Network coding for resource optimization and error correction

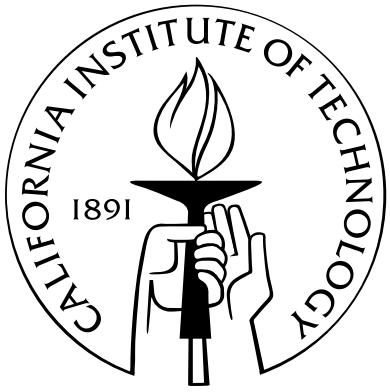
Thesis by

Sukwon Kim

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2010

(Defended May. 21, 2010)

To God, my family, Michelle, and Tracey

Acknowledgements

I would like to express my immense gratitude to my advisors, Professor Michelle Effros and Professor Tracey Ho for their support, guidance, and encouragement throughout the period of my graduate studies. I am very grateful that their constant passion, insight and curiosity leads me to memorable research experience. I have learned the most significant thing for the future of my life, the excitement of research, by working with them. I also give my deep gratitude to their kindness for frequent discussions and advices.

I would like to thank the members of my candidacy and defense committees, Professors Steven Low, Jehoshua Bruck, Michelle Effros, Tracey Ho, and Babak Hassibi, who read my thesis manuscript and gave me valuable feedback on my work. I also want to thank to Professor Salman Avestimehr for his consistent advice on network error correction.

I would like to give my deepest gratitude to my family. Their unconditional love and patience were the most significant support for my Ph.D abroad study. I thank my family for all the things they gave to me. I also wish to thank my friends Hyunsup Park, Jaeseok Bae, Kwanghyung Chung, Inmo Han, JunHwan Kim, Woojun Han, Keosan Kim, Christopher Chang, Changho Sohn, Hyungjoon Ahn, Minjang Jin, and Haekong Kim for their consistent encouragement.

My labmates Christopher Chang, Tao Cui, Mayank Bakshi, Wei-Hsin Gu, Derek Leong, Theodore Dikaliotis, and Svitlana Vyetrenko helped me with technical details and provided useful advices. Finally, I would like to give gratitude to Samsung Lee Kun Hee scholarship foundation for their financial support throughout Ph.D study. This material is also based upon work partially supported by NSF grant

CNS 0905615, Caltech's Lee Center for Advanced Networking a subcontract #069153 issued by BAE Systems National Security Solutions, Inc., and the Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare System Center (SPAWARSYSCEN), San Diego under Contract No. N66001-08-C-2013.

Abstract

In the first part of this thesis, we demonstrate the benefits of network coding for optimizing the use of various network resources.

We first study the problem of minimizing the power consumption for wireless multiple unicasts. A simple XOR-based coding strategy is considered for reducing the power consumption. We present a centralized polynomial time algorithm that approximately minimizes the number of transmissions for two unicast sessions. We extend it to a greedy algorithm for general problem of multiple unicasts.

Previous results on network coding for low-power wireless transmissions of multiple unicasts rely on opportunistic coding or centralized optimization to reduce the power consumption. Thus we propose a distributed strategy for reducing the power consumption with wireless multiple unicasts. Our strategy attempts to increase network coding opportunities without the overhead required for centralized design or coordination. We present a polynomial time algorithm using our strategy that maximizes the expected power savings with respect to the random choice of sources and sinks on the wireless rectangular grid.

We study the problem of minimum-energy multicast using network coding in mobile ad hoc networks (MANETs). The optimal subgraph can be obtained by solving a linear program every time slot, but it leads to high computational complexity. We present a low-complexity approach, network coding with periodic recomputation, which recomputes an approximate solution at fixed time intervals, and uses this solution during each time interval. We analyze the power consumption and the complexity of network with this approach.

Recently, several back-pressure type optimization algorithms with network coding

are presented for multiple unicasts and multicast. Such algorithms are distributed since decisions are made locally at each node based on feedback about the size of queues at the destination node of each link. We develop a back-pressure based distributed optimization framework, which can be used for optimizing over any class of network codes. Our approach is to specify the class of coding operations by a set of generalized links, and to develop optimization tools that apply to any network composed of such generalized links.

In the second part of this thesis, we study the capacity of single-source single-sink noiseless networks under adversarial attack on no more than z edges. Unlike prior papers, which assume equal capacities on all links, we allow arbitrary link capacities. Results include new upper bounds, general transmission strategies, and example networks where those bounds are tight. We introduce a new method for finding upper bounds on the linear coding capacities of arbitrary networks and show that there exists networks where the capacity is 50% greater than the best rate that can be achieved with linear coding. We also demonstrate examples where, unlike the equal link capacity case, it is necessary for intermediate nodes to do coding, nonlinear error detection or error correction in order to achieve the capacity. We introduce a new strategy called "guess-and-forward" and employ this strategy on a sequence of networks designed to provide increasingly complex generalizations of the cut-set bounds. The first is a two-node network with multiple feedback links. The second is a four-node acyclic network. The third is a family of 'zig-zag' networks. In the first two cases, the guess-and-forward strategy achieves the capacity. For zig-zag networks, we derive the achievable rate of guess-and-forward strategy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A new paradigm for operating a network, network coding was first introduced Ahlswede et al. in paper [1],which generalizes traditional routing by allowing each node to perform arbitrary operations on its inputs to generate the node's output. It was shown that the capacity of the network is equal to the size of the minimum cut that separates the source and any terminal. In a subsequent work, Li et al. [2] proved that linear network codes are sufficient to achieve the capacity of the network. An algebraic framework for linear network codes on directed graphs was developed by Koetter and Medard [3]. This framework was used by Ho et al. [4, 5] to construct random distributed network coding, which achieves the network capacity with probability exponentially approaching 1 with the code length. Jaggi et al. [6] proposed a polynomial-time algorithm for systematically finding feasible network codes. Opportunistic XOR coding, which allows coding between packets across different sessions, is proposed in [7].

Recently, it was shown that network coding can be used for optimizing the use of network resources. In [6, 8, 9], the authors investigate the problem of achieving the optimal throughput using network coding with multicast sessions. They show that, although computing optimal multicast throughput with routing involves solving NP-complete problems, the maximum multicast throughput and the corresponding optimal multicast strategy can indeed be computed efficiently in polynomial time. In [7,10–12] the authors demonstrate the benefits of network coding for power saving in wireless networks, which is an important advantage over traditional routing. Sev-

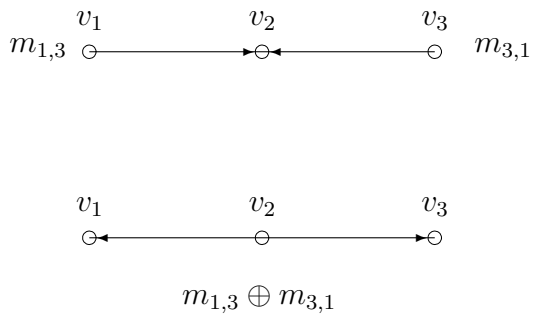


Figure 1.1: Transmitting messages $m_{1,3}$ and $m_{3,1}$ from v_1 to v_3 and v_3 to v_1, respectively, requires three transmissions with network coding and four without.

eral polynomial time algorithms are presented that minimize the power consumption in wireless networks with network coding. In [13,14], different optimization trade-offs in wireless networks with network coding are studied.

In Chapter 2, we demonstrate the benefits of network coding for optimizing the use of various network resources and propose several optimization algorithms.

In Section 2.1 and 2.2, we study a centralized and distributed design of network codes for low-power wireless multiple unicasts, respectively. It was shown that network coding is useful for information exchange [11], as illustrated in Fig 1.1. In the given example, node v_1 wishes to communicate a single packet of information to node v_3 while node v_3 wishes to communicate a single packet to node v_1. We label these packets as $m_{1,3}$ and $m_{3,1}$, respectively. Without network coding, meeting the given pair of demands requires four transmissions: each source transmits to node v_2 (2 transmissions), and then v_2 transmits first one and then the other message to its intended sink (2 more transmissions). With network coding we obtain a savings in energy – here simply measured by counting the number of transmissions required. The savings is achieved because in this case node v_2 can pass along both messages in a single coded transmission. Precisely, nodes v_1 and v_3 transmit packets $m_{1,3}$ and $m_{3,1}$ to node v_2 (two transmissions); node v_2 takes the bit-wise binary sum of its two received packets ($m_{1,3} \oplus m_{3,1}$) and then broadcasts the sum (one transmission).

Since both nodes v_1 and v_3 are within transmission range of node v_2, both receive the mixture packet $(m_{1,3} \oplus m_{3,1})$. Node v_1 decodes $m_{3,1}$ by taking the binary sum of $m_{1,3} \oplus m_{3,1}$ and its known value of $m_{1,3}$. Node v_3 similarly knows $m_{3,1}$ and receives $m_{1,3} \oplus m_{3,1}$, from which it can decode $m_{1,3}$. The given strategy generalizes from single packet transmissions across a two-hop network to information flows across a path with arbitrarily many hops. We call this strategy *reverse carpooling*. We use the word "carpooling" because the method allows two messages to effectively share a ride through the network: after an initial set-up period, every time an internal node transmits, it transmits a bit-wise binary sum of the next packet that it intends to send forward and the next packet that it intends to send backward along the path. For long paths and long sequences of packets the savings approaches a factor of two. We call it "reverse" carpooling because the strategy only applies when the information flows that want to share a ride are traveling opposite directions. In addition to the reverse carpooling advantage, network coding is useful at network crossroads, such as the bottleneck of the wireless butterfly example of [15] or other scenarios where overheard information may provide opportunities for coding [16]. An example of particular interest is shown in Fig. 1.2. Here a single packet $(x_{1,4})$ passes from node v_1 to node v_4, another $(x_{3,6})$ from node v_3 to node v_6, and a third $(x_{5,2})$ from node v_5 to node v_2, The routing solution requires a minimum of six transmissions as each node transmits its known packet to node v_7, which then sends each message along separately. In the network coding solution, here called 3-*star coding*, node v_7 finds the bit-wise binary sum $x_{1,4} \oplus x_{3,6} \oplus x_{5,2}$ and sends that value to all three receivers in a single broadcast transmission. In this case, node v_2 overhears node v_1's transmission of $x_{1,4}$ simply because it is one of v_1's neighbors; it likewise overhears $x_{3,6}$ due to its proximity to v_3. Node v_2 can therefore combine its overheard messages with the coded packet $x_{1,4} \oplus x_{3,6} \oplus x_{5,2}$ to decode the desired message $x_{5,2}$. Nodes v_4 and v_5 likewise overhear the messages that don't interest them and use those to decode their desired packets from $x_{1,4} \oplus x_{5,2} \oplus x_{3,6}$.

Each application of the 3-star coding strategy gives a savings of two transmissions. The same approach likewise applies when $k \geq 2$ paths cross, provided that the cross

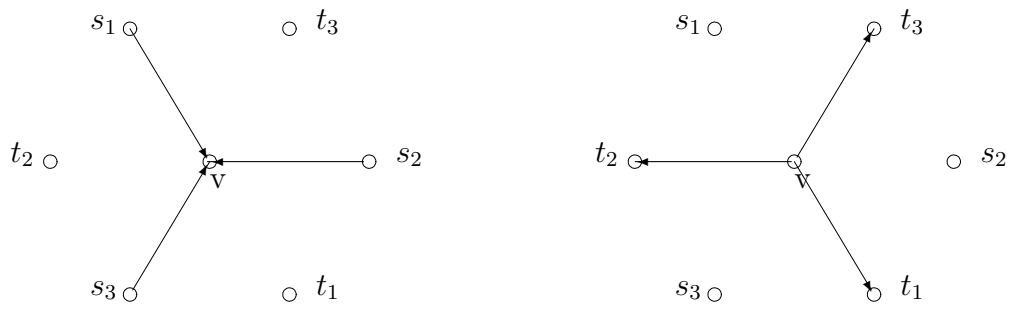


Figure 1.2: 3-star coding: s_i wants to transmit packet x_i to t_i $(1 \leq i \leq 3)$ and t_j overhears from s_i $(j \neq i)$. Node v broadcasts $x_1 \oplus x_2 \oplus x_3$ to t_1, t_2, and t_3 and it gives a savings of two transmissions.

configuration allows each of the intersection outputs to overhear the message that it doesn't require. Consider k different session's packets p_1, p_2,..,p_k at a node v that have distinct next-hop nodes v_1,..,v_k respectively. For each next-hop node v_i, if it is the previous-hop node of packet p_j or it overheard packet p_j from the transmission of its previous-hop node from opportunistic listening for $\forall j \neq i$, coded packet $p = p_1 \oplus p_2 \oplus \ldots \oplus p_k$ is broadcast to all the next-hop nodes v_1,..,v_k at node v. The savings from such a crossing, here called a k-star, is $k-1$ transmissions. The wireless butterfly network of [15] is one such example. The savings from such a crossing, here called a 2-star, is a single transmission.

