Practical Algorithms for Distributed Network Control

by

Nathaniel Matthew Jones

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2013

Author .....
Department of Aeronautics and Astronautics
August 14, 2013

Certified by .....
Eytan Modiano
Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by .....
Brooke Shrader
Technical Staff, MIT Lincoln Laboratory
Thesis Supervisor

Certified by .....
Moe Win
Professor of Aeronautics and Astronautics
Thesis Committee Member

Accepted by .....
Eytan Modiano
Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Practical Algorithms for Distributed Network Control

by

Nathaniel Matthew Jones

Submitted to the Department of Aeronautics and Astronautics
on August 14, 2013, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Optimal routing and scheduling algorithms have been studied for decades, however several practical issues prevent the adoption of these network control policies on the Internet. This thesis considers two distinct topics in distributed network control: (i) maximizing throughput in wireless networks using network coding, and (ii) deploying controllable nodes in legacy networks.

Network coding is a relatively new technique that allows for an increase in throughput under certain topological and routing conditions. The first part of this thesis considers jointly optimal routing, scheduling, and network coding strategies to maximize throughput in wireless networks. We introduce a simple network coding strategy and fully characterize the region of arrival rates supported. We propose a centralized dynamic control policy for routing, scheduling, and our network coding strategy, and prove this policy to be throughput optimal subject to our coding constraint. We further propose a distributed control policy based on random access that optimizes for routing, scheduling, and pairwise coding, where pairwise coding captures most of the coding opportunities on random topologies. We prove this second policy to also be throughput optimal subject to the coding constraint. Finally, we reduce the gap between theory and practice by identifying and solving several problems that may occur in system implementations of these policies.

Throughput optimal policies typically require every device in the network to make dynamic routing decisions. In the second part of this thesis, we propose an overlay routing architecture such that only a subset of devices (overlay nodes) need to make dynamic routing decisions, and yet maximum throughput can still be achieved. We begin by formulating an optimization problem that searches for the minimum overlay node placement that achieves maximum throughput. We devise an efficient placement algorithm which solves this problem optimally for networks not subject to interference constraints. Then we propose a heuristic control policy for use at overlay nodes, and show by simulation that this policy performs optimally in all studied scenarios.

Thesis Supervisor: Eytan Modiano
Title: Professor of Aeronautics and Astronautics

Thesis Supervisor: Brooke Shrader
Title: Technical Staff, MIT Lincoln Laboratory

Thesis Committee Member: Moe Win
Title: Professor of Aeronautics and Astronautics

Acknowledgments

I would first and foremost like to thank my advisor, Eytan Modiano, who has patiently met with me on a weekly basis since I started this program four years ago. Having an advisor who takes a keen interest in the work and lives of his students is an asset that not every graduate student can claim to have. I have been lucky to work with someone so gifted and dedicated for the past four years.

I am also very fortunate to have worked with Brooke Shrader, my co-advisor from Lincoln Laboratory. Brooke traveled between Lincoln Lab and the MIT campus countless times to help me with my research. Her ability to help me convert intuition into mathematical formulations has made me a better researcher, and I am excited to return to work with her at Lincoln Lab.

Many thanks also to Moe Win, my committee member who has supported me enthusiastically at many key crossroads in this program. He was there for my qualifying exams, my proposal defense, and now my thesis defense. I appreciate his flexibility, kindness, and support of my research.

Georgios Paschos, one of my thesis readers, came to MIT later in my career here, but his help with my research has been immeasurable. He has become a friend, an unofficial mentor, and a collaborator. I also appreciate his many failed attempts at trying to convince me to like coffee.

Thanks to Hulya Seferoglu for being a thesis reader and for sharing advice on the practical applications of my research.

Thanks also to Chih-ping Li for always being willing to help me when I was stuck on proofs.

My time here would not have been nearly so much fun were it not for Greg Kuperman, Seb Neumayer, and Matt Johnston, who somehow made our office below the Gas Turbine Lab a fun place to be with discussions of critical topics, like locating free soda and light bulb failure rates.

Sincere thanks to all the other post-docs and fellow students in CNRG for being such a talented and friendly cohort.

Thanks to my mentor in the Lincoln Scholars Program, Andrew Worthen, for frequent advice and encouragement on my academic progress, and necessary distractions on diverse topics ranging from plotting styles to gardening tips.

I wouldn't be here without the support of Lincoln Laboratory, specifically everyone in Group 63 who have been with me since I moved to the Boston area, all the support staff who helped me with various logistical issues, and the Lincoln Scholars Program, which made my return to grad school possible.

And of course, love and thanks to my family. Thanks to my parents, Jana and Rudy, for their inspiration and encouragement. Thanks to my sister and brother-in-law, Katie and Scott, for always offering a bedroom in Georgia and a cold beer in the fridge. And thanks to my sister Rachel and brother Josh for cheering me along through four years of grad school.

I thank my best friend and fiancée, Carrie Klugman, whose unrelenting support, patience, and laughter has kept me going, ever since the incident with a tomato in the elevator.

Thanks to Julie and Craig Klugman for their many, many delicious snack deliveries, and to Josh Klugman for always finding humor in academia.

This work is sponsored by the Department of the Air Force and the Assistant Secretary of Defense for Research and Engineering under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

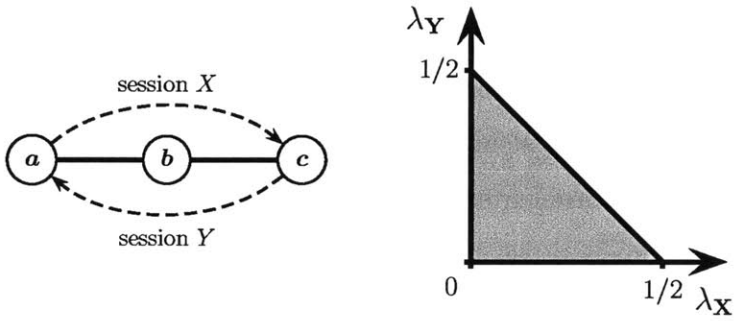
There is an ongoing proliferation of wireless networks, out of convenience and necessity. However, existing networks are often inefficient at supporting traffic demands due to poor mitigation of wireless interference and suboptimal choice of routes. Network expansion is extremely expensive due to a scarce frequency spectrum and high costs of deploying wired infrastructure. This motivates our study of network control policies that maximize throughput.

There have been many recent innovations in data networks with notable advances in scheduling, routing, and network coding. The pioneering work on network control [40] provides a joint routing and scheduling policy that maximizes the region of sustainable arrival rates versus all other routing and scheduling policies. Network coding [1] is a relatively new technique that allows for an increase in throughput under certain topological and routing conditions. Despite a wealth of rich theory, a deficiency of practical implementations limits the adoption of these advanced techniques.

This thesis moves towards practical implementation on two fronts. In Chapters 2 and 3, we consider the interactions between scheduling, routing, and network coding, and propose dynamic policies that maximize the region of sustainable throughput vectors in wireless networks. In Chapter 4, we consider integration with legacy networks by proposing an overlay approach where throughput can be maximized when optimal control policies are applied at only a subset of nodes in the network.

1.1 Background

In deployed wireless networks, throughput is commonly limited due to inefficient routing and scheduling algorithms. For example, consider the scheduling algorithm used in IEEE 802.11 protocols that attempts to give all nodes equal access to the channel. This equal access is undesirable for well-connected nodes that serve as relays for a large fraction of traffic in the network. Also, commonly used routing protocols such as OSPF try to identify the best *single-path* routes between sources and destinations, ignoring diverse multiple-path routing opportunities that can increase throughput and reduce congestion. The research area known as network control considers routing and scheduling policies that overcome these deficiencies by using multiple-path routes and choosing efficiently packed transmission schedules.



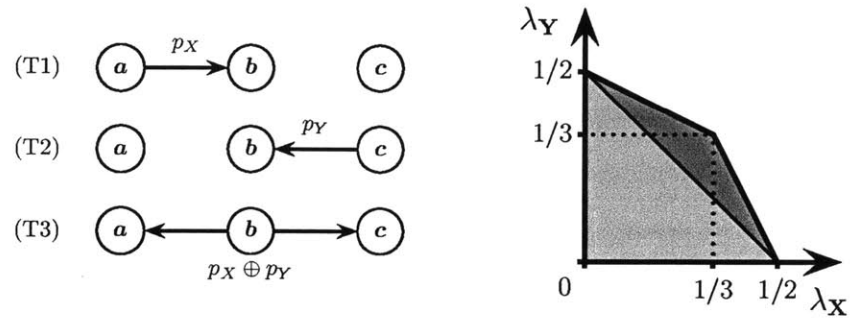
(a) Scenario (b) Stability Region

Figure 1-1: The wireless 2-way relay scenario and associated stability region. (a) Nodes a and c can communicate only through the relay at node b. Dashed arrows indicate traffic sessions. (b) Stability region for traffic sessions X and Y, assuming at most one node can transmit at a time.

The *stability region* is the set of all arrival rates that can be supported by a network, and is the largest region that any control policy can achieve. We give an example of a stability region for the wireless 2-way relay scenario shown in Figure 1-1a. Here nodes a and c would like to exchange traffic, but can only communicate via the relay at node b. For this wireless scenario, we assume interference constraints such that at most one node can transmit at a time. Session X carries traffic from node a to node c with an arrival rate of λ_X, while session Y carries traffic from c to a with arrival rate λ_Y. Either session can individually support an arrival rate of 1/2,

as it takes 2 transmissions to deliver each packet. Thus the network can support arrival rates (λ_X, λ_Y) of $(0, 1/2)$ and $(1/2, 0)$. Taking the convex hull of these two extreme points and the origin gives the stability region[1] shown as a gray triangle in Figure 1-1b.

Network coding is a coding technique that allows for increased throughput by encoding packets at intermediate nodes in the network. With network coding, data can be exchanged in fewer transmissions by strategically combining packets such that each recipient has previously seen some portion of the encoded set.



(a) Pairwise Coding (b) Stability Region

Figure 1-2: Network coding on wireless 2-way relay scenario from Figure 1-1. (a) Packets p_X and p_Y exchanged in 3 transmissions using pairwise coding. (b) Pairwise coding adds the area of the dark gray triangle to the non-coding stability region from Figure 1-1b.

A motivating example for wireless network coding is shown in Figure 1-2, where we again consider the wireless 2-way relay scenario. In Figure 1-2a, we would like to exchange one packet for each session. With network coding, these packets can be exchanged in only 3 transmissions: (T1) send packet p_X from node a to node b; (T2) send packet p_Y from node c to node b; and (T3) send coded packet $p_X \oplus p_Y$ as a binary XOR combination of the two packets from node b to both a and c simultaneously via a single wireless multicast transmission. Node a knows the value of packet p_X, so it can therefore recover packet p_Y. Likewise, node b can recover packet p_X. Without network coding, this same packet exchange takes 4 transmissions, so in this example

[1]The full stability region of this network is a 6 dimensional polytope accounting for all combinations of traffic sessions between nodes a, b, and c. For ease of exposition, we show a 2 dimensional slice of this region where only sessions X and Y are active.

network coding has increased throughput by a factor of 4/3. This example shows that network coding can support a symmetric arrival rate of $(1/3, 1/3)$, increasing the stability region from the gray triangle shown in Figure 1-1b to also include the area of the dark gray triangle in Figure 1-2b.

We would like to exploit network coding opportunities to increase throughput in wireless networks. However, naive application of network coding can actually reduce throughput by increasing interference and causing traffic to flow on undesirable paths. Therefore we wish to find strategies that seek out beneficial coding opportunities in the network while deferring to uncoded transmissions when no beneficial coding opportunities exist. This requires a routing and scheduling strategy with knowledge of network coding, as optimal routes and schedules may change when network coding is considered.

In the first part of this thesis, we jointly optimize for routing, scheduling, and network coding in wireless networks. We consider online policies that make dynamic decisions based on queue state information without requiring knowledge of the exogenous arrival rates. We develop both centralized and distributed policies, and compare their performance. We will prove that our policies are throughput optimal subject to our coding constraint, in that the policies maximize the region of arrival rates supported by the network.

In the second part of this thesis, we consider the use of throughput optimal control policies in legacy networks. We would like to enable network control policies to be deployed in existing networks, alongside legacy nodes that are unaware of our control policies. However, policies based on differential backlog routing typically require a homogeneous network where all nodes participate in the network control decisions. A problem with heterogeneous networks containing a mixture of controllable and legacy nodes is that legacy nodes might not be able to forward all traffic that they receive, creating a black hole route[2]. The legacy nodes have no means to communicate congestion information with controllable nodes, thus congested routes can attract

[2]Such nodes are sometimes called trapping nodes, and usually are explicitly assumed to not exist on backpressure networks.

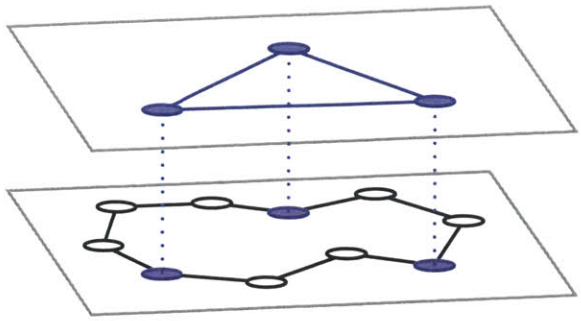


Figure 1-3: Example of a network overlay. The bottom plane shows the full network graph, while the top plane shows a subset of network nodes and their conceptual overlay connectivity.

more traffic and can cause the network queues to grow without bound.

Network overlays are frequently used to deploy new communication architectures in legacy networks [31]. To accomplish this, messages from the new technology are encapsulated in the legacy format, allowing the two methods to coexist in the legacy network. Nodes making use of the new communication methods are then connected in a conceptual network overlay that operates on top of the legacy network, as shown in Figure 1-3. The most predominant example of a network overlay is the Internet, which was previously connected as a network overlay on top of the public telephone networks (e.g., via dial-up modems). Lately, this situation has reversed such that telephone communications now largely operate as a network overlay on top of the Internet (e.g., via Voice-over-IP).

We model our heterogeneous mixture of controllable and legacy nodes such that controllable nodes are deployed in a network overlay operating on top of the legacy network. We consider two complementary parts to this problem: (1) we develop algorithms to choose which legacy nodes to replace with controllable nodes for maximizing throughput, and (2) we design a backpressure-based network control policy to be used in the network overlay setting. We then evaluate our backpressure overlay policy via simulation, and observe that maximum throughput can be attained when only a fraction of legacy nodes are replaced with controllable nodes.

1.2 Related Work

The first part of our research is at the intersection of network coding and network control. While there are many related works on each of these individual research topics, there is little existing work on the combined area of routing, scheduling, and network coding. We review literature on network coding in Section 1.2.1, routing and scheduling in Section 1.2.2, and joint routing, scheduling, and network coding in Section 1.2.3.

The second part of our research considers the use of network control policies in heterogeneous networks with a mixture of legacy and fully controllable nodes. Our approach will use a modified backpressure policy in a network overlay model. In Section 1.2.4, we review literature on modified backpressure policies that are related to ours, and on the use of network overlays to incorporate advanced routing techniques into legacy networks.

1.2.1 Network Coding

Originally introduced in 2000 by Ahlswede, Cai, Li, and Yeung [1], *network coding* can increase network throughput by allowing intermediate nodes to combine or encode received data rather than simply forwarding it. The benefit of this approach for wireless transmissions was clearly demonstrated by COPE [14], an opportunistic network coding protocol that takes advantage of wireless multicast and allows encoding of packets between multiple unicast sessions using binary XOR operations. The authors combine their coding strategy with a modified MAC protocol to show significant throughput improvements versus a standard 802.11 MAC on a wireless testbed. A more sophisticated coding scheme was considered by Traskov *et al.* [42], where a linear program is developed to identify butterfly coding opportunities throughout the network. While the proposed scheme realizes many advanced coding opportunities, the scheme is suboptimal due to high complexity of the problem. The original work on COPE [14] explored the interplay between coding and scheduling, and subsequent work in [37] motivated the need for routing protocols to be aware of COPE-style

network coding. The appropriate choice of routes can increase coding opportunities and [37] shows that significant throughput improvements are possible through such coding aware routing. This is further studied in a utility maximization framework [35] to argue that rate control and scheduling should also be coding-aware. In this work, we address the joint design and performance of routing, scheduling, and coding in a wireless network.

Network coding can be combined with packet overhearing to yield additional coding opportunities. Packet overhearing occurs when nodes receive a packet concurrently with that packet's intended recipient. These additional nodes can then use their knowledge of the overheard packet in future decoding operations. Katti *et al.* [14] use opportunistic overhearing with coding operations over 3 or more packets. A similar coding scheme is combined with energy efficient scheduling in [6]. Khreishah *et al.* [16] use pairwise coding with overhearing in a joint coding, scheduling, and rate control policy, while Paschos *et al.* [30] optimize for scheduling and pairwise coding with statistical overhearing. Finally, a policy for scheduling and coding with symmetric overhearing on star topologies is provided in [34]. We incorporate overhearing into our coding strategy in Chapter 3.

1.2.2 Routing and Scheduling

Numerous previous works have considered joint routing and scheduling in the absence of network coding. In their seminal paper on network control [40], Tassiulas and Ephremides introduce the maximum weight scheduling (MWS) and differential backlog routing policy to provide throughput optimal network control. The policy has an attractive property for dynamic control in that decisions rely only on current queue state information, without requiring knowledge of the long-term arrival rates. The authors are able to prove, using Lyapunov stability theory, that their policy can stabilize the network queues for any stochastic arrival process within the stability region of the network. Neely, Modiano, and Rohrs [27] extended this to jointly optimize for routing, scheduling, and power control in wireless networks with time-varying channels. Le, Modiano, and Shroff [17] developed routing, scheduling,

and flow control algorithms for wireless networks with finite buffers.

Several recent works combine differential backlog routing with random access schedulers in place of MWS to avoid the need for information sharing inherent to centralized control. Jiang and Walrand [10] provided a carrier sense multiple access (CSMA) policy that adaptively chooses backoff durations based on queue backlogs to achieve throughput optimality under an ideal CSMA setting. Ni and Srikant [29] relax the ideal CSMA assumption and avoid collisions of data packets by allowing collisions in control traffic while still maintaining throughput optimality. Marbach and Eryilmaz [21] provide an alternate proof to [10] and provide additional results under the primary interference model. Liu *et al.* [19] provide another proof of convergence for [10] and study the effects of collisions. Rajagopalan, Shah, and Shin [32] provided a throughput optimal slotted ALOHA policy that chooses transmission probabilities as a function of queue backlog.

Other works have also focused on distributed queue-based scheduling, and can be extended to incorporate backpressure routing. Chaporkar, Kar, and Sarkar [4] characterized performance bounds of a distributed maximal scheduler with imperfect matchings. Modiano, Shah, and Zussman [22] provided a distributed scheduler that achieves 100% throughput using a randomized gossip algorithm.

While the literature is very rich for the theoretical framework of MWS and backpressure routing, there are very few system implementations addressing the practical aspects of these policies. This is in part due to the overhead required to implement MWS under centralized control. Moeller *et al.* [23] provided the backpressure collection protocol for wireless sensor networks and evaluated the performance of this backpressure routing scheme on a testbed with 40 Mote sensor devices. Nardelli *et al.* [25] implemented their oCSMA policy as a variation of [10], and were able to evaluate the performance of oCSMA under several scenarios known to cause problems for standard 802.11 MAC protocols.

1.2.3 Joint Routing, Scheduling, and Network Coding

Network coding has been incorporated into the design of scheduling and routing schemes in recent work. A number of recent works, including [5], [16], [30], [34], [41], and [43], develop joint scheduling and coding schemes in a network control framework, either for single-hop transmissions, or under the assumption that routes are fixed and specified *a priori*. In addressing the routing problem, [42] provides a linear optimization approach for identifying network coding opportunities on butterfly subgraphs with multiple unicast sessions, while [7] develops a policy for dynamic routing and scheduling to provide stability throughout the region from [42]. The *poison-remedy* approach introduced in [7] involves opportunistically identifying coding opportunities, creating *poisoned* or coded packets, and subsequently sending a request for *remedy* or uncoded packets to be sent to the destination node to allow for decoding. In a different approach, [6] provides a distributed backpressure routing and maximum weight scheduling policy for a generalized COPE coding scheme, making opportunistic coding decisions to increase throughput. The policy in [6] exploits the use of overhearing to provide coding opportunities, optimizing for a subset of coding opportunities to reduce complexity while allowing for distributed implementation. Finally, [38] formulates a linear program for the joint routing, scheduling and pairwise coding problem and evaluates results from a computational solution to the problem.

1.2.4 Backpressure Routing in a Network Overlay

Here, we discuss works related to application of backpressure routing in network overlays. This includes general use of network overlays for improved route selection, modifications to the differential backlog routing policy for reducing delay, and separation of routing and scheduling in network control policies.

Several works have considered the use of network overlays to improve routing on the Internet. Andersen *et al.* [2] motivate the need for resilient overlay networks (RON) to find paths around network outages on a faster timescale than BGP. Their method deploys a group of RON nodes as an application-layer overlay across various

routing domains, continuously monitoring the quality of paths in the RON to decide which routes to use. Similarly, Han *et al.* [9] proposed a method for choosing placement of overlay nodes to improve path diversity in overlay routes. While both of the preceding works show that their strategies choose high quality single-path routes, we would like to go further and identify multiple-path routes that offer maximum throughput.

Delay reduction for backpressure routing has been studied in a variety of scenarios. While multiple-path routes are required to support the full stability region, the exploratory phase of backpressure routing can lead to large queues when the offered load is low and single-path routes would suffice. Neely, Modiano, and Rhors [26] propose a hybrid policy combining backpressure routing with shortest-path routing, where flows are biased towards shortest-path routes, yet still support the full stability region. Khan, Le, and Modiano [15] extend this hybrid policy to also include digital fountain codes, and show their policy to achieve minimum end-to-end delay in the presence of random link failures. Ying, Shakkottai, and Reddy [44] develop a policy that achieves a similar shortest-path result by minimizing the average path length used by flows. In a scenario with multiple clusters that are intermittently connected, Ryu, Ying, and Shakkottai [33] combine backpressure routing with source routing in a network overlay model to separate the queue dynamics of intra-cluster traffic from longer inter-cluster delays. Bui, Srikant, and Stolyar apply shadow queues [3] to allow the use of per-neighbor FIFO queues instead of per-commodity queues, as is typical with differential backlog routing, and find that this can improve network delay.

Seferoglu and Modiano develop the Diff-Max [36] policy, which separates routing and scheduling functions for backpressure networks. This separation simplifies practical implementation of network control policies, and their control function modifies the differential backlog routing policy in a similar fashion as our backpressure policy for network overlays in Chapter 4.

1.3 Contributions

Next we preview the main contributions in this thesis.

1.3.1 Centralized Control for k-Tuple Coding

In Chapter 2, we consider jointly optimal routing, scheduling, and network coding strategies to maximize throughput in wireless networks. Other works also combine network coding with optimal routing and scheduling, but have either applied network coding in an opportunistic manner [6] or considered advanced network coding strategies where the region of supported arrival rates is difficult to characterize [7]. Our approach is to instead consider a simple network coding strategy where we exactly define our coding constraint, and therefore we can characterize the stability region for our coding scheme.

The original policy for differential backlog routing with max-weight scheduling [40] is optimal for a class of routing and scheduling algorithms that support multicommodity unicast traffic. We generalize the class of supported algorithms to also include network coded transmissions via wireless multicast, subject to our coding constraint, and then we develop a policy to optimize for this more general class. We are able to jointly optimize for routing with network coding because the inherit behavior of our policy is to probe all edges and hyperedges in the network with each commodity. As the policy explores the network, queue backlogs grow until routes are found that can satisfy a given arrival rate vector. If coding is required for the rate vector to be satisfied, then backlogs will grow until the necessary coding opportunities are available for a sufficient fraction of time to satisfy the rate vector.

A generalization of pairwise network coding with next-hop decodability is introduced — called k-tuple coding. The region of arrival rates is fully characterized for which the network queues can be stabilized under this coding strategy.

We propose a dynamic control policy for routing, scheduling, and k-tuple coding, and prove that our policy is throughput optimal subject to the k-tuple coding constraint. Analytical bounds are provided for coding gain of the policy, and numerical

results are presented to support our analytical findings. The stability region is evaluated directly, using a linear program solver, both with and without network coding to calculate the coding gain along random arrival rate vectors in the stability region. We study our policy in a packet simulator to verify that the queues remain relatively small for arrival rates interior to the stability region, and observe average network queue sizes with and without network coding to compare network delays.

Simulation results show that most of the gains are achieved with pairwise coding, and that the coding gain is greater with 2-hop interference[3] than 1-hop interference[4]. Also, we find that under 2-hop interference, the policy yields median throughput gains of 31% beyond optimal scheduling and routing on random topologies with 16 nodes.

Results of Chapter 2 were presented in [12].

1.3.2 Distributed CSMA with Pairwise Coding

In Chapter 3, we consider distributed strategies for joint routing, scheduling, and network coding to maximize throughput in wireless networks. While Chapter 2 provides a *centralized* control policy for routing, scheduling, and network coding, that policy requires large overhead to share queue state information with the central controller. Moreover, the centralized policy from Chapter 2 is based on max-weight scheduling, which requires solving a computationally hard problem in every time step. Recently, a distributed CSMA policy [10] with low computational complexity was proved to be throughput optimal. In Chapter 3, we keep the pairwise network coding and differential backlog routing strategies from the policy in Chapter 2, but replace the centralized scheduler with a *distributed* CSMA scheduler similar to that from [10]. Following the proof from [10], we prove that our CSMA policy can support all arrival rates allowed by the network subject to our pairwise coding constraint.

The network coding scheme is extended to optimize for packet overhearing to increase the number of beneficial coding opportunities. The stability region is adjusted

[3]The 2-hop interference model allows simultaneous transmissions to be non-interfering as long as they are at least 2-hops apart in the network.

[4]The 1-hop interference model allows each node to transmit or receive at most one packet at a time.

to account for our overhearing scheme, and we evaluate coding gain on random topologies. The results show that overhearing provides an average throughput gain of only 2% beyond pairwise coding without overhearing, however the additional computational cost of our overhearing scheme is low. Our results show that overhearing can provide up to an additional 25% increase in throughput on random topologies, and thus find it to be a worthwhile addition.

The distributed CSMA policy has the same throughput region as that of the centralized MWS policy from Chapter 2. However, through simulation we observe that delay with the CSMA policy can be significantly worse than delay with the MWS policy. We study delay on structured and random scenarios, and find for both policies that network queue size grows quadratically with the number of nodes in tandem, where the CSMA policy has larger coefficients on this quadratic function. The network queue size from the CSMA policy is found to be inversely proportional to α, a step-size parameter, where small values of α are required to support the full stability region. We ultimately conclude that the cost of our distributed optimal control comes at the cost of increased delay.

Results of Chapter 3 were presented in [13].

1.3.3 Backpressure Routing in Overlay Networks

In Chapter 4, we consider strategies for integrating network control policies in legacy networks. The approach is to model the controllable nodes as a network overlay operating within the legacy network, and contributions are along two fronts. First, we determine where to place controllable nodes with the objective of maximizing throughput. Second, we develop a network control policy that operates on the network overlay using a modification to differential backlog routing, allowing the policy to observe the level of congestion in the legacy network.

The stability region is characterized for heterogeneous network overlays with a mixture of controllable and uncontrollable nodes, where the controllable nodes can arbitrarily re-route traffic while all other nodes are limited to forwarding traffic along shortest-path routes. An all-paths condition is identified, requiring that all paths in a

network can be constructed as a concatenation of shortest-path routes at controllable overlay nodes. This property is proved necessary and sufficient for the network overlay to have the same stability region as if all nodes in the network were controllable. We study three classes of simple graphs, and find the minimum number of overlay nodes required to provide the full stability region: (1) on tree networks, no controllable nodes are required; (2) on ring networks, exactly 3 controllable nodes are required; and (3) on clique networks, all nodes must be controllable.

Next, we develop an algorithm for placing controllable nodes to satisfy the all-paths condition on graphs where shortest-path routes are given. This overlay node placement algorithm is simulated on several models for random graphs. On power-law graphs with an exponent of $\alpha = 2.5$, considered a good model for the Internet, only 8% of nodes are required to be controllable to enable the full throughput region. A variation on the node placement algorithm is provided to maximize scaling of a specific rate vector given a fixed number of controllable nodes. In one scenario, 80% of the arrival rate vector is supported with only 4 overlay nodes, while support for the final 20% of the rate vector requires an additional 5 overlay nodes.

Finally, we propose a heuristic policy for applying differential backlog routing on network overlays. The overlay backpressure policy is simulated, comparing network queue size to that of standard differential-backlog on fully controllable networks. For the scenarios considered, the modified overlay backpressure policy stabilizes the network queues, yielding the full stability region. We observe decreased delay, relative to that of standard differential backlog routing, attributed to the reduced number of controllable nodes.

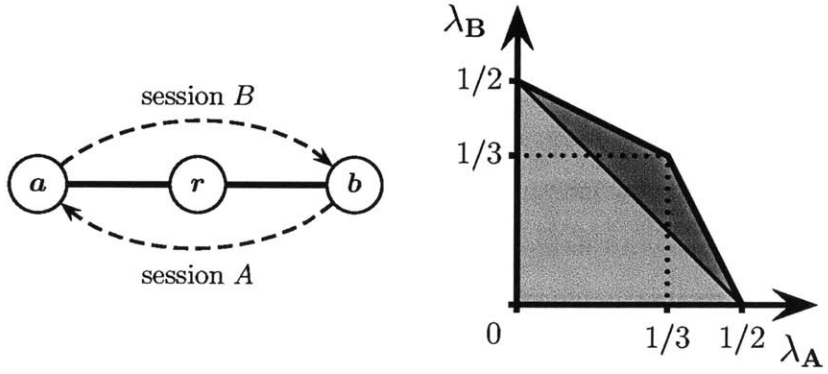
Chapter 2

Centralized Control for k-Tuple Coding

In this chapter, we consider jointly optimal routing, scheduling, and network coding strategies to maximize throughput in wireless networks. While routing and scheduling for wireless networks have been studied for decades, network coding is a relatively new technique that allows for an increase in throughput under certain topological and routing conditions. In this work, we introduce k-tuple coding, a generalization of pairwise coding with next-hop decodability, and fully characterize the region of arrival rates for which the network queues can be stabilized under this coding strategy. We propose a dynamic control policy for routing, scheduling, and k-tuple coding, and prove that our policy is throughput optimal subject to the k-tuple coding constraint. We provide analytical bounds on the coding gain of our policy, and present numerical results to evaluate performance on random topologies. We show that most of the gains are achieved with pairwise coding, and that the coding gain is greater under 2-hop than 1-hop interference. Simulations show that under 2-hop interference our policy yields median throughput gains of 31% beyond optimal scheduling and routing without coding on random topologies with 16 nodes.

2.1 Introduction

Network coding, originally introduced in [1], can increase network throughput by allowing intermediate nodes to combine or encode the data they receive, rather than simply replicating and forwarding it. For example, consider the wireless 2-way relay scenario shown in Figure 2-1a, where nodes a and b can only communicate via the intermediate node r. Here, we assume unit-rate links and that only one node can transmit at a time. If node r is limited to forwarding traffic without network coding, then the region of arrival rates that can be supported for these two sessions is shown as the light gray triangle in Figure 2-1b. However, if node r is also allowed to perform network coding for this pair of sessions, then the supported region increases to also include the dark gray triangle. Thus, network coding can allow us to increase throughput in the network. The details of our network coding scheme will be discussed in Section 2.2.3, and the stability region for this coding scheme will be characterized in Section 2.3.



(a) 2-way Relay Scenario (b) Stability Region

Figure 2-1: Pairwise coding on the 2-way relay scenario. (a) Traffic sessions are shown with dashed arrows, where session A has rate λ_A and session B has rate λ_B. (b) The stability region is the set of all arrival rates that the system can support. The region supported without network coding is shown in light gray, while the combination of light and dark gray triangles show the region supported with network coding.

The benefit of wireless network coding was clearly demonstrated with the introduction of COPE by Katti, *et al.* [14]. COPE is an opportunistic network coding protocol that takes advantage of wireless multicast and allows encoding of packets be-

tween multiple unicast sessions using binary XOR operations. The authors combine their coding strategy with a modified MAC protocol to show significant throughput improvements versus a standard 802.11 MAC on a wireless testbed. While the original work on COPE [14] explored the interplay between coding and scheduling, subsequent work by Sengupta, Rayanchu, and Banerjee [37] motivated the need for routing protocols to be aware of COPE-style network coding. The appropriate choice of routes can increase coding opportunities and [37] shows that significant throughput improvements are possible through such coding aware routing. In this chapter, we address the joint design and performance of routing, scheduling, and coding in a wireless network.

Numerous previous works have considered joint routing and scheduling in the absence of network coding. In their seminal paper on network control [40], Tassiulas and Ephremides introduce the maximum weight scheduling and differential backlog routing policy to provide throughput optimal network control. The policy has an attractive property for dynamic control in that decisions rely only on current queue state information, without requiring knowledge of the long-term arrival rates. The authors are able to prove, using Lyapunov stability theory, that their policy can stabilize the network queues for any stochastic arrival process within the stability region of the network. Neely, Modiano, and Rohrs [27] extended this to jointly optimize for routing, scheduling, and power control in wireless networks with time-varying channels.

Recently, network coding has been incorporated into the design of scheduling and routing schemes. A number of recent works, including [5], [16], [30], and [34], develop joint scheduling and coding schemes in a network control framework, either for single-hop transmissions, or under the assumption that routes are fixed and specified *a priori*. In addressing the routing problem, [42] provides a linear optimization approach for identifying network coding opportunities on butterfly subgraphs with multiple unicast sessions, while [7] develops a policy for dynamic routing and scheduling to provide stability throughout the region from [42]. The *poison-remedy* approach introduced in [7] involves opportunistically identifying coding opportunities, creating *poisoned*

or coded packets, and subsequently sending a request for *remedy* or uncoded packets to be sent to the destination node to allow for decoding. In a different approach, [6] provides a distributed backpressure routing and maximum weight scheduling policy for a generalized COPE coding scheme, making opportunistic coding decisions to increase throughput. The policy in [6] exploits the use of overhearing to provide coding opportunities, optimizing for a subset of coding opportunities to reduce complexity while allowing for distributed implementation.

