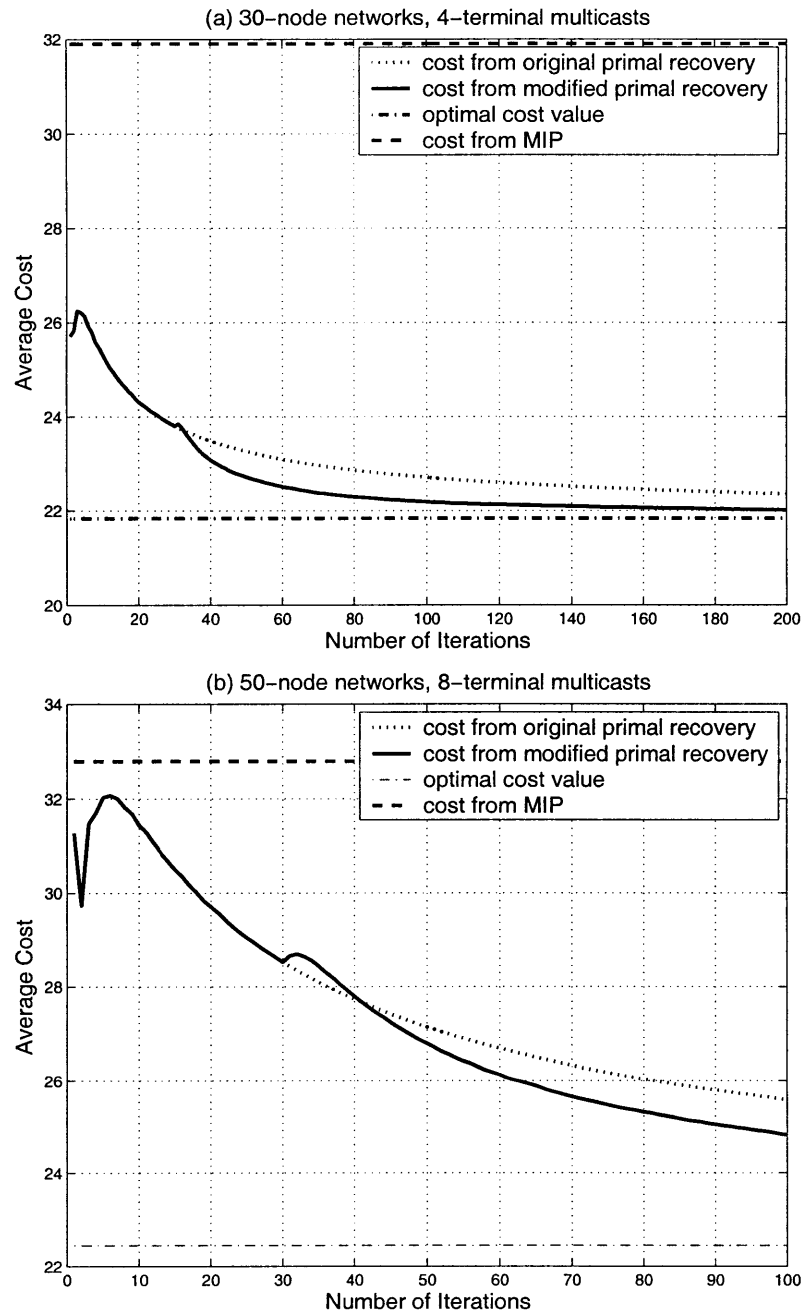


Figure 2-3: Average cost of random 4/8-terminal multicasts in 30/50-node wireless networks, using the decentralized subgraph optimization algorithms and centralized MIP algorithm. For modified primal recovery method, $N_a = 30$.

of subgraph optimization iterations performed within each static period be N_s .

We ran simulations for the various methods presented in Section 2.3.1. For dual variable initialization, we only present results based on the projection method, since it gives the best performance. Here, each node has a random speed in the interval $[0, 0.1]$ units/static period. We choose this range because the steps taken by the nodes with such speeds are relatively small as compared to r , and our assumption that the network is static between steps is valid. Also, this is a relative speed of the nodes with respect to the static period, and we can vary N_s to simulate different actual speeds of the nodes.

To illustrate the typical performance of the subgraph optimization scheme in a mobile wireless network, Fig. 2-4 shows the costs for each iteration for an instance of the multicast problem. As expected, if we flush the memory of $\{\hat{x}[n]\}$ at the end of each static period, and start accumulation for the primal cost afresh, the cost of the multicast is very spiky. On the other hand, if old $\{\hat{x}[n]\}$ values are used when they are feasible, the primal cost is usually much smoother. Of course, if node movement renders the old $\{\hat{x}[n]\}$ values infeasible, we have no choice but to start afresh, and the curves for original primal recovery and look-back primal recovery coincide (as in the 2nd static period in Fig. 2-4).

In Fig. 2-5, we show simulation results under different network and multicast settings, and for nodes with different speeds. The parameter N_a used in the modified primal recovery is set to 20. First, we compare the performance of the three options to recover primal solutions. Under the same settings, look-back primal recovery gives the lowest average cost, followed by modified and original primal recovery. We also observe that when the same methods are used, the faster the node moves, the higher the average primal cost is, owing to the lack of time for the algorithm to converge. Also, a network with more nodes or a multicast with more terminals makes convergence of the decentralized algorithm slower, and thus results in higher average primal cost.

The simulation results have shown that the decentralized subgraph optimization scheme is robust in mobile wireless networks when the nodes are moving slowly rel-

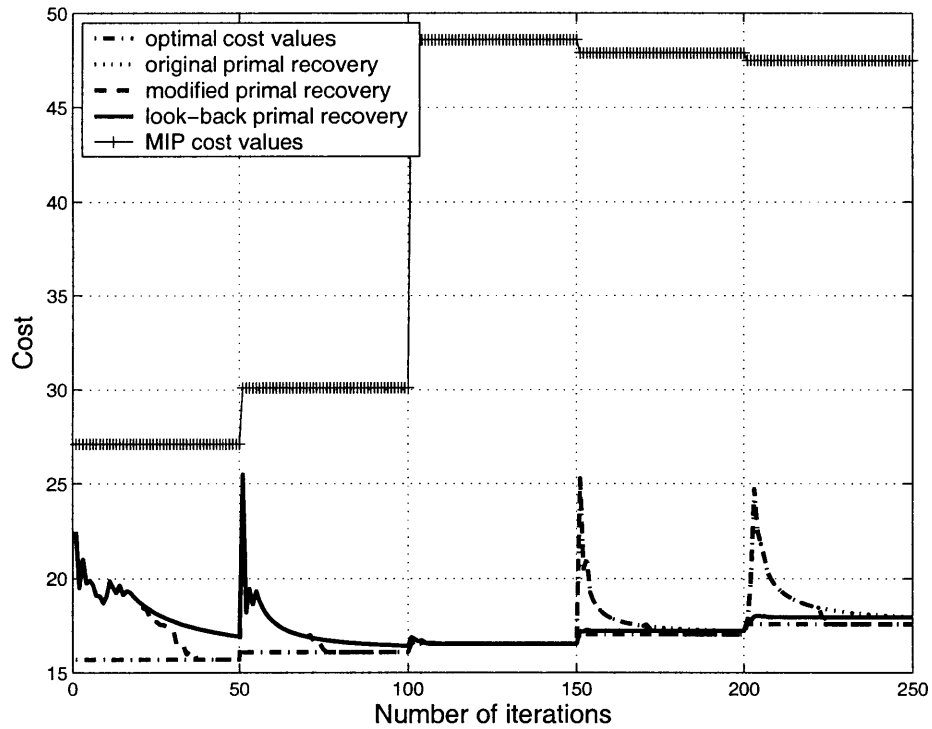


Figure 2-4: Cost of a random 4-terminal multicast in a 30-node mobile wireless network, with $N_s = 50$, under various algorithms. For the modified primal recovery method, we used $N_a = 20$ and, for the look-back primal recovery method, we used $N_a = 50$.

ative to the computation and message exchange rate of the nodes. On average, it can track the changes in the optimal value closely, and in most cases, requires lower energy for multicast than MIP even though the nodes are mobile and computation is done at each node in a distributed manner.

2.4 Min-cost subgraph algorithms for dynamic multicasts

For the dynamic multicast problem, there are two extreme cases. On the one hand, we can simply find the new optimal subgraph whenever there is an update to the multicast group, and replace the existing subgraph with this new one. In this case, users in the group will experience a lot of disruptions, but the cost of the multicast is always kept minimal. On the other hand, we can enforce that no link or code rearrangement is allowed for all existing users throughout the multicast session. In this case, users enjoy uninterrupted services, but in general, the subgraph used will deviate further and further away from the optimal one. In this section, we present one algorithm to solve the nonrearrangeable version of the dynamic multicast problem, and three algorithms for the rearrangeable version. Simulation results show that one of the rearrangeable algorithms we propose, the α -scaled algorithm, can be used to strike a balance between cost and frequency of user disturbances in a distributed manner. Although we present our algorithms based on wireline networks, they can be easily extended to wireless networks.

2.4.1 Nonrearrangeable Algorithm

For simplicity, we assume that the rate of the multicast is lower than the capacity of the links, which is generally the case in current wireline networks. In a multicast session, a source node s transmits to a group of terminal nodes T , and the group changes over time. We refer to each change of the membership of the multicast group (either an addition or a removal of a terminal node) as an *online step*. The network

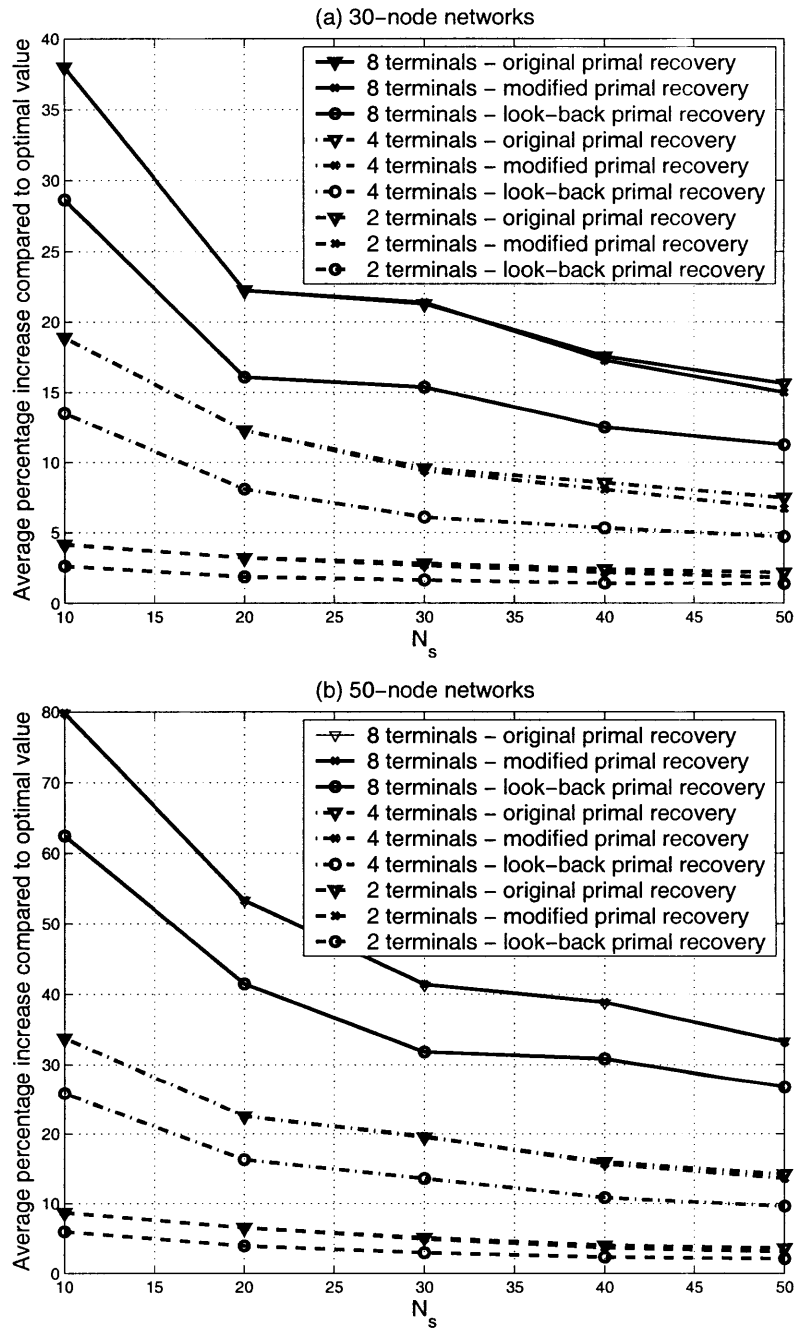


Figure 2-5: Extra energy required for multicasts in mobile wireless networks using decentralized subgraph optimization scheme in terms of percentage of the optimal value.

model and problem formulation is the same as that in Section 2.2.1. This LP problem (2.1) can be solved by a number of methods, both centrally (e.g., Simplex method) and in a distributed manner (e.g., subgradient method in Section 2.3.1). For the rest of this chapter, we denote any centralized/distributed algorithm that solves the LP problem as LP_{cent}/LP_{dist} respectively.