In Section 2.1, we first propose a centralized algorithm using reverse carpooling that minimizes the number of transmissions for two unicasts. We extend this to the polynomial time greedy algorithm for general multiple unicasts problem. In Section 2.2, we present a distributed strategy using reverse carpooling for reducing the expected power consumption for wireless multiple unicasts. Our approach increases the reverse carpooling opportunities without requiring central coordination. A wireless rectangular grid is used as a simple network model. The proposed technique designates "reverse carpooling lines" analogous to a collection of bus routes in a crowded city. Each individual unicast then chooses a route from its source to its destination independently but in a manner that maximizes the fraction of its path

spent on reverse carpooling lines. Intermediate nodes apply reverse carpooling opportunistically along these routes. Our network optimization attempts to choose the reverse carpooling lines in a manner that maximizes the expected power savings with respect to the random choice of sources and sinks.

Section 2.3 introduces a new low complexity approach, network coding with periodic recomputation. In [12, 17], it is shown that network coding can be used for achieving the minimum cost coded subgraph for multicasting in mobile ad hoc networks (MANETs). The optimal solution can be obtained by solving a linear program every time slot, but it leads to high computational complexity. This motivates us to develop our approach with low complexity. In our approach, we recompute a suboptimal coded subgraph at fixed time intervals, and use this solution during each time interval although the network topology changes in MANETs. We obtain a simple theoretical cost bound on the gap between our solution and the optimal cost. We also analyze its computational complexity with an interior-point method, and show how our results can be applied to trade off performance and complexity in a given network scenario.

In Section 2.4, we investigate a back-pressure approach for network optimization with network coding. Most of the previous work focused on developing back-pressure type algorithms for multicast routing, session scheduling, and rate allocation, which maintain a queue for each session's packets at each node, and a route based on queue gradients that form by the addition of packets to sources and their removal from sinks [18–25]. However, in the case without network coding, the algorithms are significantly more complex and harder to implement, even for wired networks. By combining network coding with back-pressure approach, several distributed polynomial-time algorithms for optimizing the network resources are presented recently in [26–29]. We propose a back-pressure based distributed optimization framework, which can be used for optimizing over any class of network codes, including reverse carpooling and star coding. Our approach is to specify the class of coding operations by a set of generalized links, and to develop optimization tools that apply to any network composed of such generalized links. We prove that our algorithm achieve the stability for any

input rates within the capacity region.

In the second part of this thesis, we study the problem of error correction for network codes when links in the network may have unequal link capacities.

Network error-correction was first studied by Yeung and Cai [30,31] in the context of multicast network coding [1–3] on networks with unit-capacity links. Mixing the information content from different packets can increase the multicast throughput, but it can also potentially increase the impact of malicious links or nodes that corrupt data transmissions, since a single corrupted packet, mixed with other packets in the network, can corrupt all of the information reaching a destination. In the equal capacity link scenario, the Singleton bound is tight and linear network error-correcting codes suffice to achieve the capacity, which equals $C - 2z$ where C is the min-cut of the network [31, Theorem 4]. The problem of network coding under Byzantine attack was also investigated in [32], which gave an approach for detecting arbitrary errors under random network coding. Construction of codes that can correct errors up to the full error-correction capability specified by the Singleton bound was presented in [33]. A variety of alternative models of adversarial attack and strategies for detecting and correcting such errors appear in the literature. Examples include [34–41].

Here we consider network error correction with unequal link capacities. (A similar model, where adversaries control a fixed number of nodes in a network of arbitrary-capacity links rather than a fixed number of edges is studied in [42].) The unequal link capacity problem is substantially different from the problem studied by Yeung and Cai in [30, 31] since the rate controlled by the adversary varies with his edge choice. For the equal link capacities case, coding only at the source and forwarding at intermediate nodes suffices to achieve the capacity for any single-source and single-sink network. In contrast, for networks with unequal link capacities, we show that network error correction is needed even for a single-source and single-sink network.

Specifically, the network error correction problem concerns reliable information transmission in a network with the adversary who arbitrarily corrupts the packets sent on some set of z links. The location of the adversarial links is fixed for all time but unknown to the network user. A z-error-correcting network code is defined as

follows: if the total number of links in the network that may be corrupted by errors is at most z, then the source message can be recovered by the sink node.

The z-error correcting network capacity, henceforth simply called the capacity, is the supremum over all rates achievable by z-error correcting codes. Here we define a z-error link correcting code for a single-source and single-sink network to be a code that can recover the source message at the sink node if there are at most z adversarial links in the network; the code must work no matter what the capacity of the adversarial links.

We propose a new cut-set upper bound for the error-correction capacity for general acyclic networks. The standard cut-set bounding approach effectively treats all nodes on the same side of a given cut as a single super node. We therefore develop our cut-set bound by first studying the two-node network shown in Fig. 1.3. In this network, the source node can transmit packets to the sink node along the forward links and the sink node can send information back to the source node along the feedback links. We begin by characterizing the capacity of this network. However, this cut-set abstraction is insufficient to fully capture the effect of network topology relative to the cut since it assumes that all feedback is available to the source node and all information crossing the cut in the forward direction is available to the sink. We therefore introduce the four-node acyclic network shown in Fig. 1.4 as a step towards generalizing the cut-set bound. In this acyclic network, source node S and its neighbor node B lie on one side of a cut that separates them from sink node U and its neighbor A. As in the cut-set model, we allow unbounded reliable communication from source S to its neighbor B on one side of the cut and from node A to sink U on the other side of the cut; this differs from the original cut-set assumption only in that the communication is undirectional. We derive the capacity of this four-node network and use our result to generalize the cut-set bound. Since the resulting bound, like its predecessor, fails to capture the general network cut capacity, we introduce the zig-zag network model shown in Fig. 1.5 to generalize our four-node acyclic network model. Nodes A_i and B_i can communicate reliably with unbounded rate to A_{i+1} and B_{i+1}, respectively. The links from A_i to B_i represent feedback across the cut. This model more accurately

captures the behavior of any cut with k feedback links across the cut.

We first propose a new cut-set upper bound which applies to general acyclic networks. For networks with unequal link capacities, this bound tightens the generalized Singleton bound given in our earlier work in [43] and independently in [42]. We consider a variety of linear and nonlinear coding strategies useful for achieving the capacity of the example networks. We present a method for upper bounding the linear coding capacity of an arbitrary network and prove the insufficiency of linear network codes to achieve the capacity. We also give examples of single-source and single-sink networks for which intermediate nodes must perform coding, nonlinear error detection or error correction in order to achieve the network capacity. We then introduce a new coding strategy called "guess-and-forward." In this strategy, an intermediate node which receives some redundant information from multiple paths guesses which of its upstream links controlled by the adversary. The intermediate node forwards its guess to the sink which tests the hypothesis of the guessing node. We investigate this strategy with a variety of example networks. We show that guess-and-forward achieves network capacity on the two-node network with multiple feedback links, as well as the proposed family of four-node acyclic networks. Finally, we apply guess-and-forward strategy to the zig-zag networks, deriving a general achievable rate region and presenting conditions under which that bound is tight. This work also appears in [43, 44].

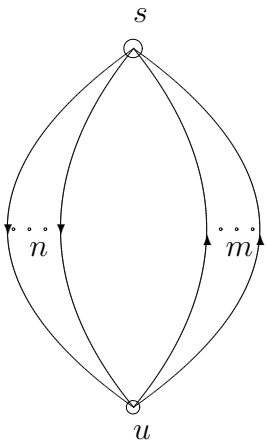


Figure 1.3: Point to point channel composed of n forward links and m feedback links.

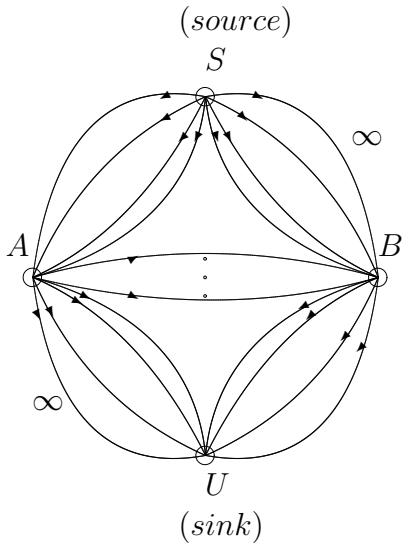


Figure 1.4: Four node acyclic networks: unbounded reliable communication is allowed from source S to its neighbor B on one side of the cut and from node A to sink U on the other side of the cut, respectively. There are feedback links from A to B.

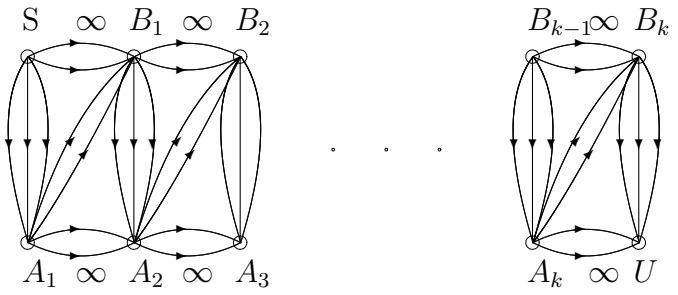


Figure 1.5: k-layer zig-zag network: A_i and B_i can communicate reliably with unbounded rate to A_{i+1} and B_{i+1}, respectively.($S = B_0$, $U = A_{k+1}$). The links from A_i to B_i represent feedback across the cut. This model more accurately captures the behavior of any cut with k feedback links across the cut.

Chapter 2

Network resource optimization

2.1 Centralized design of network codes for low-power wireless multiple unicasts

2.1.1 Introduction

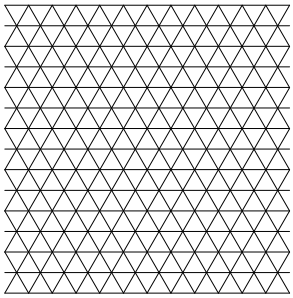


Figure 2.1: The nodes of the network lie on the vertices of a triangular lattice.

In this section, we investigate the use of reverse carpooling strategy for wireless multiple unicasts. As shown in Chapter 1, even for transmission of independent messages, reverse carpooling strategy provides a potential energy saving benefits for multiple unicasts in wireless networks. Effros *et al.* [10] present a recursive centralized algorithm that employs a dynamic programming argument for optimizing the power consumption using reverse carpooling and star-coding. However, the complexity of this algorithm makes its solution impractical for large networks and this motivates us to develop low computational complexity algorithms for the same problem.

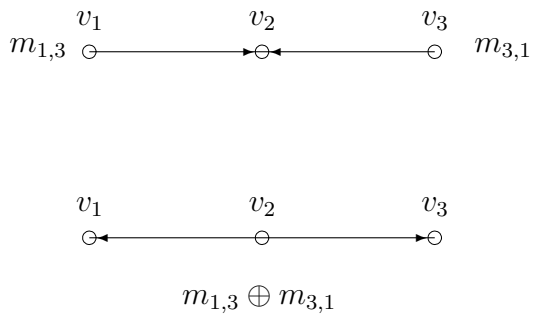


Figure 2.2: Reverse carpooling.

A wireless triangular grid is used as a simplified network model, as shown in Fig. 2.1. Each node corresponds to a vertex of a triangular grid, and it can broadcast information only to its six neighbor nodes. Each node directly receives all transmissions sent by its six neighbors. Thus there is direct communication only along edges connecting a node and its neighbor in the graph.

First, we propose a centralized algorithm that approximately minimizes the number of transmissions for two unicast sessions. This heuristic algorithm finds the minimal cost solution in $O(1)$ time. We extend this to obtain a greedy algorithm for general problem with multiple unicast sessions. The complexity of our greedy algorithm is $O(n^3)$ where n is the number of unicast sessions. We show by simulations that the algorithm reduces the power consumption significantly as the number of unicast sessions increases on the wireless triangular grid.

2.1.2 Preliminaries

2.1.2.1 Network

We define a *triangular grid* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as the set of vertices $\mathcal{V} = \{a(1,0) + b(\frac{1}{2}, \frac{\sqrt{3}}{2}) : a, b \in \mathbb{Z}\}$ where $\mathbb{Z}$ denotes the set of integers and the set of directed edges $\mathcal{E}= \{(v, v') : \|v - v'\| = 1\}$ where for any $v, v' \in \mathcal{V}$, (v, v') denotes the arc connecting v and v'. Thus each node has six incident and six outgoing edges, each of length

1. The head and tail of edge $e = (v_i, v_j)$ are denoted by $v_j = \text{head}(e)$ and $v_i = \text{tail}(e)$, respectively. Together, the edges in $\mathcal{E}$ form lines at angles 0, 60°, and 120° from the horizontal, as shown in Fig. 2.1; we call these lines grid lines. A *path* is an ordered list of connected edges. Precisely, for any path $P = (e_1, e_2, .., e_k)$, we require $e_1, e_2, .., e_k \in \mathcal{E}$ and $\text{head}(e_i) = \text{tail}(e_{i+1})$ for $1 \leq i \leq k-1$. For any $1 \leq i \leq j \leq k$, we call $P' = (e_i, e_{i+1}, .., e_j)$ a sub-path of $P = (e_1, e_2, .., e_k)$ and write $P' \subseteq P$. When $\text{tail}(e_i) = \text{head}(e_j)$ for some $i \leq j$, we call sub-path $P' = (e_i, e_{i+1}, .., e_j)$ a self-loop. We restrict our attention to paths without self loops; Lemma 2.1 shows that for our purposes, there is no loss of generality in this restriction. We use $l(P) = \sum_{e \in P} \|e\| = |P|$ to denote the length of path P. For any distinct vertices $v, v' \in \mathcal{V}$, we use $\mathcal{P}(v, v')$ to denote the set of all paths from v to v' in $\mathcal{G}$, $SP(v, v') = \arg\min_{P \in \mathcal{P}(v,v')}\{l(P)\}$ to denote the shortest path from v to v', and $d(v, v') = l(SP(v, v'))$ to denote the length of the shortest path.