This chapter proposes an inter-session network coding strategy that jointly optimizes for routing and scheduling of unicast traffic on wireless networks. The coding scheme considered does not require overhearing, but simply requires each node to keep a copy of packets it previously transmitted for some limited period of time. All coded packets must be decoded at the next hop, and when a coding opportunity is identified, the requisite conditions for decoding are already satisfied. The main contributions in this chapter are as follows. We introduce k-tuple coding, a generalization of pairwise inter-session network coding, and fully characterize the stability region under this coding strategy. We then propose a dynamic routing, scheduling, and k-tuple coding policy and prove that this policy is throughput optimal subject to the k-tuple coding constraint. Analytical bounds are provided for the throughput gain of k-tuple coding relative to optimal routing and scheduling without network coding. Finally, numerical results from simulation and linear program evaluation are given to provide a sense of the performance of our policy under various settings.

A unique attribute of our policy is that it requires keeping track of which one-hop neighbor supplies each packet; this requirement shows up both in the characterization of the stability region and in the construction of weight calculations.

This chapter is organized as follows. Section 2.2 describes our system model, and Section 2.3 characterizes the stability region under this model. In Section 2.4 we design a control policy that combines scheduling, routing, and network coding to achieve the given stability region. We provide analytical results on coding gain in Section 2.5, and describe the complexity of our coding operations in Section 2.6. In Section 2.7 we give numerical results, and offer concluding remarks in Section 2.8.

2.2 Model

2.2.1 Wireless Network

We model the wireless network as a directed hypergraph, $G = (\mathcal{N}, \mathcal{H})$, where $\mathcal{N}$ is the set of nodes in the network and $\mathcal{H}$ is the set of directed hyperedges [1] supported by the network. Hyperedge $(a, \tilde{J})$ allows head node a to communicate directly with a set of tail nodes $\tilde{J}$ using a single transmission. Standard edge (a, b) is a special case of a hyperedge, where node b is the only tail node. Let $\mathcal{H}_k \subseteq \mathcal{H}$ be the set of hyperedges that contain exactly k tail nodes. We model the network as a hypergraph to capture the effects of wireless multicast transmissions, which are needed by the network coding strategy.

We consider unicast traffic. In this context, wireless multicast is used only for the transmission of network coded packets. Time is assumed to be slotted, and for simplicity unit rate links are used with packets of a fixed size corresponding to one packet per time slot. Packets destined for node c are called *commodity* c packets. Exogenous packet arrivals are allowed from arbitrary processes with finite second moments. Let λ_a^c be the average rate of exogenous arrivals at node a for commodity c, and let $\boldsymbol{\lambda} = (\lambda_a^c)$ be a vector of arrival rates for all sources a and commodities c.

We assume non-interfering transmissions to be reliable, but otherwise allow arbitrary interference constraints. However, our numerical results consider two interference models of interest: 1-hop and 2-hop interference. In the context of wireless networks, the 1-hop interference model means that each node can receive from at most one neighbor at a time, and a node cannot receive while transmitting. The 2-hop interference model builds on the restrictions of 1-hop interference, adding a constraint such that simultaneous communications will interfere if connected by any standard edge in the network. We naturally extend the 1-hop and 2-hop interference models from [39] by allowing these models to make use of wireless multicast.

[1] We consider hyperedges composed of one or more standard edges emanating from a single node.

2.2.2 Routing and Scheduling

A wireless network requires mechanisms for *routing* packets along a series of hyperedges toward the destination node and for *scheduling* a set of hyperedges to be activated simultaneously without creating interference. Let schedule ℓ be a set of non-interfering hyperedges, and let $\mathcal{L}$ be the set of all such schedules. We consider a centralized control policy that dynamically chooses which hyperedges to activate during each time slot, and chooses which commodity to send over each hyperedge when active.

Tassiulas and Ephremides [40] provided a joint routing and scheduling policy that is throughput optimal; in the absence of network coding, their policy yields 100% throughput for all arrival rate vectors that can be supported by any policy. At each time slot $t \geq 0$ and for each edge (a, b), this policy calculates the edge weight $W^*_{ab}(t)$ as the maximum differential backlog over the edge,

$$W^*_{ab}(t) = \max_{c \in \mathcal{N}} \{U^c_a(t) - U^c_b(t)\} \,, \tag{2.1}$$

where $U^c_a(t)$ is the backlog at node a of commodity c packets at time t. Their policy then chooses the schedule with maximum total weight $\ell^*(t)$,

$$\ell^*(t) = \arg\max_{\ell \in \mathcal{L}} \sum_{(a,b) \in \mathcal{H}_1} \ell_{ab} W^*_{ab}(t) \,, \tag{2.2}$$

where $\ell_{ab} = 1$ if edge (a, b) is active in schedule ℓ, and is 0 otherwise. Finally, this policy serves the commodities that maximize Equation (2.1) for each active edge in schedule $\ell^*(t)$. While this policy optimizes for scheduling and routing, the policy as stated only considers standard edges in $\mathcal{H}_1$ and does not account for network coding. We extend this policy from Tassiulas and Ephremides to jointly optimize for scheduling, routing, and our simple network coding scheme.

2.2.3 Network Coding

We describe our k-tuple coding operations using a constructive approach by first considering the pairwise case of coding over 2 sessions, then extending this to the

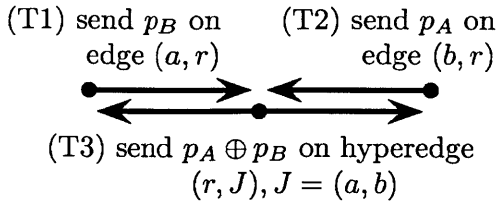


Figure 2-2: Sequence of transmissions T1, T2, and T3 for pairwise coding on the 2-way relay scenario from Figure 2-1a.

case of coding over 3 sessions, and finally generalizing to the case of coding over k sessions. We then motivate the use of coding by describing achievable throughput gains in simple scenarios. Our coding strategy depends on knowing the neighbor from which each packet is received. To accomplish this, nodes store packets in subqueues based on the one-hop source of each packet; one-hop subqueue d for commodity c holds commodity c packets received from neighbor d.

Our coding strategy considers ordered sets of hyperedge tail nodes and commodities. Let (a, J) be a hyperedge with ordered tail nodes, for $J \in \text{perms}(\tilde{J})$, where $\text{perms}(\tilde{J})$ is the set of all permutations of $\tilde{J}$. The tail node at the m^{th} position in J is denoted $J(m)$, and with an abuse of notation $J(k+1) = J(1)$ for $|J| = k$. Let $s \in \mathcal{N}^k$ be an ordered set of k commodities, and let $\mathcal{S}_k$ be the set of all ordered commodity sets of size k. The commodity at the m^{th} position of s is denoted $s(m)$, and again by abuse of notation, let $s(k+1) = s(1)$ for $|s| = k$.

Pairwise Coding

Consider again the scenario from Figure 2-1a, where nodes a and b can only communicate via node r. With pairwise coding, we can exchange one packet from each session in 3 transmissions as shown in Figure 2-2. Here, node a sends packet p_B for commodity b to node r in transmission T1, and node b sends packet p_A for commodity a to node r in transmission T2. Thus for hyperedge $(r, J), J = (a, b)$, and commodity set $s = (a, b)$, a packet for commodity $s(2) = b$ (i.e., p_B) is in one-hop subqueue $J(1) = a$ and a packet for commodity $s(1) = a$ resides in one-hop subqueue $J(2) = b$. Node r can generate a coded packet $p_{AB} = p_A \oplus p_B$, where $\oplus$ is the binary XOR operation,

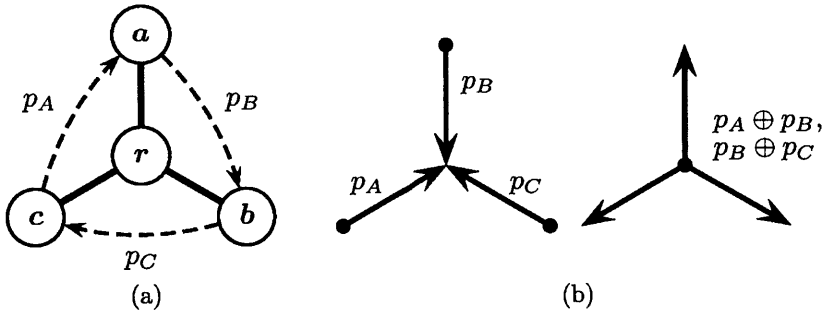


Figure 2-3: 3-tuple coding operation at node r. (a) Standard edges shown with solid lines, with all hyperedges available; traffic demands shown with dashed arrows. (b) Activations shown with solid arrows.

and then send p_{AB} to nodes a and b with a single wireless multicast transmission, labeled T3 in Figure 2-2. Node a has previously seen packet p_B, and can recover $p_A = p_{AB} \oplus p_B$. Likewise, node b can recover packet p_B. Note that we do not require packets p_A, p_B, and p_{AB} to be transmitted in consecutive time slots, but require only that p_{AB} is transmitted after both p_A and p_B have been received at node r.

The coding operation requires that each node maintain an extra buffer with uncoded copies of packets that it has previously transmitted; we call this the *side information buffer.* In the example above, upon transmitting to node r, node a keeps packet p_B and node b keeps packet p_A in their respective side information buffers. Additionally, node r adds p_A and p_B to its side information buffer upon transmitting coded packet p_{AB}. We discuss operations for removing packets from this buffer in Section 2.6.2. Note that coded packets can be discarded at the end of the coding operation, as only uncoded packets are stored in one-hop subqueues and side information buffers.

3-Tuple Coding

Now suppose that node r has received packet p_A for commodity a from neighbor c, packet p_B for commodity b from neighbor a, and packet p_C for commodity c from neighbor b. For hyperedge $(r, J), J = (a, b, c)$, and commodity set $s = (a, b, c)$, a packet for commodity $s(2) = b$ resides in the one-hop subqueue $J(1) = a$, a packet for commodity $s(3) = c$ resides in subqueue $J(2) = b$, and a packet for commodity

$s(1) = a$ resides in subqueue $J(3) = c$. Node r can encode packets p_A, p_B, and p_C using two coded packets: $p_{AB} = p_A \oplus p_B$ and $p_{BC} = p_B \oplus p_C$. Node r can then transmit coded packets p_{AB} and p_{BC} to neighbors a, b, and c using 2 wireless multicast transmissions. Each of the 3 neighbors can decode the packet destined for them using the 2 coded packets from r along with their side information copy of the uncoded packet that they respectively supplied to the encoding node. Note that even though nodes a, b, and c can decode all 3 packets, they each keep only the one packet that is destined for them and discard the rest. This scenario is shown in Figure 2-3.

Definition 2.1. *A **coding opportunity** $\big(s, (r, J)\big)$ is formed by the combination of ordered hyperedge (r, J) and ordered set of commodities s held at node r for which: (a) $|s| = |J| = k$, and (b) for each $m = 1, 2, ..., k$, a packet for commodity $s(m+1)$ resides in the one-hop subqueue $J(m)$ at node r.*

For the pairwise coding scenario, if p_A and p_B are the only packets in the one-hop subqueues at node r, then $s = (a, b)$ is the only set of commodities that forms a coding opportunity with hyperedge $(r, J), J = (a, b)$. Commodity set $s' = (b, a)$ does not form a coding opportunity with hyperedge $(r, J), J = (a, b)$, since at node r, there is no packet for commodity $s'(2) = a$ in one-hop subqueue $J(1) = a$, and no packet for commodity $s'(1) = b$ in one-hop subqueue $J(2) = b$. By assumption, a node will never transmit a packet destined for itself, so commodity set $s' = (b, a)$ and ordered hyperedge $(r, J), J = (a, b)$, will never satisfy the condition (b) for coding opportunities. However commodity set $s' = (b, a)$ and hyperedge $(r, J'), J' = (b, a)$, do form a coding opportunity, since s' and J' are formed by the same circular shift of s and J, respectively. Yet, the coding operations and packets delivered for $\big(s', (r, J')\big)$ and $\big(s, (r, J)\big)$ are identical. In general, for any coding opportunity $\big(s, (r, J)\big)$, we can ignore equivalent circular shifts $\big(s', (r, J')\big)$ in constructing a routing and scheduling policy. Furthermore, consider hyperedge $(r, J), J = (a, b)$, on a more general topology, where the transmit buffer at r contains packets in both one-hop subqueues a and b for commodities g and h. Then commodity sets $s_1 = (g, h)$, $s_2 = (h, g)$, $s_3 = (g, g)$, and $s_4 = (h, h)$ can each be combined with hyperedge (r, J) to form valid coding

opportunities $\big(s, (r, J)\big)$ for $s \in \{s_1, s_2, s_3, s_4\}$. Here, we see that a coding opportunity can be valid even when combining two packets of the same commodity, as in s_3 and s_4. While such coding scenarios indicate that packets have traveled in cycles, and therefore do not increase throughput, these coding opportunities can help recover from suboptimal paths explored by backpressure routing.

Coding Rule

A k-tuple coding operation can only be performed for a coding opportunity $\big(s, (r, J)\big)$ that satisfies Definition 2.1. A packet for commodity $s(m+1)$ that resides in the one-hop subqueue $J(m)$ at node r is delivered to neighbor $J(m+1)$.

For the 3-tuple coding example, commodity set $s = (c, b, a)$ and ordered hyperedge $(r, J), J = (a, c, b)$ also form a valid coding opportunity. In this alternate coding opportunity, node r delivers packet p_C to node a, p_B to c, and p_A to b.

k-Tuple Coding

Generalizing further, a commodity set s and hyperedge (r, J), $|s| = |J| = k$, can form a k-tuple coding opportunity for $2 \leq k \leq \text{degree}(r)$, where degree$(r)$ is the number of edges incident to node r. The encoding operation requires r to receive one packet from each of the k distinct neighbors in J, and then to transmit $k-1$ coded packets via wireless multicast to all k neighbors. To encode the uncoded packets $p_1, ..., p_k$ corresponding, in order, to commodities $s(1), ..., s(k)$, node r can generate $k-1$ coded packets as: $(p_1 \oplus p_2)$, $(p_2 \oplus p_3)$, ..., and $(p_{k-1} \oplus p_k)$. Each of the k neighbors already has in their side information buffer a copy of the packet that they respectively supplied to r. Upon receiving the $k-1$ coded packets from r, each of the k neighbors can then decode the packet destined for them. For example, assume node d supplied packet p_1 to r, and r sends packet p_k to d using a k-tuple code. Node d can recover packet p_k as: $p_2 = p_1 \oplus (p_1 \oplus p_2)$, $p_3 = p_2 \oplus (p_2 \oplus p_3)$, ... , and $p_k = p_{k-1} \oplus (p_{k-1} \oplus p_k)$. It follows that for all code sizes k, the use of binary XOR operations between pairs of packets is sufficient for both encode and decode operations for k-tuple coding.

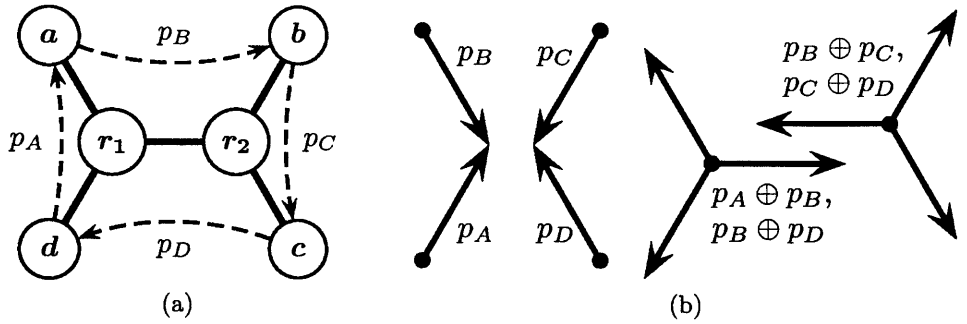


Figure 2-4: Pair of 3-tuple coding nodes, r_1 and r_2. (a) Standard edges shown with solid lines, with all hyperedges emanating from r_1 and r_2 available; traffic demands shown with dashed lines. (b) Activations shown with solid arrows.

Lemma 2.1. *If k neighbors of a node each have in their respective side information buffers at most one packet from a k-tuple coding opportunity, then under any coding strategy, $k-1$ is the fewest number of packets that the coding node must transmit to exchange all k packets.*

Proof. The proof follows because in order to solve for $k-1$ unknown packets, $k-1$ linearly independent equations are needed. □

Thus, the benefit of network coding reduces with increasing k. We next consider the throughput gain for individual and *cascading* k-tuple coding operations in Observations 2.1 and 2.2, where cascading occurs when the output of one coding operation is the input to another coding operation.

Observation 2.1. *A single k-tuple coding operation yields a throughput gain of $\frac{2k}{2k-1}$ when all hyperedges connected to the coding node mutually interfere.*

The single k-tuple coding operation requires a total of $2k-1$ time slots, consisting of k uplink transmissions and $k-1$ downlink transmissions, while the same packet exchange without coding requires $2k$ time slots, yielding the observed result. Here, we save 1 transmission per coding operation. For pairwise coding the throughput gain is 4/3, and for 3-tuple coding the gain is 6/5. While 3-tuple coding yields a lower gain than pairwise coding, notice that there is no pairwise coding opportunity in the 3-tuple coding scenario in Figure 2-3.

Observation 2.2. *Throughput gain can increase when k-tuple coding operations* cascade. *For example, a pair of* cascading *k-tuple coding operations can yield a throughput gain of* $\frac{2k-1}{2k-2}$.

Consider coding nodes r_1 and r_2 that share a single edge and where both coding nodes have degree k, with interference constraints such that all hyperedges mutually interfere. An example of this scenario is shown in Figure 2-4 for the case of $k = 3$ under 2-hop interference. For the traffic demands shown, node r_1 is a tail node for the 3-tuple coding opportunity $\big(s_2, (r_2, J_2)\big)$, $s_2 = (b, c, d)$ and $J_2 = (b, c, r_1)$, while r_2 is a tail node for the coding opportunity $\big(s_1, (r_1, J_1)\big)$, where $s_1 = (a, b, d)$ and $J_1 = (a, r_2, d)$. The coding operation at r_1 can deliver p_A to a, p_B to r_2 and p_D to d. The coding operation at r_2 can deliver p_B to b, p_C to c, and p_D to r_1. Without coding, it takes $2+3+2+3 = 10$ or $2(2k-1)$ time slots to deliver packets p_A, p_B, p_C, and p_D, while k-tuple coding can deliver the same set of packets in $4 + 2 + 2 = 8$ or $2(2k-2)$ time slots using the activations in Figure 2-4b. This yields the observed gain of $\frac{2k-1}{2k-2}$. Again, we save 1 transmission per coding operation. However, by allowing two coding operations to interact, the throughput gain has increased beyond that of a single coding operation. For pairwise coding this is a throughput gain of 3/2, while for 3-tuple coding the throughput gain is 5/4. These throughput gains require a pipeline of coding operations, where nodes r_1 and r_2 are initialized with packets from a, b, c, and d, and the activations cycle between coding operations at r_1 and r_2.

2.3 Stability Region

The stability region Λ_{NC} of our k-tuple coding strategy is the set of all arrival rate vectors (λ_a^c) that can be supported while ensuring that all packet queues in the network remain finite.

Let $f_{ab}^{d,c}$ be the rate of flow for uncoded packets of commodity c packets received from node d and sent over edge (a, b), and let f_{aJ}^{s} be the rate of flow for coded packets over ordered hyperedge (a, J) for each commodity in set s, where $\big(s, (a, J)\big)$ is a coding opportunity. For simplicity, we use the following $\hat{f}$ notation to represent

a sum over a set of underlying flow variables. Notation $\{d, c\} \rightarrow b$ means commodity c from one-hop subqueue d is sent to node b. Let $\hat{f}_{ab}^{d,c}$ be the total uncoded and coded flow rate from node a to neighbor b for commodity c from the subqueue for one-hop neighbor d. Thus,

$$\hat{f}_{ab}^{d,c} = f_{ab}^{d,c} + \sum_{\left\{\substack{(a,J)\in\mathcal{H}_k, k\geq 2, s\in\mathcal{S}_k: \\ d,b\in J, c\in s, \{d,c\}\rightarrow b}\right\}} f_{aJ}^{s}, \quad \forall a, b, c, d \in \mathcal{N}, \tag{2.3}$$

where the summation is over the set of coded flow variables f_{aJ}^{s} for all hyperedges (a, J) and commodity sets s that deliver commodity c packets from one-hop subqueue d to node b. Let $\hat{f}_{ab}^{c}$ be the total coded and uncoded flow rate from a to b for commodity c traffic from all one-hop subqueues, as shown below.

$$\hat{f}_{ab}^{c} = \sum_{d} \hat{f}_{ab}^{d,c}, \quad \forall a, b, c \in \mathcal{N} \tag{2.4}$$

We start with some efficiency assumptions: nodes don't transmit to themselves and nodes don't transmit any traffic destined for themselves. Also, all flow variables are non-negative. Next, we define several constraints from our policy.

Flow Conservation: For each node a and for each commodity $c \neq a$, all commodity c flow that enters a must leave a. To maintain this flow conservation, the exogenous arrivals for commodity c must equal the difference between total network departures for commodity c and total network arrivals for commodity c.

$$\lambda_a^c = \sum_{b} \hat{f}_{ab}^{c} - \sum_{d} \hat{f}_{da}^{c}, \quad \forall a, c \in \mathcal{N} : a \neq c \tag{2.5}$$

Coding Constraint: Our coding strategy allows node a to encode packets for commodity c that have been received directly from neighbor d, where the total flow directly from d to a for commodity c gives an upper bound on the total coded flow from a that can make use of commodity c packets in the side information buffer at

neighbor d.

$$\sum_b \left(\hat{f}_{ab}^{d,c} - f_{ab}^{d,c} \right) \leq \hat{f}_{da}^{c} , \quad \forall a, c, d \in \mathcal{N} \tag{2.6}$$

Hyperedge Rate Constraint: Let γ_ℓ be the fraction of time that schedule ℓ is active, and let $\ell_{a\tilde{J}} = 1$ if hyperedge $(a, \tilde{J})$ is active in schedule ℓ, and 0 otherwise. Let $R_{a\tilde{J}}$ be the fraction of time that hyperedge $(a, \tilde{J})$ is active. Then we find $R_{a\tilde{J}}$ as follows:

$$R_{a\tilde{J}} = \sum_{\ell \in \mathcal{L}} \ell_{a\tilde{J}} \, \gamma_\ell , \quad \forall (a, \tilde{J}) . \tag{2.7}$$

The set $(R_{a\tilde{J}})$ for all hyperedges must then be in the convex hull of the set of all schedules $\mathcal{L}$, where $\sum_{\ell \in \mathcal{L}} \gamma_\ell \leq 1$.

For uncoded traffic ($k = 1$), the fraction of time $R_{a\tilde{J}}$, $\tilde{J} = \{b\}$, that edge (a, b) is active gives an upper bound on the total flow of all commodities over that edge.

$$\sum_{d,c \in \mathcal{N}} f_{ab}^{d,c} \leq R_{a\tilde{J}} , \quad \forall (a, b) : \tilde{J} = \{b\} \tag{2.8}$$

For coded traffic ($k \geq 2$), our coding strategy imposes a factor of $\frac{1}{k-1}$ to account for the $k - 1$ time slots required to deliver one packet to each destination of a coded packet.

$$\sum_{J \in \text{perms}(\tilde{J}), s \in \mathcal{S}_k} f_{aJ}^{s} \leq \frac{R_{a\tilde{J}}}{k-1} , \quad \forall (a, \tilde{J}) : |\tilde{J}| = k, \, k \geq 2 \tag{2.9}$$

The stability region for our k-tuple coding strategy is the convex polytope bounded by the set of constraints in Equations (2.5)-(2.9).

It can be shown that Equations (2.5)-(2.9) are necessary for stability. Equation (2.5) is required by flow efficiency assumptions and reliable transmission, combined with the model of each packet traversing a single path in the network. Equation (2.6) follows from the structure of subqueues and the coding rule. Equation (2.7)

is required for schedules to be non-interfering, where violating this convexity constraint would necessitate the use of a schedule that is not allowed. Equations (2.8) and (2.9) are clear from the edge capacities and the coding rule.

Additionally, we give the following two redundant constraints that are informative about our coding strategy.

$$\lambda_a^c = \sum_b f_{ab}^{a,c}, \qquad \forall a, c \in \mathcal{N} : a \neq c \tag{2.10}$$

$$0 = \left(\sum_b \hat{f}_{ab}^{d,c} \right) - \hat{f}_{da}^{c}, \qquad \forall a, c, d \in \mathcal{N} : a \neq c,\ d \neq a \tag{2.11}$$

Equation (2.10) indicates that exogenous arrivals must be sent uncoded by the source node, while Equation (2.11) indicates that the total flow out of one-hop subqueue d for commodity c at node a must equal the total flow sent from d to a for commodity c. These two equations can be viewed as detailed flow conservation constraints for the one-hop subqueues. It can be shown that replacing Equations (2.5) and (2.6) with Equations (2.10) and (2.11) does not change the stability region Λ_{NC}.

2.4 LCM-Frame Policy for Routing, Scheduling, and k-Tuple Coding

Our control policy performs scheduling, routing, and k-tuple coding for dynamic choice of k. The transmission of each k-tuple set requires $k-1$ time slots, and we schedule for fixed size frames such that all coding operations are performed within the frame boundary. Therefore, the frame size must be an integer multiple of $k-1$ for each code size $k \in \{1, ..., K\}$.

The least common multiple framing (LCM-Frame) policy uses a fixed frame size of length $T = \text{lcm}\{1, ..., K-1\}$ time slots, where K is the maximum size of any k-tuple code used. Control decisions are made only at the beginning of each frame, so that for each hyperedge active within a frame, all packets sent over the active hyperedge contain packets using the same code size k. Let C_k represent a k-tuple coding set of

$k \in \{2, ..., K\}$ packets, which are to be encoded to form $k-1$ coded packets. For example, in the case of $K = 5$ all frames will have duration $T = \text{lcm}\{1, 2, 3, 4\} = 12$ time slots and each active hyperedge can transmit 12 uncoded packets, 12 sets of C_2, 6 sets of C_3, 4 sets of C_4, or 3 sets of C_5.

At every time slot $t = nT$, for integer $n \geq 0$, the policy operates as follows.

1. For every standard edge $(a, b) \in \mathcal{H}_1$, calculate edge weight $W^*_{ab}(t)$ as below, and choose associated commodity $s^*_{ab}(t)$ and subqueue $d^*_{ab}(t)$. Let $U^{d,c}_a(t)$ represent the backlog at node a at time t for packets received from neighbor d and destined for commodity c. This is a slight modification from the policy in Equation (2.1), in that here we use the backlog of the one-hop subqueue of each commodity instead of the total backlog for each commodity.

$$W^*_{ab}(t) = \max_{c,d} \left\{ U^{d,c}_a(t) - U^{a,c}_b(t) \right\} \tag{2.12}$$

2. For every hyperedge $(a, \tilde{J})$, for $k = |\tilde{J}| \geq 2$, calculate the weight as follows. For every ordered hyperedge (a, J), for $J \in \text{perms}(\tilde{J})$, and every commodity set $s \in \mathcal{S}_k$ such that $\big(s, (a, J)\big)$ is a coding opportunity, calculate the weight $W^s_{aJ}(t)$. First, evaluate the differential backlog for each commodity in s and the respective tail node from J as: $U^{J(m),s(m+1)}_a(t) - U^{a,s(m+1)}_{J(m+1)}(t)$, for each $m = 1, 2, ..., k$. If this differential backlog is non-positive for any position m, then it is not beneficial to encode, so set weight $W^s_{aJ}(t) = 0$. If the differential backlog is positive for all positions m, then calculate weight $W^s_{aJ}(t)$ as

$$W^s_{aJ}(t) = \frac{1}{k-1} \sum_{m=1}^{k} \left(U^{d,c}_a(t) - U^{a,c}_b(t) \right), \tag{2.13}$$

where $d = J(m), b = J(m+1)$, and $c = s(m+1)$. The factor $\frac{1}{k-1}$ accounts for $k-1$ time slots required to transmit the coded set s. Then choose optimal weight $W^*_{a\tilde{J}}(t)$ for the unordered hyperedge as below.

$$W^*_{a\tilde{J}}(t) = \max_{J,s} \left\{ W^s_{aJ}(t) \right\} \tag{2.14}$$

The optimal commodity set $s^*_{a\tilde{J}}(t)$ and tail ordering $\tilde{J}^*_a(t)$ are chosen as the values of s and J, respectively, that yield the maximum weight in Equation (2.14).

3. Choose the maximum weighted schedule, generalizing Equation (2.2) to allow for hyperedges, as below.

$$\ell^*(t) = \arg\max_{\ell \in \mathcal{L}} \sum_{(a,b)} \ell_{ab} W^*_{ab}(t) + \sum_{(a,\tilde{J})} \ell_{a\tilde{J}} W^*_{a\tilde{J}}(t) \tag{2.15}$$

4. Repeatedly activate the chosen hyperedges for the duration of the T-slot frame. If there are not enough packets in the subqueue for an active commodity, then null packets are used in place of that commodity for the remainder of the frame.

Definition 2.2. *A queue $U(t), t \geq 0$, is* stable *if*

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}[U(\tau)] < \infty \,. \tag{2.16}$$

A network is stable *when all queues in the network are stable.*

Theorem 2.1. *The LCM-Frame policy stabilizes the network for all arrival rate vectors interior to stability region Λ_{NC}.*

The proof is given in Appendix 2.A.

The policy probes all paths in the network, so it will encounter complex coding opportunities such as those in the example from Figure 2-4. As the policy makes use of coding opportunities it reduces backpressure along those paths, thereby attracting more traffic to paths that offer coding opportunities.

2.5 k-Tuple Coding Gain

We require a meaningful metric to compare performance with coding versus without coding. We identify the maximum scaling of arrival rate vector subject to stability from Equations (2.5)-(2.9). Our metric of interest is the ratio of these stable scalings

under k-tuple coding versus routing and scheduling without coding. Let Λ_{RS} be the stability region under routing-and-scheduling only.

Definition 2.3. *Given rate vector $\boldsymbol{\lambda} \in \Lambda_{RS}$, let $f_{NC}(\boldsymbol{\lambda}) = \max\{\rho : \rho\boldsymbol{\lambda} \in \Lambda_{NC}\}$ and let $f_{RS}(\boldsymbol{\lambda}) = \max\{\rho : \rho\boldsymbol{\lambda} \in \Lambda_{RS}\}$, where ρ is a scalar.* ***Coding gain*** *is the ratio $f_{NC}(\boldsymbol{\lambda})/f_{RS}(\boldsymbol{\lambda})$.*

Theorem 2.2. *Considering all possible topologies, traffic demands, and interference constraints, and with dynamic and optimal choice of code size, the coding gain from k-tuple coding is upper bounded by 2.*

We offer a sketch of the proof. First, the coding constraint is relaxed, removing the requirement that recipients of coded packets are part of a coding opportunity while still allowing encoding nodes to deliver k packets in $k-1$ time slots. Starting with Equation (2.6), take the sum over all neighbors d for both sides of the inequality, then add λ_a^c to the right side to yield the relaxed coding constraint below.

$$\sum_{b,d} \left(\hat{f}_{ab}^{d,c} - f_{ab}^{d,c} \right) \leq \lambda_a^c + \sum_{d} \hat{f}_{da}^{c}, \quad \forall a, c \in \mathcal{N} \tag{2.17}$$

By this relaxation, we have $\Lambda_{NC} \leq \Lambda_{RCC}$, where Λ_{RCC} is the stability region under the relaxed coding constraint (2.17). Next, we compare to the flow conservation constraint (2.5) and note that the relaxed constraint (2.17) is degenerate and can be removed. With coding constraint (2.6) removed, any pair of packets can be encoded and sent over a standard edge in one time slot, allowing each edge to achieve a rate of 2. This implies that $\Lambda_{RCC} = 2\Lambda_{RS}$, which in turn gives the desired result $\Lambda_{NC} \leq 2\Lambda_{RS}$.

Further, a coding gain of 2 is achievable. Consider again the scenario from Figure 2-1a, but now relax the interference constraint to allow all nodes to transmit simultaneously. This might occur, for example, when node r has multiple uplink channels while nodes a and b are in the same downlink channel. Then, without network coding the aggregate system throughput is limited to sum of the combined rates, i.e., $\lambda_A + \lambda_B \leq 1$. However, with network coding the aggregate system throughput

is limited to the maximum of the combined rates, i.e., $\max(\lambda_A, \lambda_B) \leq 1$. For symmetric arrival rates, the system without network coding supports a maximum rate of $(\lambda_A, \lambda_B) = (1/2, 1/2)$, while the system with network coding supports a maximum rate of $(1, 1)$, yielding a coding gain of 2.

2.6 Complexity and Side Information

The gain in stable throughput provided by the LCM-Frame policy comes at the expense of additional complexity in computing weights and additional side information that must be stored in the network. In this section we quantify these aspects of the policy.

2.6.1 Complexity of Weight Computation

The policies that we consider require solving the maximum weight independent set (MWIS) problem at each time slot, which is known to be NP-Hard for general interference graphs. However, polynomial time solutions for MWIS are possible for certain classes of interference graphs, such as claw-free interference graphs [24]. For these classes of graphs with polynomial MWIS solutions, we focus on the complexity of calculating weights of hyperedges for k-tuple coding. Let N be the number of nodes in the network, and let each node represent a commodity.