For the dynamic multicast problem, the initial multicast subgraph is set up by solving (2.1). If we allow complete rearrangement, we can simply solve this problem again at each online step. However, to solve the nonrearrangeable version of the dynamic multicast problem, we have to prevent link and code rearrangements from happening. To meet the no link rearrangement requirement, we basically need to make sure that the existing users still use the same path(s) for the multicast when the set T changes over time. This can be achieved by setting the cost of the links in the current subgraph G_c to zero. If the capacity of a link is larger than the rate used for the multicast, then the link is split into two virtual links, one with capacity equal to the rate used for the multicast and cost zero, and the other with the remaining capacity and cost unchanged. For example, if link (i, j) has capacity $c_{ij} = 2$ and rate of flow $z_{ij} = 1$ for the multicast, then nodes i and j treat link (i, j) as two parallel links with capacities 1 each, and one of them has cost of 0, and the other one has cost of a_{ij} . After doing this, the current multicast subgraph becomes “free”, and doing optimization on this new cost assignment will always lead to using the same path(s) to serve the existing users in the new subgraph. Therefore, link rearrangements are avoided.

One problem with the above method is that some links not necessary for the new terminal set might be included in the new subgraph after a removal of a terminal. This is because all link in the old subgraph are free, and some of these links might still be included in the solution to the LP problem even though they are not necessary in performing the multicast to the new terminal set. To solve this problem, instead of setting their costs to 0, we can set the cost of the used links to a small value ϵ , so that no extra link would be included in the optimal solution, and, at the same time, the used links are still almost free as compared to the other links.

```

node i
nodeUsed = 0
for all (j,i) ∈ A
  if (j,i) ∈ Gc
    aji = ε
    nodeUsed = 1
  end
end
if nodeUsed = 1
  for all (j,i) ∈ A
    if (j,i) ∈ Gc    aji = M
  end
end
call LPdist

```

Figure 2-6: Nonrearrangeable algorithm.

As for code rearrangements, we want to prevent the usage of new links that go into existing nodes of the subgraph. To do that, each node in the subgraph can scan through its incoming links, and sets the cost of those unused links to a very large value, M . Again, if the capacity of an incoming link is not fully used in the current subgraph, we can split it into two parallel virtual links as above. These nodes then send the new costs of its incoming links to their corresponding tail nodes, and the new high costs can prevent these links from being used.

After making these changes to the link costs, when an online step occurs, we can simply run LP_{dist} again with the new costs, and obtain a feasible subgraph for the new multicast group without any link or code rearrangements. This algorithm is summarized in Fig. 2-6.

The above algorithm can be complicated due to the splitting of physical links into parallel virtual links. This requires more processing at the nodes and more coordination between the end nodes of the links. In addition, in the non-rearrangeable solution of the dynamic multicast problem, it is inevitable that the subgraph used would deviate further and further from the optimal subgraph. This is because the no-rearrangement requirement forces the subgraph we use as close as possible to the initial subgraph. Thus, when the multicast group changes further and further away

```

node i
for all  $(j, i) \in A$ 
  if  $(j, i) \in G_c$ 
     $a_{ji} = \epsilon$ 
  end
end
call  $LP_{dist}$ 

```

Figure 2-7: MLR algorithm.

from the original group over time, our multicast subgraph becomes more and more suboptimal.

To simplify this algorithm and keep the cost of the multicast low, we may need to make some compromise and allow some rearrangements. In the next subsection, we present three such heuristic algorithms.

2.4.2 Rearrangeable algorithms

Algorithm for minimizing link rearrangement (MLR)

One way to simplify the nonrearrangeable algorithm is to focus on eliminating link rearrangement only, and ignore code rearrangement. This can be easily done by setting the used links costs to a very small value ϵ after each online step as in the nonrearrangeable algorithm, and call LP_{dist} to solve the new LP problem. This algorithm, which we call the MLR (minimal link rearrangement) algorithm, is shown in Fig. 2-7.

The motivation for this algorithm comes from the observation that the complication of splitting links into used and unused portions arises when we have a non-tree subgraph, and things would be much simpler if we only have to deal with trees. This is because, in trees, each node only has one incoming link, and it has full information of the multicast. Thus, there is no worry about code rearrangement.

Notice that once the multicast subgraph becomes a tree, it will remain as a tree through the rest of the online steps. To see this, consider addition of a new node to the multicast group. Since the original subgraph G_c is considered “free” and each node in G_c has full information of the multicast, the new terminal node only needs to

find the shortest path from any node in G_c to itself, and attach itself to the subgraph. As for the removal of a terminal, only a part of the tree may be removed, and the remaining graph should still be a tree. Since at every step, if G_c is not a tree, there is some positive probability that it will become a tree, and once it evolves into a tree, it will stay that way till the end of the multicast. Therefore, if we keep running the dynamic multicast session, the probability that we are dealing with trees goes to 1.

In addition, simulations on practical networks show that in more than 98% of cases, we do get the optimum Steiner tree at startup. Therefore, we can focus on link rearrangements only and use the MRL algorithm. This algorithm still works if the initial subgraph is not a tree, the only difference is that we cannot guarantee that there will not be any rearrangements in such cases.

Algorithm for limiting multicast cost (LMC)

If we use the MLR algorithm, it is expected that as time goes on, the subgraph used for multicast will move further and further away from the actual optimal subgraph for the current set of terminal nodes. As an alternative, we might want to introduce occasional rearrangements in order to keep the cost of the multicast close to optimal. We introduce the LMC algorithm, shown in Fig. 2-8, to do this. In this algorithm, the nodes run two programs in parallel, one of which generates the subgraph with no rearrangement using the algorithm presented above. We call this subgraph the *no-change subgraph* G_{nc} , and the cost of this subgraph C_{nc} . The other program keeps track of the optimal subgraph, G_{opt} , for the current set of multicast terminals, and the cost of G_{opt} is C_{opt} . At each step, the cost of the two subgraphs are compared, and if the cost of the no-change subgraph is higher than the optimal graph by a certain factor, β , we switch to the optimal subgraph. Using this method, we can control the trade off between the frequency of disturbances to the users and the cost of the subgraph used for the multicast by changing the value of β . However, this method requires the nodes to keep track of two subgraphs, and centralized coordination is needed to compare the costs and make the nodes switch between two subgraphs simultaneously.

```

call  $LP_{cent}$  to compute  $C_{opt}$  and  $G_{opt}$ 
for all  $(j, i) \in A$ 
  if  $(j, i) \in G_c$ 
     $a_{ji} = \epsilon$ 
  end
end
call  $LP_{cent}$  to compute  $C_{nc}$  and  $G_{nc}$ 
if  $C_{nc} > C_{opt} \times (1 + \beta)$ 
  use  $G_{opt}$  for the multicast
else
  use  $G_{nc}$  for the multicast
end

```

Figure 2-8: LMC algorithm.

```

node i
for all  $(j, i) \in A$ 
  if  $(j, i) \in G_c$ 
     $a_{ji} = \alpha \times a_{ji}$ 
  end
end
call  $LP_{dist}$ 

```

Figure 2-9: α -scaled algorithm.

α -scaled algorithm

We now present a simple approximate algorithm that can trigger “auto-switching” between G_{nc} and G_{opt} in a distributed manner. Instead of assigning a very small cost to the used links as in the MLR algorithm, we can use a scaled value of the original cost, i.e., for an existing link (i, j) in the subgraph, we use αa_{ij} as its cost in the future computations as long as it stays in the subgraph, where α is a scaling factor between 0 and 1. If $\alpha = 0$, it is the same as the MLR algorithm; and if $\alpha = 1$, we will be using the optimal subgraph every time. We refer to this algorithm as the *α -scaled algorithm*, and it is shown in Fig. 2-9.

To see why this heuristic works and how the constants α and β are related, consider the case of removal of a terminal node. The LMC algorithm compares the values of C_{nc} and $(1 + \beta)C_{opt}$, and picks the lower of the two. Since G_{opt} may overlap with the existing subgraph from before the online step, we assume the cost of this overlapping

part of the subgraph is C_{ol} and the cost of the rest of the optimal subgraph is C_{others} . Thus, the comparison is equivalent to

$$\frac{1}{1 + \beta} \times C_{nc} \geq C_{ol} + C_{others} \quad (2.25)$$

On the other hand, in the α -scaled algorithm, we are effectively choosing the lower cost between these two.

$$\alpha \times C_{nc} \geq \alpha \times C_{ol} + C_{others} \quad (2.26)$$

If we set α to $1/(1 + \beta)$, we can see that equations (2.25) and (2.26) are very similar except the first term on the right hand side. By scaling the existing link costs by α , we can satisfy the requirement that the cost of the subgraph used never goes over $(1 + \beta)C_{opt}$, but owing to the scaling factor α on C_{ol} , the approximate algorithm switches to the optimal subgraph more often than required by β . Using similar analysis, we have the same results for the case of addition of a terminal.

Thus, using an appropriate α to scale the costs of the used links, the optimization can trigger auto-switching between the two subgraphs, thus keeping the cost of the multicast low. In addition, we can make α a time-varying variable. In general, when a link is first added into the subgraph, it is likely that it will remain there for a while. However, the probability that the link remains in the optimal subgraph decreases with the online steps. To capture this characteristic, we can use a lower value for α for the first few online steps after a new link is added, and increase α gradually later on. Also, in a practical network, it may not be desirable to make back-to-back changes to the link connections, i.e., addition of a link to the multicast subgraph followed by an immediately removal of it in the next step. We can reduce the occurrence of such events by setting the α of new links to 0 for a few steps before raising it to the normal value of $1/(1 + \beta)$.

2.4.3 Simulation results

We first present simulation results for the MLR algorithm. The network topologies used in the simulations are obtained from the Rocketfuel project [2]. In each simulation, we start with a multicast from a random source to a set of 10 random terminals. Subsequently, in each online step, we first randomly decide whether there is an addition or removal of terminal, and then randomly select a terminal to add/remove based on that decision. Figs. 2-10 and 2-11 show the average increase of cost of the no-change subgraph as compared to the cost of the optimal subgraph in terms of percentage of C_{opt} . The network topology used for Figs. 2-10 and 2-11 are backbones for Exodus (US) and EBONE (Europe), respectively. As expected, the extra cost of the no-change subgraph grows approximately linearly with the online steps. In addition to the average curve, we also show the data points for each instance of the simulation in both Figs. 2-10 and 2-11. Note that there are cases when the cost of the no-change subgraph is as much as 60% higher than the optimal cost after 20 steps.