2.1.2.2 Unicast

In a unicast session, a single source vertex $s \in \mathcal{V}$ transmits information to a single destination vertex $t \in \mathcal{V}$. In this paper, we consider multiple unicast sessions on a shared triangular grid. We specify a multiple unicast problem by describing the source and the destination for each unicast. For example, $U = \{(s_1, t_1), (s_2, t_2), ..., (s_n, t_n)\}$ is an n-unicast problem.

2.1.2.3 Reverse carpooling

Given a multiple unicast problem $U = \{(s_1, t_1), (s_2, t_2), ..., (s_n, t_n)\}$, a candidate solution $S = \{P_1, ..., P_n\}$ is a list of paths such that $P_i \in P(s_i, t_i)$ for each i. For any edge $e = (v, v') \in \mathcal{E}$, we use $e^R = (v', v)$ to denote the reversal of edge e. Likewise, for any path $P = (e_1, e_2, ..e_k)$, we use $P^R = (e_k^R, e_{k-1}^R, ..., e_1^R)$ to denote the reversal of path P. In candidate solution S, the opportunity to apply reverse carpooling arises when two paths, say P_i and P_j, contain sub-paths $P_i' \subseteq P_i$ and $P_j' \subseteq P_j$ satisfying $(P_i')^R = P_j'$. Such a sub-path is called a reverse carpooling segment. Note that reverse carpooling may not actually occur between sub-paths P_i' and P_j' if one of them is involved in

Figure 2.3: Reverse carpooling solution of two unicast sessions with one reverse carpooling segment and four branches.

reverse carpooling with another sub-path. Since paths P_i and P_j may reverse carpool along multiple non-consecutive segments, we use $K(P_i, P_j)$ to denote the number of reverse carpooling segments shared by P_i and P_j and $r_k(P_i, P_j)$ to denote the kth sub-path shared by P_i and P_j. If $K(P_i, P_j) = 1$, we simplify our notation to $r(P_i, P_j) = r_1(P_i, P_j)$. The sub-paths are numbered according to the order in which they appear in the first path (path P_i in $r_k(P_i, P_j)$). Each sub-path is as long as possible, and the sub-paths are disjoint.

To make these definitions precise, let $P_i = (e_1^{(i)}, ..., e_{l(P_i)}^{(i)})$ and $P_j = (e_1^{(j)}, ..., e_{l(P_j)}^{(j)})$. The following discussion defines t_k and $l_k = l(r_k(P_i, P_j))$ to be the start point (in P_i) and length, respectively, of $r_k(P_i, P_j)$ (provided that P_i and P_j have at least k reverse carpooling segments).

Initialize $t_0 = 0$, and $l_0 = 1$. Then for each subsequent $k \geq 1$ for which $t_{k-1} + l_{k-1} \leq l(P_i)$, let

$$\begin{aligned} t_k &= \min[\{n \in \{t_{k-1} + l_{k-1}, ..., l(P_i)\} : e_n^{(i)R} \in P_j\} \\ &\cup \{l(P_i) + 1\}]. \end{aligned}$$

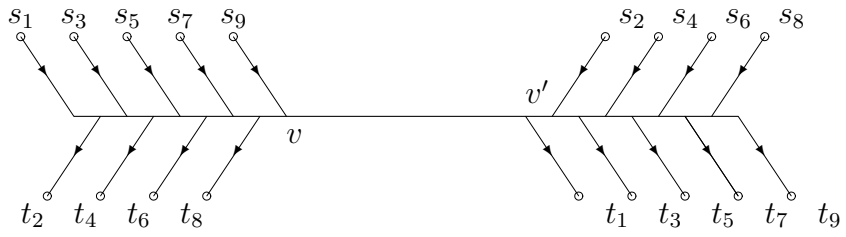


Figure 2.4: Illustration of cost of edge (v, v'): 5 unicasts use edge (v, v') and 4 unicasts use edge (v', v). Combined contribution of edges (v, v') and (v', v) to our estimated cost is 5.

If $t_k \leq l(P_i)$, let

$$l_k \quad = \quad \min\{n \in \{1, ..., l(P_i) - t_k\} : e^{(i)R}_{t_k+n} \notin P_j\}.$$

Then $r_k(P_i, P_j) = (e^{(i)}_{t_k}, ..., e^{(i)}_{t_k+l_k-1})$. We define branches B_{ijk} as $B_{ijk} = (e^{(i)}_{t_{k-1}+l_{k-1}}, ..., e^{(i)}_{t_k-1})$ if $t_k > t_{k-1} + l_{k-1}$, and $B_{ijk} = \phi$ otherwise.

If $t_k > l(P_i)$, then P_i and P_j share fewer than k reverse carpooling segments, $B_{ijk} = (e^{(i)}_{t_{k-1}+l_{k-1}}, ..., e^{(i)}_{l(P_i)})$, and the process stops. Fig. 2.3 shows an example with one reverse carpooling segment and four branches.

2.1.2.4 Network cost

The cost of a candidate solution is the energy consumed in a wireless network that transmits a single information stream along each path $P_i \in S$. When $n = 1$ (a single unicast session), we estimate the cost of candidate solution $S = \{P_1\}$ by the number of transmissions required to send a single packet from s_1 to t_1 along path P_1. Thus the cost of P_1 is the number of edges in path P_1, which equals $l(P_1)$. When $n > 1$, the opportunity for reverse carpooling may arise. We approximate the cost saved using

reverse carpooling by counting the link between nodes v and v' only once for each time we apply reverse carpooling along (v, v') and (v', v). For candidate solution S, the number of reverse carpooling opportunities along edge e using solution S is

$$R(S, e, e^R) = \min \left\{ \sum_{P \in S} 1(e \in P), \sum_{P \in S} 1(e^R \in P) \right\}.$$

Thus the resulting cost across edge e and e^R using candidate solution S is approximated as

$$\begin{aligned} C(S, e, e^R) &= \sum_{P \in S} 1(e \in P) + \sum_{P \in S} 1(e^R \in P) - R(S, e, e^R) \\ &= \max \left\{ \sum_{P \in S} 1(e \in P), \sum_{P \in S} 1(e^R \in P) \right\}, \end{aligned}$$

giving a total cost

$$C(S) = \frac{1}{2} \sum_{e \in E} \left\{ C(S, e, e^R) \right\}.$$

Fig. 2.4 gives an example. Edge(v, v') appears in five paths $(P_1, P_3, P_5, P_7, P_9)$. Edge (v', v) appears in four paths (P_2, P_4, P_6, P_8). Thus $R(S, (v, v')) = R(S, (v', v)) = 4$, and the combined contribution of edges (v, v') and (v', v) to our estimated cost $C(S)$ is 5.

The difference between the approximate cost $C(S)$ and the actual number of transmissions for a candidate solution S is at most the number of reverse carpooling segments. We show in Section 2.1.3 that the number of reverse carpooling segments is at most 1 for two unicasts problem. Fig. 2.5 illustrates the difference between cost and number of transmissions. Each network is a candidate solution in which reverse carpooling can be applied. Nodes a, b, c, and d can all apply reverse carpooling in the first network. In the second network, only nodes b and c can apply reverse carpooling. Reverse carpooling is not possible at nodes a and d in this example because nodes t_1

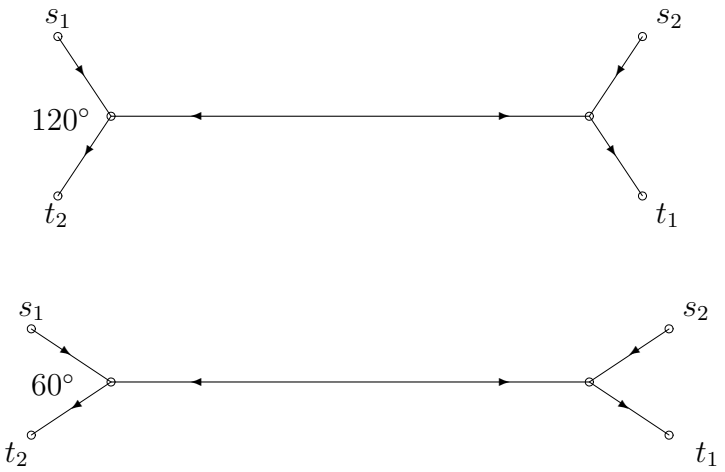


Figure 2.5: Reverse carpooling is possible at nodes a,b,c,and d in the first network. In the second network, reverse carpooling is not possible at nodes a, and d because t_2 cannot overhear the transmission from s_1 and t_1 cannot overhear the transmission from s_2. Thus the first network requires 6 transmissions, while the second network requires 8 transmissions. In both networks, we approximate the cost of S as $C(S)$ = 7. In general, for a reverse carpooling segment of n links shared by two unicast sessions, the actual number of transmissions is $n \pm 1$ while the approximate cost is n.

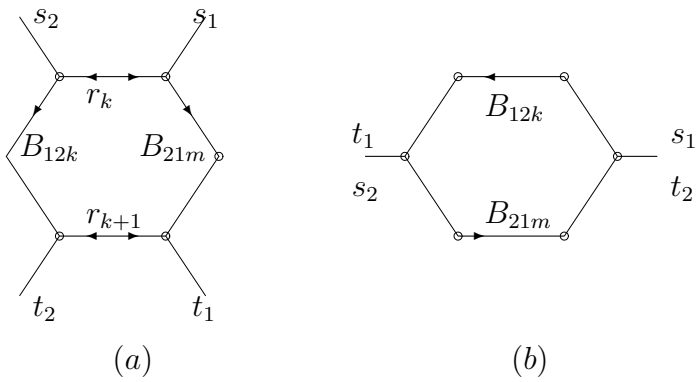


Figure 2.6: Illustration of a 4-exit loop and a 2-exit loop.

and t_2 are too far away from nodes s_2 and s_1, respectively, to overhear the information that they would need to unmix a joint transmission. Thus the first network requires 6 transmissions, while the second network requires 8 transmissions. In both networks, we approximate the cost of S as $C(S) = 7$. In general, for a reverse carpooling segment of n links shared by two unicast sessions, each of the $n-1$ intermediate nodes can satisfy both of its neighbors with a single transmission. However, the two end nodes may or may not be able to achieve a savings of this type. Thus, the actual number of transmissions is $n \pm 1$ while the approximate cost is n. We approximate the number of transmissions by the cost $C(S)$ throughout the paper. In Sections 2.1.3 and 2.1.4, we propose two polynomial time algorithms to minimize cost for two and multipe unicast sessions respectively.

2.1.2.5 Loop

In Sections 2.1.3, we show that for any two unicast sessions problem, there exists an optimal solution S^* for which no single path $P_i^* \in S^*$ contains a self-loop and for which any two paths P_i^*, $P_j^* \in S^*$ satisfy $K(P_i^*, P_j^*) \leq 1$. The following definitions are useful in that discussion. Given any two paths P_i and P_j for which $K(P_i, P_j) \geq 2$, P_i and P_j form $K(P_i, P_j) - 1$ loops between their reverse carpooling segments. These loops can

take two possible forms. In the first case, illustrated in Fig. 2.6(a), $r_k(P_i, P_j)$ and $(r_k(P_i, P_j))^R$ are both closer to the sources of their respective paths than $r_{k+1}(P_i, P_j)$ and $(r_{k+1}(P_i, P_j))^R$. In the second case, illustrated in Fig. 2.6(b), $r_k(P_i, P_j)$ is closer to the source of P_i than $r_{k+1}(P_i, P_j)$ (this must always be true by definition of $r_k(P_i, P_j)$), but $(r_{k+1}(P_i, P_j))^R$ is closer to the source of P_j than $(r_k(P_i, P_j))^R$. In the former case, the loop contains reverse carpooling segments $r_k(P_i, P_j)$ and $r_{k+1}(P_i, P_j)$ (and their reversals) and two branches B_{ijk} and B_{jim}. The two unicast sessions enter and exit the loop along four independent branches. We therefore call this loop a 4-exit loop, here denoted by $L(P_i, P_j, k, m) = (r_k(P_i, P_j),\ B_{ijk},\ r_{k+1}(P_i, P_j),\ B_{jim})$.