Standard edges: There are $\mathcal{O}(N)$ edges per node, with $\mathcal{O}(N)$ commodities and $\mathcal{O}(N)$ one-hop sources. The running time for the weight calculations of standard edges at each node is thus $\mathcal{O}(N^3)$ per time slot.

Pairwise hyperedges: There are $\mathcal{O}(N^2)$ pairwise hyperedges per node, and the required one-hop sources are given by the tail nodes of the hyperedge. Each pairwise hyperedge is composed of two standard edges, and for each of these component standard edges we can independently choose the commodity with maximum differential backlog. This gives a running time at each node of $2N^3 = O(N^3)$ per time slot.

General k-tuple hyperedges, $k \geq 2$: There are $\mathcal{O}\binom{N}{k}$ subsets of k tail nodes at each node, and when we eliminate circular shifts, there are $(k-1)!$ circular permutations

of each subset of tail nodes to consider. For each circular permutation, the choice of optimal commodity set requires a running time of $\mathcal{O}(kN)$. The running time at each node is then $\frac{N!}{(N-k)!k!}Nk(k-1)! = O(N^{k+1})$ per time slot.

2.6.2 Upper Bound on Side Information

To decode coded packets, k-tuple coding requires each node to maintain a side information buffer, where uncoded copies of previously transmitted packets are stored.

Corollary 2.1. *For every arrival rate vector (λ_a^c) strictly interior to the stability region Λ_{NC}, the LCM-Frame policy stabilizes the side information buffers in the network.*

Proof. The k-tuple coding strategy requires each node a to keep a side information copy of each packet p sent from a to neighbor b only as long as p resides in b's queue. For the LCM-Frame policy, there is only one copy of each packet in the network, and each packet has a single one-hop source. Under centralized control, the activation schedule alerts nodes when to discard packets tracked as side information. Here, the side information buffer at node a for packets sent to node b corresponds directly to the same set of packets in the subqueue at node b for packets received from node a, and both are kept in FIFO order. Thus, when a packet is sent from a specific subqueue, the same packet can be removed from the associated side information buffer. The total network queue size then gives an upper bound on the total side information in the network, and by Theorem 2.1 the LCM-Frame policy stabilizes the network queues whenever $(\lambda_a^c) \in \Lambda_{NC}$. Therefore, the side information buffers are also stable. □

2.7 Numerical Results

We use two approaches to study the LCM-Frame policy: a packet simulation to evaluate average queue size for the policy, and a linear program (LP) solver to evaluate the flow constraints of the policy to observe coding gain. For a scenario with N nodes, there are $N-1$ possible traffic demands at each node, for a total of $N(N-1)$ traffic

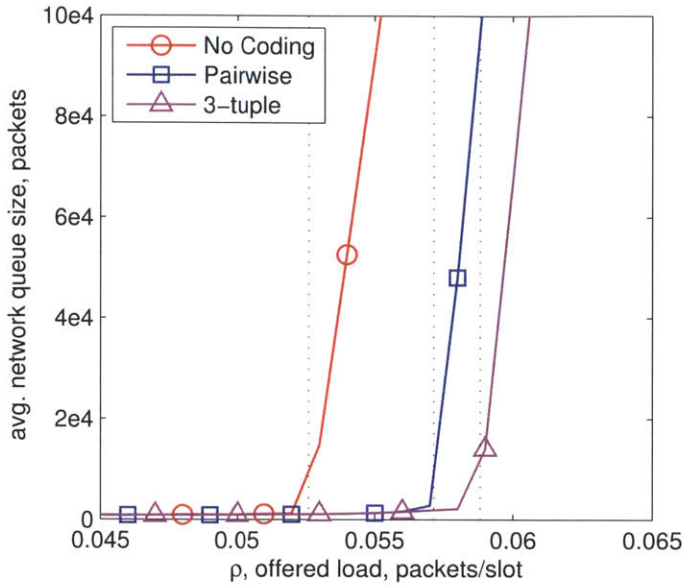


Figure 2-5: Network queue size versus offered load for three configurations of the LCM-Frame policy on 16 node topology with 11 traffic demands. Dotted vertical lines indicate bounds of the stability region for each configuration. At each value of ρ between 0.045 and 0.065 at increments of 0.001, we evaluate the LCM-Frame policy for 10 million time slots.

demands in the network. This yields 56 and 240 possible demands for $N = 8$ and $N = 16$, respectively. We generate random arrival rate vectors $\boldsymbol{\lambda}$ by activating each of these demands with probability $1/2$, where demands are specified as 1 for active and 0 for inactive. Let ρ be a value by which we scale $\boldsymbol{\lambda}$ to specify the offered load; in effect, ρ is the offered load for each active demand.

2.7.1 Simulation Results

First we consider a random 16 node topology under 2-hop interference, and we choose an arrival rate vector $\boldsymbol{\lambda}$ with 11 active traffic demands, where we scale $\boldsymbol{\lambda}$ by ρ. Exogenous arrivals are generated for each active demand using an independent Bernoulli processes; since $\boldsymbol{\lambda}$ is a vector of 0's and 1's, the scalar ρ serves as the probability of packet arrival per time slot for each active demand. We compare three configurations of the policy: no coding ($K = 1$), pairwise coding ($K = 2$), and 3-tuple coding ($K = 3$).

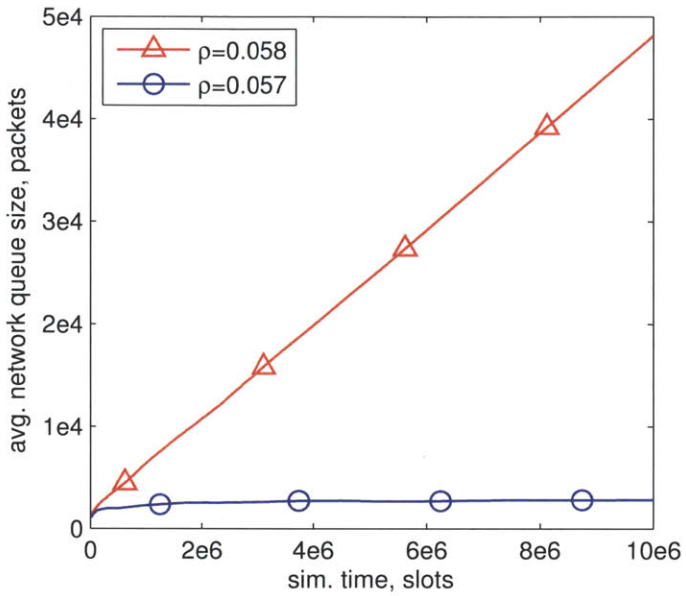


Figure 2-6: Network queue size versus simulation time for the pairwise coding configuration from Figure 2-5. Average network queue sizes shown for both stable $\rho = 0.057$ and unstable $\rho = 0.058$ values of offered load, with queue state recorded every 25 thousand time slots.

Figure 2-5 shows the time average of the total network queue size, over all nodes and commodities, as a function of offered load. Using the constraints of our stability region, Equations (2.5)-(2.9), we find the maximum stable values of offered load to be $\rho = 1/19 \approx 0.0526$ without coding, $\rho = 1/17.5 \approx 0.0571$ for pairwise coding, and $\rho = 1/17 \approx 0.0588$ for 3-tuple coding; these bounds are indicated on Figure 2-5 with vertical dotted lines. For each configuration, the policy seems to maintain bounded average queue size within the stability region.

Figure 2-6 shows average total network queue size as a function of time for pairwise coding. The queues are stable at an offered load of $\rho = 0.057$, just inside the stability region. Outside the stability region, with $\rho = 0.058$, the average network queue size grows linearly as a function of time.

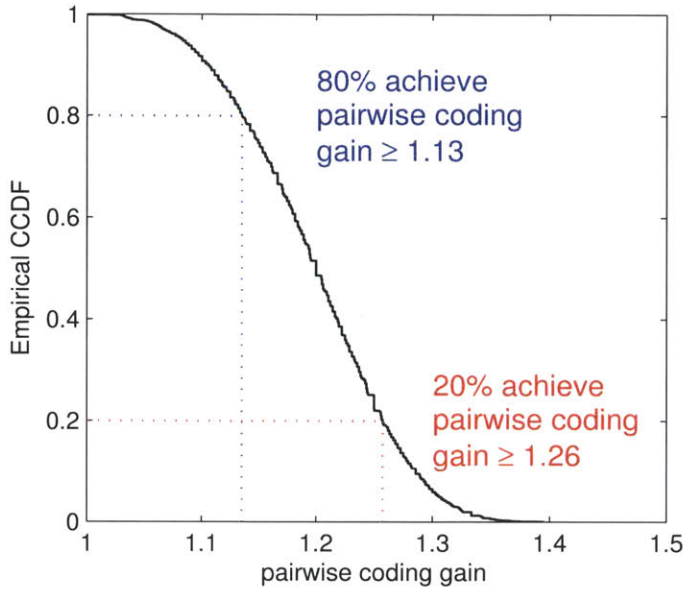


Figure 2-7: Empirical CCDF of pairwise coding gain for random 8 node topologies under 2-hop interference.

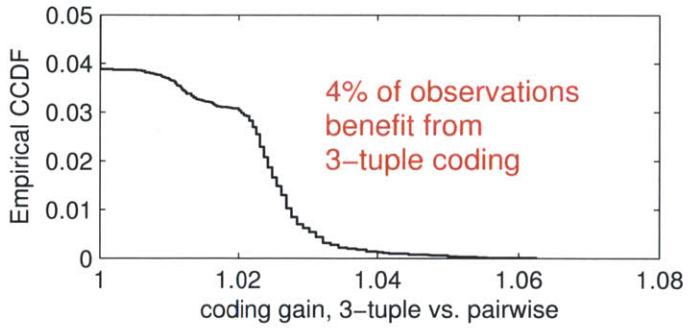


Figure 2-8: Empirical CCDF of ratio of coding gain for 3-tuple versus pairwise coding. Same topologies and arrival rate vectors as Figure 2-7.

2.7.2 Linear Program Results

We use an LP solver with constraints (2.5-2.9) of the stability region to evaluate the coding gain of our LCM-Frame policy. We consider random geometric topologies, with node placement drawn from a uniform distribution in a unit square. Node connectivity is given by a scaled unit disc model, such that two nodes are connected if they are within a certain connectivity radius of one another. In particular, we generate topologies with 8 nodes and a connectivity radius of 0.335, and topologies with 16 nodes with a connectivity radius of 0.273. For both 8 and 16 node cases, the median node degree is 3 among all nodes from the generated topologies. We consider only topologies that are connected.

First we evaluate coding gain for pairwise and 3-tuple coding. We consider 100 random topologies with 8 nodes each under 2-hop interference, and we evaluate coding gain for 100 arrival rate vectors per topology. Figure 2-7 shows an empirical complementary cumulative distribution function (CCDF) of the observed pairwise coding gain, where 80% of the observations show gain of 1.13 or more and 20% show gain of 1.26 or more. Figure 2-8 shows the ratio of 3-tuple coding gain versus pairwise coding gain. Here we see that 3-tuple coding yields additional gain in only 4% of our observations, and that this gain is limited to at most 6% and often much less.

Finally, we compare the gain of pairwise coding under 1-hop versus 2-hop interference. We evaluate 50 random topologies of 16 nodes each, with 100 random arrival rate vectors per topology. Figure 2-9 shows empirical CCDFs of pairwise coding gain for both interference models. Here, pairwise coding performs reasonably well under 1-hop interference, with a median coding gain of 1.25, and performs even better under 2-hop interference, where the median has increased to 1.31. Comparing to the 8 node scenario, there is a noticeable improvement under 2-hop interference here with 80% of observations showing gain above 1.25.

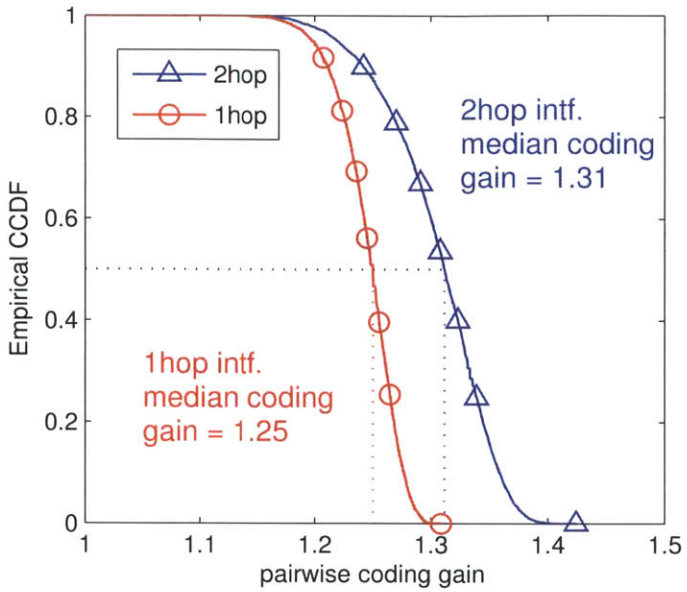


Figure 2-9: Empirical CCDFs of pairwise coding gain for random 16 node topologies under 1-hop and 2-hop interference.

2.8 Summary

In this chapter we presented a technique that dynamically optimizes for routing, scheduling, and simple network coding for wireless networks. We introduced k-tuple coding, a generalization of pairwise network coding, and provided the LCM-Frame policy, which is throughput optimal subject to the k-tuple coding constraint. We have shown achievable coding gain on simple scenarios, provided simulation results for more complex scenarios, and gave an upper bound on k-tuple coding gain for all possible scenarios.

Our main conclusion is that pairwise coding provides most of the benefit of k-tuple coding for the scenarios considered. We evaluated the LCM-Frame policy via packet simulation and LP evaluation for pairwise and 3-tuple coding. Due to the topology and traffic structure required for k-tuple coding operations, we expect limited additional gain from increasing code size k on random topologies. Note that the reduced complexity in computing weights for pairwise coding becomes significant for larger networks. We observe that the LCM-Frame policy yields greater coding gains under 2-hop interference than under 1-hop interference. Future work of interest includes

suboptimal scheduling with reduced complexity, and full system implementation.

2.A Proof of Stability for LCM-Frame Policy

Proof of Theorem 2.1. We use T-slot Lyapunov drift analysis to prove that our policy stabilizes the network for all arrival rate vectors strictly interior to the stability region. Let $A_i^{d,c}(t)$ be the number of exogenous arrivals for commodity c at node i during time slot t, where $A_i^{d,c}(t) = 0$ for any $d \neq i$. Let $\mathbf{U}(t)$ denote the matrix of queue backlogs $U_i^{d,c}(t)$ for all nodes and commodities.

At each decision time $t = nT, n \geq 0$, the LCM-Frame policy chooses which hyperedges and commodities to activate for the duration of the frame. At time $\tau \in \{t, ..., t+T-1\}$, let $\mu_{ab}^{d,c}(\tau)$ be the rate allocated for commodity c from one-hop source d over edge (a, b), where $\mu_{ab}^{d,c}(\tau) = 1$ if active and 0 otherwise. Also at time τ, let $\mu_{aJ}^{s}(\tau)$ be the rate allocated to each commodity of set s for k-tuple coded transmissions over ordered hyperedge (a, J), with traffic delivered according to the coding rule, where $\mu_{aJ}^{s}(\tau) = \frac{1}{k-1}, k = |J|$, if active and 0 otherwise. Like with flow variables, $\hat{\mu}$ represents a sum over rate allocation variables. Let $\hat{\mu}_{ab}^{d,c}(\tau)$ be the sum of rate allocation variables for coded and uncoded transmissions from a to b for commodity c packets from one-hop subqueue d, where

$$\hat{\mu}_{ab}^{d,c}(\tau) = \mu_{ab}^{d,c}(\tau) + \sum_{\left\{\substack{(a,J)\in\mathcal{H}_k, k\geq 2, s\in\mathcal{S}_k: \\ d,b\in J, c\in s, \{d,c\}\to b}\right\}} \mu_{aJ}^{s}(\tau)\,. \tag{2.18}$$

Let $\hat{\mu}_{a*}^{d,c}(\tau)$ be the sum of all transmissions at node a for packets from subqueue d for commodity c,

$$\hat{\mu}_{a*}^{d,c}(\tau) = \sum_{b} \hat{\mu}_{ab}^{d,c}(\tau)\,, \tag{2.19}$$

and let $\hat{\mu}_{ab}^{*,c}(\tau)$ be the sum of all network transmissions from a to b for commodity c

packets from all one-hop subqueues,

$$\hat{\mu}_{ab}^{*,c}(\tau) = \sum_{d} \hat{\mu}_{ab}^{d,c}(\tau)\,. \tag{2.20}$$

At decision times t_0, the queueing dynamics of the network satisfy

$$U_i^{d,c}(t_0+T) \leq \left[U_i^{d,c}(t_0) - \sum_{\tau=t_0}^{t_0+T-1} \hat{\mu}_{i*}^{d,c}(\tau)\right]^+ + \sum_{\tau=t_0}^{t_0+T-1} \left(A_i^{d,c}(\tau) + \hat{\mu}_{di}^{*,c}(\tau)\right), \tag{2.21}$$

where $[x]^+ = \max(x, 0)$. Next, we use the following result from [8, Lemma 4.3]: for $V, U, \mu, A \geq 0$ and $V \leq [U-\mu]^+ + A$, we have $V^2 \leq U^2 + \mu^2 + A^2 - 2U(\mu - A)$. Squaring both sides of Equation (2.21) and noting that $A_i^{d,c}(\tau)$, $\mu_{ab}^{d,c}(\tau)$, and $\mu_{aJ}^{s}(\tau)$ are all finite, we apply the above result to find an upper bound,

$$\left(U_i^{d,c}(t_0+T)\right)^2 \leq \left(U_i^{d,c}(t_0)\right)^2 + B_1 + 2U_i^{d,c}(t_0) \sum_{\tau=t_0}^{t_0+T-1} \left(A_i^{d,c}(\tau) + \hat{\mu}_{di}^{*,c}(\tau) - \hat{\mu}_{i*}^{d,c}(\tau)\right), \tag{2.22}$$

where B_1 is a positive finite number.

We employ the quadratic Lyapunov function,

$$L\big(\mathbf{U}(t)\big) = \sum_{i,c,d} \left(U_i^{d,c}(t)\right)^2, \tag{2.23}$$

and the following T-slot Lyapunov drift argument from [8, Lemma 4.2]: If there exists a positive integer T such that $\mathbb{E}\{\mathbf{U}(\tau)\} < \infty$ for all $\tau \in \{0, ..., T-1\}$, and if there exist positive values B and θ such that we have the following bound on T-slot Lyapunov drift for all decision times t_0,

$$\mathbb{E}\big\{L\big(\mathbf{U}(t_0+T)\big) - L\big(\mathbf{U}(t_0)\big)\big|\mathbf{U}(t_0)\big\} \leq B - \theta \sum_{i,c,d} U_i^{d,c}(t_0)\,, \tag{2.24}$$

then the network is stable according to Definition 2.2.

Using Equations (2.22) and (2.23), we find an upper bound on the T-slot difference as follows:

$$
\begin{aligned}
&L\big(\mathbf{U}(t_0+T)\big) - L\big(\mathbf{U}(t_0)\big) \\
&\leq \sum_{i,c,d}\left[B_1 + 2U_i^{d,c}(t_0)\sum_{\tau=t_0}^{t_0+T-1}\Big(A_i^{d,c}(\tau) + \hat{\mu}_{di}^{*,c}(\tau) - \hat{\mu}_{i*}^{d,c}(\tau)\Big)\right] && (2.25)\\
&\leq N^3B_1 + 2\sum_{i,c}U_i^{i,c}(t_0)\sum_{\tau=t_0}^{t_0+T-1}A_i^{i,c}(\tau) - 2\sum_{i,c,d}\left[U_i^{d,c}(t_0)\sum_{\tau=t_0}^{t_0+T-1}\big(\hat{\mu}_{i*}^{d,c}(\tau) - \hat{\mu}_{di}^{*,c}(\tau)\big)\right] && (2.26)\\
&= N^3B_1 + 2\sum_{i,c}U_i^{i,c}(t_0)\sum_{\tau=t_0}^{t_0+T-1}A_i^{i,c}(\tau) - 2\sum_{\tau=t_0}^{t_0+T-1}\sum_{i,b,c,d}\hat{\mu}_{ib}^{d,c}(\tau)\left[U_i^{d,c}(t_0) - U_b^{i,c}(t_0)\right], && (2.27)
\end{aligned}
$$

where all exogenous arrivals $A_i^{i,c}(\tau)$ are extracted from the bracketed summation in Equation (2.26). Then, the bracketed summation has been rearranged from the differential rate allocation (departures minus arrivals) for each node i in Equation (2.26) into to the differential backlog for each edge (i,b) in Equation (2.27).

At each decision time t_0, the T-slot drift is defined as

$$\Delta_T\big(\mathbf{U}(t_0)\big) \triangleq \mathbb{E}\big\{L\big(\mathbf{U}(t_0+T)\big) - L\big(\mathbf{U}(t_0)\big)\big|\mathbf{U}(t_0)\big\}. \tag{2.28}$$

Applying the upper bound from Equation (2.27) to Equation (2.28), we find an upper bound on the T-slot drift as follows.

$$
\begin{aligned}
\Delta_T\big(\mathbf{U}(t_0)\big) &\leq \mathbb{E}\left\{\begin{array}{c} N^3B_1 + 2\sum_{i,c}U_i^{i,c}(t_0)\sum_{\tau=t_0}^{t_0+T-1}A_i^{i,c}(\tau) \\ -2\sum_{\tau=t_0}^{t_0+T-1}\sum_{i,b,c,d}\hat{\mu}_{ib}^{d,c}\left[U_i^{d,c}(t_0) - U_b^{i,c}(t_0)\right] \end{array}\middle|\ \mathbf{U}(t_0)\right\} && (2.29)\\
&= N^3B_1 + 2T\sum_{i,c}U_i^{i,c}(t_0)\lambda_i^c - 2\sum_{\tau=t_0}^{t_0+T-1}\sum_{i,b,c,d}\hat{\mu}_{ib}^{d,c}(\tau)\left[U_i^{d,c}(t_0) - U_b^{i,c}(t_0)\right] && (2.30)
\end{aligned}
$$

In Equation (2.30), we have taken the expectation $\mathbb{E}\{A_i^{i,c}(\tau)|\mathbf{U}(t_0)\} = \mathbb{E}\{A_i^{i,c}(\tau)\} = \lambda_i^c$, and used a deterministic rate allocation $\hat{\mu}_{ib}^{d,c}(\tau)$ when given backlog $\mathbf{U}(t_0)$. While differential backlog routing allows for arbitrary tie-breaking when choosing rate allocation variables μ, all tie-breaking rules yield the same T-slot drift, so we assume deterministic tie-breaking for simplicity.

For any arrival rate vector $\boldsymbol{\lambda} = \left(\lambda_i^c\right)$ strictly inside of stability region Λ_{NC}, there exists a small $\epsilon > 0$ such that vector $\left(\lambda_i^c + \epsilon\right)$ is also inside the stability region. By definition of the stability region, we can identify a flow vector of $f_{ab}^{c,d}$ and f_{iJ}^{s} terms that corresponds to $\left(\lambda_i^c + \epsilon\right)$ and satisfies the constraints (2.5)-(2.9) of the stability region. For any decision time t_0 and any $\tau \in \{t_0, ..., t_0 + T - 1\}$, our policy satisfies the following inequality by choosing the set of rate allocation variables corresponding to $\hat{\mu}_{ab}^{d,c}(\tau)$ that maximize the term on the right.

$$\sum_{a,b,c,d} \hat{f}_{ab}^{d,c}\Big[U_a^{d,c}(t_0) - U_b^{a,c}(t_0)\Big] \leq \sum_{a,b,c,d} \hat{\mu}_{ab}^{d,c}(\tau)\Big[U_a^{d,c}(t_0) - U_b^{a,c}(t_0)\Big] \tag{2.31}$$

Applying the result from Equation (2.31) to Equation (2.30), we have

$$\Delta_T\big(\mathbf{U}(t_0)\big) \leq B_2 + 2T\sum_{i,c} U_i^{i,c}(t_0)\lambda_i^c - 2\sum_{\tau=t_0}^{t_0+T-1}\sum_{i,b,c,d} \hat{f}_{ib}^{d,c}\Big[U_i^{d,c}(t_0) - U_b^{i,c}(t_0)\Big] \tag{2.32}$$

$$= B_2 + 2T\sum_{i,c} U_i^{i,c}(t_0)\lambda_i^c - 2T\sum_{i,b,c,d} \hat{f}_{ib}^{d,c}\Big[U_i^{d,c}(t_0) - U_b^{i,c}(t_0)\Big] \tag{2.33}$$

$$= B_2 + 2T\Bigg[\sum_{i,c} U_i^{i,c}(t_0)\lambda_i^c - \sum_{i,c} U_i^{i,c}(t_0)\sum_b f_{ib}^{i,c} - \sum_{i,c,d\neq i} U_i^{d,c}(t_0)\Bigg(\sum_b \hat{f}_{ib}^{d,c} - \sum_g \hat{f}_{di}^{g,c}\Bigg)\Bigg] \tag{2.34}$$

$$= B_2 + 2T\Bigg[\sum_{i,c} U_i^{i,c}(t_0)\lambda_i^c - \sum_{i,c} U_i^{i,c}(t_0)\left(\lambda_i^c + \epsilon\right) - \sum_{i,c,d\neq i} U_i^{d,c}(t_0)(0)\Bigg] \tag{2.35}$$

$$= B_2 - 2T\epsilon\sum_{i,c} U_i^{i,c}(t_0)\,, \tag{2.36}$$

where $B_2 = N^3 B_1$ is finite. In Equation (2.32), the summation of time $\tau = t_0$ to

$t_0 + T - 1$ is over an argument that is fixed for the duration of the T-slot period, thus this summation of time is replaced by the scalar T. Then, from Equation (2.33), rearrange terms to yield Equation (2.34). We apply modified Flow Conservation Equations (2.10) and (2.11) to substitute $(\lambda_i^c + \epsilon)$ and (0) into Equation (2.34) to yield Equation (2.35). Finally, terms λ_i^c are canceled out to arrive at Equation (2.36), which is in the form of Equation (2.24).

We have shown that our policy satisfies Equation (2.24), and thus satisfies the conditions of [8, Lemma 4.2]. The LCM-Frame policy therefore stabilizes the network for all arrival rate vectors strictly interior to the stability region. □

Chapter 3

Distributed CSMA with Pairwise Coding

In Chapter 2, we developed a centralized control policy to jointly optimize for routing and scheduling combined with a simple network coding strategy using max-weight scheduling (MWS). In this chapter, we focus on pairwise network coding and develop a distributed carrier sense multiple access (CSMA) policy that supports all arrival rates allowed by the network subject to the pairwise coding constraint. We extend our network coding scheme by incorporating packet overhearing to increase the number of beneficial coding opportunities, and adjust our policy to also optimize for this extension. Simulation results show that the CSMA strategy yields the same throughput as the optimal centralized policy of Chapter 2, but at the cost of increased delay. Moreover, overhearing provides up to an additional 25% increase in throughput on random topologies.

3.1 Introduction

Network coding, originally introduced in [1], can increase network throughput by allowing intermediate nodes to combine or encode the data they receive, rather than simply forwarding it. The benefit of this approach for wireless transmissions was clearly demonstrated by COPE [14], an opportunistic network coding protocol that

allows encoding of packets between multiple unicast sessions using binary XOR operations. The authors combine their coding strategy with a modified MAC protocol to show significant throughput improvements versus a standard 802.11 MAC on a wireless testbed. While the original work on COPE [14] explored the interplay between coding and scheduling, subsequent work in [37] motivated the need for routing protocols to be aware of network coding by formulating an offline linear program to show that significant throughput improvements are possible. The appropriate choice of routes can increase coding opportunities and [37] shows that significant throughput improvements are possible through such coding aware routing. In this work, we address the joint design and performance of routing, scheduling, and network coding in a wireless network by developing a distributed online policy that is throughput optimal subject to our coding constraints.

Numerous previous works have considered joint routing and scheduling in the absence of network coding. In their seminal paper on network control [40], Tassiulas and Ephremides introduce the max-weight scheduling (MWS) and differential backlog routing policy to provide throughput optimal network control. The policy has an attractive property for dynamic control in that decisions rely only on current queue state information, without requiring knowledge of the long-term arrival rates. The authors are able to prove, using Lyapunov stability theory, that their policy can stabilize the network queues for any stochastic arrival process within the stability region of the network. In [27], MWS is extended to optimize for routing, scheduling, and power control in wireless networks. MWS is a very powerful scheduling technique, but the benefits do not come without cost. Even [40] notes that it can be cumbersome to collect queue state information from across a wireless network to a centralized controller. Additionally, MWS requires the solution to the maximum weight independent set (MWIS) problem, which is known to be NP-Hard under general interference constraints.

Jiang and Walrand [10] recently developed an adaptive carrier sense multiple access (CSMA) policy based on queue size information, and proved their policy to be throughput optimal. This adaptive CSMA policy is a randomized scheduler and

operates under distributed control, addressing some of the main concerns with the scalability of MWS. In [29], the adaptive CSMA scheduler is extended by relaxing some ideal assumptions from [10], maintaining throughput optimality in the presence of collisions in control traffic. An alternate proof of optimality is provided in [21] for queue-based CSMA policies on wireless networks with primary interference constraints. In [19], the authors provide another proof of CSMA rate convergence and study the effects of collisions. A throughput optimal ALOHA policy that chooses transmission probabilities as a function of queue backlog is developed in [32]. Other works ([4],[22]) have focused on distributed queue-based scheduling, and can be extended to incorporate backpressure routing. Performance bounds are characterized in [4] for a distributed maximal scheduler with imperfect matchings. A distributed scheduler that achieves 100% throughput using a randomized gossip algorithm is developed in [22].

We developed a centralized control policy based on MWS in Chapter 2 to jointly optimize for routing, scheduling, and a simple network coding scheme. Here we develop a *distributed* online queue-size based policy that is throughput optimal subject to our coding constraints. We modify the adaptive CSMA policy from [10] to incorporate a simple network coding scheme that we first proposed in [12]. We focus on pairwise coding, combined with a packet overhearing feature that can increase the number of coding opportunities with only a constant increase in algorithmic complexity. Our main contributions include:

- We propose a distributed CSMA policy for routing, scheduling, and pairwise coding that supports all arrival rates within the stability region of pairwise coding;

- We develop an extension to our coding strategy to allow for additional coding opportunities via overhearing of uncoded transmissions, and update our policy to optimize for these overhearing opportunities;

- We address several practical implementation issues, including overflow of finite precision variables and management of side information buffers;

- We provide results from packet simulation and linear program evaluation to compare the performance of our policy under various settings.

This chapter is organized as follows. We describe our system model in Section 3.2, and characterize the stability region under this model in Section 3.3. In Section 3.4 we design a distributed routing, scheduling, and pairwise coding policy. Section 3.5 adds a packet overhearing option to our coding strategy and updates the policy to take advantage of coding opportunities with overhearing. We address implementation issues in Section 3.6, provide numerical results in Section 3.7, and offer concluding remarks in Section 3.8.

3.2 Model

3.2.1 Wireless Network

We model the wireless network as a directed hypergraph, $G = (\mathcal{N}, \mathcal{H})$, where $\mathcal{N}$ is the set of nodes in the network and $\mathcal{H}$ is the set of directed hyperedges supported by the network. Hyperedge (a, J) allows node a to communicate directly with a set of tail nodes J using a single transmission, where J is always in alphabetical order. For example, in Figure 3-1a node a can transmit to nodes b and c simultaneously over hyperedge (a, J), $J = (b, c)$. Standard edge (a, b) is a special case of a hyperedge where node b is the only tail node. In this chapter we consider hyperedges with at most two tail nodes, $|J| \leq 2$ (corresponding to pairwise coding).

We consider unicast traffic, but use wireless multicast to transmit on hyperedges for network coded packets and to enable a packet overhearing feature. We assume time to be continuous, and for simplicity assume unit rate links and that exogenous arrivals are for packets of a fixed size corresponding to one time unit. Packets destined for node c are called *commodity* c packets. Let λ_a^c be the average rate of exogenous arrivals at node a for commodity c, and let $\boldsymbol{\lambda} = (\lambda_a^c)$ be a vector of arrival rates for all sources a and commodities c.

We assume that non-interfering transmissions are reliable, but otherwise allow arbitrary interference constraints. Let $\mathcal{L}$ be the set of all feasible schedules on the network. Here, schedule ℓ is a group of simultaneous (hyper)edge activations, and ℓ is feasible if these activations don't violate the network interference constraints. While our policy supports general interference models, our simulations were conducted using two simple interference models, known as 1-hop and 2-hop interference. The 1-hop interference model allows any node to transmit or receive at most one packet at a time. The 2-hop interference model requires at least two hops in the network between any simultaneous transmissions, else they will interfere.

3.2.2 Adaptive CSMA

Wireless networks are subject to packet losses from interfering transmissions, and thus benefit from a *scheduling* policy that prevents interfering transmissions from becoming simultaneously active. CSMA is a random access scheduler where each node listens to the channel for interfering transmissions, and competition for the channel is mitigated using random backoff times. Our CSMA policy is based on the policy from [10], which we extend to account for hyperedges with our coding scheme.

Jiang and Walrand [10] developed an adaptive CSMA policy that operates in continuous time, choosing exponentially distributed backoff times for each edge i as a function of the queue backlog on that edge $U_i(t)$. The policy assumes an idealized setting where each node can sense any transmission that it would interfere with and channel sensing is instantaneous. Combined with backoff times drawn from a continuous distribution[1], this ideal setting avoids packet collisions. The *backoff rate* $R_i(t)$ is updated at periodic times $t = nT$, where T is the duration of the update interval. The weight of edge i is chosen as $W_i(t) = U_i(t)$, and the backoff rate is chosen as

$$r_i(t) = \alpha \cdot W_i(nT),\ \forall t : nT \leq t < (n+1)T \tag{3.1}$$

$$R_i(t) = \exp\left(r_i(t)\right). \tag{3.2}$$

[1]The probability that any two edges choose the same backoff time from the exponential distribution is 0, independent of the edge backoff rates.