This undesirable phenomenon motivates the usage of the α -scaled algorithm. Fig. 2-12 shows the simulation results for using the α -scaled algorithm on the network used in Fig. 2-10. Here, we aim to control the cost of the sub-graph used to within $\beta = 30\%$ away from that of the optimal subgraph, thus, we use $\alpha = 0.75$. The average curve in Fig. 2-10 for the MLR algorithm is also shown here for comparison. The α -scaled algorithm provides lower cost for the multicasts as compared to the MLR algorithm. More importantly, the cost difference between the subgraph used and C_{opt} for the α -scaled algorithm is roughly constant after a while, and it does not grow over time. Of course, there is a price for this gain, which is the occasional switching from the no-change graph to the optimal graph. In this case, the average switching probability is 11.7%, which means, out of a hundred online steps, there are about 12 times when the existing users might experience disturbances to their transmissions. Furthermore, if we look closely at the data points for individual instances, we can see that, actually, none of the instance has gone over 20% higher than the optimal one. This is consistent with our discussion in Section III about the values of α and

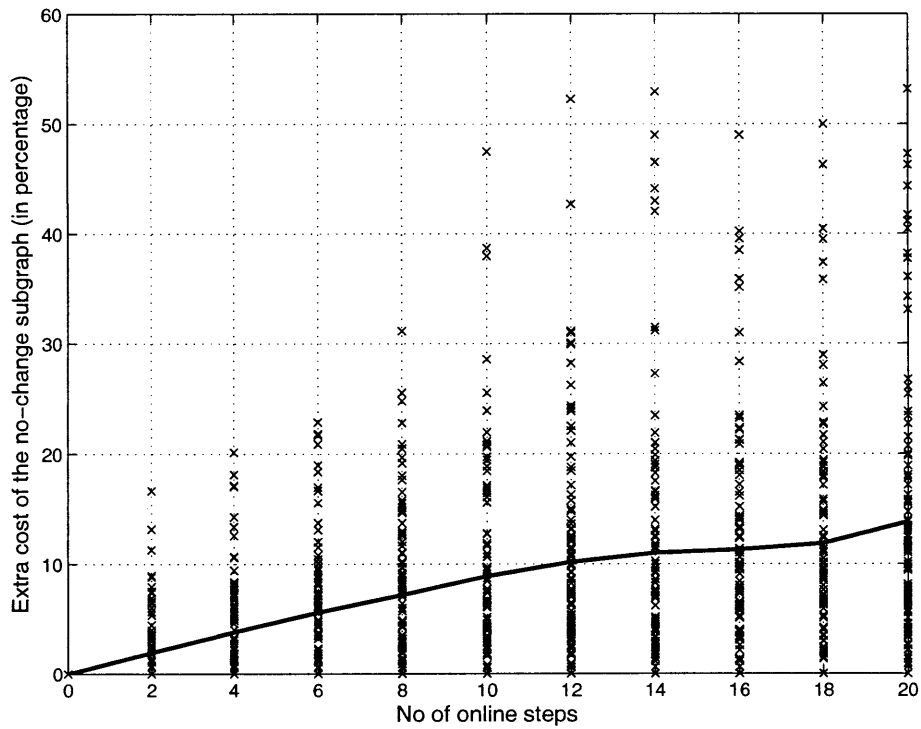


Figure 2-10: Extra cost of the multicast subgraph generated by the MLR algorithm in terms of percentage of C_{opt} for the Exodus network. We are showing both the individual data points for each trial and the average curve.

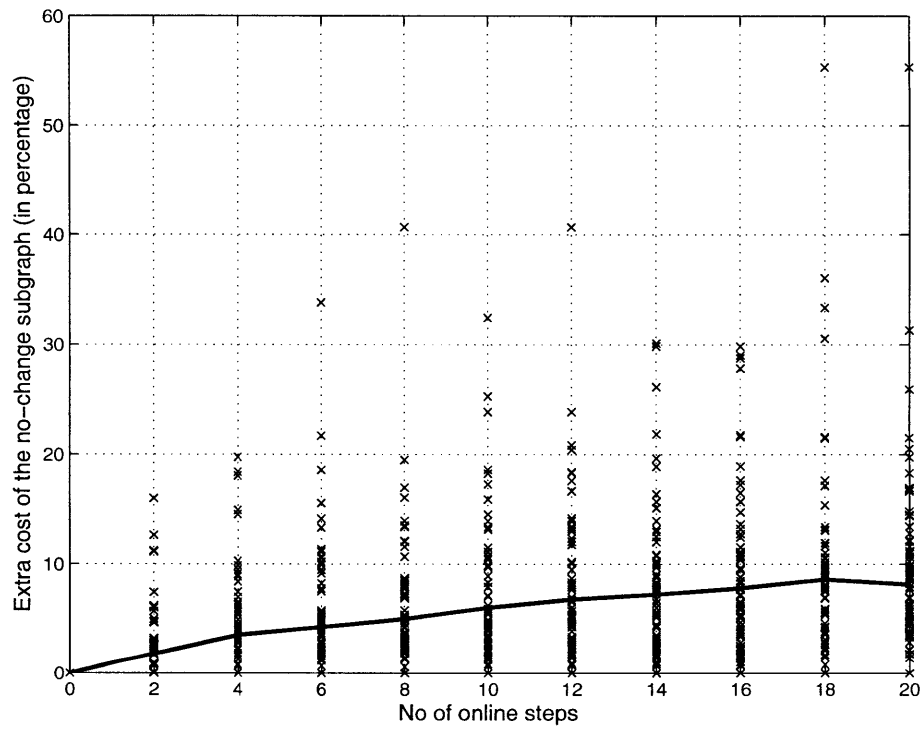


Figure 2-11: Extra cost of the multicast subgraph generated by the MLR algorithm for the EBONE network in terms of percentage of C_{opt} .

β . Therefore, if we want to have $\beta = 30\%$, we can use a lower value for α .

Finally, Fig. 2-13 shows the simulation results for the same network setup with different α values. As we can see, the higher the α value, the lower the average cost of the subgraph. At the same time, higher α values lead to higher switching rate. We observed that when α is equal to 0.5, the cost of the subgraph used is kept at around 9% higher than the optimal cost, whereas the switching probability is only 2.05%. Therefore, by selecting the α value properly, we can keep the cost of the multicast close to optimal during the multicast session while causing few disturbances to the existing users.

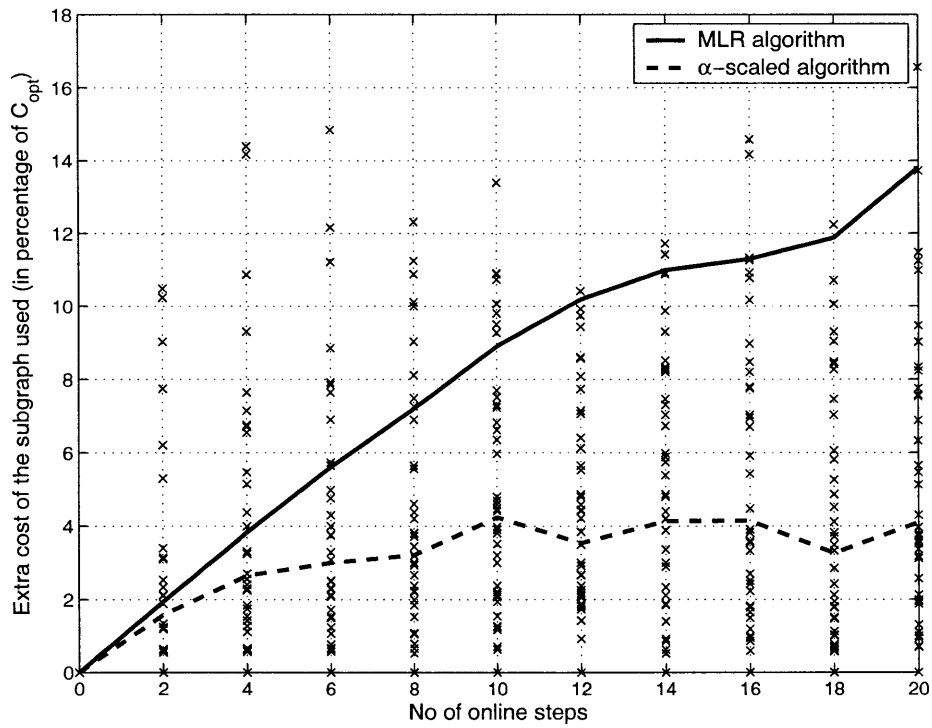


Figure 2-12: Extra cost of the multicast subgraph generated by the α -scaled algorithm with $\alpha = 0.75$ and the MLR algorithm in terms of percentage of C_{opt} , on the Exodus network. We are also showing the individual data points for each trial for the α -scaled algorithm.

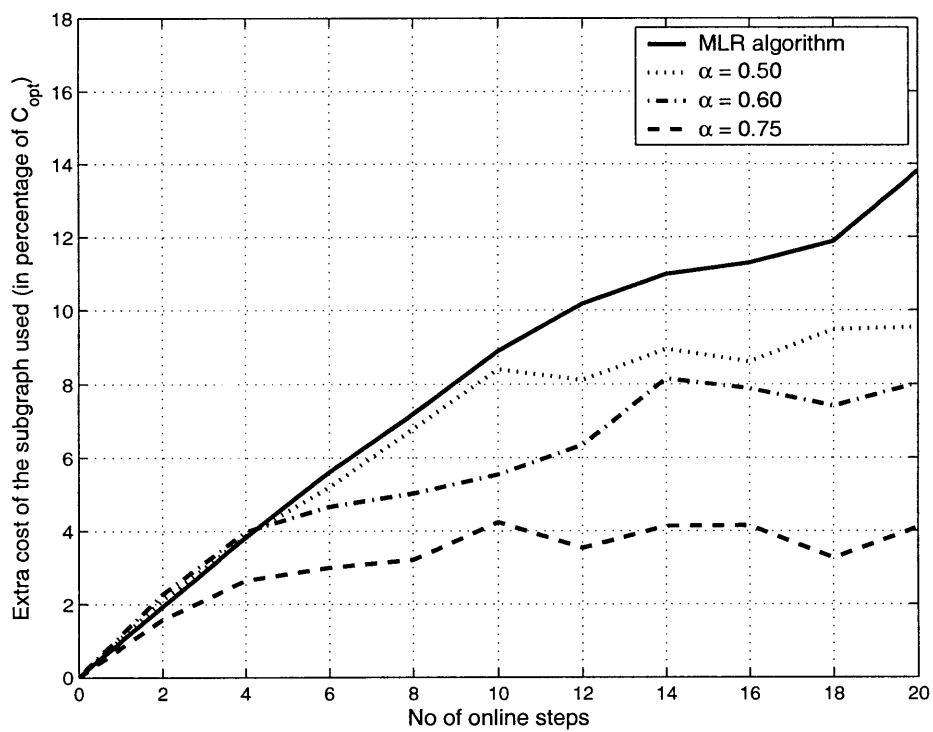


Figure 2-13: Extra cost of the multicast subgraph generated by the α -scaled algorithm with various α values, on the Exodus network.

Chapter 3

Analysis and Improvement to COPE

In this chapter, we focus on COPE [28, 29], a new architecture for wireless mesh networks that employs opportunistic network coding to improve throughput in congested networks. The COPE system, which we will introduce in details in Section 3.1 is a completely distributed system that significantly improves the throughput of ad hoc wireless networks with UDP traffic. Several attempts have been made to analyze the COPE performance and explain this big gain, however, they are not completely satisfactory. In this chapter, we give a new analysis of the COPE system, which explains all the main characteristics of the COPE performance curves observed in experiments. Furthermore, based on the analysis, we propose a simple modification to the COPE system that can further improve the network throughput.

3.1 Background

3.1.1 The COPE system

COPE, introduced by Katti et al. [28, 29], is a new forwarding architecture for wireless network that inserts a coding shim between the IP and MAC layers, which identifies coding opportunities and benefits from them by forwarding multiple packets

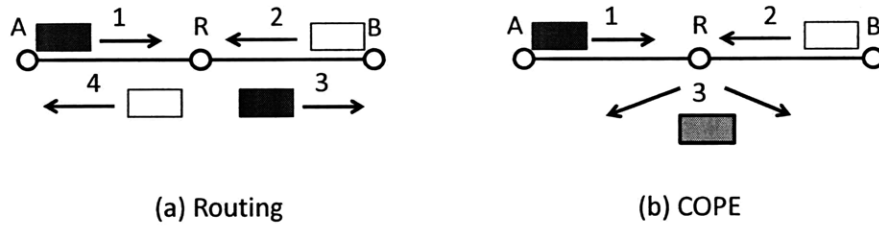


Figure 3-1: Example of how COPE increases the throughput in the Alice-and-Bob wireless network.

in a single transmission. We first explain the basic idea of this scheme by using the Alice-and-Bob network shown in Fig. 3-1. Here, Alice and Bob want to exchange a pair of packets via a router, R . In a traditional routing network, Alice and Bob would first send their packets to R , and then R forwards the two packets to their respective destinations in two time slots. This process takes 4 transmissions. However, if network coding is allowed in the router, after R has received the two packets from Alice and Bob, it can XOR the two packets together and broadcast this new packet. When Alice and Bob receive the XOR-ed packet, they can obtain each other's packet by XOR-ing again with their own packet. In this way, we utilize the broadcast nature of the medium and save one transmission, which can be used to send additional data, increasing the network throughput.