In the latter case, two branches, say B_{ijk} and B_{jim}, of P_i and P_j form a loop; reverse carpooling segments $r_k(P_i, P_j)$ and $r_{k+1}(P_i, P_j)$ (and their reversals) form the entrances and exits of this loop. We therefore call this loop a 2-exit loop. We use $L'(P_i, P_j, k, m) = (B_{ijk}, B_{jim})$ to denote this 2-exit loop.

We show that there always exists a minimal cost solution for multiple unicasts that contains no self-loops.

Lemma 2.1 *Given a n-unicast problem $U = \{(s_1, t_1), .., (s_n, t_n)\}$, there exists a minimal cost solution $S^* = (P_1, .., P_n)$ that contains no self-loops.*

Proof. Suppose that P_1 has a self-loop $P_{11} = (e_i^{(1)}, .., e_j^{(1)}) \subseteq P_1$. We define an alternative solution $S' = (P_1', P_2, .., P_n)$ with $P_1' = P_1 - P_{11}$. Then,

$$C(S') \leq C(S' \cup P_{11}) = C(S^*) \leq C(S') + l(P_{11}).$$

If $C(S') < C(S)$, we obtain a contradiction since S^* is optimal by assumption. Otherwise, $C(S') = C(S^*)$, and we can remove P_{11} without increasing cost. By repeating this argument, we can remove all self-loops while maintaining the optimal cost $C(S^*)$.

■

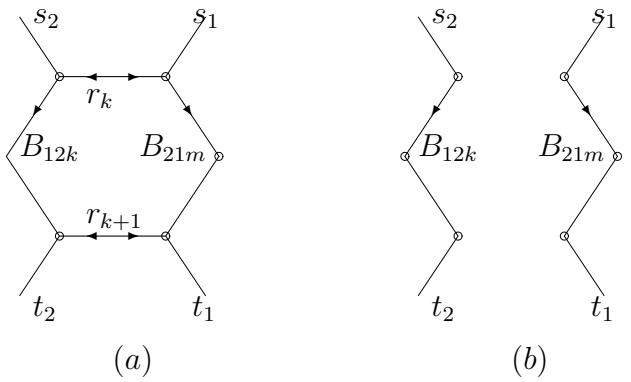


Figure 2.7: Illustration of proof of Theorem 2.2: (a) $S^* = (P_1, P_2, P_3)$ contains a 4-exit loop $L(P_1, P_2, k, m)$. (b) Redirecting both P_1 and P_2 as shown removes $r_k(P_1, P_2)$ and $r_{k+1}(P_1, P_2)$ and decreases the cost of the solution.

2.1.3 Two unicast sessions problem

This section presents a polynomial-time algorithm that finds an optimal cost solution for two unicast sessions (s_1, t_1), (s_2, t_2) on a triangular grid.

Lemma 2.1 and Theorem 2.2 help to characterize the form of an optimal solution for two unicast sessions.

Theorem 2.2 *Given a two-unicast problem $U = \{(s_1, t_1), (s_2, t_2)\}$, if $S^* = (P_1, P_2)$ is a minimal cost solution, then $K(P_1, P_2) \leq 1$, i.e., there is at most one reverse carpooling segment shared by P_1 and P_2.*

Proof. The proof is by contradiction. Suppose that $S^* = (P_1, P_2)$ is an optimal solution with $K(P_1, P_2) > 1$. Then S^* either contains a 2-exit loop or a 4-exit loop.

First, suppose that $S^* = (P_1, P_2)$ contains a 4-exit loop $L(P_1, P_2, k, m) = (r_k(P_1, P_2), B_{12k}, r_{k+1}(P_1, P_2), B_{21m})$, as shown in Fig. 2.7(a). Let $x = C(S^*) - C(L(P_1, P_2, k, m))$. Then

$$C(S^*) = x + l(r_k(P_1, P_2)) + l(r_{k+1}(P_1, P_2)) + l(B_{12k}) + l(B_{21m}).$$

As shown in Fig. 2.7(b), redirecting P_1 down B_{21m} and P_2 down B_{12k} (which re-

moves both reverse carpooling segments) decreases the cost of the solution, thereby contradicting the optimality of (P_1, P_2). Let $S' = (P_1', P_2')$ where

$$P_1' = (P_1 - (r_k(P_1, P_2) \cup B_{12k} \cup r_{k+1}(P_1, P_2))) \cup B_{21m},$$

$$P_2' = (P_2 - ((r_{k+1}(P_1, P_2))^R \cup B_{21m} \cup (r_k(P_1, P_2))^R)) \cup B_{12k}.$$

Then $C(S') = C(S^*) - l(r_k(P_1, P_2)) - l(r_{k+1}(P_1, P_2)) < C(S^*)$ since $l(r_k(P_1, P_2)) > 0$ and $l(r_{k+1}(P_1, P_2)) > 0$ by definition of a reverse carpooling segment. This gives the desired contradiction.

Now suppose that $S^* = (P_1, P_2)$ contains a 2-exit loop $L'(P_1, P_2, k, m) = (B_{12k}, B_{21m})$, as shown in Fig. 2.6(b). The following argument shows that directing both paths down one side of the loop decreases the cost. This gives a contradiction (since S^* is optimal by assumption) and therefore proves that S^* cannot contain a 2-exit loop. Without loss of generality, assume that $l(B_{12k}) \geq l(B_{21m})$ and define an alternative solution $S' = (P_1', P_2)$ with

$$P_1' = P_1 - B_{12k} \cup (B_{21m})^R.$$

Then, P_1 and P_2 can reverse carpool along B_{21m} and thus $C(S') = C(S^*) - l(B_{12k}) < C(S^*)$, which gives the desired result. ■

Given any $a, b, c \in \mathcal{V}$ that are not collinear, we use $\triangle abc$ to denote the triangle with corners at a, b, and c. We use $\angle a$ to denote the angle between lines (b, a) and (c, a), $\angle b$ to denote the angle between lines (a, b) and (c, b), and $\angle c$ to denote the angle between lines (a, c) and (b, c).

Theorem 2.3 *For any $a, b, c \in \mathcal{V}$ that are not collinear, we can find the largest set*

$$\begin{aligned} P &= \{x \in \mathcal{V} : d(a, x) + d(b, x) + d(c, x) \\ &\leq d(a, y) + d(b, y) + d(c, y), \forall y \in \mathcal{V}\} \end{aligned}$$

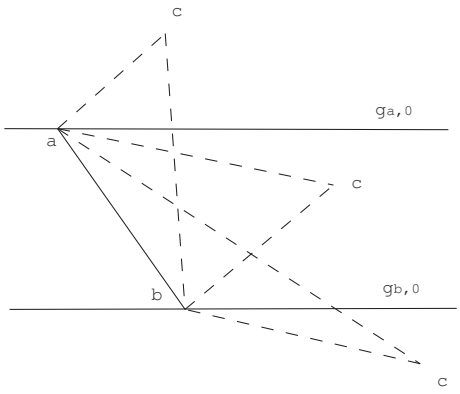


Figure 2.8: Lemma 2.4 of Theorem 2.3. First portion of proof: If c is above $g_{a,0}$, then $g_{b,0} \notin \triangle abc$. If c is below $g_{b,0}$, then $g_{a,0} \notin \triangle abc$. If c is between $g_{a,0}$ and $g_{b,0}$, then $g_{a,0}$, $g_{b,0} \notin \triangle abc$.

in $O(1)$ *time.*

Before proving the theorem, we prove a lemma that bounds the number of grid lines contained in $\angle a$, $\angle b$, and $\angle c$ of $\triangle abc$. If any side of the triangle corresponds to a grid line, then we count that grid line only once. The following notation is useful for that discussion. For each $v \in \{a, b, c\}$ and $\theta \in \{0°, 60°, 120°\}$, let $g_{v,\theta}$ denote the grid lines at angle θ through vertex v. We write $g \in \triangle abc$ to specify that grid line g is contained in one or more of the angles in $\triangle abc$ and for each $\theta \in \{0°, 60°, 120°\}$ and $v \in \{a, b, c\}$, define

$$G_\theta = \{g_{v',\theta} : v' \in \{a, b, c\}, g_{v',\theta} \in \triangle abc\}.$$

$$G^v = \{g_{v,\theta'} : \theta' \in \{0°, 60°, 120°\}, g_{v,\theta'} \in \triangle abc\}.$$

We prove the desired result by first proving that $|G_\theta| \leq 1$ and then proving that $|G_\theta| \geq 1$ for each $\theta \in \{0°, 60°, 120°\}$. (Here, $|A|$ denotes the number of *distinct* elements in set A.) Both proofs are by contradiction.

Lemma 2.4 *Given any* $a, b, c \in \mathcal{V}$ *that are not collinear,* $|G_\theta| = 1$ *for each* $\theta \in \{0°, 60°, 120°\}$*, and* $|\cup_{v \in \{a,b,c\}} G^v| = |\cup_{\theta \in \{0°,60°,120°\}} G_\theta| = 3$.

Proof.

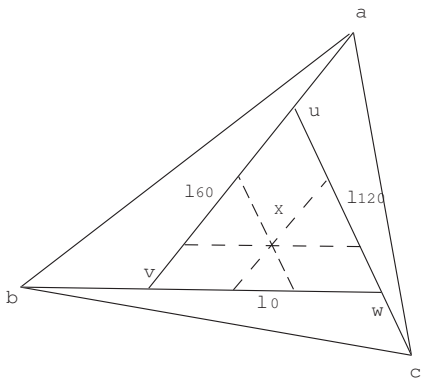


Figure 2.9: Case 1 of Theorem 2.3: $\angle a$,$\angle b$,$\angle c$ contain one gridline respectively, $g_{a,60}$, $g_{b,0}$, and $g_{c,120}$. These grid lines form an equilateral triangle $\triangle uvw$.

Suppose that $|G_\theta|>1$ for some $\theta\in\{0°, 60°, 120°\}$. Without loss of generality, we label $g_{a,\theta}$ and $g_{b,\theta}$ as distinct elements of G_θ. Grid lines $g_{a,\theta}$ and $g_{b,\theta}$ are parallel lines. Let a be the higher of the two and recall that (a, b) is one side of $\triangle abc$ (See Fig. 2.8). If c is above $g_{a,\theta}$, then $g_{b,\theta} \notin \triangle abc$, which gives a contradiction. If c is below $g_{b,\theta}$, then $g_{a,\theta} \notin \triangle abc$, which again gives a contradiction. If c lies between $g_{a,\theta}$ and $g_{b,\theta}$, then $g_{a,\theta}$, $g_{b,\theta} \notin \triangle abc$ gives the final contradiction and completes the first portion of our proof.

Now suppose that $|G_\theta| = 0$ for some $\theta\in\{0°, 60°, 120°\}$. If $|\{g_{a,\theta}, g_{b,\theta}, g_{c,\theta}\}|<3$, then assume without loss of generality, that $g_{a,\theta} = g_{b,\theta}$. This implies that $g_{a,\theta}$ is collinear with line (a, b) and is therefore contained in $\triangle abc$, which gives a contradiction. If $|\{g_{a,\theta}, g_{b,\theta}, g_{c,\theta}\}| = 3$, then $g_{a,\theta}$, $g_{b,\theta}$, and $g_{c,\theta}$ are distinct parallel grid lines. We assume without loss of generality that $g_{c,\theta}$ is the center line. (See Fig. 2.8). Then $g_{c,\theta}\in\triangle abc$, which gives the desired contradiction. ■

Proof of Theorem 2.3: Lemma 2.4 allows us to break the proof into three cases.

Case 1) $\max_{v\in\{a,b,c\}} |G^v| < 3$.

In this case, we show that the desired set contains all vertices in a triangle formed by $\cup_{v\in\{a,b,c\}} G^v$. By Lemma 2.4, $|\cup_{v\in\{a,b,c\}} G^v| = 3$ and $|G_\theta| = 1$ for each $\theta\in\{0°, 60°, 120°\}$. Since $\max_{v\in\{a,b,c\}} |G^v| < 3$, these grid lines form an equilateral triangle. We label $u, v, w \in \mathcal{V}$ are the triangle corners closest to a, b, and c, respectively and the grid lines in $\cup_{v\in\{a,b,c\}} G^v$ as g_{uv}, g_{vw}, and g_{uw}, where g_{uv} is the

grid line that contains u and v. Let l denote the side length for $\triangle uvw$. We show $d(a,x)+d(b,x)+d(c,x)$ is constant for all $x\in\triangle uvw$ (that is all $x\in\mathcal{V}$ that lie in triangle $\triangle uvw$ or on its boundary). We then show that for any $x\in\triangle uvw$ and $y\notin\triangle uvw$,

$$d(a,y)+d(b,y)+d(c,y) > d(a,x)+d(b,x)+d(c,x).$$

To prove these results, first note that for any vertex $x\in\triangle uvw$, there exists a shortest path between a and x that passes through u. The shortest paths from b to x and c to x can likewise pass through v and w, respectively. Let l_0, l_{60}, and l_{120} be the side lengths for the three equilateral triangles formed by intersecting $g_{x,\theta}$ with the sides of $\triangle uvw$. (See Fig. 2.9.) Then $l = l_0 + l_{60} + l_{120}$, and

$$\begin{aligned} & d(a,x)+d(b,x)+d(c,x) \\ = \;& (d(a,u)+l_{120}+l_{60})+(d(b,v)+l_{60}+l_0) \\ & +(d(c,w)+l_0+l_{120}) \\ = \;& d(a,u)+d(b,v)+d(c,w)+2l. \end{aligned}$$

Therefore, $d(a,x)+d(b,x)+d(c,x)$ is constant for any vertex $x\in\triangle uvw$.