Here, $r_i(t)$ is called the *transmission aggressiveness* parameter, and α is a step size parameter controlling the convergence of the algorithm. The mean backoff time $1/R_i(t)$ decreases as the backlog increases, giving preference to transmissions on edges with higher backlog. In this policy, each edge i transitions between *idle*, *wait*, and *transmit* states as follows.

- *Idle State:* Edge i remains in the *idle* state while the channel is sensed to be busy, i.e., while an interfering edge is active. When the channel is later sensed to be inactive[2], draw a backoff timer from an exponential distribution with mean $1/R_i$ and switch to the *wait* state.

- *Wait State:* Edge i remains in the *wait* state while the channel is sensed to be inactive and the backoff timer is non-zero. If the channel becomes busy, switch to the *idle* state. Else, when the backoff timer expires switch to the *transmit* state.

- *Transmit State:* Transmit packet of unit duration[3]. When the transmission has completed, switch to the *idle* state.

3.2.3 Backpressure Routing

Combined with an optimal scheduler, backpressure routing was proved to be a throughput optimal routing strategy in [40]. The idea is simple: choose the weight of each edge as the difference in backlog across the edge for the commodity that maximizes the difference. For example, edge (a, b) has weight $W_{ab}(t)$ as follows:

$$W_{ab}(t) = \max_{c \in \mathcal{N}} [U_a^c(t) - U_b^c(t)]^+, \tag{3.3}$$

where notation $[x]^+$ represents $\max(x, 0)$. Backpressure routing was combined with adaptive CSMA for multihop traffic in [10], where the weight from Equation (3.3) is

[2]We allow for multiple simultaneous activations outside of the sensing range.

[3]Both exponentially distributed and unit duration transmissions are considered in [10]. The authors cite the main result from [18], which states that for an ideal CSMA network, edge activation frequencies are insensitive to the distributions of backoff and transmit times when given their means.

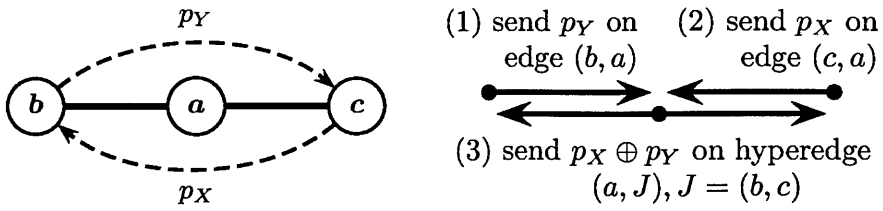
used to calculate aggressiveness parameter $r_{ab}(t)$ in Equation (3.1). The backoff rate $R_{ab}(t)$ is then calculated as in Equation (3.2).

3.2.4 Network Coding

Network coding is a technique that allows for increased throughput by encoding packets at intermediate nodes in the network. Our network coding scheme allows data to be exchanged in fewer transmissions by strategically combining packets such that each recipient has previously seen some portion of the encoded set. In Chapter 2 we described a simple network coding scheme that under specific routing conditions allows intermediate nodes to exchange k packets in $k-1$ transmissions. When evaluating this scheme on random wireless topologies, we observed that the majority of coding gains are generated by $k=2$ pairwise coding operations. Similar observations are noted in [6], [14], [20], and [38]. Therefore, here we limit our consideration to the pairwise coding case. We describe pairwise coding in the following example.

Consider the wireless network in Figure 3-1a with 1-hop interference. We would like to exchange packets p_Y and p_X between nodes b and c via a relay at node a. Without network coding it takes 2 transmissions to exchange each packet, for a total of 4 transmissions. With network coding, however, these same packets can be exchanged in only 3 transmissions: (1) send p_Y from b to a; (2) send p_X from c to a; and (3) send coded packet $p_X \oplus p_Y$ as a binary XOR combination of p_X and p_Y from node a to nodes b and c simultaneously via a single wireless multicast transmission. Using the packets that they contributed, nodes b and c can each recover the packet destined for them. In this example, network coding has increased throughput by a factor of 4/3. As in Chapter 2, here our coded transmissions are decoded hop-by-hop and each node maintains a *side information buffer* of packets that it previously transmitted (so that they can be used to decode coded transmissions).

A pairwise **coding opportunity** $\big(s,(a,J)\big)$, is formed by the combination of hyperedge (a,J), $J=(b,c)$, and commodity pair $s=(x,y)$ for which: (1) a packet of commodity x was received at node a from neighbor c, and (2) a packet of commodity y was received at node a from neighbor b. Identifying coding opportunities requires



(a) Pairwise Coding Scenario (b) Edge Activations

Figure 3-1: Pairwise coding operation at node a. (a) Standard edges shown with solid lines, with all hyperedges available; traffic demands shown with dashed arrows. (b) Edge activations shown with solid arrows.

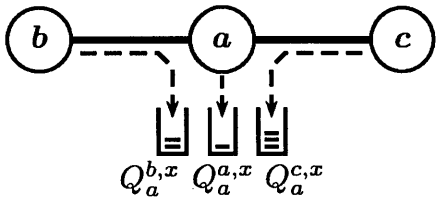


Figure 3-2: Subqueues at node a for commodity x. Subqueue $Q_a^{b,x}$ contains network arrivals from neighbor b; subqueue $Q_a^{a,x}$ contains local exogenous arrivals; subqueue $Q_a^{c,x}$ contains network arrivals from neighbor c. Packet arrivals shown in dashed arrows.

that nodes keep track of which one-hop neighbor supplied each packet. While other works on differential backlog routing (e.g., [40] and [10]) track the number of packets for each commodity at each node, we further divide the queues into *subqueues* to track the number of packets from each neighbor for each commodity. For example, subqueue $Q_a^{b,x}$ at node a contains $U_a^{b,x}$ number of packets received from neighbor b for commodity x, i.e., $U_a^{b,x} = |Q_a^{b,x}|$. This is illustrated in Figure 3-2 for commodity x packets received at node a from various sources.

3.3 Stability Region

The stability region Λ_{NC} of our network coding strategy is the set of all arrival rate vectors (λ_a^c) that can be supported while ensuring that all packet queues are stable. This region is independent of the control policy chosen, and is a special case of the stability region that we specified in Chapter 2 for network coding with maximum code size of 2. We specify the region here for convenience.

Let $f_{ab}^{d,c}$ be the rate of uncoded flow of commodity c packets, previously received from one-hop neighbor d, and sent over edge (a,b), and let f_{aJ}^{s} be the rate of coded flow over hyperedge (a,J) for each commodity in set s, where $\big(s,(a,J)\big)$ is a coding opportunity. For simplicity, we use the following $\hat{f}$ notation to represent a sum over a set of underlying flow variables. Let $\hat{f}_{ab}^{d,c}$ be the total uncoded and coded flow rate from node a to neighbor b for commodity c from subqueue $Q_a^{d,c}$, where node a received the packets from one-hop neighbor d. Thus,

$$\hat{f}_{ab}^{d,c} = f_{ab}^{d,c} + \sum_{g:s=(c,g)} f_{aJ}^{s},\ \forall a,b,c,d \in \mathcal{N},\ J=(b,d), \tag{3.4}$$

where the summation is over all commodities g such that $\big(s,(a,J)\big), s=(c,g)$, is a coding opportunity. Let $\hat{f}_{ab}^{c}$ be the total coded and uncoded flow rate from a to b for commodity c traffic from all one-hop subqueues.

$$\hat{f}_{ab}^{c} = \textstyle\sum_d \hat{f}_{ab}^{d,c}, \quad \forall a,b,c \in \mathcal{N} \tag{3.5}$$

We define the stability region by starting with some efficiency assumptions: nodes don't transmit to themselves and nodes don't transmit any traffic destined for themselves. Also, all flow variables are non-negative. The remaining constraints are equivalent to the pairwise coding configuration of the stability region from Chapter 2, as follows.

$$\lambda_a^c = \sum_b \hat{f}_{ab}^{c} - \sum_d \hat{f}_{da}^{c},\ \forall a,c \in \mathcal{N} : a \neq c \tag{3.6}$$

$$\sum_b \left(\hat{f}_{ab}^{d,c} - f_{ab}^{d,c}\right) \leq \hat{f}_{da}^{c},\ \forall\, a,c,d \in \mathcal{N} \tag{3.7}$$

$$G_{aJ} = \sum_{\ell \in \mathcal{L}} \gamma_\ell \mathcal{I}_{(a,J)\in\ell},\ \forall (a,J),\ \sum_{\ell\in\mathcal{L}} \gamma_\ell = 1,\ \gamma_\ell \geq 0\ \forall \ell \tag{3.8}$$

$$\sum_{d,c\in\mathcal{N}} f_{ab}^{d,c} \leq G_{aJ},\ \forall (a,b) : J=\{b\} \tag{3.9}$$

$$\sum_{s\in\{\mathcal{N}\}^2} f_{aJ}^{s} \leq G_{aJ},\ \forall (a,J) :\ |J|=2 \tag{3.10}$$

Equation (3.6) is the *flow conservation* constraint, stating that all flow entering any node a for commodity c must leave node a, except at the destination ($a = c$). The *coding constraint* in Equation (3.7) states that the total flow into subqueue $Q_a^{d,c}$ from node d gives an upper bound on the total *coded* flow out of $Q_a^{d,c}$ to all neighbors b. Equation (3.8) is a *convexity constraint*, stating that activation frequencies G_{aJ} for all edges and hyperedges (a, J) must be in the convex hull of the set of all feasible schedules $\mathcal{L}$. Here, indicator $\mathcal{I}_{(a,J)\in\ell} = 1$ if (a, J) is active in schedule ℓ, and 0 otherwise. The edge and hyperedge *rate constraints* in Equations (3.9)-(3.10) state that activation frequency G_{aJ} gives an upper bound on the total flow for all commodities over edge or hyperedge (a, J). The stability region for our pairwise coding strategy is the polytope bounded by the set of constraints in Equations (3.6)-(3.10).

3.4 Distributed CSMA

Our proposed policy adapts that of [10] to account for pairwise network coding as follows. The policy is parameterized for step-size α and update interval T. The policy updates backoff rate parameters every T time units and maintains *edge* timers associated with transitions between *idle*, *transmit*, and *wait* states. Each node requires backlog information only for the queues of one-hop neighbors, therefore this policy is distributed.

3.4.1 Distributed CSMA Policy for Pairwise Coding

Parameter Updates: For each edge or hyperedge (a, J), we maintain a *transmission aggressiveness* (TA) parameter $r_{aJ}(t)$ and a *backoff rate* $R_{aJ}(t)$. At times $t = nT$, for integer values of $n \geq 0$, these parameters are updated as follows.

For each standard edge (a, b), calculate edge weight $W_{ab}(t)$ as follows:

$$W_{ab}(t) = \max_{d,c}[U_a^{d,c}(t) - U_b^{a,c}(t)]^+, \tag{3.11}$$

where c^* is the optimal commodity and d^* identifies the optimal subqueue $Q_a^{d^*,c^*}$ that

maximizes Equation (3.11). TA parameter $r_{ab}(t)$ and backoff rate $R_{ab}(t)$ for edge (a, b) can then be calculated as in Equations (3.1) and (3.2).

For each hyperedge $(a, J), J = (b, g)$, calculate weight $W_{aJ}(t)$ as

$$W_{aJ}(t) = \max_x \left[U_a^{g,x}(t) - U_b^{a,x}(t)\right]^+ + \max_y \left[U_a^{b,y}(t) - U_g^{a,y}(t)\right]^+, \tag{3.12}$$

where x^* identifies the optimal commodity to send from node a to b, and y^* identifies the optimal commodity to send from node a to g. This optimal commodity pair $s = (x^*, y^*)$ and hyperedge $(a, J), J = (b, g)$, form a coding opportunity $\big(s, (a, J)\big)$ as long as (i) there is a packet at node a of commodity x^* from neighbor g, i.e., $U_a^{g,x^*}(t) > 0$, and (ii) there is a packet at node a of commodity y^* from neighbor b, i.e., $U_a^{b,y^*}(t) > 0$. Next, calculate TA parameter $r_{aJ}(t)$ and *backoff rate* $R_{aJ}(t)$ as in Equations (3.1) and (3.2).

State Transitions: The *Idle*, *Wait*, and *Transmit* states are handled as in Section 3.2.2. For a transmission on standard edge (a, b), transmit an uncoded packet p_C for optimal commodity c^* from subqueue $Q_a^{d^*,c^*}$. For a transmission on hyperedge $(a, J), J = (b, g)$, transmit a coded packet $p_{XY} = p_X \oplus p_Y$, where packet p_X is from subqueue Q_a^{g,x^*} and packet p_Y is from subqueue Q_a^{b,y^*}. If a subqueue is ever found to be empty, the policy creates a null packet to send.

3.4.2 Rate Stability

It can be shown that distributed CSMA with pairwise coding stabilizes the network for all arrival rate vectors strictly interior to the stability region Λ_{NC} specified in Equations (3.6)-(3.10). The proof follows the method shown in [10]. A sketch of this proof is given in Appendix 3.A. Whenever the packet queues are stable, the distributed CSMA policy also stabilizes all side information buffers in the network. This is clear from the discussion of maintenance operations on side information buffers in Section 3.6.4.

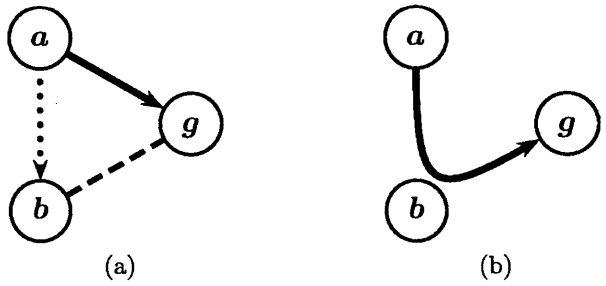


Figure 3-3: Simple packet overhearing operation. (a) Transmission from a to g, overheard by b. (b) Analogous routing scenario.

3.5 Packet Overhearing Extension

Network coding can be combined with packet overhearing to yield additional coding opportunities. Packet overhearing occurs when any nodes receive a packet concurrently with that packet's intended next-hop recipient. These additional nodes can then use their knowledge of the overheard packet in future decoding operations. The use of overhearing has been explored in [6], [14], [16], [30], and [34].

We consider a simple packet overhearing scheme to improve our network coding strategy, as shown in Figure 3-3. A transmission from node a to node g that is overheard by node b, where nodes b and g are neighbors as shown in Figure 3-3a, is analogous to a special routing operation where a transmission is sent from node a to node b to node g all at once, as shown in Figure 3-3b. We allow for overhearing of uncoded transmissions, creating two additional pairwise coding scenarios as shown in Figure 3-4. A single overhearing operation leads to the pairwise coding opportunity shown in Figure 3-4a, using edge activations in Figure 3-4c. Here, node a transmits packet p_Y to node g, and this packet is overheard by node b, allowing b to later decode the coded packet $p_X \oplus p_Y$ from g. The addition of a second overhearing operation leads to the pairwise coding opportunity shown in Figure 3-4b, using edge activations in Figure 3-4d. In addition to the overhearing at node b, node e transmits packet p_X to node g, and p_X is overheard by node c. Nodes b and c can both decode the coded packet $p_X \oplus p_Y$ from g.

A standard uncoded transmission from a to g for commodity x has weight W_{ag}

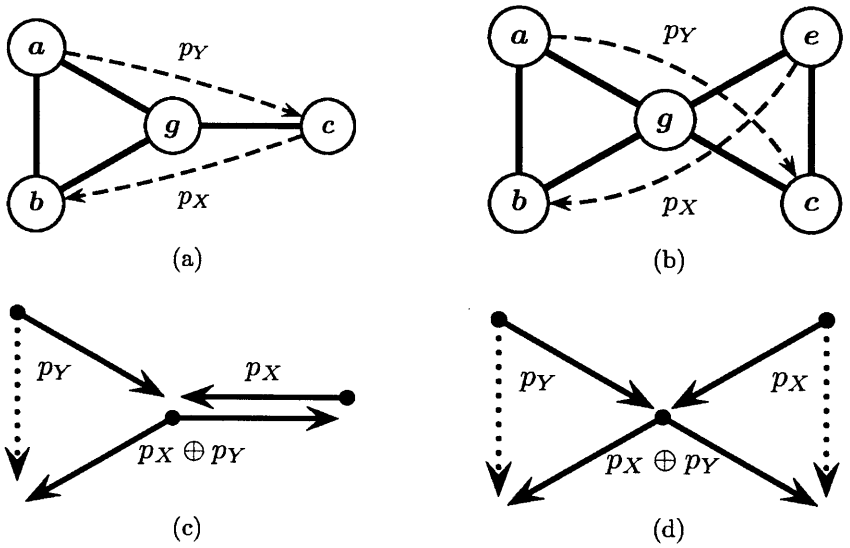


Figure 3-4: Pairwise coding scenarios with overhearing shown in (a) and (b), where solid lines indicate edges and dashed lines indicate traffic demands. Associated edge activations shown below each overhearing scenario in (c) and (d), where dotted arrows indicate overheard transmissions.

from Equation (3.13), while the same transmission overheard by node b has weight W_{abg} from Equation (3.14). (Here d is the source of the subqueue at node a containing the commodity x packet.)

$$W_{ag} = [U_a^{d,x}(t) - U_g^{a,x}(t)]^+ \tag{3.13}$$

$$W_{abg} = [U_a^{d,x}(t) - U_g^{b,x}(t)]^+ \tag{3.14}$$

The packet departs from subqueue $Q_a^{d,x}$ at node a for both the standard and overheard transmissions, while the subqueue at which the packet is received at node g depends on the transmission type. For the standard transmission from a to g, the packet enters subqueue $Q_g^{a,x}$ and a copy is stored in the side information buffer at node a. However, for the overheard transmission, the packet enters subqueue $Q_g^{b,x}$ because we treat the packet as if it was received at g from node b, as shown in Figure 3-3b. The overheard packet is then stored in the side information buffer at node b instead of at node a.

Our overhearing strategy doesn't introduce new types of network coding edge

activations, only a new type of uncoded hyperedge activation. Therefore, Theorem 2 from Chapter 2 still applies, giving an upper bound of 2 for the maximum possible coding gain from our pairwise coding strategy with overhearing.

3.5.1 Updated Stability Region for Overhearing

Overhearing leads to minor changes to the stability region. We represent the overhearing transmission as flow variable $\dot{f}^{j,c}_{dab}$, which is the flow from subqueue $Q^{j,c}_d$ at node d to node b and overheard by node a. We introduce an *Overhearing Constraint* as a prerequisite for our overhearing strategy: overhearing flow variables can only represent positive flow for hyperedges (d, J), $J = (a, b)$, where edge (a, b) is also available in the network; otherwise the overhearing flow variable must take the value of zero flow. The total uncoded and coded flows $\hat{f}^{d,c}_{ab}$ from Equation (3.4) becomes

$$\hat{f}^{d,c}_{ab} = f^{d,c}_{ab} + \sum_j \dot{f}^{j,c}_{dab} + \sum_{g:s=(c,g)} f^s_{aJ}, \quad \forall a, b, c, d \in \mathcal{N}, \quad J = (b, d)\,. \tag{3.15}$$

Equations (3.5) and *Flow Conservation* (3.6) incorporate the addition of overhearing from Equation (3.15) but otherwise remain unchanged. The *Coding Constraint* (3.7) changes to account for outgoing overheard transmissions, as follows:

$$\textstyle\sum_b \left(\hat{f}^{d,c}_{ab} - f^{d,c}_{ab} - \sum_g \dot{f}^{d,c}_{abg}\right) \;\leq\; \hat{f}^c_{da}\,, \quad \forall\, a, c, d \in \mathcal{N}\,. \tag{3.16}$$

The *Hyperedge Rate Constraints* in Equations (3.8)-(3.10) remain unchanged. However, note that we have generalized the hyperedge activation rate G_{aJ} in Equation (3.10) to include both pairwise coding and uncoded overhearing, as these both operate over hyperedges. The stability region with overhearing is then given by the constraints in Equations (3.6), (3.8)-(3.10), and (3.16).

3.5.2 Policy Modification for Overhearing

The overhearing extension requires only minor changes to how hyperedge rate parameters are handled by our distributed CSMA policy. Parameter updates for standard

edges remain unchanged, and the state transitions behave exactly as without the overhearing feature.

Parameter Updates for Hyperedges: At each time $t = nT$, for integer $n \geq 0$, for each hyperedge (a, J), $J = (b, g)$, calculate weights W_{aJ}^1, W_{aJ}^2, and W_{aJ}^3 as follows.

1. For transmissions from a to g overheard by b, calculate W_{aJ}^1 as

$$W_{aJ}^1 = \begin{cases} \max\limits_{c,d} \left[U_a^{d,c}(t) - U_g^{b,c}(t)\right]^+, & \text{if edge } (b,g) \in \mathcal{H}\,, \\ 0\,, & \text{otherwise.} \end{cases}$$

2. For transmissions from a to b overheard by g, calculate W_{aJ}^2 as

$$W_{aJ}^2 = \begin{cases} \max\limits_{c,d} \left[U_a^{d,c}(t) - U_b^{g,c}(t)\right]^+, & \text{if edge } (g,b) \in \mathcal{H}\,, \\ 0\,, & \text{otherwise.} \end{cases}$$

3. For network coded transmissions from a to b and g, calculate W_{aJ}^3 as $W_{aJ}(t)$ from Equation (3.12).

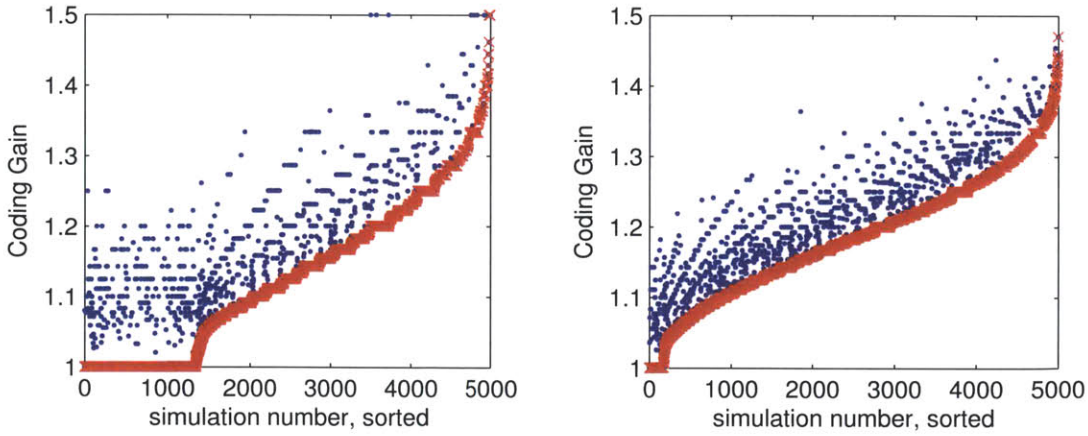
Then, choose the network coding or overhearing operation that maximizes the weight of the hyperedge,

$$W_{aJ}(t) = \max\left\{W_{aJ}^1, W_{aJ}^2, W_{aJ}^3\right\}. \tag{3.17}$$

TA parameter $r_{aJ}(t)$ and *backoff rate* $R_{aJ}(t)$ are calculated as in Equations (3.1) and (3.2).

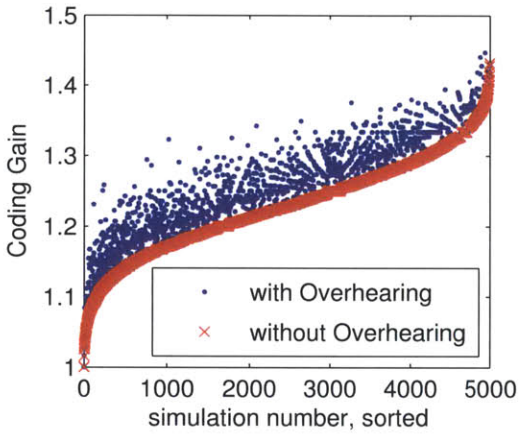
3.5.3 Linear Program Results

We compare coding gains directly by evaluating the bounds of the stability region using an LP solver. We generate 100 random 16 node topologies, where there are $16 \times 15 = 240$ possible traffic demands on each topology. We choose traffic demand vector $\boldsymbol{\lambda} \in \{0,1\}^{240}$, where each demand is activated with probability p, and find

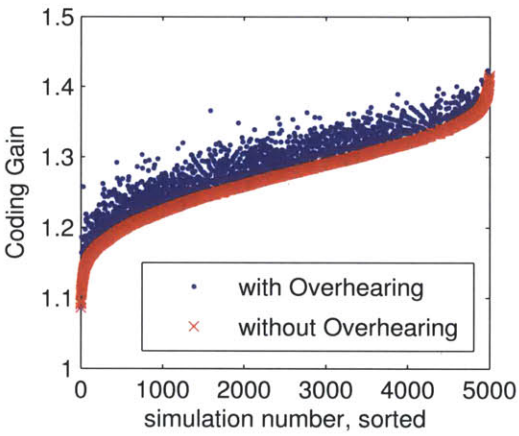


(a) $p = 1/32$

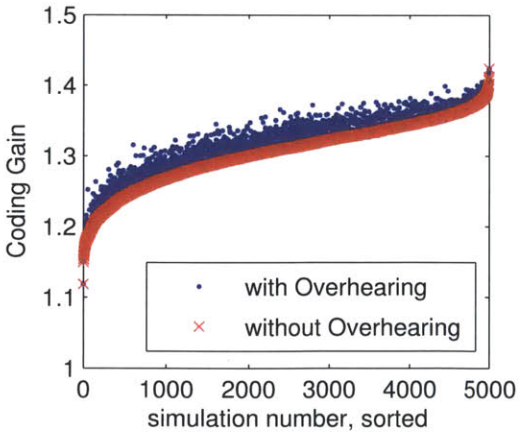
(b) $p = 1/16$



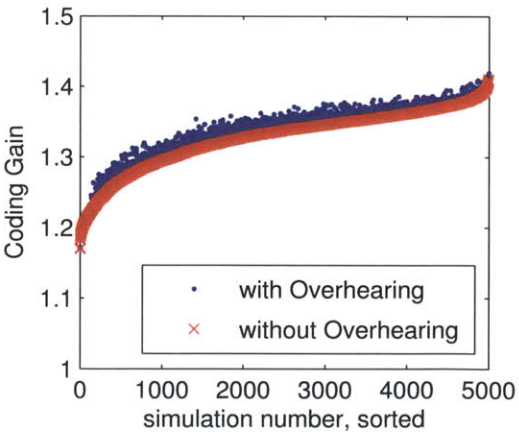
(c) $p = 1/8$



(d) $p = 1/4$



(e) $p = 1/2$



(f) $p = 3/4$

Figure 3-5: Comparison of pairwise coding gains with and without overhearing for individual traffic vectors. Random traffic demands with activation probability p. Results sorted in order of increasing gain for coding without overhearing. All graphs share same legend.

the maximum offered load without coding ρ_1 such that $\boldsymbol{\lambda} \cdot \rho_1 \in \Lambda$ and the maximum offered load with network coding ρ_{NC} such that $\boldsymbol{\lambda} \cdot \rho_{NC} \in \Lambda_{NC}$. Coding gain is then the ratio ρ_{NC}/ρ_1. These topologies are evaluated with 2-hop interference constraints.

Figure 3-5 shows coding gains for traffic demand activation probabilities p={1/32, 1/16, 1/8, 1/4, 1/2, 3/4}. For each activation probability, 5000 individual arrival rate vectors are generated (50 per topology). Coding gain is then evaluated for each arrival rate vector, both with and without overhearing. For each activation probability, the vectors are sorted in increasing order of coding gain for pairwise coding without overhearing, and the values for coding gain are plotted in that order. In Figure 3-5a where $p = 1/32$, we observe up to 25% additional gain from overhearing, although these additional gains are only present in 21% of our observations. The maximum additional gain from overhearing decreases as the probability p increases, while the frequency of occurrence increases with p. In Figure 3-5e where $p = 1/2$, the additional gain from overhearing is at most 5%, and these additional gains are present in 50% of our observations. For each activation probability p, the median additional coding gain from overhearing is less than 2%, however the small computational cost to include overhearing and the potential increase in coding gain make it a worthwhile extension. It is interesting to note that the gain from overhearing is greatest when the traffic vector is sparse. Additional traffic demands increase the likelihood of coding opportunities without the need for overheard transmissions, so overhearing provides only small incremental gains when the traffic vector is dense.

3.6 Implementation Considerations

Next we discuss some details related to implementation of the distributed CSMA policy.

3.6.1 Backoff Times

Backoff rate R_i grows exponentially with aggressiveness parameter r_i, and for any finite precision computation this can lead to overflow of variable R_i. This occurs,

Data Type of R_i	Max. R_i before overflow	Max. Associated $r_i = \log(R_i)$
Double Precision Floating Point	1.7977e+308	709.78
Single Precision Floating Point	3.4028e+38	88.72
Unsigned 64-bit Integer	1.8447e+19	44.36

Table 3.1: Maximum values for r_i before overflow of rate R_i.

for example, in the case of a bursty source node, and is exacerbated on systems that require the use of fixed-point arithmetic. Table 3.1 shows values of r_i that lead to overflow for various data types of variable R_i. When the differential backlog is large, multiple outgoing edges i can be assigned backoff rate $R_i = \infty$ and the node will not be able to correctly discriminate between exponentially distributed backoff times $B_i \sim \text{Exp}(R_i = \infty) = 0$.

Larger values of r_i can be supported by comparing logarithms of the backoff times instead of comparing the backoff times directly. We use the inverse transform method to generate backoff times $B_i \sim \text{Exp}(R_i)$ as follows. Generate random variable $Z \sim \text{Uniform}[0, 1]$, where the CDF of Z is $F_Z(z) = \mathbb{P}(Z \leq z) = z$. Then choose backoff times using the function $B_i = -\log(Z)/R_i$. The CDF of B_i is $F_{B_i}(b_i) = \mathbb{P}(B_i \leq b_i) = \mathbb{P}(-\log(Z)/R_i \leq b_i) = \mathbb{P}(Z \geq e^{-b_i R_i}) = 1 - e^{-b_i R_i}$, so B_i is exponentially distributed with rate R_i. Taking the logarithm of b_i and using $R_i = e^{r_i}$, we find

$$\log(b_i) = \log\left(\frac{-\log(z)}{e^{r_i}}\right) = \log\big(-\log(z)\big) - r_i, \tag{3.18}$$

which allows almost the full range of values supported by variable r_i, except when z is extremely close to 0 or 1. The earliest of a group of backoff times can then be chosen as

$$\min_i b_i = \exp\big(\min_i \log(b_i)\big). \tag{3.19}$$

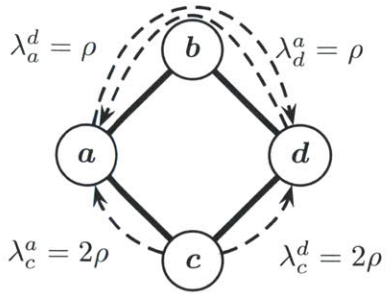
A node can then choose the minimum backoff time between interfering edges with the correct activation probabilities, or a simulation engine can choose between all waiting edges in the network. New backoff times can be drawn at each comparison due to the

memoryless property of the exponential distribution.

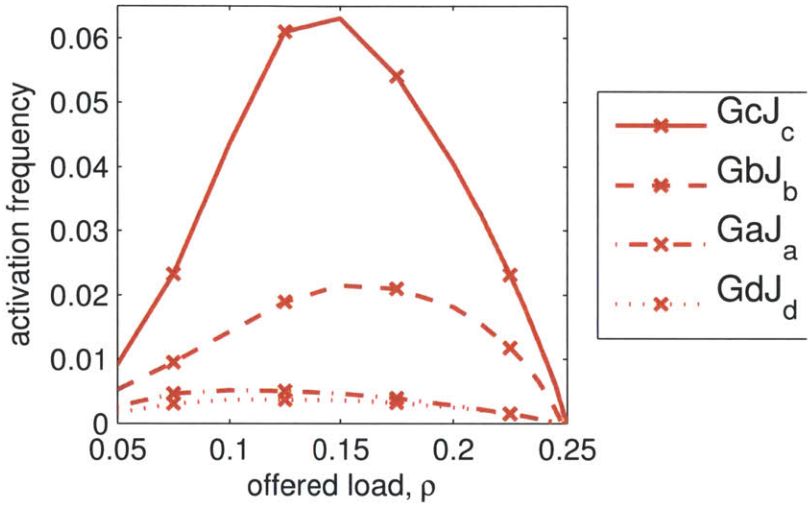
3.6.2 Avoiding Greedy Application of Network Coding

It may be tempting to opportunistically promote edge activations into hyperedge activations. However, it is known that greedy application of network coding can reduce throughput [5]. In this section, we show that our policy avoids greedy application of network coding on one such scenario, the 4 node diamond topology with 1-hop interference and arrival rates as indicated in Figure 3-6a. Here the network can be stabilized for offered loads $\rho < 1/4$. With 1-hop interference, edges (c, a) and (c, d) mutually interfere with all hyperedges in the network, (a, J_a), (b, J_b), (c, J_c), and (d, J_d), where $J_a = (b, c)$, $J_b = (a, d)$, $J_c = (a, d)$, and $J_d = (b, c)$. Thus, a greedy application of network coding on any hyperedge reduces the fraction of time that edges (c, a) and (c, d) can be active. This problem is illustrated as follows. Without loss of generality, assume that traffic only flows on efficient paths (e.g., traffic from c to a doesn't go the long way around the diamond), and let ρ be feasible. By Equation (3.9) we find activation frequency $G_{ca} \geq f_{ca}^{c,a} = 2\rho$ and likewise $G_{cd} \geq 2\rho$. Using the convexity of schedules from Equation (3.8), $G_{ca} + G_{cd} + G_{aJ_a} + G_{bJ_b} + G_{cJ_c} + G_{dJ_d} \leq 1$, and thus $G_{aJ_a} + G_{bJ_b} + G_{cJ_c} + G_{dJ_d} \leq 1 - 4\rho$. Therefore, as the offered load ρ approaches the stability bound $1/4$, all hyperedge activation frequencies must go to 0 as a prerequisite for stability.