Even larger coding gain can be obtained when more packets are coded together. For example, consider the cross network shown in Fig. 3-2. Here, nodes 1, 2, 4, 5 each has a packet to be sent to the opposite node via node 3 in the middle. In addition, when node 1 sends, nodes 2 and 4 can overhear the transmission. Same goes for nodes 2, 4, and 5. It is easy to see that in conventional routing network, it takes 8 transmission for the 4 packets to be delivered. However, in COPE, we can first let the four source nodes send their packets to 3, and then 3 XOR all of them together and broadcast the code packet. Since every node has its own packet and the two packets overheard from the transmissions of their neighbors, it can derive the packet destined to it from the XOR-ed packet. Therefore, in COPE, the process only takes 5 transmissions, and we save $3/8$ of the bandwidth as compared to the routing case.

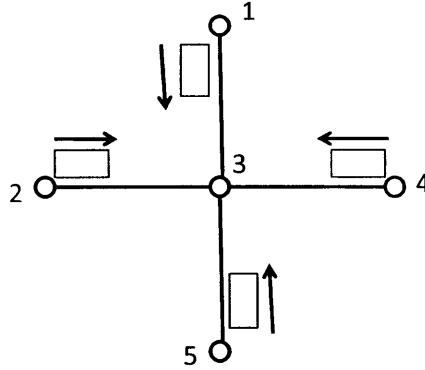


Figure 3-2: Cross network.

In summary, COPE employs network coding to utilize the broadcast nature of the wireless channel. The coding here is simple XOR, and the decoding is done at the next hop, i.e., there is no forwarding of the coded packets. Implementation of the COPE system involves many practical issues, as explained in [28, 29]. Here, we only summarize the three main techniques incorporated in COPE:

1. **Opportunistic Listening:** Since the wireless channel is a broadcast medium, and the nodes are equipped with omni-directional antennae, COPE makes the nodes snoop on all communications over the wireless medium and store the overheard packets for a limited period. In addition, each node broadcasts reception report to its neighbors about which packets it has stored, to enable their neighbors to find coding opportunities.
2. **Opportunistic Coding:** Based on its knowledge of what the neighbors have, a node decides on what packets to code together. The rule it follows is to maximize the number of native packets delivered in a single transmission, while ensuring that each intended nexthop has enough information to decode its native packet.
3. **Learning Neighbor State:** In addition to using the reception reports to find out what packets a neighbor has, a node may also need to guess whether a neighbor has a particular packet. This is done intelligently by leveraging the routing computation. In the absence of deterministic information, COPE

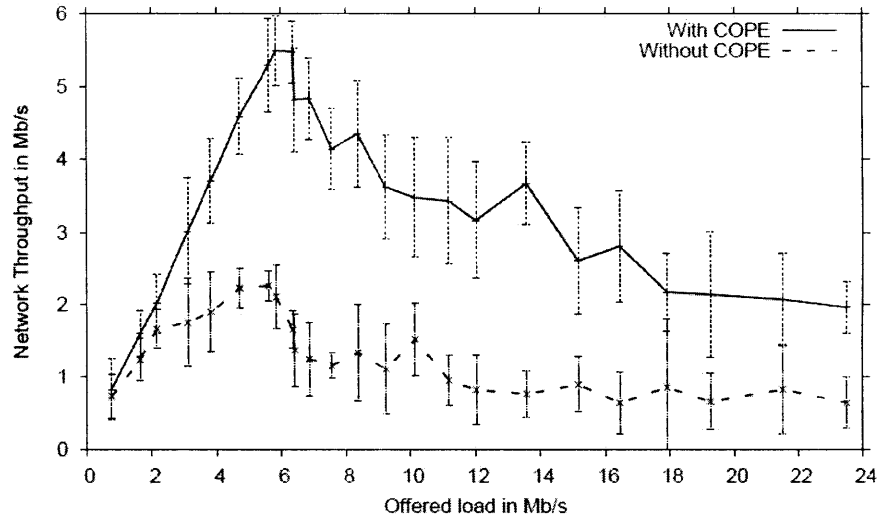


Figure 3-3: COPE can provide a several-fold (3-4x) increase in the throughput of wireless ad hoc networks with UDP flows. This figure is taken from [29].

estimates the probability that a particular neighbor has a packet as the delivery probability of the link between the packet's previous hop and the neighbor.

COPE has been implemented by Katti *et al.* [29] in a 20-node wireless mesh network, and tested with both TCP and UDP traffic. In their tests, TCP does not show any significant improvement with coding, and this is due to TCP's reaction to collision-related losses. Due to collisions at the bottleneck nodes, the TCP flows suffer timeouts and excessive back-off. Thus, the bottleneck nodes never see enough traffic to make use of coding. Few coding opportunities arise, and hence the throughput performance is the same with and without coding.

On the other hand, with UDP traffic, COPE can provide a several-fold increase in the throughput of wireless ad hoc networks. Fig. 3-3, taken from [29], demonstrates the throughput gain of the COPE system for the 20-node testbed for UDP traffic with randomly picked source-destination pairs, Poisson arrivals, and heavy-tail size distribution.

3.1.2 Existing analysis on COPE

The good performance of COPE with UDP traffic has attracted the attention of many researchers, and several attempts have been made to model COPE and explain the huge throughput gain.

Sengupta *et al.* formulated the throughput computation in a wireless network coding system into a linear programming (LP) problem [49]. Their formulation only considers the coding cases involving two packets, the scenario illustrated in Fig. 3-1. However, from the statistics in [29], we see that coded packets consisting of more than two native packets plays an important role in the throughput gain. In addition, the LP formulation does not capture the interaction between COPE and the MAC protocol very well. The LP problem enforces fairness among the overall flows, whereas in reality, fairness is enforced by the MAC protocol on a local scale. These differences lead to discrepancies between the experimental results and that predicted by the theoretical formulation. Also, the authors suggested that routing be made aware of network coding opportunities rather than, as in COPE, being oblivious to it. We will discuss in the following sections that this may not be a good idea.

Le *et al.* [37] tries to understand COPE by focusing on one *coding structure* at a time. A *coding structure* includes one coding node as well as the one-hop predecessor nodes and the one-hop successor nodes of the associated coding flows. The networks in Fig. 3-1 and Fig. 3-2 are both examples of single coding structures. The key performance measure they use is the encoding number, i.e., the number of packets that can be encoded by a coding node in each transmission. They upper bound the throughput gain in COPE by $2n/(n+1)$ for a general wireless network, where n is the maximum encoding number in one of its coding structures. Clearly, this upper bound is less than 2, which is much smaller than the throughput gain observed in Fig. 3-3. This is due to the fact that the analysis in [37] only deals with coding gain, but does not take into consideration the coding+MAC gain.

3.2 COPE performance analysis

The existing analysis of COPE fails to address two important aspects of the performance curves.

1. The magnitude of the throughput gain in COPE experiments is much larger than that predicted by the analysis;
2. As shown in Fig. 3-3, for both the COPE and the non-COPE systems, the throughput first increases linearly with the offered load. After reaching a *peak* point, the throughput decreases with increased load, and finally settles down to a *saturation* level. On the contrary, the performance curves derived from the existing theoretical formulations have a different shape. The throughput rises with increased load until it reaches a saturation level, and further increase in load does not affect the total throughput by much.

These discrepancies motivate us to take a closer look at the COPE system. We find out that the key to explain the COPE performance curves lies in the interaction between coding and the MAC protocol, and the local fairness enforced by the MAC protocol when it assigns bandwidth to competing nodes.

To understand this, we first look at the simple Alice-and-Bob network shown in Fig. 3-1. Here, we assume that the three nodes in the network share the wireless channel, and the total bandwidth is 1. Flows of size 0.01 are originated from node A/B , and are to be sent to node B/A , respectively. The relay node, R , does not generate any traffic. By increasing the number of flows, the total offered load to the network is increased. Also, the probability that a flow is generated by A or B is equal. We denote the bandwidth allocated to nodes A , B , and R as BW_a , BW_b , and BW_r , respectively. The wireless channel is lossless.

- **Routing (non-COPE) case:** When the offered load is very small, every node can get enough bandwidth to transfer what they have, and the total throughput grows linearly with the offered load. The bandwidth demand for the relay node is equal to the sum of the sending rates at A and B , and the total throughput

of the system is always equal to BW_r . The throughput reaches its peak when the channel bandwidth is completely used up, i.e.,

$$BW_a = 0.25, BW_b = 0.25, BW_r = 0.5.$$

In this case, the total throughput of the system is 0.5.

As the offered load increases beyond 0.5, the system will not be able to handle all the traffic. Queues at some of the nodes will grow, and packets are going to be dropped. Consider the *saturation* case, when the offered load is very large, all the nodes have backlogs. They are constantly competing for the channel. In this case, the MAC protocol will allocate the channel fairly among the three nodes, i.e.,

$$BW_a = BW_b = BW_r = 1/3.$$

Note that the total throughput of this system is always equal to BW_r , thus, the saturation throughput is $1/3$.

Now, we look at the transition stage where the offered load is between $1/2$ and $2/3$. For simplicity, assume symmetric load for nodes A and B , and we denote it by l . If the assigned bandwidth for A and B is less than that required to clear their queues, they would be constantly requesting for the channel. This situation is the same as that in the saturation stage, and they will be allocated $1/3$ bandwidth each. However, since their demand is less than $1/3$, they won't have a backlog in this case. This is a contradiction, therefore, nodes A and B will have bandwidths equal to l , and the bandwidth assigned to R is then equal to $1 - 2l$. The complete throughput curve is shown in Fig. 3-4.

- **Coding (COPE) case:** When coding is allowed at the relay node, if the loads at A and B are perfectly symmetric, every packet delivered by R generates a throughput of 2 packets. Similar to the routing case, when the offered load is small, every node gets its required bandwidth, and the throughput grows

linearly with offered load until the total bandwidth is used up, i.e.,

$$BW_a = BW_b = BW_r = 1/3.$$

Since every transmission by R delivers two packets, the peak throughput of the COPE system is $2/3$. When the offered load is increased further, packets starts to get dropped by A and B , however, the bandwidth allocation remains the same in the saturation stage, and the total throughput stays at $2/3$. This throughput curve is also plotted in Fig. 3-4.

We next consider the cross network shown in Fig. 3-2. Assume symmetric traffic is generated at the four side nodes, 1, 2, 4, and 5, and to be delivered to the opposite nodes. Also, when node 1 transmits, nodes 2 and 4 can overhear the transmission, and they store the overheard packets for future decoding. Same goes for the three other nodes. The curves in Fig. 3-5 show the throughput performance for this network with and without coding.

- **Routing (non-COPE) case:** Similar to the Alice-and-Bob network, the total throughput first increases linearly with the offered load until it reaches the peak where $BW_1 = BW_2 = BW_4 = BW_5 = 1/8$, $BW_3 = 1/2$, and total throughput is $1/2$. The throughput then drops when offered load is increased further until it reaches the saturation stage where each node is allocated a bandwidth of $1/5$, and the total throughput is also equal to $1/5$.
- **Coding (COPE) case:** When the load at the four side nodes are perfectly symmetric, the relay node 3 can code four packets together every time it transmits. The throughput of this system peaks at $4/5$ when the offered load at each side node is equal to $1/5$. The throughput then remains at this level when offered load is further increased.