For any $y\notin\triangle uvw$, two of the three grid lines in $\{g_{y,0}, g_{y,60}, g_{y,120}\}$ form an equilateral triangle with the nearest one of g_{uv}, g_{vw}, and g_{uw}, as shown in Fig. 2.10. We assume that y is above g_{uw}. We use l_y to denote the side length of that triangle and y' to denote one of the nearest corner of triangle from $\triangle uvw$. Since $d(a,u)+d(c,u)=d(a,c)$ as shown in Fig. 2.10 and Fig. 2.11,

$$d(a,y)+d(c,y) \geq d(a,c) = d(a,u)+d(c,u).$$

In addition, since y is above g_{uw}, $d(b,y')\geq d(b,u)$. Thus,

$$d(b,y) = d(b,y')+l_y > d(b,y') \geq d(b,u).$$

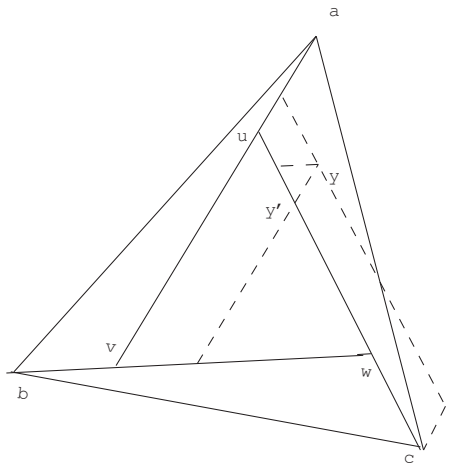


Figure 2.10: Case 1 of Theorem 2.3: Any point outside of the triangle uvw cannot be a solution.

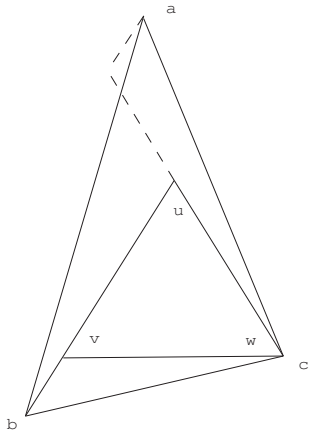


Figure 2.11: Case 1 of Theorem 2.3: Any point outside of the triangle uvw cannot be a solution.

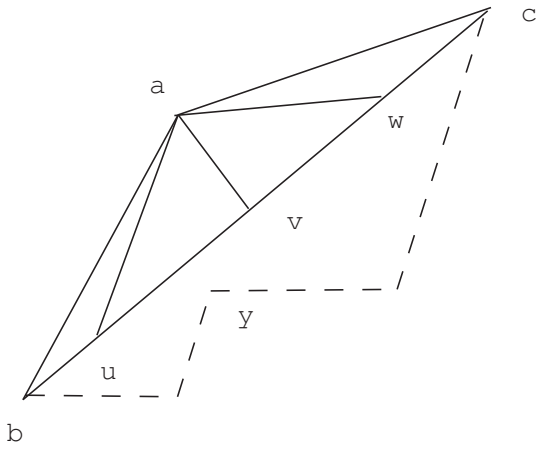


Figure 2.12: Case 2 of Theorem 2.3: $\angle a$ contains three grid lines. a is the unique solution point.

Then

$$\begin{aligned} & d(a,y) + d(b,y) + d(c,y) \\ \geq\ & d(a,u) + d(b,y') + d(c,u) + l_y \\ >\ & d(a,u) + d(b,u) + d(c,u) \\ =\ & d(a,u) + d(b,v) + d(c,w) + 2l. \end{aligned}$$

Case 2) One of $\angle a$, $\angle b$, and $\angle c$ contains three grid lines.

In this case, we show that P contains only the corner of $\triangle abc$ that contains the triangle's three grid lines. We assume that $\angle a$ contains three grid lines $g_{a,0}$, $g_{a,60}$, and $g_{a,120}$, and b is below $g_{a,0}$ and above $g_{a,60}$ as shown in Fig. 2.12. Then $\angle b$ and $\angle c$ contain no grid lines. We show that for any $y \neq a$,

$$d(a,y) + d(b,y) + d(c,y) > d(a,b) + d(a,c).$$

Since b is below $g_{a,0}$ and above $g_{a,60}$ and c is above $g_{a,0}$ and below $g_{a,60}$, there exists a shortest path between b and c that passes through a. Thus, $d(b,c) = d(b,a) + d(a,c)$, and

$$d(b,y) + d(c,y) \geq d(b,c) = d(a,b) + d(a,c).$$

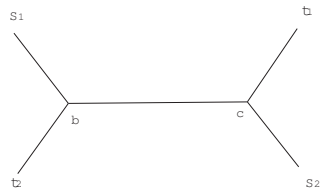


Figure 2.13: Reverse carpooling solution $S(b, c)$.

In addition, $d(a, y)>0$. Therefore,

$$\begin{aligned} & d(a, y) + d(b, y) + d(c, y) \\ > \quad & d(a, b) + d(a, c). \end{aligned}$$

This gives the desired result.

In two cases, the result follows from the fact that the time complexity of finding G_θ is $O(1)$.

In theorem 2.5, we find locations of $rc(s_1, t_2)$ and $rc(s_2, t_1)$ to obtain the optimal cost reverse carpooling solution and compare its cost with the cost of the shortest paths solution, $l = d(s_1, t_1) + d(s_2, t_2)$. If l is less than the cost of the optimal reverse carpooling solution, the optimal solution is the shortest paths solution. Otherwise, we choose the optimal cost reverse carpooling solution.

Theorem 2.5 *Given a two unicast problem $U = \{(s_1, t_1), (s_2, t_2)\}$, we can find a minimal cost solution $S^* = (P_1, P_2)$ in $O(1)$ time.*

Proof. By Theorem 2.2, for two unicast sessions (s_1, t_1) and (s_2, t_2) , there exists an minimal cost solution $S^* = (P_1, P_2)$ for which $K(P_1, P_2)\leq 1$. If $K(P_1, P_2) = 0$, using a shortest path for each unicast session gives the optimal solution. We use ($SP(s_1, t_1)$, $SP(s_2, t_2)$) to denote the shortest path solution. Otherwise, S^* contains exactly one reverse carpooling segment $r(P_1, P_2)$. Let $r(P_1, P_2) = (e_k^{(1)}, e_{k+1}^{(1)}, ..., e_{k+m-1}^{(1)})$.

We use $rc(s_1, t_2) = \text{tail}(e_k^{(1)})$ and $rc(t_1, s_2) = \text{head}(e_{k+m-1}^{(1)})$ to denote the two endpoints of $r(P_1, P_2)$. As shown in Fig. 2.13, let b and c denote $rc(s_1, t_2)$ and $rc(t_1, s_2)$ respectively for brevity. Given locations of b and c, the reverse carpooling solution is

$S(b,c) = (P_1, P_2)$ where

$$P_1 = SP(s_1, b) \cup r(P_1, P_2) \cup SP(c, t_1),$$

$$P_2 = SP(s_2, c) \cup (r(P_1, P_2))^R \cup SP(b, t_2),$$

and

$$\begin{aligned} C(S(b,c)) &= l(SP(s_1,b)) + l(SP(t_2,b)) + l(SP(b,c)) \\ &+ l(SP(s_2,c)) + l(SP(t_1,c)). \end{aligned} \tag{2.1}$$

We use $S' = (P_1', P_2')$ to denote the minimal cost reverse carpooling solution. We compare $C(S')$ with $l = d(s_1,t_1) + d(s_2,t_2)$. If $C(S') \leq l$, then $S^* = S'$. Otherwise, $S^* = (SP(s_1,t_1),\ SP(s_2,t_2))$. For a reverse carpooling solution with given b to be optimal, the location of c must satisfy

$$d(c,b) + d(c,t_1) + d(c,s_2) \leq d(p,b) + d(p,t_1) + d(p,s_2)$$

for all $p \in \mathcal{V}$. By the construction in the proof of Theorem 2.3,

(i) If $\angle b$ in $\triangle bs_2t_1$ contains more than one grid line,then $c = b$. In this case, $b = rc(s_1,t_2) = rc(t_1,s_2)$ for some b. The corresponding reverse carpooling solution is $S' = (P_1', P_2')$ where $P_1' = (SP(s_1,b) \cup SP(b,t_1)$, and $P_2' = (SP(s_2,b) \cup SP(b,t_2)$. Then, the reverse carpooling segment length $l(r(P_1,P_2)) = 0$. Since there is no cost reduction for reverse carpooling, the shortest paths solution is at least as good.

(ii) If $\angle s_2$ in $\triangle bs_2t_1$ contains more than one grid line,then $c = s_2$.

(iii) If $\angle t_1$ in $\triangle bs_2t_1$ contains more than one grid line,then $c = t_1$.

(iv) Otherwise, when each angle in $\triangle bs_2t_1$ contains one grid line as shown in Fig. 2.14, we assume that $\angle s_2$ in $\triangle bs_2t_1$ contains $g_{s_2,\alpha}$ and $\angle t_1$ in $\triangle bs_2t_1$ contains $g_{t_1,\beta}$ where $\alpha, \beta \in (0°, 60°, 120°)$, $\alpha \neq \beta$. We use $a_{\alpha,\beta}$ to denote the intersection between $g_{s_2,\alpha}$ and $g_{t_1,\beta}$. Then, by Theorem 2.3, we can get $c = a_{\alpha,\beta}$.

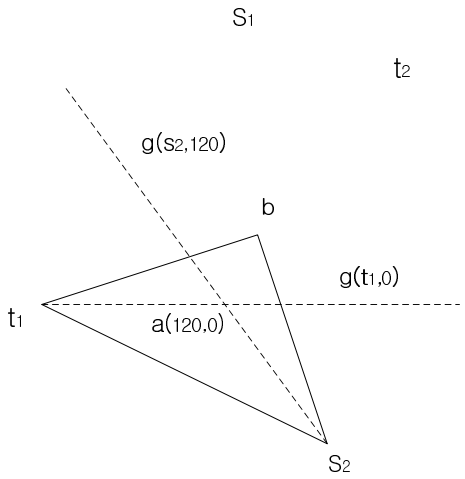


Figure 2.14: Each angle in $\triangle bs_2t_1$ contains one grid line.

Thus, we only need to consider the following set of possible locations for c : I = $\{s_2,\ t_1,\ a_{0^\circ,60^\circ},\ a_{0^\circ,120^\circ},\ a_{60^\circ,0^\circ},\ a_{60^\circ,120^\circ},\ a_{120^\circ,0^\circ},\ a_{120^\circ,60^\circ}\}$. For each $c \in I$, we use $b(c) \in \mathcal{V}$ to denote the corresponding location for the other end point b of the reverse carpooling segment. Then, we can find $b(c) \in \mathcal{V}$ such that

$$d(b(c), c) + d(b(c), s_1) + d(b(c), t_2) \leq d(p, c) + d(p, s_1) + d(p, t_2)$$

for all $p \in \mathcal{V}$ in $O(1)$ time, by Theorem 2.3. By comparing the costs of the reverse carpooling solutions, given by (2.1), for these 8 possibilities for c, we can obtain the minimum cost reverse carpooling solution, i.e. $C(S') = \min_{C \in I}\{C(S(b(c), c)\}$. Finally, we compare $C(S')$ with l and obtain an optimal cost solution. ■

2.1.4 General multiple unicasts problem

In this section, we generalize our problem to general multiple unicasts problem and introduce a greedy algorithm to obtain an approximate solution. In Section 2.1.3, we described a polynomial time algorithm which finds an optimal cost solution for the two unicast sessions problem. Based on this algorithm, we present a greedy algorithm to obtain a suboptimal solution for n-unicast sessions (s_1, t_1),...,(s_n, t_n) on the triangular grid.

We define a metric m_{ij}, $(i, j \in \{1, ..n\}, i \neq j)$ and a selection function I. We use

$S_{(i,j)}$ to denote an optimal cost solution obtained by applying Theorem 2.5 to two unicast sessions (s_i, t_i) and (s_j, t_j). Let $m_{ij} = d(s_i, t_i) + d(s_j, t_j) - C(S_{ij})$. Given (s_i, t_i) and (s_j, t_j), m_{ij} denotes the difference between the cost of the shortest paths solution and the optimal cost. The selection function $I : N \subset (1, 2, ..n) \rightarrow N\text{x}N$ chooses a pair of indices in N which maximizes the metric m_{ij}. Precisely,

$$I(N) = \arg\max_{i,j \in N}\{m_{ij}\}.$$

We use CS to denote the current solution and $N \subset (1, 2, ..n)$ to denote a set of indices which are not used in the current solution at each step. In each step of the algorithm, we update two sets CS and N using the selection function I. Then, we remove $I(N)$ from N and add $S_{I(N)}$ to CS at each step. Finally, at the end of the algorithm, we obtain a suboptimal solution $CS = S = (P_1, .., P_n)$.