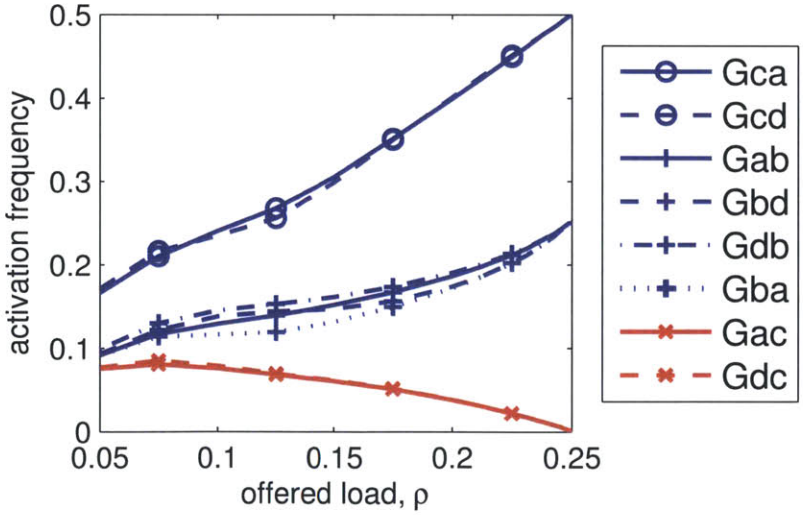
We evaluate distributed CSMA with pairwise coding on the scenario from Figure 3-6a by simulating our policy using Poisson arrivals, $\alpha = 1/10$, and $T = 10$. The simulations are run for 10 million time units for each value of offered load considered. Figure 3-6b shows the activation frequency of each hyperedge versus offered load ρ, while Figure 3-6c shows activation frequencies for standard edges in the same scenario. As ρ approaches $1/4$, we observe that G_{ca} and G_{cd} each converge to $1/2 = 2\rho$, G_{ab}, G_{bd}, G_{db}, and G_{ba} all converge to $1/4 = \rho$, and all other edges and hyperedges converge to 0, as desired. Therefore, our policy avoids greedy application of network coding for this scenario.



(a) Scenario



(b) Hyperedge Activation Frequencies



(c) Edge Activation Frequencies

Figure 3-6: The 4 node diamond scenario. Under 1-hop interference, greedy application of network coding can reduce throughput. Stability requires all activation frequencies $G_i \to 0$ for each hyperedge i as offered load approaches stability bound $\rho = 0.25$. Our policy satisfies this condition.

3.6.3 Minimum Queue Size with Network Coding

As the arrival rate vector approaches the upper bound of the stability region, our policy requires use of small values of step-size α to achieve necessary service rates. However, we observe that queues grow large for small values of α. As a function of step-size α and offered load ρ, we find a lower bound on the average network queue size required for rate convergence on the 3 node scenario in Figure 3-1a. In particular we show that the queue size must be inversely proportional to α. For simplicity of this example, let the arrival rates be symmetric, i.e., $\rho = \lambda_b^c = \lambda_c^b$, and let ρ be in the range $1/4 < \rho < 1/3$ such that pairwise coding is required to stabilize the network.

Using the result from [18], we model schedule activations of our policy as a Markov chain. In this simple 3 node scenario, at most one edge can be active at a time, so activation frequency π_i of each schedule i is the service rate for edge i. Note the convexity constraint is $\pi_\emptyset + \pi_{ba} + \pi_{ca} + \pi_{ac} + \pi_{ab} + \pi_{aJ} = 1$, where $J = (b, c)$, $\pi_i \geq 0$, and $\pi_\emptyset$ is the activation frequency of the empty schedule. By symmetry, $\pi_{ca} = \pi_{ba}$ and $\pi_{ac} = \pi_{ab}$. Combine the convexity constraint with service requirements $\pi_{ba} = \pi_{ca} \geq \rho$, and $\pi_{ac} + \pi_{aJ} \geq \rho$, we find upper bound $\pi_{ab} \leq 1 - 3\rho$. Applying this bound to service requirement $\pi_{ab} + \pi_{aJ} \geq \rho$, we find lower bound $\pi_{aJ} \geq 4\rho - 1$. Taking the ratio between π_{aJ} and π_{ab},

$$\frac{4\rho - 1}{1 - 3\rho} \leq \frac{\pi_{aJ}}{\pi_{ab}} = \frac{\pi_\emptyset R_{aJ}}{\pi_\emptyset R_{ab}} = \frac{e^{r_{ab}+r_{ac}}}{e^{r_{ab}}} = e^{r_{ac}}, \tag{3.20}$$

where $\pi_i = \pi_\emptyset R_i$ is given by the stationary distribution of the Markov chain, $R_{aJ} = \exp(r_{aJ})$, and $r_{aJ} = r_{ab} + r_{ac}$. Solving for r_{ac} yields $r_{ac} \geq \log \frac{4\rho-1}{1-3\rho}$. By a similar method, we find $r_{ba} \geq \log \frac{\rho}{1-3\rho} + r_{ac}$. By symmetry, $r_{ca} = r_{ba}$ and $r_{ac} = r_{ab}$.

When rate parameters are stable, average queue sizes can be found as follows. Applying Equation (3.1), $U_a^{b,c} = U_a^{c,b} = \frac{r_{ac}}{\alpha}$, and accounting for differential backlog, $U_b^{b,c} = U_c^{c,b} = \frac{r_{ba}+r_{ac}}{\alpha}$. The policy back-fills packets to learn the forward direction of traffic flow, so $U_b^{a,c} = U_c^{a,b} = U_a^{b,c}$. Taking a sum over all queues, a lower bound on

average network queue size is found as follows.

$$\sum_{i,c,d} U_i^{d,c} \geq \frac{2}{\alpha} \log \frac{\rho}{1-3\rho} + \frac{8}{\alpha} \log \frac{4\rho - 1}{1-3\rho} \tag{3.21}$$

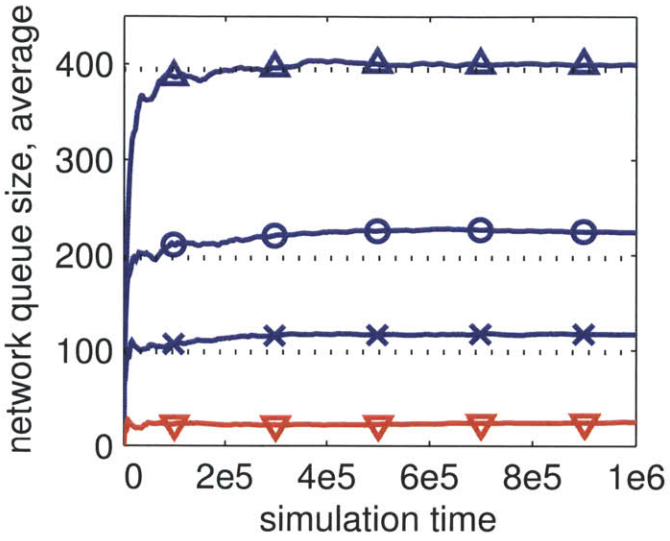
Considering offered load $\rho = 0.32$ in Equation (3.21), we find that convergence of service rates requires a minimum network queue size of $19.73/\alpha$, which is inversely proportional to α as expected from Equation (3.1). We evaluate this lower bound on network queue size for various values of α, as shown in Figure 3-7a. Simulations for this scenario are discussed in Section 3.7.

3.6.4 Managing Side Information Buffers

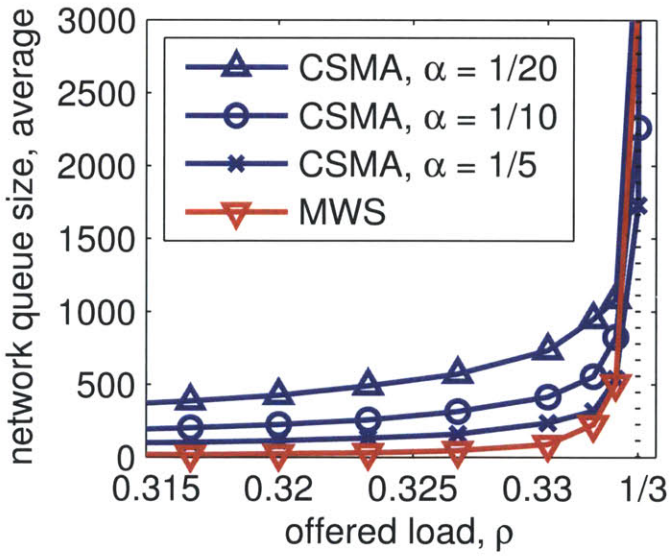
This subsection describes a distributed method to determine when packets can be discarded from side information buffers. Let $S_a^{b,c}$ be the size of the side information buffer at node a for packets sent to neighbor b for commodity c. Recall that even with packet overhearing, a copy of each transmitted packet is stored in exactly one side information buffer; in the case of packet overhearing, the side information copy is stored at the overhearing node instead of the transmitting node. The policy exchanges backlog information with neighbors every T units of time. Side information buffers are kept in FIFO order, so when node b sends backlog information $U_b^{a,c}$ to node a, the associated side information buffer at node a can be reduced such that it contains only the most recent $S_a^{b,c} = U_b^{a,c}$ packets. Without loss of generality, assume node b can transmit at most one packet at a time. Therefore node b can transmit at most T packets between sending backlog updates to node a. Thus, $S_a^{b,c} \leq U_b^{a,c} + T$, and the side information buffers are stable whenever the queues are stable.

3.7 Numerical Results

We simulate our policy using Poisson arrivals, and compare distributed CSMA with our MWS policy from Chapter 2. All configurations were simulated for 10 million time units.



(a) Queue Size vs. Time



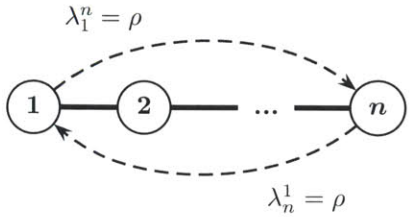
(b) Queue Size vs. Offered Load

Figure 3-7: Simulations on 3 node scenario from Figure 3-1a. Legend applies to both subplots. (a) Offered load $\rho = 0.32$ for various α. Dotted lines show lower bound on stable queue size from Equation (3.21). (b) Stability bound at $\rho = 1/3$.

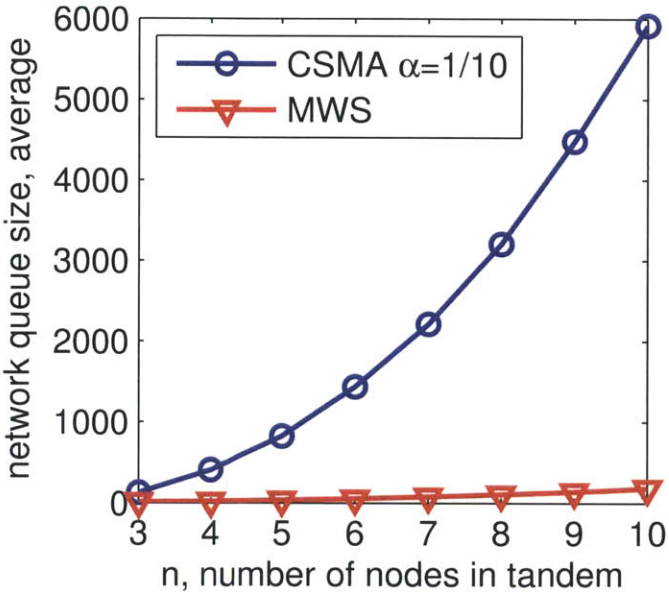
We first consider the performance of our CSMA policy on the 3 node scenario from Figure 3-1a with symmetric offered load $\rho = \lambda_b^c = \lambda_c^b$. We simulate CSMA with $\alpha = \{1/5, 1/10, 1/20\}$ and update interval $T = 10$. Figure 3-7a shows the network queue size as a function of time for offered load $\rho = 0.32$. Here we see that CSMA operates with queue size at roughly $1/\alpha$ times that of MWS. The lower bound on CSMA queue size from Equation (3.21), shown as a dotted horizontal lines, appears reasonably close to actual network queue size in this scenario. However, the distance between the bound and actual queue size will vary based on offered load and arrival process. Figure 3-7b shows average network queue size versus offered load, where the bound of the stability region is indicated with a dashed vertical line at $\rho = 1/3$. For all configurations, we see that queues remain relatively small when the offered load is interior to the stability region, and the queues grow large as the offered load approaches the stability bound.

We next consider how queue sizes scale with the number of nodes n on a tandem configuration with symmetric end-to-end traffic, as shown in Figure 3-8a. We configure CSMA with $\alpha = 1/10$, $T = 10$, $\rho = \lambda_1^n = \lambda_n^1 = 0.3$, and evaluate this scenario under the 1-hop interference model. Figure 3-8b shows average network queue size for our CSMA and MWS policies on networks with $n = \{3, 4, ..., 10\}$ nodes. For both policies we observe that the queues grow quadratically with number of nodes n due to differential backlog routing, which is consistent with findings from [3]. The ratio between CSMA and MWS network queue sizes is roughly 10 for $n = 3$ nodes and increases to around 30 for $n = 10$ nodes.

Finally, we consider queue size versus offered load for a 16 node scenario with 11 traffic demands as shown in Figure 3-9a, with 2-hop interference. (This is the same scenario considered in Figure 2-5 of Chapter 2.) MWS results are shown on Figure 3-9b, while CSMA results are shown on Figure 3-9c. The dotted vertical lines indicate the bounds of the stability region (computed using an LP solver) at $\rho = 1/19$ without coding, $\rho = 1/17.5$ for pairwise coding, and at $\rho = 1/16$ for pairwise coding with overhearing. This yields a pairwise coding gain of $19/17.5 = 1.086$ without overhearing and $19/16 = 1.188$ with overhearing. We see that the queues remain

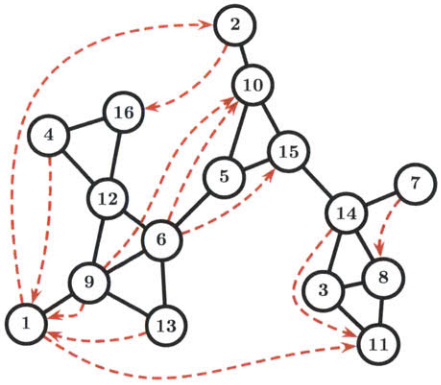


(a) n Node Tandem Scenario

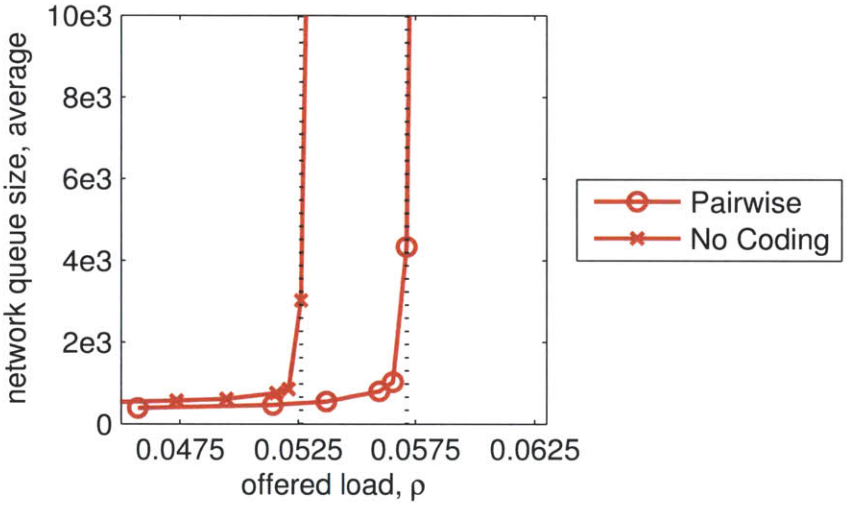


(b) Packet Simulation

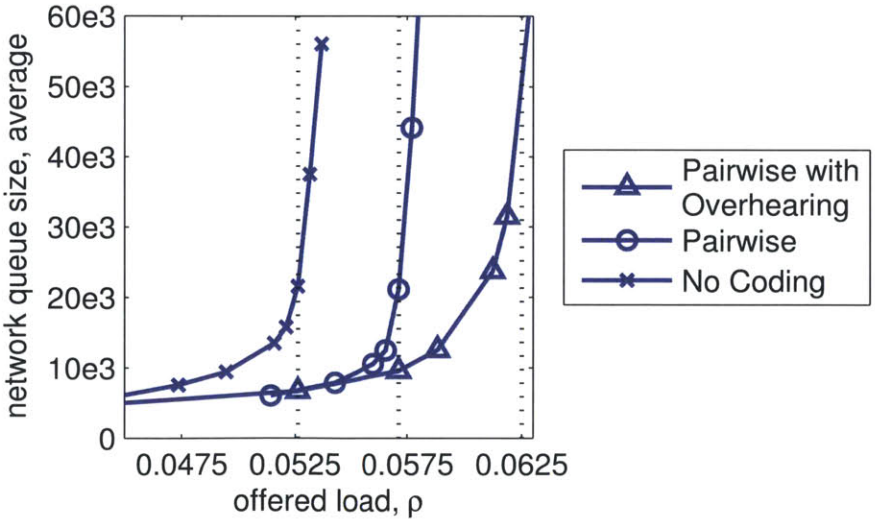
Figure 3-8: Average network queue size for pairwise coding on tandem network with increasing number of intermediate nodes with symmetric end-to-end traffic, as shown in (a). Results in (b) show quadratic growth in queue size as a function of the number of nodes, n, for both MWS and CSMA policies.



(a) Scenario



(b) MWS



(c) CSMA

Figure 3-9: Comparing MWS and CSMA for a 16 node scenario. (a) Traffic demands as dashed lines with arrows. (b)&(c) Stability bounds as dotted vertical lines.

relatively small for values of ρ interior to the stability bound, and the queues grow rapidly when ρ exceeds the bound. We also observe that CSMA queues operate at between 10 and 20 times those for MWS, although this will vary with α.

3.8 Summary

In this chapter, we considered distributed techniques for joint routing, scheduling, and pairwise network coding to maximize throughput in wireless networks. We presented the distributed CSMA policy for pairwise coding, and showed that this policy can come arbitrarily close to supporting the full stability region allowed by our coding constraint. We developed a packet overhearing extension to increase the number of beneficial coding opportunities and evaluated our policy with and without overhearing on multiple scenarios. On random scenarios we find the additional gains from our overhearing scheme are low on average at around 2%, but occasionally we observe larger gains of up to 25% that make this simple extension worthwhile.

In comparing performance of our CSMA and MWS policies, we find that the distributed control of the CSMA policy comes at the expense of growth in average queue size. For a simple pairwise coding scenario, we provide a lower bound on stable CSMA queue size as a function of the offered load and α. This bound is inversely proportional to α, and we found it useful for approximating the network queue size in our simulations. We evaluated stable queue size as a function of the number of nodes in a tandem network, and observe quadratic growth in stable CSMA queue size. While MWS also experiences quadratic growth in queue size, the growth rate is noticeably faster for CSMA.

3.A Rate Stability

Using appropriate choices for parameters α and T, we wish to show that for any strictly feasible arrival rate vector λ and any flow decomposition $\hat{f}$, the distributed CSMA policy chooses TA parameters r_i such that service $s_i(r)$ dominates arrivals $\hat{f}_i$

for each edge i. Here, λ is strictly feasible if $(\lambda + \epsilon) \in \Lambda_{NC}$, for $\epsilon \geq 0$, and $\hat{f}$ is a flow decomposition of λ according to Equations (3.6)-(3.10). First, we show that if a solution is attainable for finite r^*, then $s_i(r^*) \geq \hat{f}_i$, $\forall i$. Second, we show that the solution is attainable whenever the arrival rate is strictly feasible. Combining the first and second steps gives the desired result.

Let γ_ℓ be an activation probability for schedule ℓ satisfying flow decomposition $\hat{f}_i$, and let $\pi_\ell(r)$ be the actual activation frequency of each schedule ℓ according to service rates $s_i(r)\forall i$. Indicator $\mathcal{I}_{i\in\ell} = 1$ if edge i is active in schedule ℓ, and 0 otherwise. Then $\hat{f}_i = \sum_\ell \gamma_\ell \mathcal{I}_{i\in\ell}$. Again using the result from [18], we model schedule activations of our policy as a continuous time Markov chain, where the schedule activation frequencies conditioned on r_i are given by Equations (3.22) and (3.23).

$$\pi_\ell(r) = \exp(\textstyle\sum_i r_i \mathcal{I}_{i\in\ell}) \,/\, C(r) \tag{3.22}$$

$$C(r) = \textstyle\sum_j \exp(\sum_i r_i \mathcal{I}_{i\in j}) \tag{3.23}$$

We can minimize the Kullback-Leibler divergence between distributions γ and $\pi(r)$ by solving $\sup_{r\geq 0} F(r)$, where $F(r)$ is non-positive for $r \geq 0$ and is defined as

$$F(r) = \textstyle\sum_\ell \gamma_\ell \log\left(\pi_\ell(r)\right) = \sum_i \hat{f}_i r_i - \log(C(r))\,. \tag{3.24}$$

Note that $\frac{\partial}{\partial r_i} F(r) = \hat{f}_i - s_i(r)$, so a *distributed* gradient algorithm to solve $\sup_{r\geq 0} F(r)$ is shown in Equation (3.25).

$$r_i(n+1) = [r_i(n) + \alpha(n)(\hat{f}_i - s_i(r(n)))]^+,\ \forall i \tag{3.25}$$

Choosing $r_i(0) = 0$, $\alpha(n) = \alpha$, interval n of duration T, and observing that $\hat{f}_i$ and $s_i(r)$ correspond to queue arrivals and departures, respectively, we obtain $r_i(nT) = \alpha U_i(nT)$. This is in the form of Equation (3.1).

Existence Proposition: If $r^* \geq 0$ exists such that $F(r^*) = \sup_{r \geq 0} F(r)$, then $s_i(r^*) \geq \hat{f}_i$, $\forall i$. Dualize each constraint $r_i \geq 0$ with dual variables $d_i \geq 0$:

$$\mathcal{L}(r, d) = F(r) + \sum_i d_i r_i \,. \tag{3.26}$$

At solution r^* we have

$$\begin{aligned} \frac{\partial}{\partial r_i} \mathcal{L}(r^*, d^*) &= \hat{f}_i - s_i(r^*) + d_i^* \\ &= 0 \,. \end{aligned} \tag{3.27}$$

We know $d_i \geq 0$, so $s_i(r^*) \geq \hat{f}_i \; \forall i$.

Attainability Proposition: If λ is strictly feasible, then $F(r^*) = \sup_{r \geq 0} F(r)$ is attainable. In [11], the dual of $\sup_{r \geq 0} F(r)$ was found as the following program.

$$\begin{aligned} \max_u \quad & -\sum_\ell u_\ell \log(u_\ell) \\ \text{s.t.} \quad & \sum_\ell (u_\ell \mathcal{I}_{i \in \ell}) \geq \hat{f}_i, \forall i \\ & \sum_\ell u_\ell = 1, \; u_\ell \geq 0 \end{aligned} \tag{3.28}$$

Strict feasibility of λ satisfies the Slater condition, giving existence of finite values for Lagrangian dual variables of all constraints in (3.28): $y_i^* \geq 0$, $w_\ell^* \geq 0$, and z^*. The optimal value for Equation (3.28) occurs when

$$u_\ell^* = \frac{\exp(\sum_i y_i^* \mathcal{I}_{i \in \ell})}{\sum_j \exp(\sum_i y_i^* \mathcal{I}_{i \in j})}, \quad \forall \ell, \tag{3.29}$$

where y_i is the dual variable for constraint $\sum_\ell (u_\ell \mathcal{I}_{i \in \ell}) \geq \hat{f}_i$. Observe that u_ℓ^* is in the form of $\pi_\ell(r^*)$ from Equations (3.22)-(3.23), where $y_i^* = r_i^* \; \forall i$. Then the optimal value for Equation (3.28) equals $F(r^*)$ and is obtained whenever λ is strictly feasible.

Combining the two propositions: If λ is strictly feasible, then $s_i(r) \geq \hat{f}_i$, $\forall i$. Note that for fixed values of parameters α and T, we are only guaranteed that the

service rates will converge to the neighborhood of the link arrivals $\hat{f}_i$. For rate stability, it is sufficient for the convergence neighborhood to be fully contained in the stability region. By assumption, arrival rates are strictly feasible, so there always exists a value of α small enough that the neighborhood of convergence is fully within the stability region. Thus, the parameterized policy can come arbitrarily close to supporting the full stability region.

Note that for pairwise coding, e.g., in Figure 3-1a, we have assumed that TA parameter for hyperedge $(a, J), J = (b, c)$, is $r_{aJ} = r_{ab} + r_{ac}$. This assumption is confirmed by verifying that the total service rate $s_{ab}(r)$ on edge (a,b) is $\pi_{ab} + \pi_{aJ}$:

$$
\begin{aligned}
s_{ab}(r) &= \frac{\partial}{\partial r_{ab}} \log\big(C(r)\big) && (3.30)\\
&= \big(\exp(r_{ab}) + \exp(r_{ab} + r_{ac})\big)/C(r) && (3.31)\\
&= \pi_{\emptyset}\big(R_{ab} + R_{aJ}\big) && (3.32)\\
&= \pi_{ab} + \pi_{aJ}, && (3.33)
\end{aligned}
$$

where $\pi_{\emptyset} = 1/C(r)$.

Chapter 4

Backpressure Routing in Overlay Networks

In Chapters 2 and 3, we considered the use of network coding to increase the stability region for wireless networks. In this chapter, we consider routing strategies for maximizing throughput in legacy networks where a subset of legacy nodes is replaced by nodes aware of network control policies.

While differential backlog routing is known to be a throughput optimal routing policy, it typically requires a homogeneous network where all nodes participate in control decisions. We model a set of controllable nodes as a network overlay operating within a legacy network, and characterize the throughput region of the network as a function of this set of controllable nodes. Our goal is to increase achievable throughput in the network by using control policies in the network overlay to enable multiple-path routing. As a motivation, we find that ring networks require exactly 3 controllable nodes to enable the same throughput region as when all nodes are controllable, independent of the total number of nodes in the network. We develop algorithms to choose the minimum number of controllable nodes required to enable the full throughput region of networks with shortest-path routing, and these algorithms are evaluated on several classes of regular and random graphs. In the case of random networks with a power-law degree distribution, which is a common model for the Internet, we find that fewer than 80 out of 1000 nodes are required to be

controllable to enable the full throughput region.

Since standard backpressure routing cannot be directly applied to the overlay setting, we develop a heuristic extension to backpressure routing that determines how to route packets between overlay nodes. Simulation results confirm that maximum throughput can be attained with our policy in several scenarios, when only a fraction of legacy nodes are replaced by controllable nodes. Moreover, we observe reduced delay relative to the case where all nodes are controllable and operate under backpressure routing.

4.1 Introduction

Backpressure routing has been studied for decades, however adoption of this throughput optimal policy has not been embraced for general use on the Internet. This is due, in part, to a difficulty for backpressure routing to coexist with legacy routing protocols. With few exceptions, backpressure routing is studied in homogeneous networks, where all nodes are dynamically controllable and implement a backpressure policy across all nodes uniformly. As will be shown, backpressure routing — also known as differential backlog routing, as proposed in [40] — is suboptimal when applied only to a subset of nodes in the network. A key problem with deployment in heterogeneous networks is the possibility of encountering black hole routes, which are typically assumed to not exist in a homogeneous network control scenario.

We would like to enable network control policies to be deployed in existing networks, alongside legacy nodes that are unaware of our control policies. There are many reasons to integrate controllable nodes into heterogeneous networks, not the least of which is the financial cost of replacing all nodes at once. Other reasons include a need to maintain compatibility with current applications and special purpose hardware, a lack of ownership to decommission legacy equipment, and a lack of administrative privilege to modify existing software.

Conceptually, we model controllable nodes as operating in a network overlay on top of a legacy network. Network overlays are frequently used to deploy new commu-

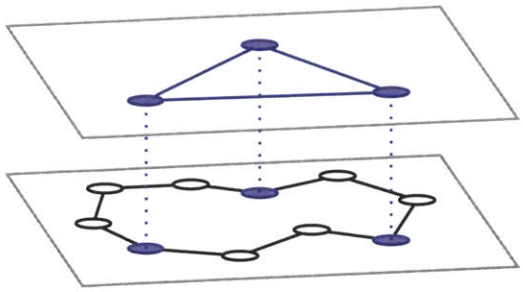


Figure 4-1: Example of a network overlay. The bottom plane shows the full network graph, while the top plane shows a subset of network nodes and their conceptual overlay connectivity.

nication architectures in legacy networks [31]. To accomplish this, messages from the new technology are encapsulated in the legacy format, allowing the two methods to coexist in the legacy network. Nodes making use of the new communication methods are then connected in a conceptual network overlay that operates on top of the legacy network, as shown in Figure 4-1. The most predominant example of a network overlay is the Internet, which was previously connected as a network overlay on top of the public telephone networks (e.g., via dial-up modems). Lately, this situation has reversed such that telephone communications now largely operate as a network overlay on top of the Internet (e.g., via Voice-over-IP).

Several works have considered the use of network overlays to improve routing on the Internet. Andersen *et al.* [2] motivate the need for resilient overlay networks (RON) to find paths around network outages on a faster timescale than BGP. Their method deploys a group of RON nodes as an application-layer overlay across various routing domains, continuously monitoring the quality of paths in the RON to decide which routes to use. Similarly, Han *et al.* [9] proposed a method for choosing placement of overlay nodes to improve path diversity in overlay routes. While both of the preceding works show that their strategies choose high quality single-path routes, we would like to go further and identify multiple-path routes that offer maximum throughput.

Delay reduction for backpressure routing has been studied in a variety of scenarios. While multiple-path routes are required to support the full throughput region, the ex-

ploratory phase of backpressure routing can lead to large queues when the offered load is low and single-path routes would suffice. Neely, Modiano, and Rhors [26] propose a hybrid policy combining backpressure routing with shortest-path routing, where flows are biased towards shortest-path routes, yet still support the full throughput region. Khan, Le, and Modiano [15] extend this hybrid policy to also include digital fountain codes, and show their policy to achieve minimum end-to-end delay in the presence of random link failures. Ying, Shakkottai, and Reddy [44] develop a policy that achieves a similar shortest-path result by minimizing the average path length used by flows. In a scenario with multiple clusters that are intermittently connected, Ryu, Ying, and Shakkottai [33] combine backpressure routing with source routing in a network overlay model to separate the queue dynamics of intra-cluster traffic from longer inter-cluster delays. Bui, Srikant, and Stolyar apply shadow queues [3] to allow the use of per-neighbor FIFO queues instead of per-commodity queues, as is typical with differential backlog routing, and find that this can improve network delay.

In this chapter, we consider two problems in the area of control for heterogeneous networks. First, we develop algorithms for choosing the placement of controllable nodes, where our goal here is to allocate the minimum number of controllable nodes such that the full network throughput region is available. Given a graph G with nodes $\mathcal{N}$ supporting shortest-path routes between each pair of nodes, we wish to identify a set of controllable nodes $\mathcal{V} \subseteq \mathcal{N}$ to maximize throughput. Ideally, we would like to solve P1,

$$\begin{aligned} V_1^* = \min_{\mathcal{V} \subseteq \mathcal{N}} \quad & |\mathcal{V}| \\ \text{s.t.} \quad & \Lambda_G(\mathcal{V}) = \Lambda_G(\mathcal{N}) \ , \end{aligned} \tag{P1}$$

where $\Lambda_G(\mathcal{V})$ is the throughput region of graph G when only nodes $\mathcal{V}$ are controllable, while $\Lambda_G(\mathcal{N})$ is the throughput region when all nodes are controllable. Note that comparing throughput regions directly can be difficult, so instead we identify a condition that is sufficient to guarantee the full throughput region, and minimize placement of controllable nodes subject to this condition. Second, we develop a backpressure

routing policy that avoids black hole routes in heterogeneous networks, with only mild assumptions about the behavior of the underlying scheduling algorithm. Our solutions for the first and second problems are complementary, in the sense that they can be used together to solve the joint problem. However, our node placement algorithm can be used with other policies, and our backpressure policy does not depend on the results of the node placement algorithm.

This chapter is organized as follows. Section 4.2 describes our system models, and Section 4.3 characterizes the throughput region under these models. We develop a node placement algorithm in Section 4.4, and offer simulation results for this algorithm in Section 4.5. A backpressure policy for heterogeneous networks is designed in Section 4.8, and we offer concluding remarks in Section 4.9.

4.2 Model

4.2.1 Network

We model the network as a directed graph $G = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N}$ is the set of nodes in the network and $\mathcal{E}$ is the set of edges. Networks drawn as undirected graphs are implied to have both directed edges (i, j) and (j, i) for every pair of nodes i and j shown as being connected. For simplicity, we assume slotted time and we use unit rate links with fixed packet sizes corresponding to one packet per time slot. Our simulation results will assume wired networks that are interference free, such that all edges can be activated simultaneously. Finally, packet transmissions are assumed to be reliable.

4.2.2 Uncontrollable Nodes

We assume that the underlay network provides a fixed realization for shortest-path routes between all pairs of nodes, and that uncontrollable nodes will forward traffic only along the given shortest-path routes. Further, we assume that only one path is provided between each pair of nodes. Note that shortest-paths are necessarily

acyclic[1] provided non-negative edge costs. Let P_{ab}^{SP} be the path from a to b provided by the underlay network, and let $\mathcal{P}^{\text{SP}} = (P_{ab}^{\text{SP}})$ for all $a, b \in \mathcal{N}$ be the set of all paths provided by the underlay network. Assume the underlay provides all edges $(i, j) \in \mathcal{E}$ as single-hop paths P_{ij}^{SP}.