As we can see, when more flows are involved in the coding structure, the throughput gain becomes larger. This gain is not just due to coding, but also due to the

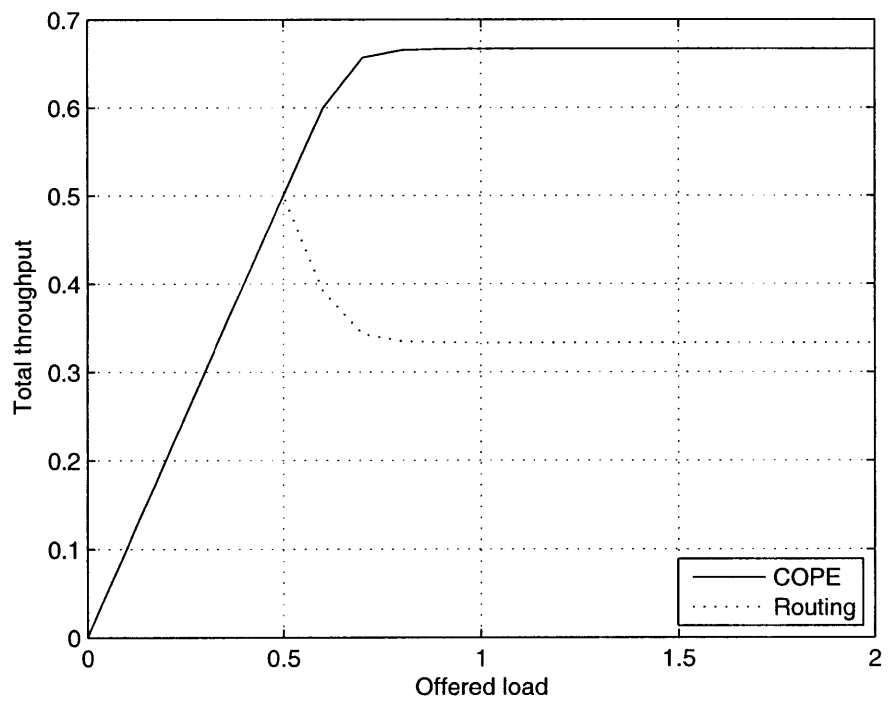


Figure 3-4: Throughput for COPE and non-COPE systems in an Alice-and-Bob network with cross traffic only.

fact that the MAC protocol allocates bandwidth among competing node fairly. The throughput of the system is limited by the bandwidth at the bottleneck node. With coding, the bottleneck node drains its queue multiple times faster than that in the non-coding case, thus resulting in the significant throughput gain. Even larger throughput gain can be obtained if the coding structure involves more than four flows, but they rarely happen in a practical system.

In the above simple models, we only considered cross traffic and all flows can be coded together. What happens when there exist unicast flows that cannot be coded with any other flow? To answer this question, we consider the cross network where in addition to the traffic generated by the side nodes, there are also flows generated by the center node, node 3, to be sent to one of the side nodes. The throughput performance of the COPE and non-COPE systems for this scenario is plotted in Fig. 3-6. As we can see, in the coded system, the total throughput drops after reaching the peak. This is because the traffic generated by the center node cannot be coded with any other packets, and the bandwidth used to send these ‘unicast’ packets are less efficiently used as compared to that used for sending the coded packets. As more and more flows are generated by node 3, these ‘unicast’ packets take up more and more bandwidth at the bottleneck node, reducing the total throughput.

The curves in Fig. 3-6 resembles that in Fig. 3-3 both in shape and in the magnitude of gain. As in the experiments, the largest gains are observed when the non-COPE curve has started dropping from the peak, and the COPE curve has yet to drop. Although our analysis only focuses on one coding structure, we believe the performance of COPE in a general network follows the same trend. This is because in a practical network, the throughput is limited by a few bottleneck nodes (coding structures). Therefore, by closely examining one coding structure, we can understand what really causes the COPE system behave in such a way. As mentioned previously, the key factor here is the interaction between COPE and the MAC and also the local fairness enforced by the MAC protocol.

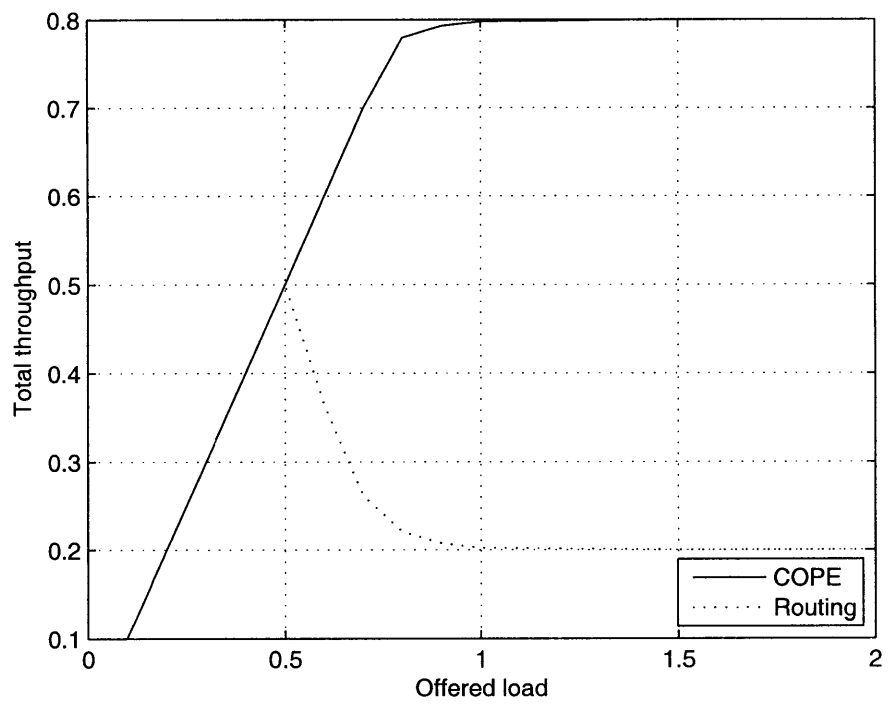


Figure 3-5: Throughput for COPE and non-COPE systems in a cross network with cross traffic only.

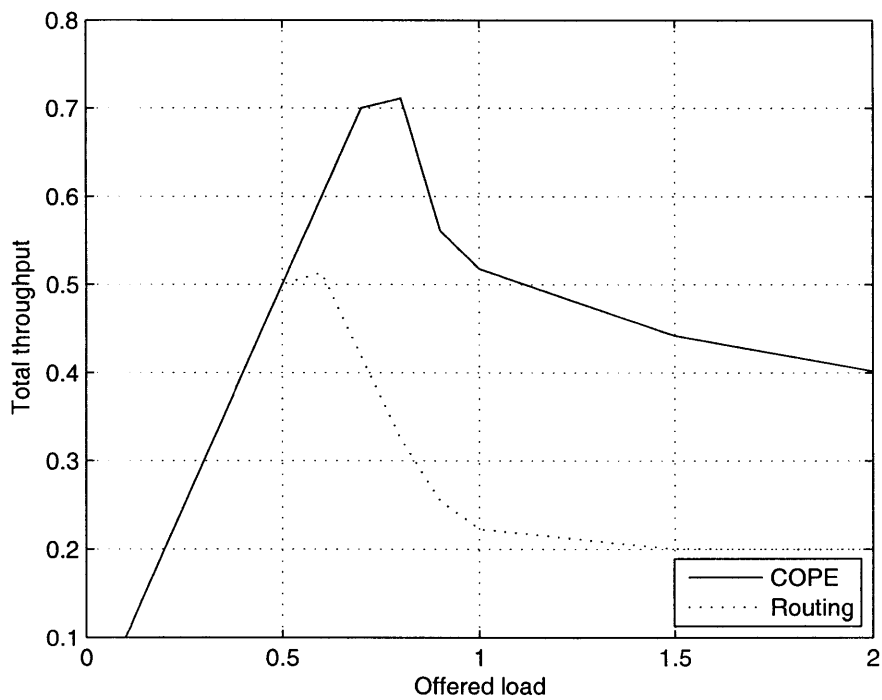


Figure 3-6: Throughput for COPE and non-COPE systems in a cross network with cross traffic and traffic generated at the center node.

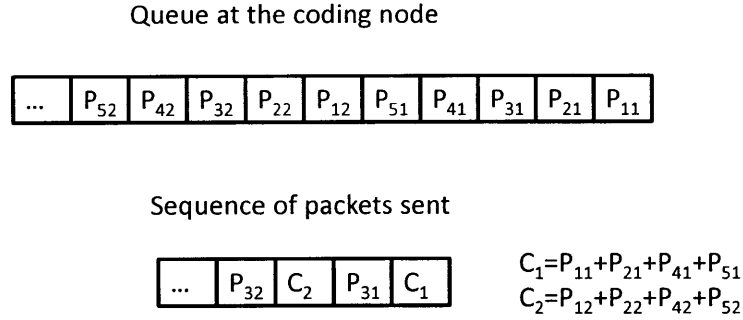


Figure 3-7: An example of queue status and packets sent in a cross network with COPE.

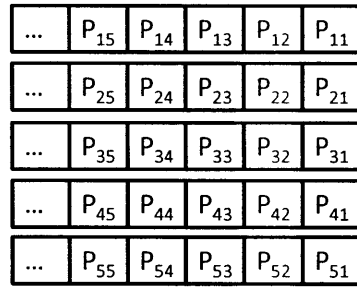
3.3 Improvements on COPE

Our observation in the previous section leads to a simple improvement of COPE that can further increase the network throughput. Recall that in the case when there are both cross traffic and traffic originated from the center (Fig. 3-6), the reason why the COPE curve drops is because the center node has to use some of its bandwidth to take care of the ‘unicast’ packets, which are less efficient in terms of throughput. To improve the total throughput, we would like to give higher priority to coded packets, as they help to drain the queue at the bottleneck node at a faster rate. A simple way to do this is to have virtual queues for each input-output pair at the coding node, and packets are sent from these virtual queues in a round-robin manner.

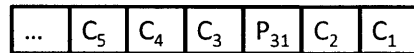
In the current COPE system, only one queue is maintained at a node. Every time there is a transmission opportunity, the node dequeues the first packet, and checks if it can be coded with any other packets currently in its queue. If yes, the packets would be coded together and sent out; otherwise, the native packet will be sent alone. Fig. 3-7 illustrates the sequence of packets sent by the center node in a cross network with COPE. Here, P_{ij} denotes the j -th packet from node i . As we can see, in this case, ‘coded’ and ‘uncoded’ packets share the bandwidth equally, which is very inefficient and unfair, as the coded packets serve more users than the uncoded one.

If we keep separate virtual queues for each input-output pair and serve them in a round robin manner, what would happen in the cross network case is illustrated in

Virtual queues at the coding node



Sequence of packets sent



$$\begin{aligned}
 C_1 &= P_{11} + P_{21} + P_{41} + P_{51} \\
 C_2 &= P_{12} + P_{22} + P_{42} + P_{52} \\
 C_3 &= P_{13} + P_{23} + P_{43} + P_{53} \\
 C_4 &= P_{14} + P_{24} + P_{44} + P_{54} \\
 C_5 &= P_{15} + P_{25} + P_{45} + P_{55}
 \end{aligned}$$

Figure 3-8: An example of queue status and packets sent in a cross network with modified COPE.

Fig. 3-8. Here, the uncoded packets take up a much smaller fraction of the bandwidth, and the total throughput of the system improves.

We simulated the cross network with this simple modification, and the results are shown in Fig. 3-9. This modification leads to about 50% gain in the network throughput as compared to the original COPE system.

We would also like to point out that coding-aware routing has been suggested as a method to improve COPE performance. The main idea is to route traffic in such a way that generates more coding opportunities. However, doing so would only increase the ‘coding gain’ of COPE over routing under the same routing choice. What we really want is not this ‘coding gain’, but rather the total throughput gain. We should not sacrifice throughput just to increase coding opportunities. Therefore, coding-aware routing is not really a good way to improve COPE.

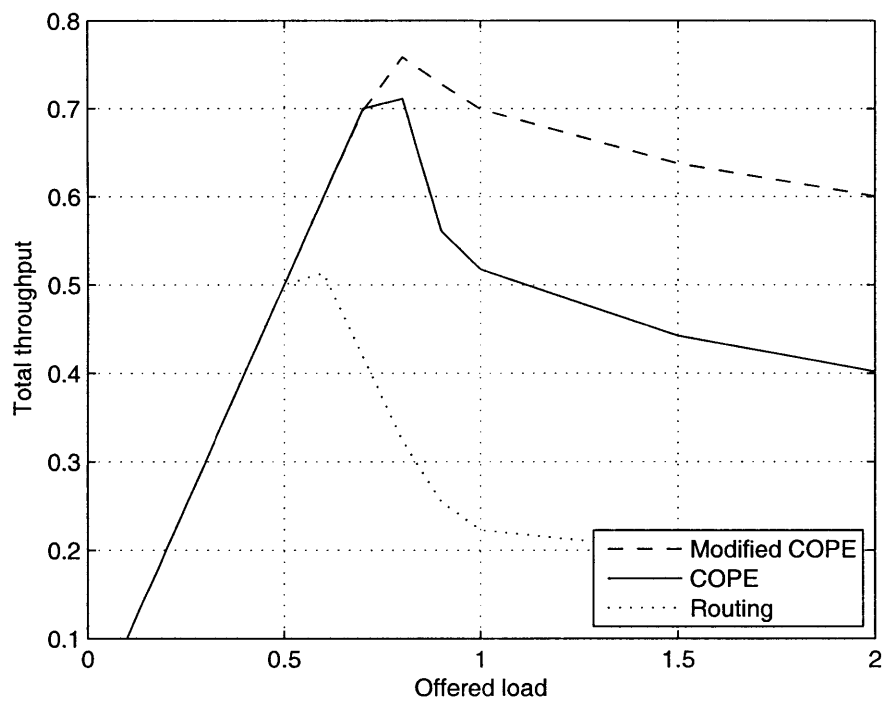


Figure 3-9: Throughput for COPE, modified COPE, and non-COPE systems in a cross network with cross traffic and traffic generated at the center node.