$N \leftarrow \{1, 2, ..., n\}$
$CS \leftarrow \emptyset$
  While $|N|>1$
    $N = N - I(N)$
    $CS = CS \cup_{I(N)}$
  endwhile
 If $N = \{k\} \quad (1 \leq k \leq n)$
   return $CS = CS \cup SP(s_k, t_k)$
 else
    return CS
  endif

Theorem 2.6 *The time complexity of the above greedy algorithm is* $O(n^3)$.

Proof. If $|N| = 1$, greedy algorithm returns the shortest paths solution. Otherwise, in each step, the selection function I chooses any pairs of indices among N to compute a metric corresponding these indices and selects a pair of indices which maximizes a metric. By Theorem 2.5, it takes $O(1)$ time to compute a metric for given two unicast sessions. Thus, the time-complexity is $O\left(\binom{|N|}{2}\right)$ in each step. In the worst case, the maximum metric is 0 and thus $|N|$ is decreased by one at each

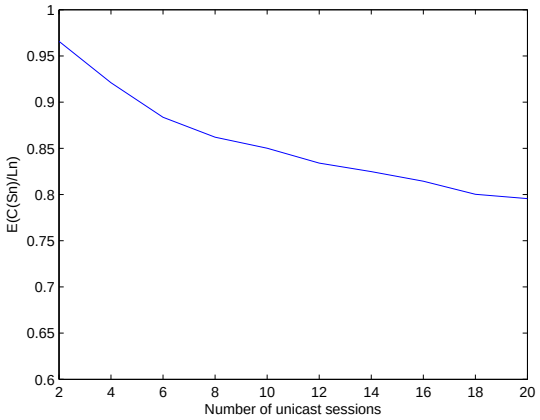


Figure 2.15: Simulation result: As the number of unicast sessions is increased, $E(\frac{C(S_n)}{L_n})$ is decreased.

step. In this case, the time-complexity is

$$O(\sum_{i=1}^{n} \binom{n-i}{2}) = O(n^3).$$

■

2.1.5 Simulation

In order to determine the effectiveness of reverse carpooling, we constructed a simulation environment which models operation on the wireless triangular grid. Given a wireless triangular grid, we choose the locations of unicast sessions uniformly at random on the grid and compare the average network cost between a suboptimal solution obtained by the greedy algorithm of Section 2.1.4 and the shortest paths solution without network coding.

Given n-unicast problem $U = ((s_1, t_1), .., (s_n, t_n))$, we use S_n to denote a suboptimal solution obtained by our greedy algorithm. Let $L_n = \sum_{i=1}^{n} l(SP(s_i, t_i))$ denote the cost of the shortest paths solutions. Our evaluation uses the performance metric $E(\frac{C(S_n)}{L_n})$ which is the average ratio between the cost of the suboptimal solution obtained by our greedy algorithm and the cost of the shortest paths solution.

We use a *triangular grid* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as the set of vertices $\mathcal{V} = \{a(1,0) + b(\frac{1}{2}, \frac{\sqrt{3}}{2})$

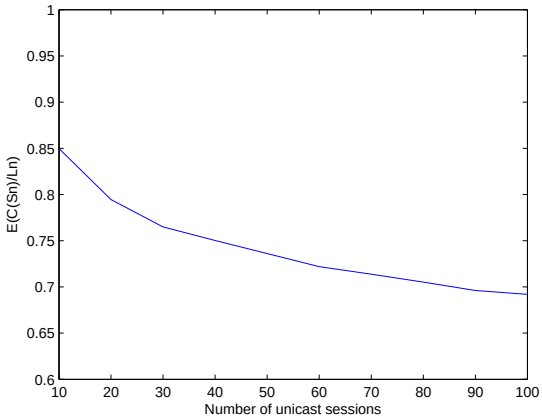


Figure 2.16: Simulation result: We extend the result in Fig. 2.8. When 100 unicast sessions are chosen uniformly at random on the grid, $E(\frac{C(S_n)}{L_n})$ is 0.69.

: $-5 \leq a \leq 5, -10 \leq b \leq 10\}$ and randomly choose the locations of a given number of unicast sessions. Fig. 2.15 indicates that when 20 unicasts are chosen uniformly at random on the graph G, the average cost of the greedy reverse carpooling solution is 0.79 times that of the shortest paths solution. As the number of unicast sessions increases, $E(\frac{C(S_n)}{L_n})$ decreases. This result agrees with our intuition that the number of opportunities to apply reverse carpooling increases with the number of unicast sessions in a given network. As shown in Fig. 2.16, when $n = 100$, $E(\frac{C(S_n)}{L_n})$ is 0.69. When n is sufficiently large, $E(\frac{C(S_n)}{L_n})$ approaches 0.5, which is the best possible ratio between the reverse carpooling solution and the shortest path solution.

2.1.6 Conclusion

In this section, we have studied the problem of low-power transmission for wireless multiple unicasts on triangular grid. We have presented $O(1)$ time algorithm to obtain approximately optimal solutions which minimize the number of transmissions for two unicast sessions. By extending the algorithm for two unicasts problem, we presented a $O(n^3)$ time greedy algorithm to obtain an approximate solution for n-unicast sessions problem. From simulations, we have shown that our algorithm reduces the power consumption significantly for multiple unicasts on a wireless triangular grid.

2.2 Distributed design of network codes for low-power wireless multiple unicasts

2.2.1 Introduction

Previous results on network coding for low-power wireless transmissions of multiple unicasts rely on opportunistic coding or centralized optimization to reduce the power consumption. While our greedy algorithm in Section 2.1 runs in polynomial time, it requires a central controller.

In this section, we develop a distributed strategy for reducing the expected power consumption for multiple unicasts in a network coded wireless network. Our strategy attempts to increase network coding opportunities without the overhead required for centralized design or coordination. A wireless rectangular grid is used as a simple network model. As in [45], a single node sits on each vertex of a rectangular grid, and each node can broadcast information only to its four nearest neighbors. The goal is to transmit a distinct data stream from each transmitter to its corresponding receiver in this shared network environment. Power savings are achieved using the reverse carpooling strategy again.

Our strategy is to attempt to increase the number of coding opportunities by designating "reverse carpooling routes" in central locations and choosing unicast routes to maximize the fraction of the path spent on carpooling routes without increasing individual path lengths. The hope is that careful route choice will maximize the expected number of reverse carpooling opportunities. Intermediate nodes apply reverse carpooling opportunistically along these routes. Our network optimization attempts to choose the reverse carpooling lines in a manner that maximizes the expected power savings with respect to the random choice of sources and sinks.

This section is organized as follows. In Section 2.2.3, for each network model, we design a reverse carpooling edge set $E_1 \subseteq E$. Together, the edges in E_1 form reverse carpooling routes. We first design E_1 so that each reverse carpooling route is a horizontal grid line. We call this reverse carpooling route a row reverse carpooling

line and this network model a row model. We begin by proposing a distributed route choice algorithm for an arbitrary row model and analyzing the edge use distribution of our algorithm. We then optimize the reverse carpooling line placement to minimize the resulting expected cost.

In Section 2.2.4, we design E_1 to contain both horizontal and vertical grid lines. We again propose an algorithm, analyze the resulting edge use distribution, and optimize the line choice.

2.2.2 System Model

We define a *rectangular grid* $G_m = (V, E)$ as the set of vertices $V = \{a(1,0)+b(0,1)\}$: $0 \leq a, b \leq m\}$ and the set of directed edges E= $\{(v, v') : \|v - v'\| = 1\}$ where for any $v, v' \in V$, (v, v') denotes the arc connecting v and v'. The head and tail of edge $e = (v_i, v_j)$ are denoted by $v_j = \text{head}(e)$ and $v_i = \text{tail}(e)$, respectively. We call the horizontal and vertical lines formed by E grid lines. A *path* is an ordered list of connected edges. Precisely, for any path $P = (e_1, e_2, .., e_k)$, we require $e_1, e_2, .., e_k \in E$ and $\text{head}(e_i) = \text{tail}(e_{i+1})$ for $1 \leq i \leq k-1$. We use $l(P) = \sum_{e \in P} \|e\| = |P|$ to denote the length of path P. For any distinct vertices v,$v' \in V$, we use $\mathcal{P}(v, v')$ to denote the set of all paths from v to v' in G_m, $\mathcal{P}^*(v, v') = \arg\min_{P \in \mathcal{P}(v,v')} l(P)$ to denote the set of the shortest paths from v to v', and $d(v, v')$ to denote the length of the shortest path from v to v'.

We use the same definition for the network cost with reverse carpooling as that in Section 2.1.

2.2.3 Row Models

The optimal configuration of the reverse carpooling lines may depend on factors like the size of the network, the number of unicasts, the distribution on unicasts, etc. We assume that n unicasts $U = \{(s_1, t_1), .., (s_n, t_n)\}$ are chosen uniformly at random on the wireless rectangular grid G_m and begin by exploring simple row models. Given a wireless rectangular grid G_m, we use a t-tuple $(0 \leq h_1 < h_2 < .. < h_t \leq m)$

to denote the locations of t row reverse carpooling lines. (For convenience, $h_0 = 0$ and $h_{t+1} = m + 1$.) We define the reverse carpooling edge set $E_1 = \{((i, h_j), (i + 1, h_j)), ((i + 1, h_j), (i, h_j)) : 0 \leq i \leq n - 1, 1 \leq j \leq t\}$. Then edges in E_1 form t row reverse carpooling lines.

2.2.3.1 Path Choice Algorithm

The proposed algorithm finds a shortest path $P_i \in \mathcal{P}^*(s_i, t_i)$ that maximizes the fraction of the path spent on the row reverse carpooling lines. Let $s_i = (s_{ix}, s_{iy})$ and $t_i = (t_{ix}, t_{iy})$ and choose $0 \leq p, q \leq t$ so that $h_p \leq s_{iy} < h_{p+1}$ and $h_q \leq t_{iy} < h_{q+1}$.

Case 1) $p = q$. Here P_i is the unique path in $\mathcal{P}^*(s_i, u_i) \times \mathcal{P}^*(u_i, t_i)$ where $u_i = (t_{ix}, s_{iy})$.

Case 2) $p < q$. Here P_i is the unique path in $\mathcal{P}^*(s_i, v_i) \times \mathcal{P}^*(v_i, w_i) \times \mathcal{P}^*(w_i, t_i)$, where $v_i = (s_{ix}, h_{p+1})$ and $w_i = (t_{ix}, h_{p+1})$.

Case 3) $p > q$. Here P_i is the unique path in $\mathcal{P}^*(s_i, x_i) \times \mathcal{P}^*(x_i, y_i) \times \mathcal{P}^*(y_i, t_i)$, where $x_i = (s_{ix}, h_p)$ and $y_i = (t_{ix}, h_p)$.

2.2.3.2 Edge Use Distribution

Together, the edge set and path choice strategy impose a traffic distribution $r^i(e)$ for each $e \in E$ where $r^i(e)$ is the probability that $e \in P_i$. Since each unicast session is chosen uniformly at random and follows the same strategy to determine a path, $r^i(e) = r^1(e)$ for all $1 \leq i \leq n$. To obtain $r^1(e)$, we calculate the fraction of possible unicasts $(s_1, t_1) \in V^2$ for which $e \in P_1$. Fix $0 \leq p, q \leq t$ so that $h_p \leq s_{1y} < h_{p+1}$, and $h_q \leq t_{1y} < h_{q+1}$.

Case 1) $e = ((a, b), (a + 1, b))$, $e \notin E_1$. Since $e \notin E_1$, $h_i < b < h_{i+1}$ for some $0 \leq i \leq t$. Thus, $e \in P_1$ if and only if $0 \leq s_{1x} \leq a$, $s_{1y} = b$, $a + 1 \leq t_{1x} \leq m$, and $h_i \leq t_{1y} < h_{i+1}$. Therefore, $r^1(e) = \frac{(a+1)(m-a)(h_{i+1}-h_i)}{(m+1)^4}$.