Optimal substructure is assumed for shortest-paths, such that if shortest-path P_{ac}^{SP} from node a to c includes node b, then path P_{ac}^{SP} includes shortest-paths P_{ab}^{SP}, from a to b, and P_{bc}^{SP}, from b to c. This optimal substructure is consistent with shortest-paths in OSPF, a widely used routing protocol based on Dijkstra's shortest-path algorithm [31], where OSPF allows for the use of lowest next-hop router ID as a method for choosing between multiple paths of equal length.

4.2.3 Controllable Nodes

Next, we consider the subset of nodes $\mathcal{V} \subseteq \mathcal{N}$, called *overlay* or *controllable* nodes, which can perform dynamic routing decisions to direct packets to the destination or other controllable nodes along the provided shortest-path routes. Intuitively, these nodes $\mathcal{V}$ can improve throughput performance by generating new paths and enabling multi-path routing. The remaining uncontrollable nodes $u \in \mathcal{N} \setminus \mathcal{V}$ provide only shortest-path routing in the underlay network, with an exception that any uncontrollable node u can participate in dynamic routing for all traffic that originates at u or is destined for u; this may occur, for example, in the source and destination applications at uncontrollable nodes, or in a shim-layer between the network-layer and application-layer. Without such an exception, all sources and destinations may be required to be controllable nodes (e.g., in a ring network, where a source may need to bifurcate traffic between the clockwise and counterclockwise paths to a destination), in which case supporting the full throughput region would necessitate that $\mathcal{V} = \mathcal{N}$. However, only controllable nodes $\mathcal{V}$ can participate in dynamic routing for traffic received over the network.

Controllable nodes can increase the achievable throughput region by admitting new paths to the network as concatenations of existing paths from shortest-path

[1] Acyclic paths are also known as simple paths.

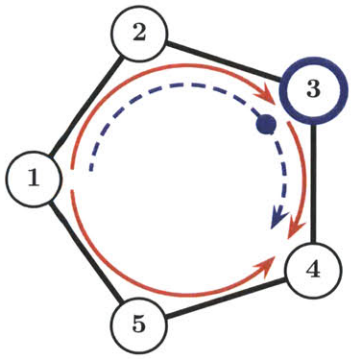


Figure 4-2: Example of 2-concatenation on a 5 node ring network, where node 3 is controllable. A subset of the shortest-paths $\{P_{13}^{\text{SP}}, P_{14}^{\text{SP}}, P_{34}^{\text{SP}}\} \in \mathcal{P}^{\text{SP}}$ are shown as solid red arrows. Path P_{14}^{concat} is a 2-concatenation of paths P_{13}^{SP} and P_{34}^{SP} at controllable node 3, and is shown as a dashed blue arrow. Here, $P_{14}^{\text{concat}} \in \mathcal{P}(\mathcal{V})$, but $P_{14}^{\text{concat}} \notin \mathcal{P}^{\text{SP}}$.

routing. A *2-concatenation* of shortest-paths P_{av}^{SP} and P_{vb}^{SP} is an acyclic path from a to b, P_{ab}, where $v \in \mathcal{V}$ is a controllable node and v is the only node shared between shortest-paths P_{av}^{SP} and P_{vb}^{SP}. Note that a 2-concatenation of acyclic paths will always be acyclic, as we only allow the concatenated paths to share the overlay node v at which concatenation is performed. An *n-concatenation* is then the concatenation of n shortest-paths at $n-1$ controllable nodes, performed as a succession of $(n-1)$ 2-concatenations. Following the rule that a 2-concatenation can be performed only on acyclic paths that share only the concatenation node, an n-concatenation is also always acyclic. Consider the set of paths $\mathcal{P}(\mathcal{V})$, which contains all underlay paths $\mathcal{P}^{\text{SP}}$ as well as all possible n-concatenations of these paths at the controllable nodes $\mathcal{V}$. We will see that this set $\mathcal{P}(\mathcal{V})$ plays a role in the achievability of the throughput region. An example of a 2-concatenation path in $\mathcal{P}(\mathcal{V})$ is shown in Figure 4-2.

4.3 Throughput Region

The *throughput region* $\Lambda_G(\mathcal{V})$ is the set of all arrival rates that can be achieved by any policy implemented at controllable nodes $\mathcal{V}$ on graph G. For the case where all nodes are controllable, i.e., $\mathcal{V} = \mathcal{N}$, the throughput region equals the stability region of graph G. This section characterizes the throughput region that corresponds to the set of paths $\mathcal{P}(\mathcal{V})$, i.e., all shortest-paths and all acyclic concatenations of shortest-

paths at controllable nodes $\mathcal{V}$. The throughput region is the set of all arrival rate vectors that can be supported by such paths on the network. Recall the following two properties about underlay paths: (i) a shortest-path P_{ab}^{SP} exists for every pair of nodes a and b, and (ii) edge $(a,b) = P_{ab}^{\text{SP}}$ for all edges $(a,b) \in \mathcal{E}$, i.e., all 1-hop paths are in the set $\mathcal{P}^{\text{SP}}$.

Packets destined for node c are called *commodity* c packets. Let λ_a^c be the rate of exogenous arrivals at node a for commodity c, and let $\boldsymbol{\lambda} = (\lambda_a^c)$ be the multicommodity arrival rate vector for all sources a and commodities c. All flow variables are non-negative, where $f_{ij}^{ab,c}$ is the *edge-flow* rate for commodity c on edge (i,j) along the shortest-path from node a to b. Flow for a path is allowed only on the edges along that path, i.e., $f_{ij}^{ab,c} = 0$ unless $(i,j) \in P_{ab}^{\text{SP}}$. Let $\bar{f}_{ab}^c$ be *path-flow* rate for commodity c along shortest-path P_{ab}^{SP}, from node a to node b. Decision variable $v_i = 1$ if node i is controllable, and $v_i = 0$ otherwise, for all nodes $i \in \mathcal{N}$. The capacity of edge (i,j) is R_{ij}. The controllable throughput region $\Lambda_G(\mathcal{V})$ is then the set of all arrival rate vectors $\boldsymbol{\lambda} = (\lambda_a^c)$ such that Equations (4.1)-(4.6) can be satisfied.

Flow Conservation:

$$\lambda_v^c = \sum_{b \in \{c, \mathcal{V} \setminus v\}} \bar{f}_{vb}^c - \sum_{d \in \mathcal{V} \setminus v} \bar{f}_{dv}^c \ , \quad \forall\, v \in \mathcal{V}, c \in \mathcal{N} \setminus v \tag{4.1}$$

$$\lambda_u^c = \sum_{b \in \{c, \mathcal{V}\}} \bar{f}_{ub}^c \ , \quad \forall\, u \in \mathcal{N} \setminus \mathcal{V}, c \in \mathcal{N} \setminus u \tag{4.2}$$

Path Constraint:

$$\bar{f}_{ab}^c = f_{ij}^{ab,c} \ , \quad \forall\, (i,j) \in P_{ab}^{\text{SP}},\ \forall\, a,b,c \in \mathcal{N} \tag{4.3}$$

Overlay Neighbor Constraints:

$$f_{ij}^{ab,c} \leq (1 - v_i) R_{ij} \ , \quad \forall\, (i,j) \in P_{ab}^{\text{SP}},\ a \neq i,\ \forall c \in \mathcal{N} \tag{4.4}$$

$$f_{ij}^{ab,c} \leq (1 - v_j) R_{ij} \ , \quad \forall\, (i,j) \in P_{ab}^{\text{SP}},\ b \neq j,\ \forall c \in \mathcal{N} \tag{4.5}$$

Edge Rate Constraint:

$$\sum_{a,b,c} f_{ij}^{ab,c} \leq R_{ij} \ , \quad \forall\, (i,j) \in \mathcal{E} \tag{4.6}$$

Equation (4.1) represents flow conservation of commodity c packets at controllable node v. Here, exogenous arrivals at node v equal network departures minus (endogenous) network arrivals at v. Similarly, Equation (4.2) represents flow conservation for exogenous arrivals at uncontrollable nodes. The exogenous arrivals for commodity c at uncontrollable node u are equal to network departures on the shortest-path to destination c plus network departures along shortest-paths to controllable nodes. This is the special case where uncontrollable node u is a source, in that u can dynamically route exogenous arrivals but not endogenous network arrivals. Equation (4.3) is a path constraint for each commodity c along the shortest-path from node a to node b, where the path-flow equals the edge-flow for each edge along path P_{ab}^{SP}. Equations (4.4)-(4.5) force edge-flow $f_{ij}^{ab,c} = 0$ if node i or j is a controllable node intermediate to path P_{ab}^{SP}, i.e., for $i \neq a$ and $j \neq b$, as such paths remove routing ability from intermediate controllable nodes. Equations (4.4)-(4.5) are necessary to allow for dynamic choice of controllable nodes, and are redundant with Equation (4.6) when nodes i and j both are uncontrollable. Finally, Equation (4.6) is an edge rate constraint for every edge (i, j), such that total flow over an edge is upper bounded by the edge capacity.

If there are no controllable nodes, i.e., $\mathcal{V} = \emptyset$, then Equation (4.2) simplifies to

$$\lambda_a^c = \bar{f}_{ac}^c \,, \quad \forall\, a, c \in \mathcal{N}, a \neq c \,. \tag{4.7}$$

Here, Equations (4.4) and (4.5) can be ignored as they are always redundant with Equation (4.6). The throughput region without controllable nodes is then limited to the set of arrival rate vectors $\boldsymbol{\lambda}$ such that Equations (4.7), (4.3), and (4.6) can be satisfied. Indeed, these equations specify the shortest-path formulation for the throughput region on graph G, defined as $\Lambda_G^{\text{SP}} \equiv \Lambda_G(\emptyset)$.

If all nodes are controllable, i.e., $\mathcal{V} = \mathcal{N}$, then there are no constraints from underlay paths and all dynamic routing decisions are allowed. Equations (4.1) and (4.6)

simplify to Equations (4.8) and (4.9).

$$\lambda_a^c = \sum_{b:(a,b)\in\mathcal{E}} \bar{f}_{ab}^c - \sum_{d:(d,a)\in\mathcal{E}} \bar{f}_{da}^c \,, \quad \forall\, a, c \in \mathcal{N}, a \neq c \tag{4.8}$$

$$\sum_c \bar{f}_{ab}^c \leq R_{ab} \,, \quad \forall (a,b) \in \mathcal{E} \tag{4.9}$$

Here, there are no uncontrollable nodes, so Equation (4.2) is unused, and Equations (4.3), (4.4), and (4.5) are redundant with Equations (4.8) and (4.9). The full region $\Lambda_G \equiv \Lambda_G(\mathcal{N})$ is then defined as the set of arrival rate vectors $\boldsymbol{\lambda}$ that satisfy Equations (4.8) and (4.9). This is the largest region supported by network G.

Any work-conserving policy with shortest-path routing can support the region $\Lambda_G(\emptyset)$, while differential backlog (backpressure) routing is known to support the full region $\Lambda_G(\mathcal{N})$. However, how to achieve the heterogeneous region $\Lambda_G(\mathcal{V})$ with a dynamic routing policy is not generally known. For heterogeneous networks, converting an uncontrollable node u into a controllable node v relaxes the constraints for node u from Equation (4.2) into Equation (4.1). Note that when node v becomes controllable, the overlay neighbor constraints from Equations (4.4) and (4.5) become active.

Recall that we assume optimal substructure for shortest-paths. We use this structure to find an additional property about the throughput region. Any path P_{ab}^{SP} that passes through a controllable node v can be split into two sub-paths P_{av}^{SP} and P_{vb}^{SP}, where optimal substructure guarantees that both sub-paths are in the set of underlay routes $\mathcal{P}^{\text{SP}}$. Node v can then concatenate these sub-paths to form the original path P_{ab}^{SP}. Therefore, if there exists a flow decomposition of $\boldsymbol{\lambda}$ that uses path P_{ab}^{SP}, then there is also a flow decomposition that uses sub-paths P_{av}^{SP} and P_{vb}^{SP}. Thus, with shortest-path routing, adding controllable nodes can allow the throughput region to grow, but never causes the region to shrink. This implies a subset relationship in the throughput region with shortest-path underlay routing, such that for any overlay node sets $\mathcal{V}_1$ and $\mathcal{V}_2$, if $\mathcal{V}_1 \subseteq \mathcal{V}_2$ then $\Lambda_G(\mathcal{V}_1) \subseteq \Lambda_G(\mathcal{V}_2)$. More generally, we have the

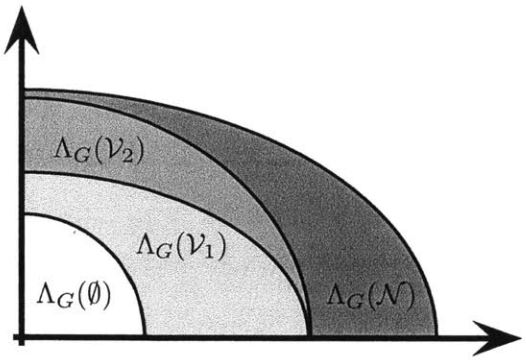


Figure 4-3: Notional diagram showing the subset relationship from Equation (4.10) in a 2-dimensional projection of throughput region $\Lambda_G(\cdot)$, for sets of controllable nodes $\mathcal{V}_1$ and $\mathcal{V}_2$ such that $\emptyset \subseteq \mathcal{V}_1 \subseteq \mathcal{V}_2 \subseteq \mathcal{N}$.

subset relationship from Equation (4.10), as pictorially shown in Figure 4-3.

$$\Lambda_G^{\text{SP}} \equiv \Lambda_G(\emptyset) \subseteq \Lambda_G(\mathcal{V}_1) \subseteq \Lambda_G(\mathcal{V}_2) \subseteq \Lambda_G(\mathcal{N}) \equiv \Lambda_G \,, \quad \forall\, \mathcal{V}_1 \subseteq \mathcal{V}_2 \subseteq \mathcal{N} \tag{4.10}$$

We next wish to find the smallest set of nodes $\mathcal{V} \subseteq \mathcal{N}$ such that $\Lambda_G(\mathcal{V}) = \Lambda_G(\mathcal{N})$.

4.4 Placement of Overlay Nodes

We would like to place controllable nodes to solve P1, but the constraint $\Lambda_G(\mathcal{V}) = \Lambda_G(\mathcal{N})$ is difficult to evaluate directly. A simple implementation for P1 can use the fact that Λ_G is a convex polytope, choosing the minimum number of controllable nodes to satisfy all points in the throughput region, as

$$\begin{aligned} V_2^* = \min_{\mathcal{V} \subseteq \mathcal{N}} \quad & |\mathcal{V}| \\ \text{s.t.} \quad & \boldsymbol{\lambda}^{(i)} \in \Lambda_G(\mathcal{V}) \,, \forall \boldsymbol{\lambda}^{(i)} \in \Lambda_G \,, \end{aligned} \tag{P2}$$

where $\boldsymbol{\lambda}^{(i)}$ enumerates all extreme points of Λ_G. It is clear that P2 is equivalent to P1, and therefore $V_2^* = V_1^*$. However, enumerating all extreme points of region Λ_G may be impractical, as Λ_G has $N(N-1)$ dimensions, i.e., one dimension for each source-destination pair, and the number of extreme points potentially grows with the power set of all dimensions, $\mathcal{O}\big(2^{N^2}\big)$.

Instead of evaluating P2, we propose a surrogate condition that is easier to evaluate while still leading to the same optimal solution. Recall that the set of paths $\mathcal{P}(\mathcal{V})$ includes all underlay paths $\mathcal{P}^{\text{SP}}$ and all n-concatenations (for any n) of these paths at controllable nodes $\mathcal{V}$. Let $\mathcal{P}_G$ be the set of all acyclic paths between all pairs of nodes in G. A first observation is that $\mathcal{P}(\mathcal{N}) = \mathcal{P}_G$. This holds by the assumption that all 1-hop paths are included in the set $\mathcal{P}^{\text{SP}}$, and since all nodes are controllable we can produce any path in G as a concatenation of 1-hop paths. Next, we define an important condition.

Condition 4.1 (All-paths). *A set of controllable nodes $\mathcal{V}$ is said to satisfy the* all-paths condition *if $\mathcal{P}(\mathcal{V}) = \mathcal{P}_G$.*

Theorem 4.1. *Given a placement of controllable nodes $\mathcal{V}$, satisfying the all-paths condition is necessary and sufficient for maximizing the throughput region, i.e.,*

$$\Lambda_G(\mathcal{V}) = \Lambda_G \text{ if and only if } \mathcal{P}(\mathcal{V}) = \mathcal{P}_G.$$

The proof is given in Appendix 4.A.1. Using the all-paths condition, we define P3 as below.

$$\begin{aligned} V_3^* = \min_{\mathcal{V} \subseteq \mathcal{N}} \quad & |\mathcal{V}| \\ \text{s.t.} \quad & \text{all-paths condition} \end{aligned} \tag{P3}$$

Corollary 4.1. *P1 $\iff$ P3, therefore $V_1^* = V_3^*$.*

4.4.1 Overlay Node Placement Algorithm

We design an algorithm to choose the placement of overlay nodes $\mathcal{V} \subseteq \mathcal{N}$ on a given graph $G = \{\mathcal{N}, \mathcal{E}\}$ such that the choice of overlay nodes is sufficient to satisfy the full throughput region of the network, i.e., $\Lambda_G(\mathcal{V}) = \Lambda_G(\mathcal{N})$. The overlay node placement algorithm consists of three phases: (1) removal of degree-1 nodes; (2) constraint pruning; and (3) overlay node placement. These phases are explained below.

Phase 1: Remove Degree-1 Nodes

An *attached tree* is a tree that is connected to the rest of graph G by only a single edge. An intuitive observation is that the throughput region does not increase by installing controllable nodes on attached trees. Thus, at this preparatory phase, we remove all attached trees by removing degree-1 nodes recursively, as follows. Start with original graph $G = (\mathcal{N}, \mathcal{E})$, and initialize $\mathcal{N}' := \mathcal{N}$ and $\mathcal{E}' := \mathcal{E}$. While there exists any node $n \in \mathcal{N}'$ such that $\text{degree}(n) = 1$, set $\mathcal{N}' := \mathcal{N}' \setminus n$ and set $\mathcal{E}' := \mathcal{E}' \setminus e$, where e is the only edge that connects to node n. Repeat until no degree-1 nodes remain. All remaining nodes have a degree of at least 2, thus all attached trees have been removed. The graph that remains is $G' = (\mathcal{N}', \mathcal{E}')$.

Lemma 4.1. *Assume that a placement $\mathcal{V}$ that satisfies the all-paths condition includes some node n on an attached tree. If node n is removed, the remaining placement $\mathcal{V} \setminus n$ still satisfies the all-paths condition.*

Proof. To show $\mathcal{P}(\mathcal{V}) = \mathcal{P}(\mathcal{V} \setminus n)$, we will show that for any pair $a, b \in \mathcal{N}$, each acyclic path $P_{ab} \in \mathcal{P}(\mathcal{V})$ falls into one of four cases. For each case we see that P_{ab} does not require any overlay nodes on attached trees, proving the lemma.

1. Nodes a and b are both on the same attached tree: There is only one path from node a to node b, and this is the shortest path.

2. Node a is on a specific attached tree and node b is not on that tree: There is only one path from node a to the node $c \in G'$ that connects to the attached tree, and this is the shortest path. Path P_{ab} must include shortest-path P_{ac}^{SP}.

3. Node b is on a specific attached tree and node a is not on that tree: There is only one path from $c \in G'$ to b, where c connects to the attached tree. Then path P_{ab} must include shortest-path P_{cb}^{SP}.

4. Nodes a and b are both on G': No possible acyclic path from node a to node b can go through attached trees, as entering and exiting an attached tree forms a cycle.

Therefore, if placement $\mathcal{V}$ satisfies the all-paths condition, then placement $\mathcal{V} \setminus n$ also satisfies the all-paths condition. □

By induction, it suffices to allocate overlay nodes in G' to satisfy the all-paths condition.

Phase 2: Constraint Pruning

In this phase, we define the destination trees which will be used to find the constraints for node placement. Exploiting a necessary condition from Lemma 4.2 regarding the placement of controllable nodes, we show that proper pruning of these destination trees will identify the set of constraints over which we minimize the allocation of controllable nodes.

By optimal substructure, the union of shortest paths P_{xn}^{SP} to any destination n from all nodes $x \in \mathcal{N}' \setminus n$ forms destination tree D_n. Define $\{P_{xn}^{\text{SP}}\} \setminus n$ to be the set of nodes on the shortest path from x to n, excluding node n. We have the following.

Lemma 4.2. *If the degree of node x on tree D_n is less than the degree of x on graph G', and there is no overlay node along the shortest path from x to n (i.e., $\nexists v \in \mathcal{V} : v \in \{P_{xn}^{SP}\} \setminus n$), then the all-paths condition is not satisfied.*

Proof. Let (b, x) be an edge in G' but not in D_n, where such an edge exists by the premise of Lemma 4.2. Consider path p formed from the concatenation of (b, x) and shortest-path P_{xn}^{SP}. We will show that this path cannot be formed if there are no controllable nodes in the shortest path from x to n, and thus the all-paths condition is not satisfied.

First, observe that since edge (b, x) is not on tree D_n, shortest-path P_{bn}^{SP} does not include this edge (b, x). Thus, the path p requires a concatenation of two or more shortest-paths. Such a concatenation must occur at a controllable node on path P_{xn}^{SP}. However, this is impossible since there are no controllable nodes on path P_{xn}^{SP}. Thus, the all-paths condition is not satisfied. □

For Phase 2, we prune destination trees D_n at nodes with degree in D_n that is less than their degree in G' to obtain *pruned trees* D'_n. By Lemma 4.2, for the all-

paths condition to be satisfied it is necessary to have at least one overlay node on the shortest path to n from every leaf node of pruned tree D'_n. The pruned trees D'_n and this necessary condition from Lemma 4.2 will be used as constraints in Phase 3.

Phase 3: Overlay Node Placement

Consider binary program P4 for placing the minimum number of overlay nodes to satisfy Lemma 4.2 for all nodes on all pruned trees D'_n.

$$\begin{aligned} V_4^* = \min & \sum_n v_n \\ \text{s.t.} & \sum_{a \in \{P_{bn}^{\text{SP}}\} \setminus n} v_a \geq 1, \ \forall b \in \text{LeafNodes}(D'_n), \ \forall n \\ & v_n \in \{0,1\}, \ \forall n \end{aligned} \tag{P4}$$

Here, $\text{LeafNodes}(D'_n)$ is the set of all leaf nodes on pruned tree D'_n, and $\{P_{bn}^{\text{SP}}\} \setminus n$ is defined in Phase 2.

Next, we show that the placement determined by the solution of P4 satisfies the all-paths condition.

Lemma 4.3. *The overlay node placement of P4 satisfies the all-paths condition for graph G'.*

The proof of Lemma 4.3 is given in Appendix 4.A.2.

The following main result establishes the performance of the proposed overlay node placement algorithm.

Theorem 4.2. *Let $\mathcal{V}^*$ be the solution to P4. Then $\mathcal{V}^*$ is an optimal solution to P3, and therefore also P1. Thus, $\Lambda_G(\mathcal{V}^*) = \Lambda_G$.*

Proof. By Lemma 4.2, the constraint of P4 is necessary for the all-paths condition. By Lemma 4.1 and Lemma 4.3 it is also sufficient. Thus, we have P4 $\iff$ P3. By Theorem 4.1, P3 $\iff$ P1, thus P4 $\iff$ P1. Then by P1, $\mathcal{V}^*$ has the minimum cardinality to satisfy $\Lambda_G(\mathcal{V}^*) = \Lambda_G$, where $\Lambda_G \equiv \Lambda_G(\mathcal{N})$. □

Overlay Node Placement Algorithm

Phase 1: Recursively remove all degree-1 nodes $\mathcal{N}_1$ and associated edges $\mathcal{E}_1$ from graph G, until no degree-1 nodes remain. The remaining graph is $G' = \{\mathcal{N}', \mathcal{E}'\}$, where $\mathcal{N}' = \mathcal{N} \setminus \mathcal{N}_1$ and $\mathcal{E}' = \mathcal{E} \setminus \mathcal{E}_1$. This removes all attached trees from G.

Phase 2: Consider the *destination tree* D_n for each node $n \in \mathcal{N}'$, and consider the degree of all nodes $b \in \mathcal{N}' \setminus n$ on tree D_n. If the degree of b on D_n is less than the degree of b on G', then prune destination tree D_n at node b by removing all edges to children of node b on D_n, and remove any nodes and edges that become disconnected from n. The remaining subgraph is the *pruned tree* D'_n.

Phase 3: Solve P4, and place an overlay node at each node n where the solution P4 has the result $v_n = 1$.

Figure 4-4: Summary of the overlay node placement algorithm.

A summary of the overlay node placement algorithm is shown in Figure 4-4.

Phases 1 and 2 of the algorithm have complexity $\mathcal{O}(N^2)$. P4 solves a vertex cover problem, which is known to be NP-Hard. However, note that the constraints of our problem have optimal substructure, which might be exploitable. For our experiments on graphs with 1000 nodes, the solver found most solutions to P4 within 5 seconds, and we only rarely encountered scenarios that required more than a few minutes to solve. Thus, we find this algorithm to be practical.

We will assume hop-count as the shortest-path metric for results that follow. However, note that the proof of optimality for the overlay node placement algorithm makes no assumption about the shortest-path metric, thus alternate metrics are also supported.

4.5 Overlay Node Placement Results

We provide results for various types of graphs representing wired networks, including specific families of graphs and random graphs. By Theorem 4.2, the full throughput region is provided by the placement of our algorithm on all of these cases.

4.5.1 Simple Scenarios

Trees and Forests

Consider trees with shortest-path underlay routes $\mathcal{P}^{\text{SP}}$ for every pair of nodes a and b. A tree is loop free, and thus each path $p \in \mathcal{P}^{\text{SP}}$ is the unique acyclic path from node a to b. Thus, the all-paths condition is automatically satisfied, and $\Lambda_G(\emptyset) = \Lambda_G(\mathcal{N})$.

It follows that no controllable nodes are required for a forest, which is a disjoint union of trees.

Cycles and Rings

Next, we provide a lower bound that is fundamental to the performance of P4 on graphs with cycles.

Lemma 4.4. *Every cycle requires at least 3 controllable nodes to satisfy the all-paths condition.*

Proof. Consider controllable nodes $v_1, v_2 \in \mathcal{V}$ on a cycle, and without loss of generality assume shortest-path $P^{\text{SP}}_{v_1 v_2}$ is on the cycle. Then path $P^{\text{SP}}_{v_1 v_2}$ allows one direction of flow on the cycle, and at least one additional controllable node is required to allow flow in the counter direction on the cycle. Note that the same problem occurs in scenarios with 0 or 1 controllable node on the cycle, and when path $P^{\text{SP}}_{v_1 v_2}$ is not on the cycle. Thus, at least 3 controllable nodes are required on each cycle in the network. □

Further, the result from Lemma 4.4 is exact for the case of a ring, where the entire graph is a single cycle.

Lemma 4.5. *Exactly 3 controllable nodes are required to satisfy the all-paths condition for a ring network with $N \geq 5$ nodes and hop-count as the metric for shortest-path routing.*

The proof is given in Appendix 4.A.3.

Cliques

Consider a clique, which is a fully connected graph where edge (a, b) exists for all pairs of nodes $a, b \in \mathcal{N}$. We require all edges (a, b) to be included in the underlay routes, thus all underlay routes are single edges, i.e., $P_{ab}^{\text{SP}} = (a, b)$ for all pairs $a, b \in \mathcal{N}$. A Hamiltonian path, traversing all nodes, will require all intermediate nodes to be controllable. Such paths can start and end at any node, therefore it is clear that a clique requires all nodes to be controllable to satisfy the all-paths condition, i.e., $\mathcal{V} = \mathcal{N}$.

Regular Grids

Consider regular grid networks as a tiling of nodes connected in squares of 2×2 nodes with total $N = L \times W$ nodes, for $L \geq 2$ and $W \geq 2$. Here, L is the number of nodes per row while W is the number of nodes per column. Assume that we get to choose how shortest-path ties are broken, i.e., the network is designed such that ties are broken in favor of minimizing the number of controllable nodes required. Each 2×2 square tile is a cycle, so by Lemma 4.4 each cycle requires at least 3 controllable nodes. Let $\mathcal{T}_j$ be the set of four nodes on tile j. Then a simple program to place overlay nodes on a grid is given by P5.

$$\begin{aligned} &\min \sum_n v_n \\ &\text{s.t.} \sum_{n \in \mathcal{T}_j} v_n \geq 3, \quad \text{for each tile } j, \\ &\quad v_n \in \{0, 1\}, \ \forall n \end{aligned} \tag{P5}$$

In Figure 4-5, we see that P5 chooses controllable nodes $\mathcal{V}$ in a crosshatch pattern.

We can apply this pattern to grids of arbitrary size by choosing all nodes on even rows and even columns to be controllable. Note that no two uncontrollable nodes are adjacent in the crosshatching pattern. By assumption, the shortest-path tie-breaking rule can be chosen to prefer uncontrollable nodes, if available, as the next-hop when constructing shortest-paths. Then, for any uncontrollable node u and any pair of controllable nodes a and b that neighbor u on the grid, the 2-hop shortest-path between node a and b is guaranteed to pass through node u by the tie-breaking rule. All such 2-hop paths are then in the set $\mathcal{P}^{\text{SP}}$, and it is easy to verify that the set of paths $\mathcal{P}(\mathcal{V}) = \mathcal{P}_G$. Therefore, the crosshatch allocation from P5 satisfies the all-paths condition, and by Theorem 4.1 we have $\Lambda_G(\mathcal{V}) = \Lambda_G$. As a confirmation, the shortest-paths from the above discussion can be used with the overlay node placement algorithm (using P4) to arrive at the same crosshatch allocation in Figure 4-5.

For the crosshatch overlay node allocation, the ratio of controllable nodes to total nodes, V/N, is shown in Equation (4.11).

$$\frac{V}{N} = \frac{L\lfloor W/2 \rfloor + \lfloor L/2 \rfloor \lceil W/2 \rceil}{L \times W}, \quad \text{for } L \geq 2 \text{ and } W \geq 2 \tag{4.11}$$

This ratio is exactly $3/4$ when both L and W are even. If either L or W or both are odd, then V/N is minimized on a 3×3 grid at $V/N = 5/9$, and asymptotically approaches $3/4$ as L and W grow large.

4.5.2 Random Networks

This section considers placement of overlay nodes to support the full throughput region on various types of random graphs. For all scenarios, N is the total number of nodes in the network, and V is the number of overlay or controllable nodes that the algorithm chooses.

Power-Law Degree Distribution

The degree distribution of nodes in the Internet follows roughly a power-law distribution [28], such that a histogram of node degrees follows a straight line when both the

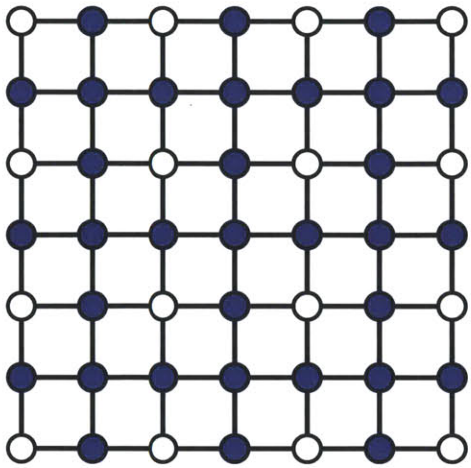


Figure 4-5: Minimal placement of overlay nodes to support full throughput region on a 7×7 grid. Overlay nodes indicated in blue. Node placement from P5.

value and frequency axes are logarithmic. We construct random networks that have power-law degree distributions using the configuration model and a truncated Zipf distribution [28]. Zipf is a discrete distribution with parameters α and Z, where α is the power-law exponent and Z is a truncation parameter indicating the maximum degree of the distribution. The Zipf PMF is shown in Equation (4.12).

$$\mathbb{P}(D = d) = \frac{d^{-\alpha}}{\sum_{k=1}^{Z} k^{-\alpha}}, \quad \text{for } d = 1, ..., Z \tag{4.12}$$

For a given number of nodes N, the configuration model attaches a number of *stubs* to each node according to the Zipf distribution, where a stub is half of an edge. Pairs of unconnected stubs are then chosen randomly and connected to form edges. Thus, node degree follows a power-law distribution. While self-loops and multi-edges are possible, these can both be ignored for our purposes; self-loops don't increase throughput, and our node placement algorithm is agnostic to capacity changes from multi-edges.

Figure 4-6 shows results from the overlay node placement algorithm for random power-law graphs with $N = 1000$ nodes, averaged over 10 realizations per data point. Values of α between 2 and 3 are considered, with $\alpha = 2.5$ being a frequent estimate for the Internet [28]. For $\alpha = 2.5$, the overlay node placement algorithm finds less

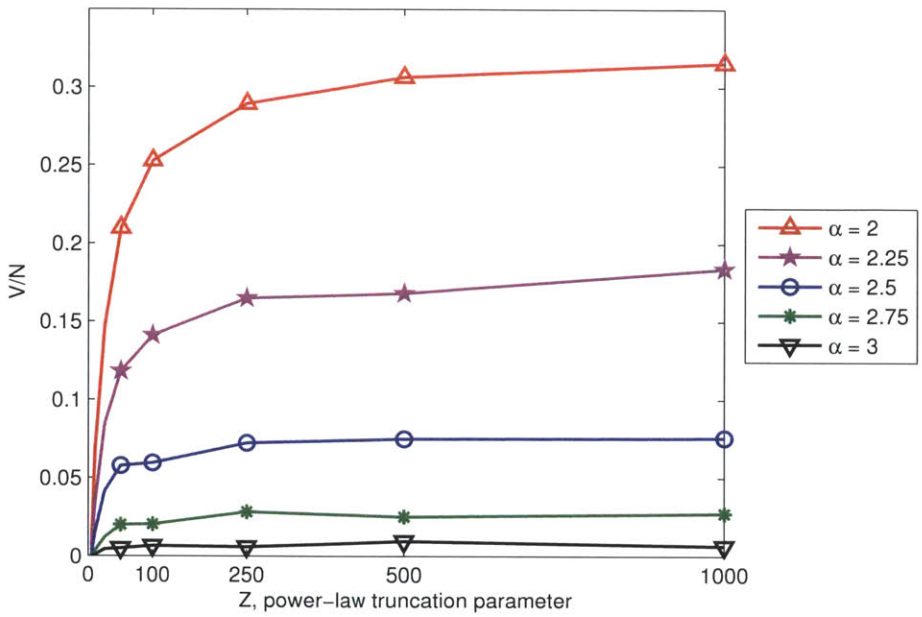


Figure 4-6: Results of overlay node placement algorithm on random graphs where node degree follows a power-law distribution with exponent α. These power-law graphs were generated with configuration model and truncated Zipf distribution. Value V/N is fraction of overlay nodes to total nodes.

than 8% of nodes are needed to be controllable for the full throughput region to be achievable.