Chapter 4

Signatures for Content

Distribution with Network Coding

In this chapter, we turn our attention to the content distribution problem in peer-to-peer (P2P) networks, where network coding can be used to improve the distribution speed of large files.

4.1 Background

4.1.1 Network coding in P2P networks

After the introduction of network coding, several researchers explored the use of it in content distribution and distributed storage systems [3, 17]. Traditionally, the solutions for content distribution are based on a client-server model, where a central server sends the entire file to each client that requests it. This kind of approach becomes inefficient when the file size is large or when there are many clients, as it takes up a large amount of bandwidth and server resources. In recent years, P2P networks have emerged as an alternative to traditional content distribution solutions to deliver large files. A P2P network has a fully distributed architecture, and the peers in the network form a cooperative network that shares the resources, such as storage, CPU, and bandwidth, of all the computers in the network. This architecture

offers a cost-effective and scalable way to distribute software updates, videos, and other large files to a large number of users.

The best example of a P2P cooperative architecture is the BitTorrent system [1], which splits large files into small blocks, and after a node downloads a block from the original server or from another peer, it becomes a server for that particular block. Although BitTorrent has become extremely popular for distribution of large files over the Internet, it may suffer from a number of inefficiencies which decrease its overall performance. For example, scheduling is a key problem in BitTorrent: it is difficult to efficiently select which block(s) to download first and from where. If a rare block is only found on peers with slow connections, this would create a bottleneck for all the downloaders. Several *ad hoc* strategies are used in BitTorrent to ensure that different blocks are equally spread in the system as the system evolves. References [3, 17] propose the use of network coding to increase the efficiency of content distribution in a P2P cooperative architecture. The main idea of this approach is the following (see Fig. 4-1). The server breaks the file to be distributed into small blocks, and whenever a peer requests a file, the server sends a random linear combination of all the blocks. As in BitTorrent, a peer acts as a server to the blocks it has obtained. However, in a linear coding scheme, any output from a peer node is also a random linear combination of all the blocks it has already received. A peer node can reconstruct the whole file when it has received enough degrees of freedom to decode all the blocks. This scheme is completely distributed, and eliminates the need for a scheduler, as any block transmitted contains partial information of all the blocks that the sender possesses.

Several authors have evaluated the performance of network coding in P2P networks. Gkantsidis *et al.* [17] propose a scheme for content distribution of large files in which nodes make forwarding decisions solely based on local information. This scheme improves the expected file download time and the robustness of the system. Reference [3] compare the performance of network coding with traditional coding measures in a distributed storage setting with very limited storage space with the goal of minimizing the number of storage locations a file-downloader connects to, to

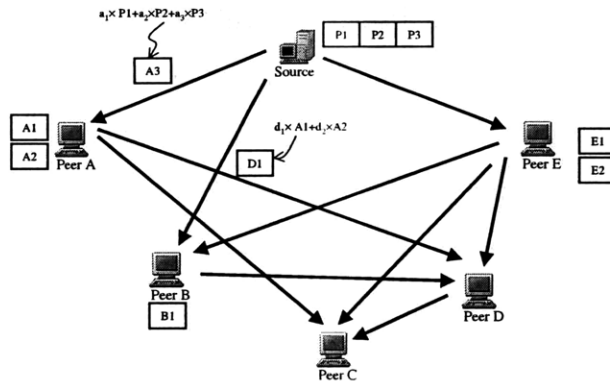


Figure 4-1: Content distribution with network coding. Assume the file being distributed is broken into three blocks, P_1 , P_2 , and P_3 . Any packet being transmitted is a random linear combination of all the blocks the sender has. For example, the packet sent from the source to peer A is a combination of P_1 , P_2 , and P_3 , whereas the packet sent from peer A to D is a combination of blocks A_1 and A_2 . A peer is able to decode the whole file when it receives 3 linearly independent blocks.

retrieve a file. They show that RLNC performs well without the need for a large amount of additional storage space. Dimakis *et al.* [15] introduce a graph-theoretic framework for P2P distributed system, and show that RLNC minimizes the required bandwidth to maintain the distributed storage architectures.

A major concern for any network coding system is the protection against malicious nodes. Take the above content distribution system for example.

Despite their desirable properties, network coded P2P systems are particularly susceptible to *Byzantine attacks* [47, 11, 35] – the injection of corrupted packets into the information flow. Since network coding relies on mixing of packets, a single corrupted packet may easily corrupt the entire information flow [26, 18]. Furthermore, in P2P networks, there is typically no security control over the nodes that join the network and the blocks that they redistribute. If a node in the P2P network behaves maliciously, it can create a polluted block with valid coding coefficients, and then sends it out. Here, coding coefficients refer to the random linear coefficients used to generate this block. If there is no mechanism for a peer to check the integrity of a received block, a receiver of this polluted block would not be able to decode anything for the file at all, even if all the other blocks it has received are valid.

To make things worse, the receiver would mix this polluted block with other blocks and send them out to other peers, and the pollution can quickly propagate to the whole network. This makes coding based content distribution even more vulnerable than the traditional P2P networks, such as BitTorrent. Similar security problems arise in all systems that use network coding, such as multicast networks. Several attempts were made to address this problem. Several authors address these problems in network coded P2P networks, which we shall discuss in detail in Section 4.1. Most of the countermeasures can be divided into two main categories: (i) end-to-end error correction, and (ii) misbehavior detection, which can be carried out either packet by packet or in generation based fashion.

Motivated by these observations, we propose a new signature scheme that is not based on elliptic curves, and is designed specifically for random linear coded systems. In this scheme, we view all blocks of the file as vectors, as in any network coding scheme, and make use of the fact that all valid vectors transmitted in the network should belong to the subspace spanned by the original set of vectors from the file. We design a signature that can be used to easily check the membership of a received vector in the given subspace, and at the same time, it is hard for a node to generate a vector that is not in that subspace but passes the signature test. We show that this signature scheme is secure, and that the overhead for the scheme is negligible for large files.

4.1.2 Byzantine detection scheme for network coded systems

End-to-end error correction scheme

Several papers address the problem of Byzantine adversaries in network coded systems. One approach is to correct the injected errors using *network error correction* [55]. Reference [55] bounds the maximum achievable rate in an adversarial setting, and generalizes the Hamming, Gilbert-Varshamov, and Singleton bounds. Jaggi *et al.* [26] introduce the first distributed polynomial-time rate-optimal network codes that work in the presence of Byzantine nodes and is information-theoretically secure.

The adversarial nodes are viewed as a secondary source. The source judiciously adds redundancy to help the receivers distill out the source information from the received mixtures. Given an adversary who can eavesdrop on all links and jam z links, their algorithm achieves a rate of $C - 2z$, where C is the network capacity; given an adversary who can observe only z_e links and jam z links where $z_e < C - 2z$, the algorithm achieves a rate of $C - z$. These rates are the maximum achievable rate given the power of the adversary. This work is generalized in [32, 51].

Generation-based Byzantine detection scheme

Ho *et al.* [24] introduce an information-theoretic approach for detecting Byzantine adversaries, which only assumes that the adversary did not see all linear combinations received by the destinations. Their detection probability varies with the length of the hash, field size, and the amount of information unknown to the adversary. A polynomial hash is added to each packet in the generation. Once the destination node receives enough packets to decode a generation, it can probabilistically detect errors. The intuition behind this scheme is that if a packet is valid, then its data and hash are consistent with its coding vector; and a linear combination of valid packets is also valid. This generation based scheme is very cheap and sensitive. For example, with 2% overhead ($k = 50$), $\log q = 7$, $s = 5$, the detection probability is at least 98.9%. Furthermore, this scheme does not require Public Key Infrastructure (PKI). However, this is a block code; therefore, will require a priori decision on the rate. In addition, the detection can only occur at a node with enough packets from a generation – thus, can incur large delays.

Packet-based Byzantine detection scheme

There are several signature schemes that have been presented in the literature. For instance, [3, 34, 56] use homomorphic hash functions to detect contaminated packets. Reference [18] suggests the use of a Secure Random Checksum (SRC) which requires less computation than the homomorphic hash function, but requires a secure channel to transmit the SRCs. In addition, [12] proposes a signature scheme for network

coding based on Weil pairing on elliptic curves. The signature scheme that we propose in this chapter is a packet-based detection scheme.

4.2 Problem Setup

In this section, we introduce the framework for a random linear coding based content distribution system. This framework can also be easily modified to be used for distributed storage systems. We model the network by a directed graph $G_d = (N, A)$, where N is the set of nodes, and A is the set of communication links. A source node $s \in N$ wishes to send a large file to a set of client nodes, $T \subset N$. In this chapter, we refer to all the clients as *peers*. The large file is divided into m blocks, and any peer receives different blocks from the source node or from other peers. In this framework, a peer is also a server to blocks it has downloaded, and always sends out random linear combinations of all the blocks it has obtained so far to other peers. When a peer has received enough degrees of freedom to decode the data, i.e., it has received m linearly independent blocks, it can re-construct the whole file.

Specifically, we view the m blocks of the file, $\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_m$, as elements in n -dimensional vector space \mathbb{F}_p^n , where p is a prime. The source node augments these vectors to create vectors $\mathbf{v}_1, \dots, \mathbf{v}_m$, given by

$$\mathbf{v}_i = (0, \dots, 1, \dots, 0, \bar{v}_{i1}, \dots, \bar{v}_{in}),$$

where the first m elements are zero except that the i th one is 1, and $\bar{v}_{ij} \in \mathbb{F}_p$ is the j th element in $\bar{\mathbf{v}}_i$. Packets received by the peers are linear combinations of the augmented vectors,

$$\mathbf{w} = \sum_{i=1}^m \beta_i \mathbf{v}_i,$$

where β_i is the weight of \mathbf{v}_i in \mathbf{w} . We see that the additional m elements in the front of the augmented vector keeps track of the β values of the corresponding packet, i.e.,

$$\mathbf{w} = (\beta_1, \dots, \beta_m, \bar{w}_{i1}, \dots, \bar{w}_{in}),$$

where $(\bar{w}_{i1}, \dots, \bar{w}_{in})$ is the payload part of the packet, and $(\beta_1, \dots, \beta_m)$ is the code vector that is used to decode the packets.

As mentioned in the previous section, this kind of network coding scheme is vulnerable to pollution attacks by malicious nodes [14, 40], and the pollution can quickly spread to other parts of the network if the peer just unwittingly mixes this polluted packet into its outgoing packets. Unlike uncoded systems where the source knows all the blocks being transmitted in the network, and therefore, can sign each one of them, in a coded system, each peer produces “new” packets, and standard digital signature schemes do not apply here. In the next section, we introduce a novel signature scheme for the coded system.

4.3 Signature scheme for network coding

We note that the vectors $\mathbf{v}_1, \dots, \mathbf{v}_m$ span a subspace V of \mathbb{F}_p^{m+n} , and a received vector \mathbf{w} is a valid linear combination of vectors $\mathbf{v}_1, \dots, \mathbf{v}_m$ if and only if it belongs to the subspace V . This is the key observation for our signature scheme. In the scheme described below, we present a system that is based upon standard modulo arithmetic (in particular the hardness of the Discrete Logarithm problem) and upon an invariant signature $\sigma(V)$ for the linear span V . Each node verifies the integrity of a received vector \mathbf{w} by checking the membership of \mathbf{w} in V based on the signature $\sigma(V)$.

Our signature scheme is defined by the following ingredients, which are independent of the file(s) to be distributed:

- q : a large prime number such that p is a divisor of $q - 1$. Note that standard techniques, such as that used in Digital Signature Algorithm (DSA), apply to find such q .
- g : a generator of the group G of order p in \mathbb{F}_q . Since the order of the multiplicative group \mathbb{F}_q^* is $q - 1$, which is a multiple of p , we can always find a subgroup, G , with order p in \mathbb{F}_q^* .
- Private key: $\mathbf{K}_{pr} = \{\alpha_i\}_{i=1, \dots, m+n}$, a random set of elements in \mathbb{F}_p^* . \mathbf{K}_{pr} is only

known to the source.