Case 2) $e = ((a, b), (a+1, b))$ and $e \in E_1$. Since $e \in E_1$, $b = h_i$ for some $1 \leq i \leq t$. If $h_{i-1} \leq s_{1y} < h_i$, then $e \in P_1$ if and only if $0 \leq s_{1x} \leq a$, $a + 1 \leq t_{1x} \leq m$, and $h_i \leq t_{1y} \leq m$, as shown in Fig. 2.17(a). If $s_{1y} = h_i$, then $e \in P_1$ if and only if $0 \leq s_{1x} \leq a$, $a + 1 \leq t_{1x} \leq m$, and $0 \leq t_{1y} < h_{i+1}$. If $h_i < s_{1y} < h_{i+1}$, then $e \in P_1$ if

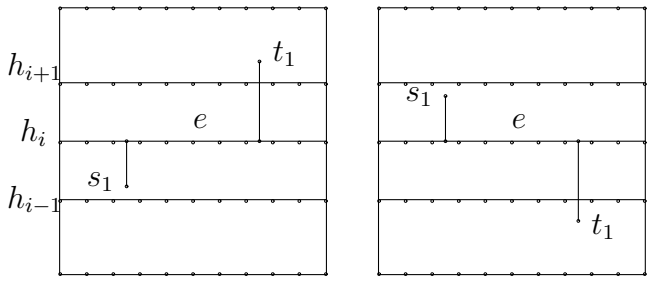


Figure 2.17: Case 2 in the calculation of edge use distribution $r^1(e)$. (a) $h_{i-1} \leq s_{1y} < h_i$ and (b) $h_i < s_{1y} < h_{i+1}$.

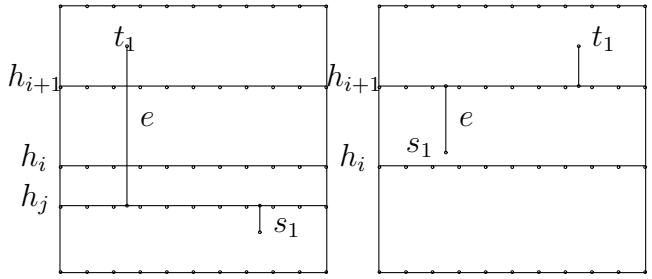
and only if $0 \leq s_{1x} \leq a$, $a+1 \leq t_{1x} \leq m$, and $0 \leq t_{1y} < h_i$, as shown in Fig. 2.17(b). If $s_{1y} < h_{i-1}$ or $s_{1y} \geq h_{i+1}$, then $e \notin P_1$. Thus,

$$r^1(e) = \left[\frac{(a+1)(m-a)(h_i - h_{i-1})(m+1-h_i)}{(m+1)^4} + \frac{(a+1)(m-a)(h_{i+1}-h_i)(h_i+1)}{(m+1)^4}\right].$$

Case 3) $e = ((a+1,b),(a,b))$.

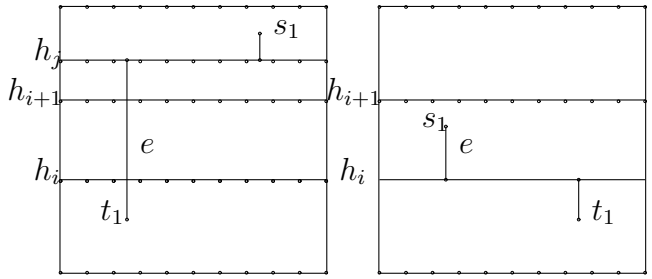
By the symmetry of our algorithm, $r^1((a+1,b),(a,b)) = r^1((m-a-1,b),(m-a,b))$. By cases 1 and 2 above, $r^1((m-a-1,b),(m-a,b)) = r^1((a,b),(a+1,b))$. Therefore, $r^1((a+1,b),(a,b)) = r^1((a,b),(a+1,b))$.

Case 4) $e = ((a,b),(a,b+1))$. Fix $0 \leq i \leq t$ so that $h_i \leq b < h_{i+1}$. In this case, $e \in P_1$ only if $0 \leq s_{1y} \leq b$ and $b+1 \leq t_{1y} \leq m$. If $0 \leq s_{1y} < h_i$, then $e \in P_1$ if and only if $0 \leq s_{1x} \leq m$, $t_{1x} = a$, and $b+1 \leq t_{1y} \leq m$, as shown in Fig. 2.18(a). If $h_i \leq s_{1y} \leq b$ and $h_{i+1} \leq t_{1y} \leq m$, then $e \in P_1$ if and only if $s_{1x} = a$, $0 \leq t_{1x} \leq m$, as shown in Fig. 2.18(b). If $h_i \leq s_{1y} \leq b$ and $b+1 \leq t_{1y} < h_{i+1}$, then $e \in P_1$ if and only



(a) (b)

Figure 2.18: Case 4 in the calculation of $r^1(e)$. (a) $0 \leq s_{1y} < h_i$ $(j \leq i)$ and (b) $h_i \leq s_{1y} \leq b$ and $h_{i+1} \leq t_{1y} \leq m$.



(a) (b)

Figure 2.19: Case 5 in the calculation of $r^1(e)$. (a) $h_{i+1} \leq s_{1y} \leq m$ $(j \geq i+1)$ and (b) $b+1 \leq s_{1y} < h_{i+1}$ and $0 \leq t_{1y} < h_i$.

if $0 \leq s_{1x} \leq m$, $t_{1x} = a$.

$$\begin{aligned} r^1(e) &= \frac{(b+1)(m-b)}{(m+1)^4}\left[\frac{h_i(m+1)}{(b+1)}\right. \\ &\quad + \frac{(b+1-h_i)}{(b+1)}\left(\frac{(m+1-h_{i+1})(m+1)}{(m-b)}\right. \\ &\quad \left.\left. + \frac{(m+1)(h_{i+1}-b-1)}{(m-b)}\right)\right] \\ &= \frac{(b+1)(m-b)}{(m+1)^3}. \end{aligned}$$

Case 5) $e = ((a, b+1), (a, b))$. Fix $0 \leq i \leq t$ so that $h_i \leq b < h_{i+1}$. In this case,

$e \in P_1$ only if $b+1 \leq s_{1y} \leq m$ and $0 \leq t_{1y} \leq b$. If $h_{i+1} \leq s_{1y} \leq m$, then $e \in P_1$ if and only if $0 \leq s_{1x} \leq m$, $t_{1x} = a$, and $0 \leq t_{1y} \leq b$, as shown in Fig. 2.19(a). If $b+1 \leq s_{1y} < h_{i+1}$ and $0 \leq t_{1y} < h_i$, then $e \in P_1$ if and only if $s_{1x} = a$, $0 \leq t_{1x} \leq m$, as shown in Fig. 2.19(b). If $b+1 \leq s_{1y} < h_{i+1}$ and $h_i \leq t_{1y} \leq b$, then $e \in P_1$ if and only if $0 \leq s_{1x} \leq m$, $t_{1x} = a$.

$$\begin{aligned} r^1(e) &= \frac{(m-b)(b+1)}{(m+1)^4}\left[\frac{(m+1-h_i)(m+1)}{(m-b)}\right. \\ &\quad + \frac{(h_{i+1}-b-1)}{(m-b)}\left(\frac{h_i(m+1)}{(b+1)}\right. \\ &\quad \left.\left. + \frac{(b+1-h_i)(m+1)}{(b+1)}\right)\right] \\ &= \frac{(b+1)(m-b)}{(m+1)^3}. \end{aligned}$$

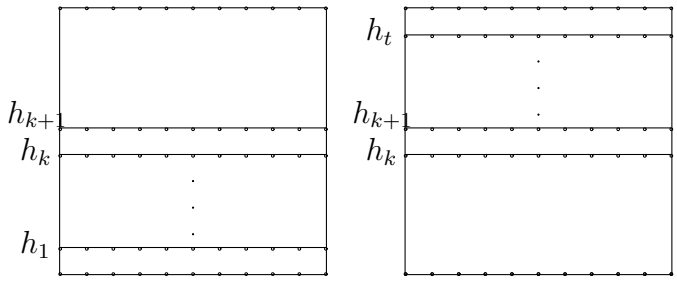
2.2.3.3 Expected Network Cost

We compute the expected network cost for the row model when n unicasts $U = \{(s_1,t_1),..,(s_n,t_n)\}$ are chosen uniformly at random on G_m. We use S to denote the candidate solution for U obtained by our strategy and $t(n,i,j,e)$ to denote the probability that i unicasts traverse e and j unicasts traverse e^R $(0 \leq i+j \leq n)$. No unicast can contain both e and e^R in its path using our path choice algorithm. Thus each unicast either uses edge e (with probability $r^1(e)$), uses edge e^R (with probability $r^1(e) = r^1(e^R)$), or uses neither (with probability $1 - r^1(e) - r^1(e^R) = 1 - 2r^1(e)$). Thus

$$t(n,i,j,e) = \binom{n}{i}\binom{n-i}{j}(r^1(e))^{i+j}(1-2r^1(e))^{n-i-j}.$$

We compute the expected network cost $EC(S)$ as follows.

For any (i,k) satisfying $0 \leq 2k+i \leq n$, $k+i$ unicasts use e and k unicasts use e^R with probability $t(n,k+i,k,e)$. Likewise, k unicasts use e and $k+i$ unicasts use e^R with probability $t(n,k,k+i,e) = t(n,k+i,k,e)$. In both cases, $C(S,e,e^R) = k+i$. Since we considered $i = 0$ in both cases, by the definition of the network cost, we



(a) (b)

Figure 2.20: Given h_k and h_{k+1}, optimizing $(h_1, ..., h_t)$ is equivalent to optimizing $(h_1, .., h_{k-1})$ in (a) and $(h_{k+2}, .., h_t)$ in (b).

obtain

$$\begin{aligned} EC(S) &= \sum_{e \in E} \frac{1}{2} EC(S, e, e^R) \\ &= \sum_{e \in E} \frac{1}{2} \left[\sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} \left(2 \sum_{i=0}^{n-2k} (k+i) t(n, k, k+i, e) \right) \right. \\ &\quad \left. - k \cdot t(n, k, k, e) \right]. \qquad (1) \end{aligned}$$

2.2.3.4 Results

In this section, we first present a low-complexity algorithm that optimizes the reverse carpooling line placement to minimize the expected cost given a number of unicasts. Then, we demonstrate the performance of our algorithm.

Our goal is to optimize the locations of t reverse carpooling lines ($0 \leq h_1 < h_2 < .. < h_t \leq m$) for n unicasts on G_m. We use E_h and E_v to denote the sets of all horizontal and vertical edges, respectively. Since the edge use distribution of any vertical edge is independent of $(h_1, ..., h_t)$, from (1), $EC(S) = f(h_1, ..., h_t) + M$ where $f(h_1, ..., h_t) = \sum_{e \in E_h} \frac{1}{2} EC(S, e, e^R)$ and $M = \sum_{e \in E_v} \frac{1}{2} EC(S, e, e^R)$. Finding $(h_1, ..., h_t)$ to minimize $f(h_1, ..., h_t)$ minimizes $EC(S)$. The following algorithm finds this optimal $(h_1, ..., h_t)$ by recursively dividing the problem into smaller and smaller independent subproblems.

Fix $q \geq 1$. Let $t = 2^{q+1} - 2$ and $r = 2^q - 1$. For convenience, $h_{-1} = h_0 = 0$ and $h_{t+1} = h_{t+2} = m + 1$.[1] Suppose that $k = \frac{t}{2}$ and the locations of the kth and $(k+1)$th reverse carpooling lines are given by $h_k = i_1$ and $h_{k+1} = i_1 + d_1$ $(d_1 \geq 1)$. Since $r^1(e)$ for each $e \in E$ depends on at most three closest reverse carpooling lines, $r^1(e)$ is a function of either $(h_1, \ldots, h_{k+1})$ or $(h_k, \ldots, h_t)$ for each $e \in E$. Thus, given h_k and h_{k+1}, the objective function f can be decomposed as

$$f(h_1, .., h_t) = f_1^{(1)}(h_{-1}, \ldots, h_{k+1}) + f_2^{(1)}(h_k, \ldots, h_{t+2}),$$

where functions $f_1^{(1)}$ and $f_2^{(1)}$ are independent when h_k and h_{k+1} are fixed. Here

$$f_1^{(1)}(0, 0, h_1, \ldots, h_{k+1}) = \sum_{e \in E_1^{(1)}} \frac{1}{2} EC(S, e, e^R) \quad \text{and}$$

$$f_2^{(1)}(h_k, \ldots, h_t, m+1, m+1) = \sum_{e \in E_2^{(1)}} \frac{1}{2} EC(S, e, e^R)$$

where $E_1^{(1)} = \{((a, b), (a+1, b)), ((a+1, b), (a, b)) : 0 \leq a < m, 0 \leq b < h_{k+1}\}$ and $E_2^{(1)} = \{((a, b), (a+1, b)), ((a+1, b), (a, b)) : 0 \leq a < m, h_{k+1} \leq b \leq m\}$.