Erdős-Rényi Model

The classic model for random graphs is known as the Erdős-Rényi (ER) model [28], where edges are independent and each edge is equally likely. ER graphs are also known as Poisson random graphs. We generate random graphs using the $G(N, p)$ formulation of the ER model, where parameter N is the number of nodes and p is the edge-connection probability. Random graphs are then generated from this model by considering every pair of nodes $a, b \in \mathcal{N}$, and creating edge (a, b) with probability p.

Figure 4-7 shows results of the overlay node placement algorithm on ER graphs with $N = 1000$ nodes. The edge-activation probability p is varied from 0 to 1, and results are averaged over 10 realizations for each probability p considered. For the ER model, a giant component forms at $p = 1/N$ with high probability, and the network becomes connected at $p = \log(N)/N$ with high probability. At around

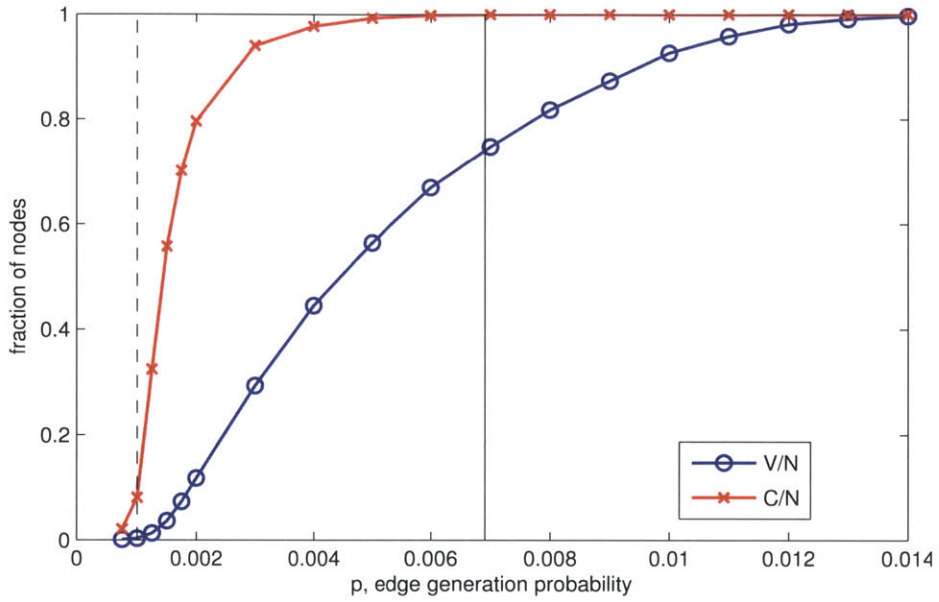


Figure 4-7: Results of overlay node placement algorithm on Erdős-Rényi random graphs with $N = 1000$ nodes. The blue curve indicates fraction V/N of overlay nodes, while the red curve indicates the ratio C/N of size for largest connected component to total nodes. Dashed vertical line at probability $p = 1/N$ at which giant component begins forming. Solid vertical line at probability $p = log(N)/N$ at which network becomes connected.

$p = 1.25/N = 0.00125$, we see the allocated controllable nodes V is only about 1% of N while the size of the largest connected component G is about 33% of N. At $p = 7/N = 0.007$, the graph is connected while V is still only 75% of N.

Two characteristics of the ER model are low clustering of nodes, such that few triangles are formed, and a high incidence of long edges yielding a low average length for shortest-paths. We next consider a random graph model with high node clustering.

Watts-Strogatz Small-World Model

Small-world graphs are characterized by high clustering of nodes, and the Watts-Strogatz (WS) model generates random graphs with small-world properties using parameters K for initial node degree and β for edge rewiring probability. Initial node degree K is limited to even values and K is also the average node degree for the random graph. Graphs are initialized as a ring lattice, where N nodes are arranged in increasing order around a ring and each node is connected to the closest K neighbors, $K/2$ in each direction. Each edge (a, b) with $a < b$ is rewired with probability β as edge (a, c), where node c is chosen uniformly amongst all nodes not directly connected to node a. At $\beta = 0$, the WS model produces a ring lattice, while at $\beta = 1$ all edges are rewired and the result approaches an ER graph.

We generate random graphs according to the WS small-world model with $N = 500$ nodes for initial edge degrees of $K = 2, 4$, and 6, and edge rewiring probability β is varied from 0 to 1. Figure 4-8 shows results of the overlay node placement algorithm on these WS graphs, where results are averaged over 20 realizations. For initial edge degree $K = 2$, the ratio V/N is less than 13% for all probabilities β. At $K \geq 4$, the initial ring lattice at $\beta = 0$ requires all nodes to be controllable. As the rewiring probability β grows, the occurrence of triangles decreases, where the minimum ratio V/N is around 50% for $K = 4$ and is around 75% for $K = 6$.

A limitation of the small-world model is that it generates too few nodes with very low-degree or high-degree distribution. The next section considers an elegant graph model that accounts for a growth process from which power-law graphs are formed.

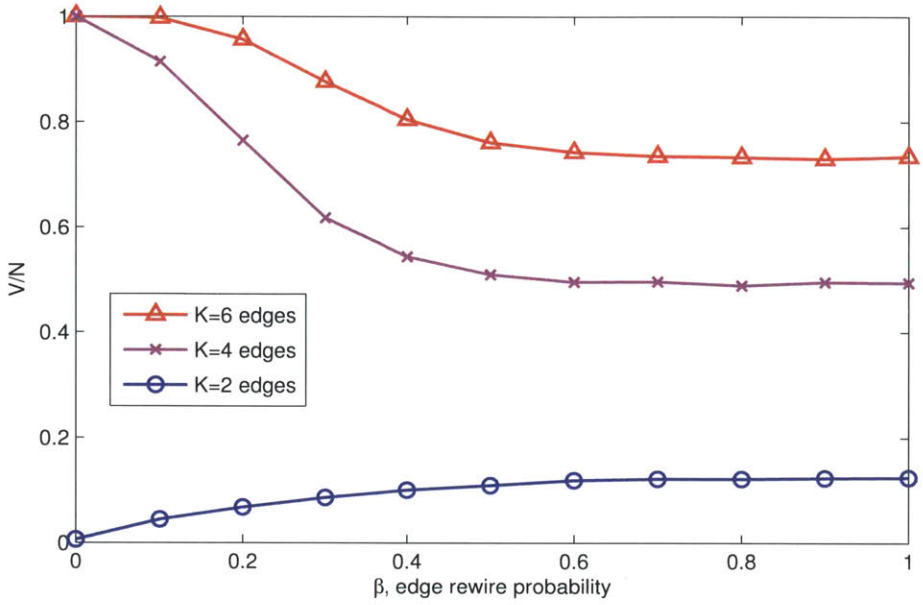


Figure 4-8: Results of overlay node placement algorithm on Watts-Strogatz small-world graphs with $N = 500$ nodes. Plot shows ratio of overlay nodes to total nodes, V/N, as a function of β, the edge rewiring probability. Curves are shown for $K = 2, 4$, and 6, where K is the average node degree.

Barabási-Albert Scale-Free Model

The Barabási-Albert (BA) model for random graphs, named for Albert-lászló Barabási and Réka Albert, is a scale-free model in the sense that nodes have a power-law degree distribution. Unlike the power-law model considered earlier in Section 4.5.2, the BA model is a growth model where nodes are added over time using a preferential attachment scheme. This model uses only one parameter, M, representing the initial degree of nodes as they are added to the network. Starting with a small connected network, the model adds nodes one-by-one, attaching each new node to M of the previously added nodes. The current degree of node i is k_i, and the probability of attaching to node i is proportional to k_i, i.e., probability $p_i = k_i / \sum_j k_j$, giving preference for attaching to nodes with high degree.

Figure 4-9 shows results of the overlay node placement algorithm on Barabási-Albert scale-free graphs, and results are averaged over 20 realizations. For $M = 1$, the model always generates a tree, giving $V = 0$. For $M = 2$, V decreases with increasing N, and V/N hits 50% at around $N = 750$. For $M = 3$, V again decreases

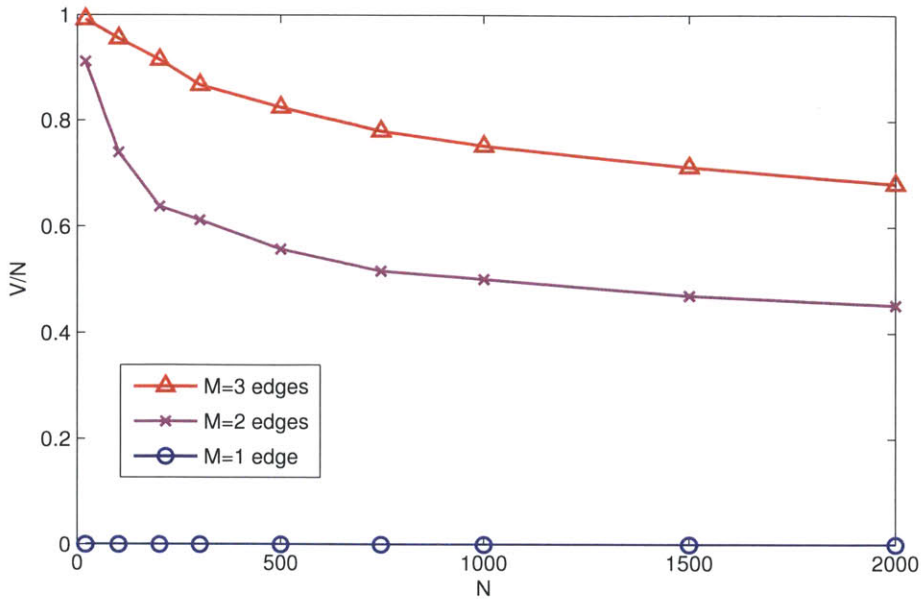


Figure 4-9: Results of overlay node placement algorithm on Barabási-Albert scale-free graphs. This is a growth model, where the number of nodes N increases over time. Curves are shown for $M = 1, 2$, and 3, where M is the initial degree of new nodes as they are added.

with N, and V/N hits 75% at around $N = 1000$. While this preferential attachment scheme from Barabási and Albert provides growth model for networks with a power-law degree distribution, we observe very different results on BA graphs in Figure 4-9 versus results on power-law graphs with a Zipf distribution in Figure 4-6. In our earlier results with the Zipf distribution, a high fraction of nodes have a degree of one. However, in BA graphs, parameter M is the minimum degree of each node in the graph. Thus, for any $M \geq 2$ with the BA preferential attachment scheme, the probability grows with each node placed that a new node will attached to two or more nodes that are already otherwise connected, in which case a new cycle is formed. Each new cycle can require additional controllable nodes, therefore BA graphs with parameter $M \geq 2$ have a high ratio of overlay nodes to total nodes, V/N.

4.6 Limited Number of Controllable Nodes

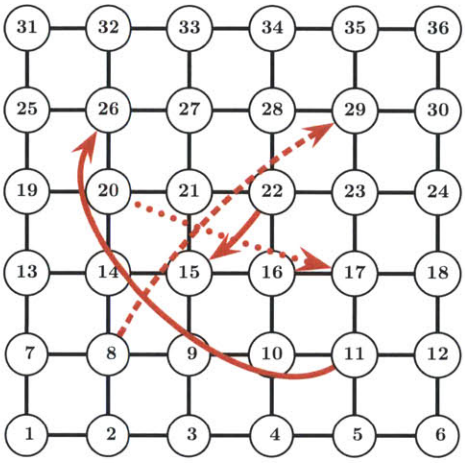
A program in the form of P2 is useful in scenarios where only a small set of arrival rate vectors require support, such that the constraints are limited to the specific vectors $\boldsymbol{\lambda}^{(i)}$ of interest. For example, this can be used to minimize the number of controllable nodes required to allow maximum flow between a specific source and destination. A similar formulation can be used to maximize the achievable flow when the maximum number of controllable nodes is upper bounded by some number X, as shown in P6. This can be useful in scenarios where resource limitations don't allow enough controllable nodes to achieve maximum throughput. As in P2, multiple rate vectors $\boldsymbol{\lambda}^{(i)}$ can be supported with additional constraints $\rho\boldsymbol{\lambda}^{(i)} \in \Lambda_G(\mathcal{V})$.

$$\begin{aligned} \max_{\mathcal{V}\subseteq\mathcal{N}} \quad & \rho \\ \text{s.t.} \quad & \rho\boldsymbol{\lambda} \in \Lambda_G(\mathcal{V}) \\ & |\mathcal{V}| \leq X \end{aligned} \tag{P6}$$

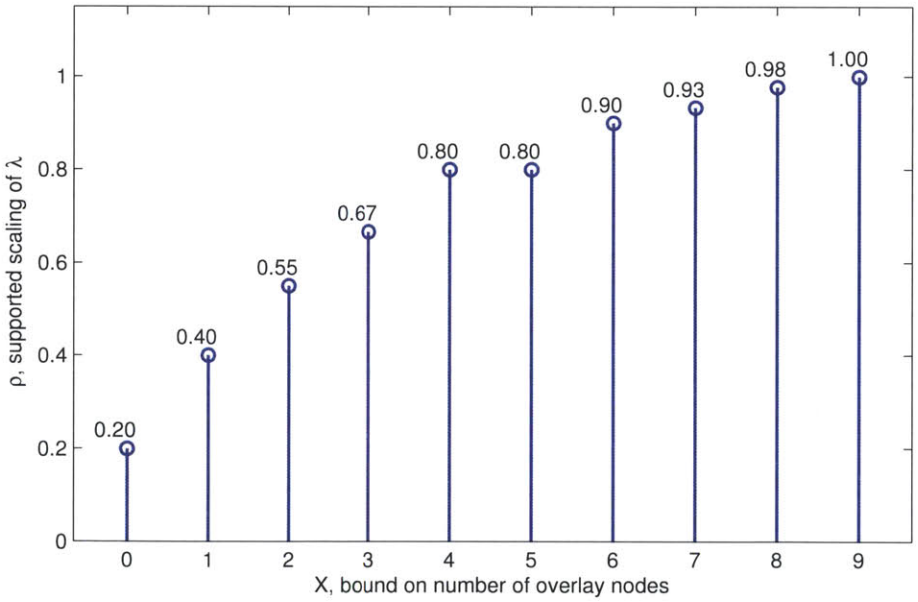
Figure 4-10 shows results of P6 on a 6×6 grid for a specific arrival rate vector $\boldsymbol{\lambda}$ with four equal traffic demands. Here, $\boldsymbol{\lambda}$ is the maximum scaling of the four traffic demands for $\boldsymbol{\lambda} \in \Lambda_G$, which occurs at $(\lambda_8^{29}, \lambda_{11}^{26}, \lambda_{20}^{17}, \lambda_{22}^{15}) = (2.5, 2.5, 2.5, 2.5)$. Figure 4-10b shows that a 20% scaling of vector $\boldsymbol{\lambda}$ is be supported by shortest-path routing alone, i.e., when $X = 0$ and no nodes are allowed to be controllable. When a single controllable node is allowed, i.e. $X = 1$, the supported throughput doubles to 40% of $\boldsymbol{\lambda}$, and when $X = 4$ controllable nodes are allowed, the supported rate quadruples to 80% of $\boldsymbol{\lambda}$. There are diminishing returns as X increases further, and maximum throughput is supported when we are limited to placing $X = 9$ controllable nodes.

4.7 Overlay Nodes in Wireless Networks

The all-paths condition 4.1 is sufficient to achieve $\Lambda_G(\mathcal{V}) = \Lambda_G$ in all networks, but this condition is not always a necessary condition in wireless networks. In other words,



(a) Scenario



(b) Node Placement Results

Figure 4-10: Placing limited number of overlay nodes. Results of P6 for chosen rate vector on a 6×6 grid. (a) Rate vector with four traffic demands, each indicated with an arrow. (b) Fraction of rate vector supported when limited to $|\mathcal{V}| \leq X$ controllable nodes.

satisfying the all-paths condition may over allocate controllable nodes under certain wireless interference models. To see why, consider a clique where all edges have unit-capacity and all transmissions mutually interfere. Due to interference, the maximum network sum throughput in this scenario is one, and this maximum throughput can only be achieved when each source a sends to destination b directly over edge (a, b). Thus no multi-hop paths are required, and the all-paths condition is sufficient but not necessary for this scenario.

To illustrate an overlay network in a wireless scenario, we study the performance of the overlay node placement algorithm on random geometric graphs, which is a simple model for wireless networks with omnidirectional antennas. The geometric model has parameters N and r, where N is the number of nodes and r is the edge range. Random graphs are then generated by randomly placing N nodes in a unit square, and creating all edges (a, b) for which the Euclidean distance between nodes a and b is within range r. Figure 4-11 shows results of the overlay node placement algorithm on random graphs with $N = 500$, averaged over 10 realizations per data point. Here, we see for the geometric model that the number of overlay nodes, V, placed by our algorithm grows much faster than the size of the largest connected component, C. The reason is twofold: (i) triangles appear in minor components[2], and (ii) multiple large components grow simultaneously. This is in contrast to the behavior of the Erdős-Rényi model, as discussed in Section 4.5.2, where minor components tend to be loop-free and a single major component appears with high probability.

The results for random geometric graphs show that the overlay node placement algorithm chooses most nodes to be controllable. However, as noted above, the placement of controllable nodes by this algorithm is sufficient but may not be necessary for wireless networks. Thus, the minimum number of controllable nodes required to provide full throughput in wireless networks is unclear. A topic for future work is a study of the necessary conditions for $\Lambda_G(\mathcal{V}) = \Lambda_G$ under various interference models.

[2]If edges (a, b) and (a, c) exist at range r, then the distance between b and c is at most $2r$. Thus, every degree-2 node at range r is on a triangle at range $2r$.

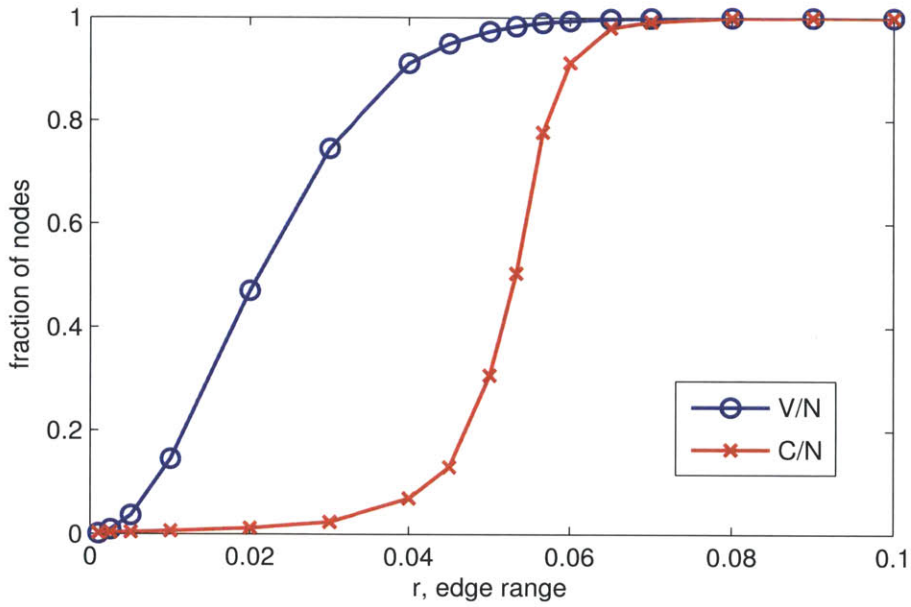


Figure 4-11: Results of overlay node placement algorithm on random geometric graphs with $N = 500$ nodes. The blue curve shows the ratio V/N of overlay nodes to total nodes. The red curve shows the ratio C/N for size of largest connected component to total nodes.

4.8 Backpressure Overlay Policy

Next, we study the problem of throughput maximization using a dynamic routing policy for a fixed placement $\mathcal{V}$ of overlay nodes. For this section, uncontrollable nodes are assumed to use a fair scheduling algorithm that serves each flow proportionally to queue backlog. This includes first-in first-out (FIFO), round robin, and proportionally fair random service on a packet-by-packet basis, the last of which is used in our simulation results. Our random scheduler at node a transmits a packet of commodity c with probability p_a^c, given by

$$p_a^c = \frac{U_a^c}{\sum_x U_a^x}\,, \tag{4.13}$$

where U_a^x is the queue backlog for commodity x at uncontrollable node a. All packet simulations of policies considered use Poisson arrivals, and simulations are run for 1 million time steps.

4.8.1 The Control Problem

We are interested in a dynamic policy that is stable for any arrival vector in the region $\Lambda_G(\mathcal{V})$, i.e., achieves maximum throughput in the overlay network. Controlling such a system is non-trivial since legacy nodes $\mathcal{N} \setminus \mathcal{V}$ lack controllability. While backpressure (BP) routing [40] is known to be throughput optimal, this assumes a homogeneous setting where all nodes are controllable and thus BP doesn't directly apply to our overlay setting. In fact, Section 4.8.2 shows an example where BP is suboptimal when applied only at overlay nodes. A primary issue is that BP cannot account for queues at uncontrollable nodes, so we modify BP to infer this queue size information.

Under policy π, let $\mu_{vn}^c(t, \pi)$ be the service function on the link $(v, n) \in \mathcal{E}$ for commodity c packets at time t, where $v \in \mathcal{V}$ and $n \in \mathcal{N}$. The edge rate constraint (4.6) implies $\sum_c \mu_{vn}^c(t) \leq R_{vn}$ must be satisfied at every slot. Thus, at each overlay node, the policy chooses the number of packets to be sent to any outgoing neighbor by assigning values to these functions. Uncontrollable nodes $\mathcal{N} \setminus \mathcal{V}$ are assumed to only forward the packets on pre-specified shortest-paths.

We define two different types of queues: (i) At each controllable node $v \in \mathcal{V}$, the queue for commodity c is denoted by $Q_v^c(t)$, and (ii) at each uncontrollable node $m \in \mathcal{N} \setminus \mathcal{V}$, queue $U_{mn}^{ab,c}(t)$ denotes commodity c packets for edge $(m, n) \in \mathcal{E}$ on path P_{ab}^{SP}. Note that queues in the underlay network have inherent directionality, since traffic going to opposite directions must be distinguished. Overlay nodes cannot directly observe queues at uncontrollable nodes, so instead we count *packets-in-flight* $F_{ab}^c(t)$ as the number of commodity c packets that have departed overlay node a but have not yet reached overlay neighbor b. The number of packets-in-flight on overlay edge (a, b) is then is the sum of uncontrollable node queues along path P_{ab}^{SP}, as shown in Equation (4.14).

$$F_{ab}^c(t) = \sum_{(m,n) \in P_{ab}^{\text{SP}}} U_{mn}^{ab,c}(t) \tag{4.14}$$

In practice, counting packets-in-flight $F_{ab}^{c}(t)$ can be realized by looking at the difference in cumulative count for packets of commodity c sent from a to b versus the cumulative count for these packets received at b. Alternatively, $F_{ab}^{c}(t)$ can be tracked by acknowledging packet serial numbers from b to a, where the in-flight count is estimated based on network delays for delivering the acknowledgment packets.

4.8.2 Insufficiency of Traditional Backpressure Routing

For an interference-free wired network, the backpressure (BP) routing policy [40] is as follows. For each edge $(a, b) \in \mathcal{E}$, define the differential backlog $W_{ab}^{c}(t)$,

$$W_{ab}^{c}(t) = Q_{a}^{c}(t) - Q_{b}^{c}(t), \quad \forall\,(a,b) \in \mathcal{E}, \forall\, c \in \mathcal{N}, \tag{4.15}$$

and define commodity $c_{ab}^{*}(t)$ that maximizes this weight,

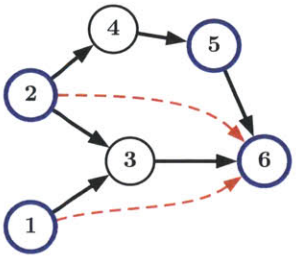
$$c_{ab}^{*}(t) \in \arg\max_{c \in \mathcal{N}} W_{ab}^{c}\,, \quad \forall\,(a,b) \in \mathcal{E}. \tag{4.16}$$

Then, the BP policy chooses

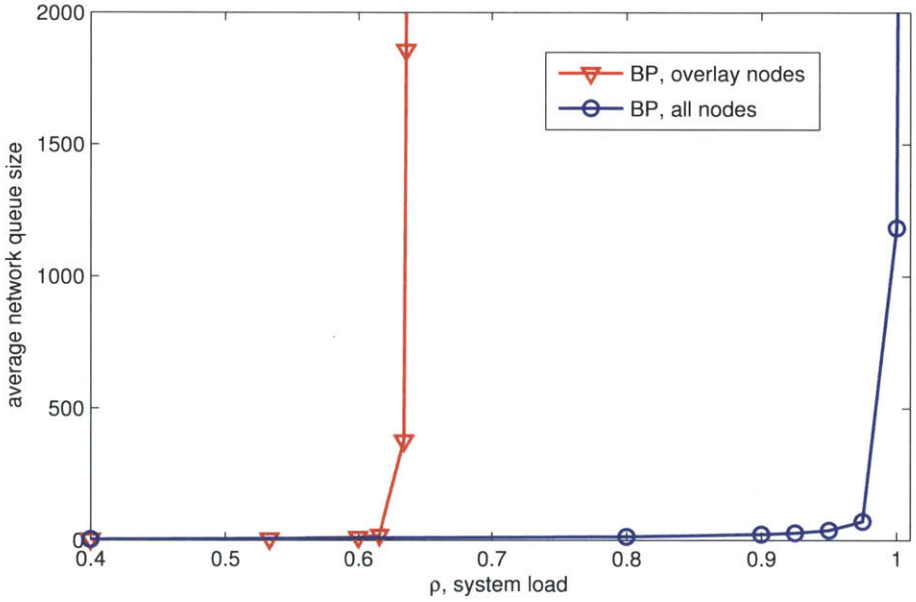
$$\mu_{ab}^{c_{ab}^{*}}(t, \mathrm{BP}) = \begin{cases} R_{ab}\,, & \text{if } W_{ab}^{c_{ab}^{*}} > 0 \\ 0\,, & \text{otherwise,} \end{cases} \tag{4.17}$$

where $\mu_{ab}^{c}(t, \mathrm{BP}) = 0,\ \forall c \neq c_{ab}^{*}(t)$. In [40], this policy is shown to stabilize any point in the region $\Lambda_{G}(\mathcal{N})$.

The intuition behind the optimality of BP is that congestion information propagates through the network via queue backlogs. The policy balances neighboring backlogs, such that when node n becomes congested, any upstream neighbors of n also become congested. Since uncontrollable nodes do not use BP, they do not propagate congestion information to BP sources. This is the primary reason why we do not expect BP to perform well in our system. The secondary reason is based on an assumption that legacy nodes cannot provide information about their backlog sizes.



(a) Scenario



(b) Simulation Results

Figure 4-12: Insufficiency of BP in overlay networks. (a) Scenario with contention at uncontrollable node 3. (b) Queue size of BP in overlay vs. BP in underlay.

Consider the example of Figure 4-12a, where the (controllable) overlay nodes $\mathcal{V} = \{1, 2, 5, 6\}$ are indicated in blue, with directed unit-rate links. It can easily be verified that the all-paths condition is satisfied for this setting, thus $\Lambda_G(\mathcal{V}) = \Lambda_G(\mathcal{N})$. The dashed red arrows show two traffic demands with symmetric arrival rates λ. With unit-rate links, offered load $\rho = \lambda$, where $\rho < 1$ is required for this network to be stable. We examine two different cases. First, we run BP at all nodes; this achieves maximum throughout and it is stable for all $\rho < 1$. Second, we run BP only at overlay nodes, computing differential backlogs across the overlay edges, e.g., node 2 computes $W^6_{2,5} = Q^6_2 - Q^6_5$ and $W^6_{2,6} = Q^6_2 - Q^6_6$. Simulation results in Figure 4-12b show that BP at the overlay nodes cannot stabilize $\rho > 2/3$, i.e., it is throughput suboptimal. The intuition is as follows. Note that $Q^6_6 = 0$, since node 6 is a destination. Then, any congestion at uncontrollable node 3 cannot be detected by source node 2, leading to positive traffic flow from source 2 through node 3 which is detrimental to traffic from source 1. This motivates our policy in the following section.

4.8.3 The Proposed OBP Policy

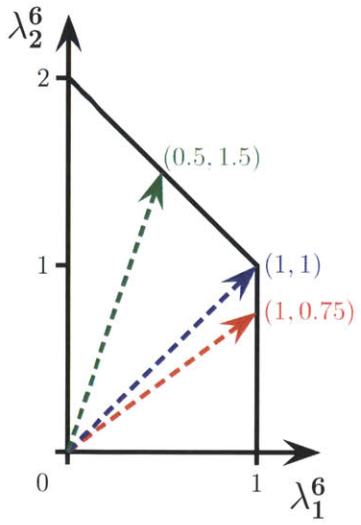
We propose the following policy, both dynamic and distributed, to account for packets-in-flight.

Overlay Backpressure (OBP). Let $\overline{\mathcal{E}}$ represent the set of edges in the overlay network. Redefine the differential backlog from Equation (4.15) as

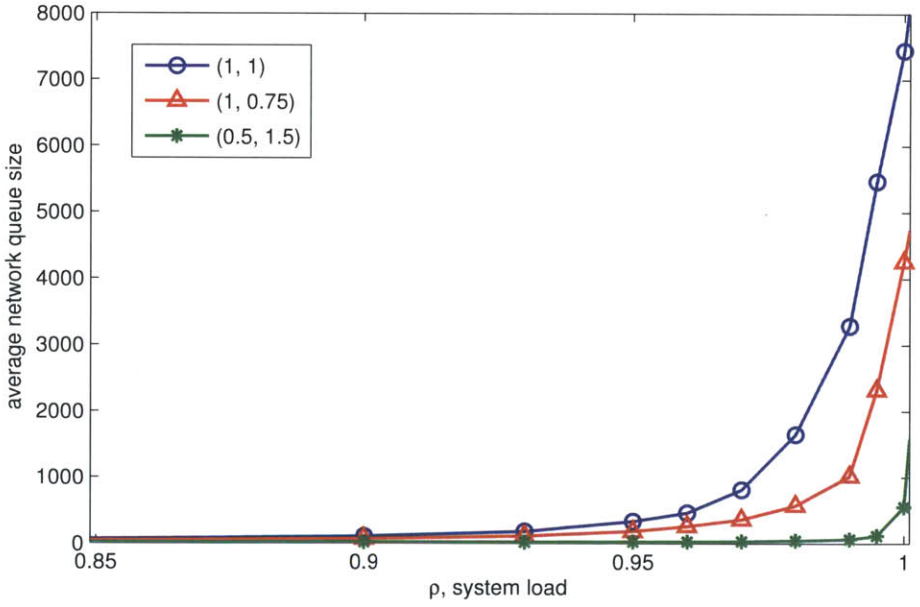
$$W^c_{ab}(t) = Q^c_a(t) - Q^c_b(t) - F^c_{ab}(t), \quad \forall\, (a,b) \in \overline{\mathcal{E}}, \forall\, c \in \mathcal{N}, \tag{4.18}$$

then determine $c^*_{ab}(t)$ and $\mu^c_{ab}(t, \text{OBP})$ as in Equations (4.16) and (4.17).

Intuitively, the OBP policy takes into account both the packet accumulation at the neighboring overlay node b, as well as any packets-in-flight on the path P^{S}_{ab} in the form of *negative pressure*. Through simulation we observe the following properties of the OBP policy: (i) OBP maximizes throughput in all examined scenarios, including the one of Figure 4-12a, (ii) OBP outperforms BP applied only at overlay nodes, and (iii) OBP has good delay properties, outperforming BP applied at all nodes.



(a) Throughput Region



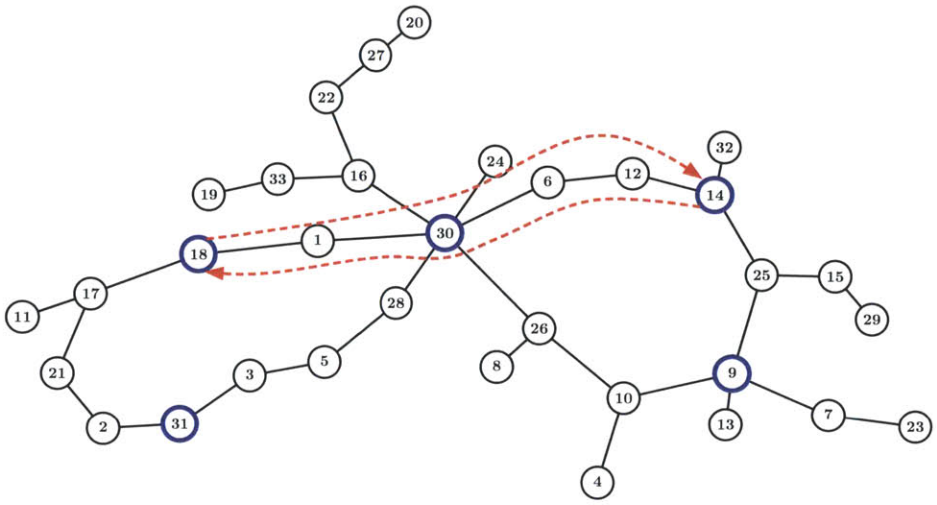
(b) Simulation Results

Figure 4-13: Evaluation of OBP policy on scenario from Figure 4-12a. (a) Throughput region of Figure 4-12a, with select rate vectors indicated. (b) Average queue backlog of OBP, after 1e6 time steps, for rate vectors indicated in (a).