- Public key: $\mathbf{K}_{pu} = \{h_i = g^{\alpha_i}\}_{i=1, \dots, m+n}$. \mathbf{K}_{pu} is signed by some standard signature scheme, e.g., DSA, and published by the source.

To distribute a file in a secure manner, the signature scheme works as follows.

1. Using the vectors $\mathbf{v}_1, \dots, \mathbf{v}_m$ from the file, the source finds a vector $\mathbf{u} = (u_1, \dots, u_{m+n}) \in \mathbb{F}_p^{m+n}$ orthogonal to all vectors in V . Specifically, the source finds a non-zero solution, \mathbf{u} , to the equations

$$\mathbf{v}_i \cdot \mathbf{u} = 0, \quad i = 1, \dots, m.$$

2. The source computes vector $\mathbf{x} = (u_1/\alpha_1, u_2/\alpha_2, \dots, u_{m+n}/\alpha_{m+n})$.
3. The source signs \mathbf{x} with some standard signature scheme and publishes \mathbf{x} . We refer to the vector \mathbf{x} as the signature, $\sigma(V)$, of the file being distributed.
4. The client node verifies that \mathbf{x} is signed by the source.
5. When a node receives a vector \mathbf{w} and wants to verify that \mathbf{w} is in V , it computes

$$d = \prod_{i=1}^{m+n} h_i^{x_i w_i},$$

and verifies that $d = 1$.

To see that d is equal to 1 for any valid \mathbf{w} , we have

$$\begin{aligned}
d &= \prod_{i=1}^{m+n} h_i^{x_i w_i} \\
&= \prod_{i=1}^{m+n} (g^{\alpha_i})^{u_i w_i / \alpha_i} \\
&= \prod_{i=1}^{m+n} g^{u_i w_i} \\
&= g^{\sum_{i=1}^{m+n} (u_i w_i)} \\
&= 1,
\end{aligned}$$

where the last equality comes from the fact that \mathbf{u} is orthogonal to all vectors in V .

Next, we show that the system described above is secure. In essence, the theorem below shows that given a set of vectors that satisfy the signature verification criterion, it is provably as hard as the Discrete Logarithm problem to find new vectors that also satisfy the verification criterion other than those that are in the linear span of the vectors already known.

Definition 1. Let p be a prime number and G be a multiplicative cyclic group of order p . Let k and n be two integers such that $k < n$, and $\Gamma = \{h_1, \dots, h_n\}$ be a set of generators of G . Given a linear subspace, V , of rank k in \mathbb{F}_p^n such that for every $\mathbf{v} \in V$, the equality $\Gamma^{\mathbf{v}} \triangleq \prod_{i=1}^n h_i^{v_i} = 1$ holds, we define the (p, k, n) -Diffie-Hellman problem as the problem of finding a vector $\mathbf{w} \in \mathbb{F}_p^n$ with $\Gamma^{\mathbf{w}} = 1$ but $\mathbf{w} \notin V$.

By this definition, the problem of finding an invalid vector that satisfies our signature verification criterion is a $(p, m, m+n)$ -Diffie-Hellman problem. Note that in general, the $(p, n-1, n)$ -Diffie-Hellman problem has no solution. This is because if V has rank $n-1$ and a \mathbf{w}' exists such that $\Gamma^{\mathbf{w}'} = 1$ and $\mathbf{w}' \notin V$, then $\mathbf{w}' + V$ spans the whole space, and any vector $\mathbf{w} \in \mathbb{F}_p^n$ would satisfy $\Gamma^{\mathbf{w}} = 1$. This is clearly not true, therefore, no such \mathbf{w}' exists.

Theorem 1. For any $k < n-1$, the (p, k, n) -Diffie-Hellman problem is at least as hard as the Discrete Logarithm problem.

Proof. Assume that we have an efficient algorithm to solve the (p, k, n) -Diffie-Hellman

problem, and we wish to compute the discrete algorithm $\log_g(z)$ for some $z = g^x$, where g is a generator of a cyclic group G with order p . We can choose two random vectors $\mathbf{r} = (r_1, \dots, r_n)$ and $\mathbf{s} = (s_1, \dots, s_n)$ in \mathbb{F}_p^n , and construct $\Gamma = \{h_1, \dots, h_n\}$, where $h_i = z^{r_i} g^{s_i}$ for $i = 1, \dots, n$. We then find k linearly independent (and otherwise random) solution vectors $\mathbf{v}_1, \dots, \mathbf{v}_k$ to the equations

$$\mathbf{v} \cdot \mathbf{r} = 0 \text{ and } \mathbf{v} \cdot \mathbf{s} = 0.$$

Note that there exist $n - 2$ linearly independent solutions to the above equations. Let V be the linear span of $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$, it is clear that any vector $\mathbf{v} \in V$ satisfies $\Gamma^{\mathbf{v}} = 1$. Now, if we have an algorithm for the (p, k, n) -Diffie-Hellman problem, we can find a vector $\mathbf{w} \notin V$ such that $\Gamma^{\mathbf{w}} = 1$. This vector would satisfy $\mathbf{w} \cdot (x\mathbf{r} + \mathbf{s}) = 0$. Since \mathbf{r} is statistically independent from $(x\mathbf{r} + \mathbf{s})$, with probability greater than $1 - 1/p$, we have $\mathbf{w} \cdot \mathbf{r} \neq 0$. In this case, we can compute

$$\log_g(z) = x = \frac{\mathbf{w} \cdot \mathbf{s}}{\mathbf{w} \cdot \mathbf{r}}.$$

This means the ability to solve the (p, k, n) -Diffie-Hellman problem implies the ability to solve the Discrete Logarithm problem. \square

This proof is an adaptation of a proof that appeared in an earlier publication by Boneh *et. al* [8].

4.4 Discussion

Our signature scheme nicely makes use of the linearity property of random linear network coding, and enables the peers to check the integrity of packets without the requirement for a secure channel, as in the case of hash function or SRC schemes [3, 18, 34]. Also, the computation involved in the signature generation and verification processes is very simple.

Next, we examine the overhead incurred by this signature scheme. Let the file

size be M and let the file be divided into m blocks, each one of which is a vector in \mathbb{F}_p^n . The size of each block is $B = n \log(p)$ and we have $M = mn \log(p)$. The size of each augmented vector (with coding vectors in the front) is $B_a = (m + n) \log(p)$, and thus, the overhead of the coding vector is m/n times the file size. Note that this is the overhead pertaining to the linear coding scheme, not to our signature scheme, and any practical network coding system would make $m \ll n$. The initial setup of our signature scheme involves the publishing of the public key, \mathbf{K}_{pu} , which has size $(m + n) \log(q)$. In typical cryptographic applications, the size of p is 20 bytes (160 bits), and the size of q is 128 bytes (1024 bits), thus, the size of \mathbf{K}_{pu} is approximately equal to $6(m + n)/mn$ times the file size.

For distribution of each file, the incremental overhead of our scheme consists of two parts: the public data, \mathbf{K}_{pu} , and the signature vector, \mathbf{x} .

For the public key, \mathbf{K}_{pu} , we note that it cannot be fully reused for multiple files, as it is possible for a malicious node to generate a invalid vector that satisfies the check $d = 1$ using information obtained from previously downloaded files. Specifically, let \mathbf{x}_1 be the signature of File 1, and \mathbf{w}_1 be a valid received vector for File 1, we have

$$d = \prod_{i=1}^{m+n} h_i^{x_{1i} w_{1i}} = 1.$$

If the source then distribute File 2 using the same public key, \mathbf{K}_{pu} , and a different signature, \mathbf{x}_2 , a malicious node can construct a vector \mathbf{w}_2 , where $w_{2i} = x_{1i} w_{1i} / x_{2i}$, which satisfies the signature check

$$d = \prod_{i=1}^{m+n} h_i^{x_{2i} w_{2i}} = \prod_{i=1}^{m+n} h_i^{x_{1i} w_{1i}} = 1.$$

However, \mathbf{w}_2 is not a valid linear combination of the vectors of File 2. To prevent this from happening, we can publish a public key for each file, and as mentioned above, the overhead is about $6(m + n)/mn$ times the file size, which is small as long as $6 \ll m \ll n$. Note that if we republish \mathbf{K}_{pu} for every new file, we can reuse the signature vector \mathbf{x} . Let \mathbf{u}_2 be a vector that is orthogonal to all vectors in File 2, the

source can compute a new private key, $\mathbf{K}_{pr} = \{\alpha_1, \dots, \alpha_{m+n}\}$, given by

$$\alpha_i = u_{2i}/x_i, \quad i = 1, \dots, m+n.$$

The source then publishes the new public key, $\mathbf{K}_{pu} = \{h_i = g^{\alpha_i}\}_{i=1, \dots, m+n}$. In this way, we do not need to publish new \mathbf{x} vectors for the subsequent files.

Alternatively, for every new file, we can randomly pick an integer i between 1 and $m+n$, select a new random value for α_i in the private key, and publish the new $h_i = g^{\alpha_i}$. The overhead for this method is $(m+n)$ times smaller than that described in the previous paragraph, i.e., this overhead is only $6/mn$ times the file size. As an example, if we have a file of size 10MB, divided into $m = 100$ blocks, the value of n would be in the order of thousands, and thus, this overhead is less than 0.01% of the file size. This method should provide good security except in the case where we expect the vector \mathbf{w} to have low variability, for example, has many zeros. Security can be increased by changing more elements in the private key for each new file.

However, if we only change one element in the public key, for each new file distributed, we also have to publish a new signature \mathbf{x} , which is computed from a vector \mathbf{u} that is orthogonal to the subspace V spanned by the file. Since the V has dimension m , it is sufficient to only replace m elements in \mathbf{u} to generate a vector orthogonal to the new file. Since the first m elements in the vectors $\mathbf{v}_1, \dots, \mathbf{v}_m$ are always linearly independent (they are the code vectors), it suffices to just modify the entries u_1 to u_m . Assume that the i th element in the private key is the only one that has been changed for the distribution of the new file, and that i is between 1 and m , then we only need to publish x_1 to x_m for the new signature vector. This part of the overhead has size $m \log(p)$, and the ratio between this overhead and the original file size N is $1/n$. Again, take a 10MB file for example, this overhead is less than 0.1% of the file size.

Therefore, after the initial setup, each additional file distributed only incurs a negligible amount of overhead using our signature scheme.

Reference [30] analyzes the overhead in terms of bandwidth associated with our

signature scheme, and compare it to that of various Byzantine detection schemes. It is shown that our scheme is the most bandwidth efficient if the probability of attack is high.

Finally, we would like to point out that, under our assumptions that there is no secure side channel from the source to all the peers and that the public key is available to all the peers, our signature scheme has to be used on the original file vectors not on hash functions. This is because to maintain the security of the system, we need to use a one-way hash function that is homomorphic, however, we are not aware of any such hash function. Although [3] and [18] suggested usage of homomorphic hash functions for network coding, [3] assumed that the intermediate nodes do not know the parameters used for generating the hash function, and [18] assumed that a secure channel is available to transmit the hash values of all the blocks from the source node to the peers. Under our more relaxed assumptions, these hash functions would not work.

Chapter 5

Conclusion and Future Work

This chapter summarizes the work presented in this thesis. Network coding offers a new paradigm for network communications, and at the same time, leads to many new networking problems that require distributed solutions. In this thesis, we focused on three aspects of distributed control of coded networks:

- **Subgraph optimization for multicast in coded networks**

Subgraph optimization is an important problem in performing multicast with network coding. We studied the algorithms that solve this optimization problem for both static and dynamic multicasts. For static multicast, we presented two distributed subgradient algorithms, Algorithms A and B, to find the min-cost subgraph, and examined their convergence rate. Using the special structure of the network coding problem, we showed that with appropriately chosen step sizes, the dual problem converges to a neighborhood of the optimal solution linearly. For Algorithm B, we showed that the convergence rate of the primal solutions is $O(1/n)$. On the other hand, for Algorithm A, since the physical flow variables are decoupled from the dual iterations, we can obtain a feasible primal solution in each iteration. We also proposed various heuristics for dual variable initialization and primal solution recovery to further improve the convergence performance. Simulation results show that the subgradient method produces significant reductions in multicast energy as compared to centralized routing

algorithms after just a few iterations. Moreover, the algorithm is robust to changes in the network and can converge to new optimal solutions quickly as long as the rate of change in the network is slow as compared to the speed of computation and transmission.