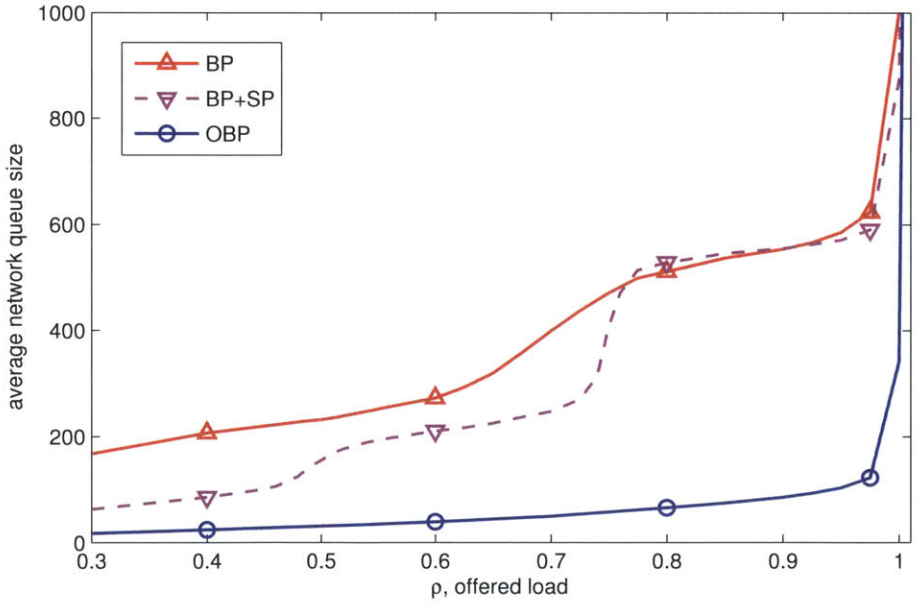
In Figure 4-13, we study different arrival vectors for the network of Figure 4-12a. The simulation results in Figure 4-13b show that all studied vectors are supported by the OBP policy.

In Figure 4-14, we show simulation results from three policies: BP, BP with shortest-path bias (BP+SP) from [26], and OBP. The simulations show that BP is stable for all values $\rho < 1$. Comparing OBP results to BP and BP+SP, both of which are throughput optimal policies, we see that OBP yields superior network delay. The reason is threefold: (i) the quadratic network queue size of BP is proportional to the number of controllable nodes used (in this scenario, OBP uses only 5 controllable nodes), (ii) no packets are sent to attached trees in case of OBP, and (iii) under light traffic, packets under BP perform random walks.

In Figure 4-15, we study a directed tandem network for the purpose of illustrating the delay properties of OBP. From [3] it is known that for BP on a tandem network, per-node queues grow linearly with distance from the destination, and thus network queue size grows quadratically with the total number of nodes. However, for the OBP policy we observe this linear growth of per-node queues only at controllable nodes, implying smaller total network queues size and improved delay performance when there are few controllable nodes. In this particular example, only the source is controllable for OBP, with $n-1$ legacy nodes, corresponding to the maximum benefit. Figure 4-15a compares BP and OBP queue size versus number of nodes n for a fixed offered load of $\rho = 0.8$. Here, we see BP queues grow quadratically in n, while OBP queues grow linearly in n. Figures 4-15c through 4-15f compare BP and OBP queue size versus offered load for $n = 10, 25, 50$, and 100 nodes. The BP policy only transmits on edges with positive differential backlog, i.e. when $W_{ab} = Q_a - Q_b > 0$ from Equation (4.15). Thus, BP can satisfy any offered load $\rho < 0.5$ with network queue size of approximately $n \times \rho$ by spacing packets at least 2 nodes apart at intermediate edges on the directed tandem, such that weight $W_{ab} = 1$ for any intermediate node a with a packet. However, to support any offered load $\rho > 0.5$, BP requires a positive differential backlog on all edges of the directed tandem. A lower bound on network queue size that allows all edges to have a positive differential

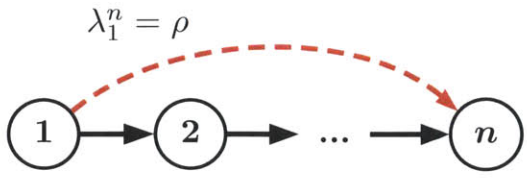


(a) Scenario

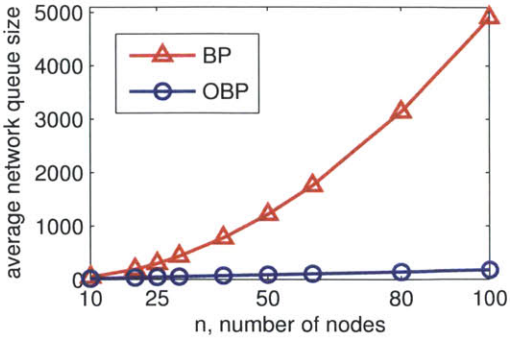


(b) Simulation Results

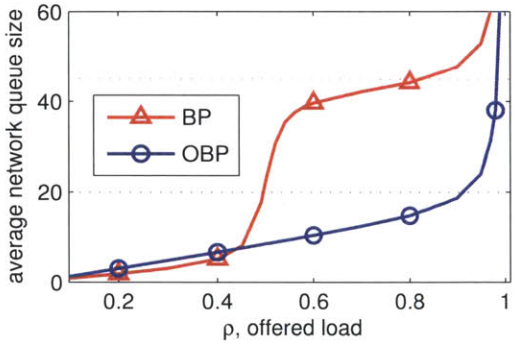
Figure 4-14: Comparing OBP with BP on a random graph. (a) Scenario with two symmetric traffic demands, indicated in red arrows. (b) Average queue size for BP, BP+SP, and OBP.



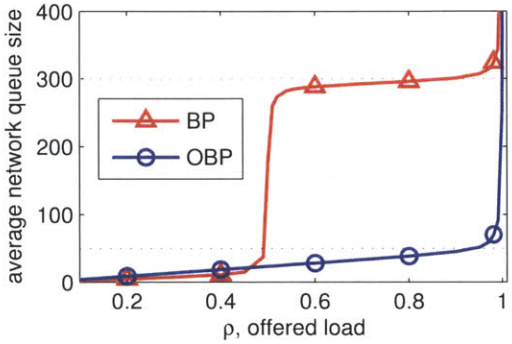
(a) Scenario



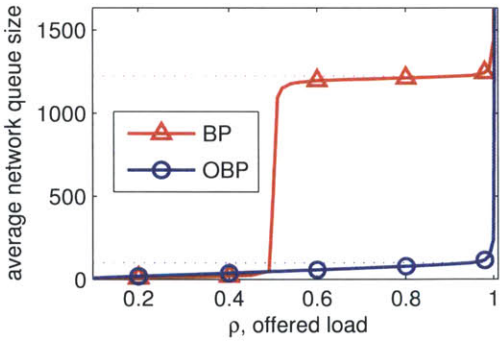
(b) Fixed load $\rho = 0.8$



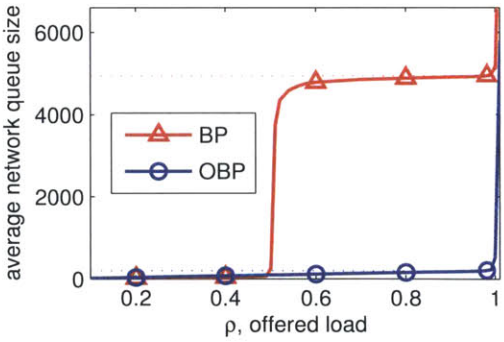
(c) Fixed $n = 10$ nodes



(d) Fixed $n = 25$ nodes



(e) Fixed $n = 50$ nodes



(f) Fixed $n = 100$ nodes

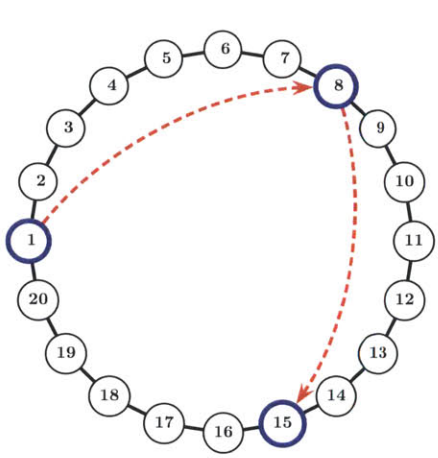
Figure 4-15: Directed tandem with n nodes. (a) Directed tandem scenario with single traffic demand from node 1 to n. (b) BP versus OBP for fixed offered load $\rho = 0.8$. BP queues grow quadratically with n, while OBP queues grow linearly with n. (c)-(f) BP versus OBP for fixed $n = \{10, 25, 50, 100\}$ number of nodes. Note sharp growth in BP backlog at $\rho = 0.5$. Dotted red horizontal lines estimate BP queue size at $n(n-1)/2$. Dotted blue horizontal lines estimate OBP queue size at $2n$.

backlog can be found as $(n-1)+(n-2)+...+2+1+0 = n(n-1)/2$, which is quadratic in the number of nodes n. Therefore, for each value of n, the BP queue size sharply transitions from a linear regime of around $n \times \rho$ for $\rho < 0.5$ to a quadratic regime of around $n(n-1)/2$ for $0.5 < \rho < 0.95$. For the same scenarios, OBP queue size is approximately $2n \times \rho$ for $\rho < 0.95$, where OBP network queues are split evenly between node 1 and total packets-in-flight F_{1n}^n.

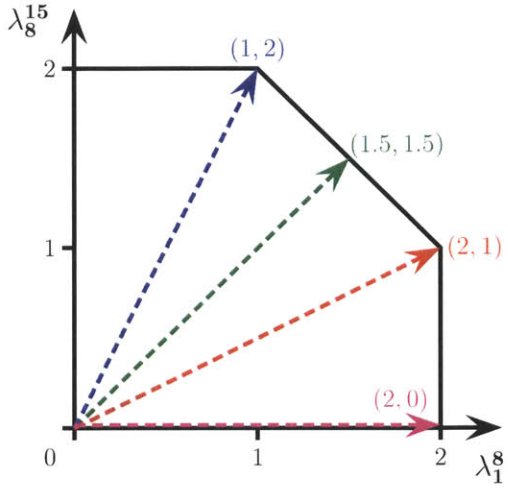
Finally, we consider the performance of OBP on a ring network with $N = 20$ nodes and $V = 3$ overlay nodes, where $V = 3$ was proved sufficient to achieve $\Lambda_G(\mathcal{V}) = \Lambda_G$ by Lemma 4.5. The scenario is shown in Figure 4-16a, with two competing traffic demands indicated with red arrows. Figure 4-16b shows the throughput region for these two traffic demands, with 4 rate vectors identified, and results for the OBP policy on these rate vectors is shown in Figure 4-16c. For each rate vector, we see the queues remain small for all points internal to the throughput region, indicating that OBP can stabilize the system for these vectors.

4.9 Summary

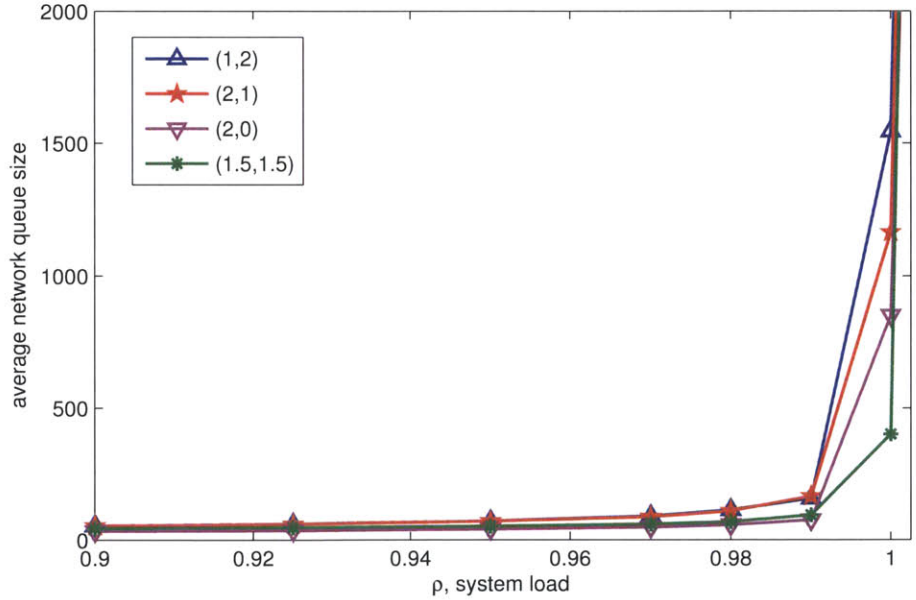
In this chapter, we have studied optimal routing in legacy networks where only a subset of nodes can make dynamic routing decisions, while the legacy nodes can forward packets only on pre-specified shortest-paths. This model captures evolving heterogeneous networks where intelligence is introduced on a fraction of nodes. We propose a sufficient condition for optimality, which is also necessary for interference-free networks with shortest-path routing. Based on this condition, we devise an optimal algorithm for placing controllable nodes. Finally, we propose a dynamic backpressure routing policy to be implemented in a network overlay, and show that this policy demonstrates superior performance in terms of throughput and delay.



(a) Scenario



(b) Stability Region



(c) Simulation Results

Figure 4-16: Evaluation of OBP on a ring with $N = 20$ nodes. (a) Overlay nodes indicated in blue; two traffic demands shown with red arrows. (b) Throughput region for competing traffic demands λ_1^8 and λ_8^{15}. Various rate vectors identified for simulation. (c) Queue size of OBP policy after 1 million time steps for rate vectors indicated in (b).

4.A Proofs

4.A.1 Proof of Theorem 4.1

Theorem 4.1. *Given a placement of controllable nodes $\mathcal{V}$, satisfying the all-paths condition is necessary and sufficient for maximizing the throughput region, i.e.,*

$$\Lambda_G(\mathcal{V}) = \Lambda_G \text{ if and only if } \mathcal{P}(\mathcal{V}) = \mathcal{P}_G.$$

Proof of Sufficiency: Consider a multicommodity vector $\boldsymbol{\lambda} \in \Lambda_G$. Feasibility of $\boldsymbol{\lambda}$ implies existence of a feasible flow decomposition of $\boldsymbol{\lambda}$. Without loss of generality, choose any one component of $\boldsymbol{\lambda}$ that sends flow from node a to node b with corresponding arrival rate λ_a^b. This arrival rate λ_a^b is supported by flow $f_{ab}^{\boldsymbol{\lambda}}$, where $f_{ab}^{\boldsymbol{\lambda}}$ can be decomposed into subflows $f_{ab}^{\boldsymbol{\lambda}}(p)$ for paths $p \in \mathcal{P}_{ab}$. Thus, if all paths are available to each source via the all-paths condition, then the feasible flow decomposition can be constructed with a stationary policy using underlay routes and the given set of controllable overlay nodes. □

Proof of Necessity: Support of the full throughput region requires support for all arrival rate vectors interior to the rate region allowed by the network. Assume $\Lambda_G(\mathcal{V}) = \Lambda_G$ and some path P_{ab}^{X} is unavailable, both as a shortest-path and as an n-concatenation of shortest-paths at controllable nodes $\mathcal{V}$. Without loss of generality, assume that this unavailable path does not traverse any controllable nodes. Otherwise, split the unavailable path at controllable nodes and choose an unavailable segment induced from the split as path P_{ab}^{X}; such an unavailable segment must exist, otherwise the original path could be formed as an n-concatenation of the induced segments. We will show that there exists a feasible arrival rate vector that requires the use of the unavailable path P_{ab}^{X}.

Construct an arrival rate vector $\boldsymbol{\lambda}$ that includes component λ_a^b equal to the maximum flow allowed for path P_{ab}^{X}, plus edge rate R_{ab} if edge (a, b) exists. In vector $\boldsymbol{\lambda}$, also include one-hop traffic demands for all edges $(i, j) \in \mathcal{E} \setminus (a, b)$ by choosing λ_i^j

to equal any remaining capacity on edge (i,j). This rate vector $\boldsymbol{\lambda}$ is then feasible by construction.

Let $\mathcal{N}_{ab}^{\mathrm{X}}$ be the set of nodes on path P_{ab}^{X}. For every node j not on path P_{ab}^{X}, i.e., $j \in \mathcal{N} \setminus \mathcal{N}_{ab}^{\mathrm{X}}$, the arrival rate vector $\boldsymbol{\lambda}$ was constructed such that $\sum_i \lambda_i^j = \sum_i R_{ij}$. Applying the edge rate constraints from Equation (4.6) at node j and taking the sum over all neighbors i, we have $\sum_i \sum_{x,y,c} f_{ij}^{xy,c} \leq \sum_i R_{ij} = \sum_i \lambda_i^j$ for all $j \in \mathcal{N} \setminus \mathcal{N}_{ab}^{\mathrm{X}}$, where the last equality comes from the previous equation. Then flow conservation requires that $f_{ij}^{xy,c} = 0$ for all commodities $c \neq j$. Thus, no feasible flow decomposition of $\boldsymbol{\lambda}$ can route flow for λ_a^b through any nodes in $\mathcal{N} \setminus \mathcal{N}_{ab}^{\mathrm{X}}$. Therefore, it remains to consider only nodes in $\mathcal{N}_{ab}^{\mathrm{X}}$ to support λ_a^b.

If P_{ab}^{X} is the only path from node a to b using nodes from the set $\mathcal{N}_{ab}^{\mathrm{X}}$, then P_{ab}^{X} is clearly necessary to support flow λ_a^b. Otherwise, recall that by assumption there are no controllable nodes intermediate to path P_{ab}^{X}. Then it remains only to consider the case where the shortest-path from node a to b uses a strict subset of nodes in $\mathcal{N}_{ab}^{\mathrm{X}}$, as no controllable nodes are available for path concatenation. Consider edge (i,j) such that nodes i and j are on path P_{ab}^{X}, where edge (i,j) is on P_{ab}^{SP} but not on P_{ab}^{X}. Here, $P_{ij}^{\mathrm{SP}} = (i,j)$ is the only available path from i to j with unused capacity, because no controllable nodes are available. Then, $f_{ij}^{ij,j} = \lambda_i^j = R_{ij}$, and Equation (4.6) requires $f_{ij}^{ab,b} = 0$. Therefore, there is no unused capacity on path P_{ab}^{SP}, so λ_a^b and λ_i^j cannot be supported simultaneously. There are no other paths to consider from node a to b for a feasible flow decomposition of $\boldsymbol{\lambda}$.

Therefore, $\Lambda_G(\mathcal{V}) \subset \Lambda_G$ if any path is not available. Thus, we have proved the necessity of the all-paths condition for wired networks with shortest-path routing. □

4.A.2 Proof of Lemma 4.3

Lemma 4.3. *The overlay node placement of P4 satisfies the all-paths condition for graph G'.*

Proof. Let $\mathcal{V}$ be a overlay node placement chosen by P4, and consider every acyclic path P_{ab} between all pairs of nodes a and b in graph G'. For all such paths P_{ab}, we will

show that either (1) P_{ab} is a shortest-path or (2) P_{ab} can be formed as a concatenation of shortest-paths at overlay nodes $\mathcal{V}$. Thus, $P_{ab} \in \mathcal{P}(\mathcal{V})$ for all paths P_{ab} on graph G', proving that $\mathcal{P}(\mathcal{V}) = \mathcal{P}_{G'}$, i.e., that the all-paths condition is satisfied.

Define overlay neighbor tree D''_n to be the union of shortest path routes to node n from all overlay neighbors of n, where the overlay nodes are defined by $\mathcal{V}$. Because P4 places overlay nodes on the shortest paths from the leaf nodes of D'_n to n, we have the relationship $D''_n \subseteq D'_n \subseteq D_n$. The leaf nodes of D''_n are the closest overlay nodes to n, and we will make use of this construction.

For each path P_{ab}, one of two cases must hold.

(1) The entire path P_{ab} is contained in overlay neighbor tree D''_b. In this case, $P_{ab} = P^{\text{SP}}_{ab}$, so $P_{ab} \in \mathcal{P}(\mathcal{V})$.

(2) There exists an overlay node $v \in D''_b$ such that path P_{ab} is a concatenation of paths P_{av} and P^{SP}_{vb} at overlay node v.

Path P^{SP}_{vb} is provided by a shortest-path route, so it only remains to show that path P_{av} is either (1) a shortest-path or (2) can be formed as a concatenation of shortest-paths at overlay nodes $\mathcal{V}$, i.e., $P_{av} \in \mathcal{P}(\mathcal{V})$. To show this, first note that neighbor tree D''_b includes all neighbors of node b, and that v is at least one hop away from b. Then path P^{SP}_{vb} has a positive length, and thus the length of path P_{av} is strictly less than the length of path P_{ab}. We can then iteratively repeat the above two-case argument by letting $b' = v$, and consider sub-path $P_{ab'}$, repeatedly shortening the path until case (1) holds.

Therefore, every path P_{ab} on graph G' is also in the set of paths $\mathcal{P}(\mathcal{V})$. Thus, $\mathcal{P}(\mathcal{V}) = \mathcal{P}_{G'}$, and the all-paths condition is satisfied. □

4.A.3 Proof of Lemma 4.5

Lemma 4.5. *Exactly 3 controllable nodes are required to satisfy the all-paths condition for a ring network with $N \geq 5$ nodes and hop-count as the metric for shortest-path routing.*

Proof. Lemma 4.4 establishes the necessity of at least 3 controllable nodes, so it only remains to show that 3 controllable nodes are sufficient to satisfy the all-paths condition.

Starting from any node x, consider nodes y and z that are neighbors, i.e., $(y,z) \in \mathcal{E}$, where shortest-paths P_{xy}^{SP} and P_{xz}^{SP} are disjoint. Without loss of generality assume $|P_{xy}^{\text{SP}}| \leq |P_{xz}^{\text{SP}}|$ where $|p|$ is the length of path p. With hop-count as the shortest-path metric, the length of these disjoint shortest-paths can differ at most by 1. Otherwise, there would exist a contradiction, as the path formed as a concatenation of P_{xy}^{SP} with edge (y,z) would be shorter than shortest-path P_{xz}^{SP}. Then the following inequality holds for any number of nodes $N \geq 5$.

$$|P_{xy}^{\text{SP}}| \geq \left\lfloor \frac{N-1}{2} \right\rfloor \geq \frac{N}{3} \tag{4.19}$$

Therefore, any node can reach a minimum of $N/3$ nodes in either direction around the ring using shortest-path routing. Conversely, any node can be reached by a minimum of $N/3$ nodes in either direction. Then we can place 3 controllable nodes, v_1, v_2, and v_3, such that shortest-paths $P_{v_i v_j}^{\text{SP}}$ and $P_{v_i v_k}^{\text{SP}}$ are edge-disjoint for all permutations $i,j,k \in \{1,2,3\}$. The overlay edges between these controllable nodes then form a bidirectionally connected ring as shown in Figure 4-1, making use of all paths between the controllable nodes. Every uncontrollable u is on the shortest-path between two controllable nodes v_i and v_j; thus, by optimal substructure, paths $P_{uv_i}^{\text{SP}}$ and $P_{uv_j}^{\text{SP}}$ are edge-disjoint paths from u to v_i and v_j, and paths $P_{v_i u}^{\text{SP}}$ and $P_{v_j u}^{\text{SP}}$ are edge-disjoint paths from v_i and v_j to node u. Then every path in the network is either a shortest-path or can be formed as an n-concatenation of shortest paths, and the all-paths condition is satisfied with exactly 3 controllable nodes. □

Chapter 5

Conclusions

In this thesis, we have considered practical applications of distributed network control policies in two distinct areas. In the first part of this thesis, we studied the use of network coding to increase the stability region of wireless networks. In the second part of this thesis, we studied the use of backpressure routing in interference-free legacy networks based on shortest-path routing, where only a subset of devices are allowed to make dynamic routing decisions.

In Chapters 2 and 3, we introduced a simple network coding scheme, and characterized the stability region subject to our network coding constraints. In Chapter 2, we developed a centralized network control policy that jointly optimizes for routing, scheduling, and our simple network coding scheme, and proved that this policy achieves maximum throughput subject to our coding constraint. In Chapter 3, we extended the policy from Chapter 2 by replacing the centralized max-weight scheduler with a distributed CSMA scheduler for use in random access networks, and showed that this CSMA policy can come arbitrarily close to supporting the full stability region allowed by our network coding constraint. We also developed a method for maintaining stable side information buffers without decreasing throughput. Analytical and empirical results were provided on throughput and delay. A main conclusion is that pairwise network coding captures most of the throughput gains on random topologies, and we showed simulation results for random graphs where pairwise network coding provides a median throughput gain of 31% beyond optimal routing and

scheduling without network coding.

In Chapter 4, a network overlay architecture was considered for deploying controllable nodes in networks based on shortest-path routing. We developed an algorithm to find the minimum placement of controllable overlay nodes while maximizing throughput region of the network. A motivating result showed that large ring networks require exactly 3 controllable nodes to achieve the full throughput region of the network. We evaluated our overlay node placement algorithm on various random graph models, and found, for example, that on 1000 node random graphs with power-law degree distribution using exponent $\alpha = 2.5$ — a common model for the Internet — the full throughput region was achieved with fewer than 8% of nodes made controllable. We showed that the use of traditional backpressure (BP) routing at overlay nodes can be throughput suboptimal due to an inability to detect congestion at uncontrollable nodes. We then developed the overlay backpressure (OBP) routing policy to detect congestion at uncontrollable nodes by tracking the packets-in-flight between controllable nodes, and showed OBP to achieve maximum throughput in all scenarios considered.

There are many avenues for future work on these topics. For network coding, it would be interesting to characterize the stability region with unreliable transmissions, and to combine our network coding scheme with suboptimal – but computationally less complex – greedy maximal scheduling. Then, a system implementation of our policies could be built. We would like to find the minimum placement of controllable nodes for maximizing throughput in interference networks. It would also be interesting to consider the behavior of our OBP policy in networks with mixed data rates, and to evaluate OBP on top of live networks.

Bibliography

[1] R. Ahlswede, N. Cai, S.Y.R. Li, and R.W. Yeung. Network information flow. *IEEE Trans. on Info. Theory*, 46(4):1204–1216, 2000.

[2] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proc. ACM SOSP*, Oct. 2001.

[3] L. Bui, R. Srikant, and A. Stolyar. Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing. In *Proc. IEEE INFOCOM*, April 2009.

[4] P. Chaporkar, K. Kar, and S. Sarkar. Throughput guarantees through maximal scheduling in wireless networks. In *Proc. Allerton*, Sept. 2005.

[5] P. Chaporkar and A. Proutiere. Adaptive network coding and scheduling for maximizing throughput in wireless networks. In *Proc. ACM MobiCom*, Sept. 2007.

[6] T. Cui, L. Chen, and T. Ho. Energy efficient opportunistic network coding for wireless networks. In *Proc. IEEE INFOCOM*, April 2008.

[7] A. Eryilmaz and D. S. Lun. Control for inter-session network coding. In *NetCod*, Jan. 2007.

[8] L. Georgiadis, M. J. Neely, and L. Tassiulas. *Resource allocation and cross-layer control in wireless networks*, volume 1. Foundations and Trends in Networking, 2006.

[9] J. Han, D. Watson, and F. Jahanian. Topology aware overlay networks. In *Proc. IEEE INFOCOM*, March 2005.

[10] L. Jiang and J. Walrand. A distributed CSMA algorithm for throughput and utility maximization in wireless networks. In *Proc. Allerton*, Sept. 2008.

[11] L. Jiang and J. Walrand. A distributed algorithm for maximal throughput and optimal fairness in wireless networks with a general interference model. *EECS Department, Univ. of California, Berkeley, Tech. Rep*, 2008.

[12] N. M. Jones, B. Shrader, and E. Modiano. Optimal routing and scheduling for a simple network coding scheme. In *Proc. IEEE INFOCOM*, March 2012.

[13] N. M. Jones, B. Shrader, and E. Modiano. Distributed CSMA with Pairwise Coding. In *Proc. IEEE INFOCOM*, April 2013.

[14] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. XORs in the air: Practical wireless network coding. *IEEE/ACM Trans. on Networking*, 16(3):497–510, 2008.

[15] W. Khan, L. B. Le, and E. Modiano. Autonomous routing algorithms for networks with wide-spread failures. In *Proc. IEEE MILCOM*, Oct. 2009.

[16] A. Khreishah, C.C. Wang, and N.B. Shroff. Cross-layer optimization for wireless multihop networks with pairwise intersession network coding. *IEEE Journal on Selected Areas in Comm.*, 27(5):606–621, 2009.

[17] L.B. Le, E. Modiano, and N.B. Shroff. Optimal control of wireless networks with finite buffers. In *Proc. IEEE INFOCOM*, March 2010.

[18] S. C. Liew, C. Kai, J. Leung, and B. Wong. Back-of-the-envelope computation of throughput distributions in csma wireless networks. In *Proc. IEEE ICC*, June 2009.

[19] J. Liu, Y. Yi, A. Proutiere, M. Chiang, and H. V. Poor. Towards utility-optimal random access without message passing. *Wireless Communications and Mobile Computing*, 10(1):115–128, 2010.

[20] P. Mannersalo, G. S. Paschos, and L. Gkatzikis. Geometrical bounds on the efficiency of wireless network coding. In *Proc. WiOpt*, May 2013.

[21] P. Marbach and A. Eryilmaz. A backlog-based csma mechanism to achieve fairness and throughput-optimality in multihop wireless networks. In *Proc. Allerton*, Sept. 2008.

[22] E. Modiano, D. Shah, and G. Zussman. Maximizing throughput in wireless networks via gossiping. In *Proc. ACM SIGMETRICS*, June 2006.

[23] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali. Routing without routes: The backpressure collection protocol. In *Proc. IPSN*, April 2010.

[24] D. Nakamura and A. Tamura. A revision of Minty's algorithm for finding a maximum weight stable set of a claw-free graph. *J. Oper. Res. Soc. Japan*, 44(2):194–204, 2001.

[25] B. Nardelli, J. Lee, K. Lee, Y. Yi, S. Chong, E. W. Knightly, and M. Chiang. Experimental evaluation of optimal CSMA. In *Proc. IEEE INFOCOM*, April 2011.

[26] M. J. Neely, E. Modiano, and C. E. Rohrs. Dynamic power allocation and routing for time varying wireless networks. In *Proc. IEEE INFOCOM*, April 2003.

[27] M. J. Neely, E. Modiano, and C. E. Rohrs. Dynamic power allocation and routing for time-varying wireless networks. *IEEE Journal on Selected Areas in Comm.*, 23(1):89–103, 2005.

[28] M. E. J Newman. *Networks: An Introduction.* Oxford University Press, Inc., New York, NY, USA, 2010.

[29] J. Ni and R. Srikant. Distributed CSMA/CA algorithms for achieving maximum throughput in wireless networks. In *Proc. ITA Workshop*, Feb. 2009.

[30] G. S. Paschos, L. Georgiadis, and L. Tassiulas. Optimal scheduling of pairwise XORs under statistical overhearing and feedback. In *Proc. IEEE RAWNET*, May 2011.

[31] L. L. Peterson and B. S. Davie. *Computer Networks: A Systems Approach.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 4th edition, 2007.

[32] S. Rajagopalan, D. Shah, and J. Shin. Network adiabatic theorem: an efficient randomized protocol for contention resolution. In *Proc. ACM SIGMETRICS*, June 2009.

[33] J. Ryu, L. Ying, and S. Shakkottai. Back-pressure routing for intermittently connected networks. In *Proc. IEEE INFOCOM*, March 2010.

[34] Y.E. Sagduyu, D. Guo, and R. Berry. Throughput and stability of digital and analog network coding for wireless networks with single and multiple relays. In *Proc. WICON*, Nov. 2008.

[35] H. Seferoglu, A. Markopoulou, and U. Kozat. Network coding-aware rate control and scheduling in wireless networks. In *Proc. IEEE ICME*, June 2009.

[36] H. Seferoglu and E. Modiano. Diff-max: Separation of routing and scheduling in backpressure-based wireless networks. In *Proc. IEEE INFOCOM*, April 2013.

[37] 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.

[38] S. Shabdanov, C. Rosenberg, and P. Mitran. Joint routing, scheduling, and network coding for wireless multihop networks. In *Proc. WiOpt*, May 2011.

[39] G. Sharma, R.R. Mazumdar, and N.B. Shroff. On the complexity of scheduling in wireless networks. In *Proc. ACM MobiCom*, Sept. 2006.

[40] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. Auto. Control*, pages 1936–1948, Dec. 1992.

[41] D. Traskov, M. Medard, P. Sadeghi, and R. Koetter. Joint scheduling and instantaneously decodable network coding. In *Proc. IEEE GLOBECOM*, 2009.

[42] D. Traskov, N. Ratnakar, D. S. Lun, R. Koetter, and M. Médard. Network Coding for Multiple Unicasts: An Approach based on Linear Optimization. In *Proc. ISIT*, July 2006.

[43] C.C. Wang and N. B. Shroff. On wireless network scheduling with intersession network coding. In *Proc. CISS*, March 2008.

[44] L. Ying, S. Shakkottai, and A. Reddy. On combining shortest-path and back-pressure routing over multihop wireless networks. In *Proc. IEEE INFOCOM*, April 2009.