For dynamic multicasts, in order to characterize the disturbances to users caused by the changes in the multicast subgraph, we introduced the concepts of link rearrangement and code rearrangement. We proposed both nonrearrangeable and rearrangeable algorithms for the dynamic multicast problem, and used simulation results to show that the α -scaled algorithm can effectively bound the growth of the multicast cost without causing too many disturbances to existing users.

- **Analysis and improvement to COPE**

The second problem we studied is network coding for multiple unicast in mesh wireless networks. Specifically, we examined the COPE system that employs opportunistic coding to improve the network throughput. We showed that the outstanding performance of COPE stems from the interaction between network coding and the MAC protocol. A key factor here is the local fairness enforced by the MAC protocol among competing nodes. Based on these observations, we also proposed a simple modification to the COPE system that can further improve the throughput performance.

- **Signature scheme for content distribution**

Finally, we studied the content distribution system in P2P networks using network coding. Security problem is a main obstacle in the implementation of content distribution networks using random linear network coding. To tackle this problem, instead of trying to fit an existing signature scheme to network coding based systems, we proposed a new signature scheme that is made specifically for such systems. We introduced a signature vector for each file distributed, and the signature can be used to easily check the integrity of all the packets received for this file. We have shown that the proposed scheme is as hard as

the Discrete Logarithm problem, and the overhead of this scheme is negligible for a large file.

5.1 Future work

For future work, we would like to continue working on some of the above-mentioned problems. For instance, it would be interesting to implement our modification to COPE to see its effect in a real system. Also, we can extend our analysis method to other COPE-like systems, and provide theoretical analysis of their performances.

In addition, there are many other distributed network coding problems that are yet to be solved. One example is the multi-resolution multicast problem. Multi-resolution codes enable multicast at different rates to different receivers, a setup that is often desirable for graphics or video streaming. In a network coded system, we need a distributed algorithm to let the intermediate nodes know which layers they can code together such that the receivers can successfully decode. We propose a simple, distributed, two-stage message passing algorithm to generate network codes for single-source multicast of multi-resolution codes. Initial simulation results are promising, and directions for future work include introducing an additional stage in the algorithms to further improve the performance and modifying this algorithm to work in multi-source multicast.

Bibliography

- [1] Bittorrent file sharing protocol. <http://www.BitTorrent.com>.
- [2] The rocketfuel project. www.cs.washington.edu/research/networking/rocketfuel.
- [3] S. Acedański, S. Deb, M. Médard, and R. Koetter. How good is random linear coding based distributed network storage? In *Proc. 1st Workshop on Network Coding, Theory, and Applications (Netcod'05)*, April 2005.
- [4] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. Inf. Theory*, 46(4):1204–1216, July 2000.
- [5] D. Bertsekas and R. Gallager. *Data network*. Prentice Hall, 1992.
- [6] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1995.
- [7] K. Bharath-Kumar and J. M. Jaffe. Routing to multiple destinations in computer networks. *IEEE Trans. Commun.*, 31(3):343–351, March 1983.
- [8] D. Boneh and M. Franklin. An efficient public key traitor tracing scheme. In *Lecture Notes in Computer Science*, volume 1666, pages 338–353. Springer-Verlag, 1999.
- [9] J. V. Burke and M. C. Ferris. Weak sharp minima in mathematical programming. *SIAM J. Control Optim.*, 31(5):1340–1359, Sept. 1993.
- [10] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wirel. Commun. Mob. Comput.*, 2(5):483–502, Aug. 2002.
- [11] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *Symposium on Operating Systems Design and Implementation (OSDI)*, February 1999.
- [12] D. Charles, K. Jain, and K. Lauter. Signatures for network coding. In *Proceedings of Conference on Information Sciences and Systems*, March 2006.
- [13] P. A. Chou, Y. Wu, and K. Jain. Practical network coding. In *Proc. of the 41th Annual Allerton Conference on Communication, Control, and Computing*, October 2003.

- [14] N. Christin, A. Weigend, and J. Chuang. Content availability, pollution and poisoning in peer-to-peer file sharing networks. In *Proc. ACM E-Commerce Conference (EC'05)*, June 2005.
- [15] Alexandros G. Dimakis, P. Brighten Godfrey, Martin J. Wainwright, and Kannan Ramchandran. Network coding for distributed storage systems. In *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, May 2007.
- [16] C. Fragouli and E. Soljanin. *Network coding fundamentals*. Now Publishers, 2007.
- [17] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *Proc. IEEE Infocom*, March 2005.
- [18] Christos Gkantsidis and Pablo Rodriguez. Cooperative security for network coding file distribution. In *Proceedings of IEEE INFOCOM*, April 2006.
- [19] T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D Karger. A random linear network coding approach to multicast. *IEEE Trans. Inf. Theory*, 52(10):4413–4430, October 2006.
- [20] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *Proc. 2003 IEEE International Symposium on Information Theory (ISIT'03)*, Yokohama, Japan, June–July 2003.
- [21] T. Ho, B. Leong, M. Médard, R. Koetter, Y.-H. Chang, and M. Effros. On the utility of network coding in dynamic environments. In *Proc. 2004 International Workshop on Wireless Ad-hoc Networks (IWVAN'04)*, 2004.
- [22] T. Ho and D. Lun. *Network coding: an introduction*. Cambridge University Press, 2008.
- [23] T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger. On randomized network coding. In *Proc. of the 41th Annual Allerton Conference on Communication, Control, and Computing*, October 2003.
- [24] Tracey Ho, Ben Leong, Ralf Koetter, Muriel Médard, and Michelle Effros. Byzantine modification detection in multicast networks using randomized network coding. In *Proceedings of IEEE ISIT*, June 2004.
- [25] M. Imase and B. M. Waxman. Dynamic steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, August 1991.
- [26] S. Jaggi, M. Langberg, S. Katti, Tracey Ho, Dina Katabi, and Muriel Médard. Resilient network coding in the presence of byzantine adversaries. In *Proceedings of IEEE INFOCOM*, pages 616 – 624, March 2007.
- [27] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. M. G. M. Tolhuizen. Polynomial time algorithms for multicast network code construction. *IEEE Trans. Inf. Theory*, 5(6):1973–1982, June 2005.

- [28] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Médard. The importance of being opportunistic: practical network coding for wireless environments. In *Proc. 43th Annual Allerton Conference on Communication, Control, and Computing*, Oct. 2005.
- [29] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. Xors in the air: practical wireless network coding. In *Proc. ACM SIGCOMM '06*, 2006.
- [30] M. Kim, L. Lima, F. Zhao, J. Barros, M. Médard, R. Koetter, T. Kalker, and K. Han. On counteracting byzantine attacks in network coded peer-to-peer networks. *to appear in IEEE JSAC on Mission Critical Networks*, 2009.
- [31] K. C. Kiwiel, T. Larsson, and P. O. Lindberg. Lagrangian relaxation via ballstep subgradient methods. *Mathematics of Operations Research*, 32(3):669–686, Aug. 2007.
- [32] R. Koetter and F. Kschischang. Coding for errors and erasures in random network coding. *IEEE Transactions on Information Theory*, 54:3579–3591, 2008.
- [33] R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Trans. Netw.*, 11(5):782–795, October 2003.
- [34] Maxwell Krohn, Michael Freedman, and David Mazières. On-the-fly verification of rateless erasure codes for efficient content distribution. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2004.
- [35] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4:382–401, 1982.
- [36] T. Larsson, M. Patriksson, and A. Strömberg. Ergodic primal convergence in dual subgradient schemes for convex programming. *Mathematical Programming*, 86:283–312, 1999.
- [37] J. Le, J. Lui, and D. M. Chiu. How many packets can we encode? - an analysis of practical wireless network coding. In *Proc. IEEE Infocom*, April 2008.
- [38] S.-Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Trans. Inf. Theory*, 49(2):371–381, Feb. 2003.
- [39] Z. Li and B. Li. Network coding: the case of multiple unicast sessions. In *Proc. 44th Annual Allerton Conference on Communication, Control, and Computing*, Sept.–Oct. 2004.
- [40] X. Lou and K. Hwang. Prevention of index-poisoning ddos attacks in peer-to-peer file-sharing networks. *IEEE Trans. on Multimedia, Special Issue on Content Storage and Delivery in P2P Networks*, Nov. 2006.

- [41] S. Low and D. E. Lapsley. Optimization flow control, i: Basic algorithm and convergence. *IEEE/ACM Trans. Netw.*, 7(6):861–874, Dec. 1999.
- [42] D. S. Lun, M. Médard, and D. R. Karger. On the dynamic multicast problem for coded networks. In *Proceedings of WINMEE, RAWNET and NETCOD 2005 workshops*, April 2005.
- [43] D. S. Lun, N. Ratnakar, R. Koetter, M. Médard, E. Ahmed, and H. Lee. Achieving minimum cost multicast: A decentralized approach based on network coding. In *Proc. IEEE Infocom*, volume 3, pages 1607–1617, March 2005.
- [44] D. S. Lun, N. Ratnakar, M. Médard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao. Minimum-cost multicast over coded packet networks. *IEEE Trans. Inf. Theory*, 52(6):2608–2623, June 2006.
- [45] A. Nedić. *Subgradient methods for convex minimization*. PhD thesis, Massachusetts Institute of Technology, June 2002.
- [46] A. Nedić and A. Ozdaglar. Approximate primal solutions and rate analysis for dual subgradient methods. *to appear in SIAM Journal on Optimization*, 2009.
- [47] Ron Perlman. *Network layer protocols with Byzantine robustness*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, October 1988.
- [48] S. Raghavan, G. Manimaran, and C. Siva Ram Murthy. A rearrangeable algorithm for the construction delay-constrained dynamic multicast trees. *IEEE/ACM Trans. Netw.*, 7(4):514–529, August 1999.
- [49] S. Sengupta, S. Rayanchu, and S. Banerjee. An analysis of wireless network coding for unicast sessions: the case for coding-aware routing. In *Proc. IEEE Infocom*, May 2007.
- [50] H. D. Sherali and G. Choi. Recovery of primal solutions when using subgradient optimization methods to solve lagrangian duals of linear programs. *Oper. Res. Lett.*, 19:105–113, 1996.
- [51] D. Silva and F. Kschischang. Adversarial error correction for network coding: Models and metrics. In *Proceedings of Annual Allerton Conf. on Commun., Control, and Computing*, Monticello, IL, September 2008.
- [52] B. M. Waxman. Routing of multicast connections. *IEEE J. Sel. Areas Commun.*, 6(9):1617–1622, Dec. 1988.
- [53] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. Energy-efficient broadcast and multicast trees in wireless networks. *Mobile Networks and Applications*, 7:481–492, 2002.
- [54] Y. Wu, P. A. Chou, and S.-Y. Kung. Minimum-energy multicast in mobile ad hoc networks using network coding. *IEEE Trans. Commun.*, 53(11):1906–1918, November 2005.

- [55] Raymond W. Yeung and Ning Cai. Network error correction. *Communications in Information and Systems*, 6(1):19–54, 2006.
- [56] Zhen Yu, Yawen Wei, B. Ramkumar, and Yong Guan. An efficient signature-based scheme for securing network coding against pollution attacks. In *Proceedings of IEEE INFOCOM*, Pheonix, AZ, April 2008.
- [57] F. Zhao, T. Kalker, M. Médard, and K. Han. Signatures for content distribution with network coding. In *Proc. 2006 IEEE International Symposium on Information Theory (ISIT'07)*, Nice, France, June 2007.
- [58] F. Zhao, D. S. Lun, M. Médard, and E. Ahmed. Decentralized algorithms for operating coded wireless networks. In *Proc. IEEE Information Theory Workshop 2007*, Lake Tahoe, CA, Sept. 2007.
- [59] F. Zhao and M. Médard. Online network coding for the dynamic multicast problem. In *Proc. 2006 IEEE International Symposium on Information Theory (ISIT'06)*, Seattle, July 2006.
- [60] F. Zhao, M. Médard, D. Lun, and A. Ozdaglar. Convergence study of decentralized min-cost subgraph algorithms for multicast in coded networks. *submitted to IEEE Trans. Information Theory*, Dec. 2008.
- [61] F. Zhao, M. Médard, D. Lun, and A. Ozdaglar. Minimum-cost subgraph algorithms for static and dynamic multicasts with network coding. *New Directions in Wireless Communications Research*, Sept. 2009.