The given formulation breaks our optimization problem into two subproblems. The first subproblem contains $k+1$ reverse carpooling lines $(0 \leq h_1 < h_2 < .. < h_{k+1} \leq m)$ with $h_k = i_1$ and $h_{k+1} = i_1 + d_1$, as shown in Fig. 2.20(a). The goal here is to choose $(h_1, \ldots, h_{k-1})$ to minimize $f_1^{(1)}(0, 0, h_1, \ldots, h_{k-1}, i_1, i_1 + d_1)$. The second subproblem contains $k+1$ reverse carpooling lines $(0 \leq h_k < h_{k+1} < \ldots < h_t \leq m)$ with $h_k = i_1$ and $h_{k+1} = i_1 + d_1$, as shown in Fig. 2.20(b). The goal here is to choose $(h_{k+2}, \ldots, h_t)$ to minimize $f_2^{(1)}(i_1, i_1 + d_1, h_{k+2}, \ldots, h_t, m+1, m+1)$. The added parameters $h_{-1} = h_0 = 0$ and $h_{t+1} = h_{t+2} = m + 1$ are included so that each subproblem is bounded above and below by two reverse carpooling lines. Searching over all possible values of i_1 and d_1 and then optimizing $f_1^{(1)}(0, 0, h_1, \ldots, h_{k-1}, i_1, i_1 + d_1)$ and $f_2^{(1)}(i_1, i_1 + d_1, h_{k+2}, \ldots, h_t, m+1, m+1)$ guarantees the optimal solution.

[1] We include h_{-1}, h_0, h_{t+1}, and h_{t+2} in this characterization for symmetry, as will become clear in the following discussion.

Function $l_q(a, b, c, d)$
if $i = 0$
return $l_0(a, b, c, d) = \sum_{e \in E_{b,d}} \frac{1}{2} EC(S, e, e^R)$ (2)
where $E_{b,d} = \{((j, k), (j + 1, k)),((j + 1, k), (j, k)):$
$0 \leq j < m, b \leq k < d\}$ and the expected cost
$EC(S, e, e^R)$ for each $e \in E_{b,d}$ is calculated
assuming reverse carpooling lines only at locations
a, b, c, and d.
else
return $\min_{(x,y)} [l_{i-1}(a, b, x, y) + l_{i-1}(x, y, c, d)]$
over all (x, y) s.t. $b + 2^i - 2 < x < y < c - 2^i + 2$.

Figure 2.21: Function $l_q(a, b, c, d)$ finds the optimal $2^{q+1} - 2$ reverse carpooling lines between two upper reverse carpooling lines at locations c and d and two lower reverse carpooling lines at locations a and b and returns its expected cost.

To optimize $f_1^{(1)}(0, 0, h_1, \ldots, h_{k-1}, i_1, i_1 + d_1)$ and $f_2^{(1)}(i_1, i_1 + d_1, h_{k+2}, \ldots, h_t, m + 1, m+1)$, we again apply the same approach – first fixing the two central line locations and then breaking each problem into two independent subproblems

$$
\begin{aligned}
& f_1^{(1)}(0, 0, h_1, \ldots, h_{k-1}, i_1, i_1 + d_1) \\
&= f_1^{(2)}(0, 0, h_1, \ldots, h_{l-1}, i_2, i_2 + d_2) \\
&\quad + f_2^{(2)}(i_2, i_2 + d_2, \ldots, i_1, i_1 + d_1), \\
& f_2^{(1)}(i_1, i_1 + d_1, h_{k+2}, \ldots, m + 1) \\
&= f_3^{(2)}(i_1, i_1 + d_1, h_{k+2}, \ldots, h_{k+l-2}, i_3, i_3 + d_3) \\
&\quad + f_4^{(2)}(i_3, i_3 + d_3, h_{k+l+1}, \ldots, m + 1).
\end{aligned}
$$

Function $l_q(a, b, c, d)$ shown in Fig. 2.21 captures the recursive approach. Running $l_q(a, b, c, d)$ finds the optimal $2^{q+1} - 2$ reverse carpooling lines between two upper reverse carpooling lines at locations c and d and two lower reverse carpooling lines at locations a and b and returns its expected cost.

Theorem 2.7 *When n unicasts are chosen uniformly at random on G_m, $l_q(0, 0, m + 1, m + 1)$ finds the optimal locations for $t = 2^{q+1} - 2$ row reverse carpooling lines in*

Table 2.1: Optimal reverse carpooling lines placement on the G_{10}.

n	t^*	$(h_1^*, .., h_t^*)$
$n < 55$	2	(3,7)
$n \geq 55$	3	(2,5,8)

Table 2.2: Optimal reverse carpooling lines placement on the G_{12}.

n	t^*	$(h_1^*, .., h_t^*)$
$n < 40$	2	(4,8)
$40 \leq n < 110$	3	(3,6,9)
$n > 110$	3	(2,5,9)

time $O(qm^6 + n^2m^6)$.

Proof. The optimality of our algorithm follows immediately from its search of all possible line placements. The run-time relies on the storage of all intermediate values $l_i(a, b, c, d)$ used in calculating $l_q(0, 0, m+1, m+1)$; since many of these values are used repeatedly, we avoid repeated computation by keeping a table of known values and calling the function only when the value is unknown. We calculate the run time as follows. For each $1 \leq i \leq q$ and each needed (a, b, c, d), we find $l_i(a, b, c, d)$ as $l_{i-1}(a, b, x, y)+l_{i-1}(x, y, c, d)$ for the optimal choice (x, y) of the two central carpooling line locations. Since there are q values of i, $O(m^4)$ values of (a, b, c, d), and $O(m^2)$ values for (x, y), these calculations run in time $O(qm^6)$. From (1) and (2), calculation of $l_0(a, b, c, d)$ for each (a, b, c, d) runs in time $O(m^2n^2)$, giving total run-time $O(qm^6 + n^2m^6)$. ■

Tables 2.1 and 2.2 show the optimal number of reverse carpooling lines (t^*) and their optimal locations $(h_1^*, .., h_{t^*}^*)$ for n unicasts chosen uniformly at random on G_{10} and G_{12}, respectively.

Fig. 2.22 plots the normalized cost $EC(S^*)/ED(U)$ as a function of the number of unicasts, n, where S^* is the solution given by our algorithm and $ED(U) = E\sum_{i=1}^{n} d(s_i, t_i)$ is the expected distance between sources and sinks for the unicasts $U = \{(s_1, t_1), .., (s_n, t_n)\}$. In both cases, the normalized cost decreases as the number of unicast sessions increases. Also included are the corresponding normalized costs when no reverse carpooling lines are included and pure opportunistic coding is em-

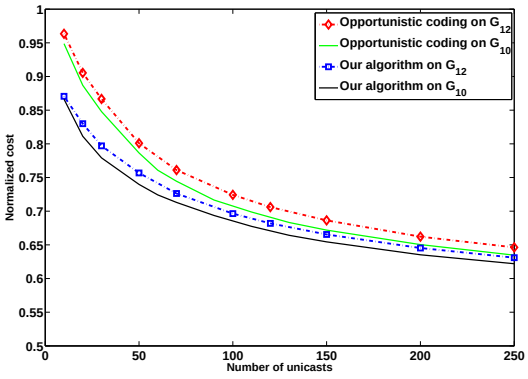


Figure 2.22: Normalized cost on G_{10} and G_{12}.

ployed. Our algorithm yields as much as 7% improvement over pure opportunistic coding when $n < 40$ in both cases. When n is large, traffic is sufficiently large that reverse carpooling opportunities arise even without the introduction of reverse carpooling lines. As a result, the percentage improvement over opportunistic coding decreases as n increases.

2.2.4 Row and Column Model

To increase the opportunities to apply reverse carpooling, we next add column reverse carpooling lines to the previous model. Given a wireless rectangular grid G_m, we use a t-tuple $(0 \leq h_1 < h_2 < .. < h_t \leq m)$ and a k-tuple $(0 \leq r_1 < r_2 < .. < r_k \leq m)$ to denote the locations of t row and k column reverse carpooling lines, respectively. (For convenience, $h_0 = r_0 = 0$ and $h_{t+1} = r_{k+1} = m + 1$.) The reverse carpooling edge set is $E_1 = \{((i, h_j), (i+1, h_j)), ((i+1, h_j), (i, h_j)), ((r_p, i), (r_p, i+1)), ((r_p, i+1), (r_p, i)) : 0 \leq i \leq m-1, 1 \leq j \leq t, 1 \leq p \leq k\}$.

2.2.4.1 Path Choice Algorithm

The proposed algorithm finds a shortest path $P_i \in \mathcal{P}^*(s_i, t_i)$ that maximizes the fraction of the path spent on the reverse carpooling lines. Choose $0 \leq c, d \leq k$ and $0 \leq f, g \leq t$ so that $r_c \leq s_{ix} < r_{c+1}$, $h_f \leq s_{iy} < h_{f+1}$, $r_d \leq t_{ix} < r_{d+1}$, and $h_g \leq t_{iy} < h_{g+1}$.

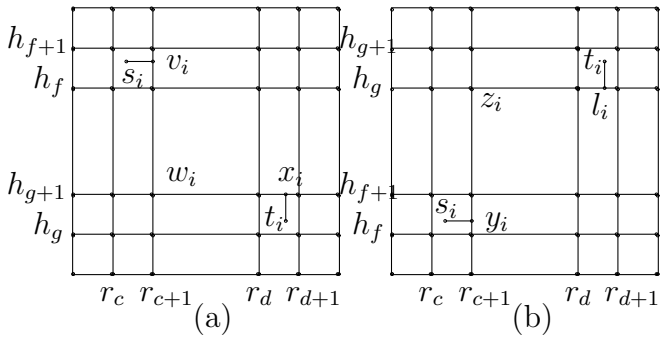


Figure 2.23: Case 2 in the path choice algorithm (a) when $f > g$ and (b) when $f < g$.

Case 1) $c = d$ or $f = g$. P_i is the unique path in $\mathcal{P}^*(s_i, u_i) \times \mathcal{P}^*(u_i, t_i)$ where $u_i = (t_{ix}, s_{iy})$.

Case 2) $c < d$ and $f \neq g$. If $f > g$, P_i is the unique path in $\mathcal{P}^*(s_i, v_i) \times \mathcal{P}^*(v_i, w_i) \times \mathcal{P}^*(w_i, x_i) \times \mathcal{P}^*(x_i, t_i)$ where $v_i = (r_{c+1}, s_{iy})$, $w_i = (r_{c+1}, h_{g+1})$, and $x_i = (t_{ix}, h_{g+1})$, as shown in Fig. 2.23(a).

If $f < g$, P_i is the unique path in $\mathcal{P}^*(s_i, y_i) \times \mathcal{P}^*(y_i, z_i) \times \mathcal{P}^*(z_i, l_i) \times \mathcal{P}^*(l_i, t_i)$ where $y_i = (r_{c+1}, s_{iy})$, $z_i = (r_{c+1}, h_g)$, and $l_i = (t_{ix}, h_g)$, as shown in Fig. 2.23(b).

Case 3) $c_1 > c_2$ and $d_1 \neq d_2$. We define unicast (s'_i, t'_i) for which $s'_i = t_i$ and $t'_i = s_i$. Then, by case 2), we can obtain a shortest path P'_i for (s'_i, t'_i). In this case, we set $P_i = (P'_i)^R$.

2.2.4.2 Edge Use Distribution

As in Sec. III-B, we determine $r^1(e)$ for $e \in E$.

Case 1) $e = ((a, b), (a + 1, b))$, $e \in E_1$. Since $e \in E_1$, $r_p \leq a < r_{p+1}$ and $b = h_q$ for some $0 \leq p \leq k$ and $1 \leq q \leq t$, respectively. If $0 \leq s_{1x} < r_p$ and $s_{1y} \leq t_{1y}$, $e \in P_1$ if and only if $0 \leq s_{1y} \leq b$, $a+1 \leq t_{1x} \leq m$, and $b \leq t_{1y} < h_{q+1}$, as shown in Fig. 2.24(a). If $0 \leq s_{1x} < r_p$ and $s_{1y} > t_{1y}$, $e \in P_1$ if and only if $b \leq s_{1y} \leq m$, $a + 1 \leq t_{1x} \leq m$, and $h_{q-1} \leq t_{1y} < b$, as shown in Fig. 2.24(b). If $r_p \leq s_{1x} \leq a$, $e \in P_1$ if and only if $s_{1y} = b$, $a + 1 \leq t_{1x} \leq m$, and $0 \leq t_{1y} \leq m$, as shown in Fig. 2.25(a). If $s_{1x} > a$, then

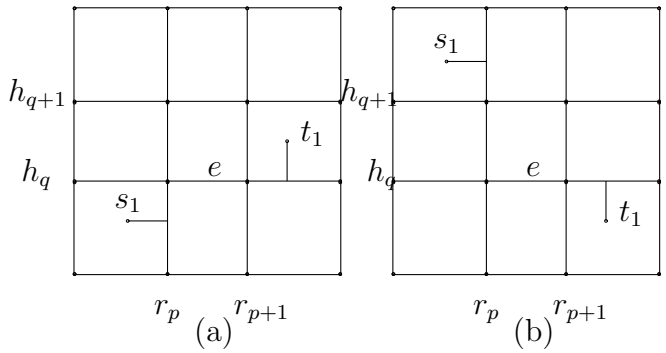


Figure 2.24: Case 1 in the calculation of edge use distribution $r^1(e)$ when $0 \leq s_{1x} < r_p$. (a) $s_{1y} \leq t_{1y}$ and (b) $s_{1y} > t_{1y}$.

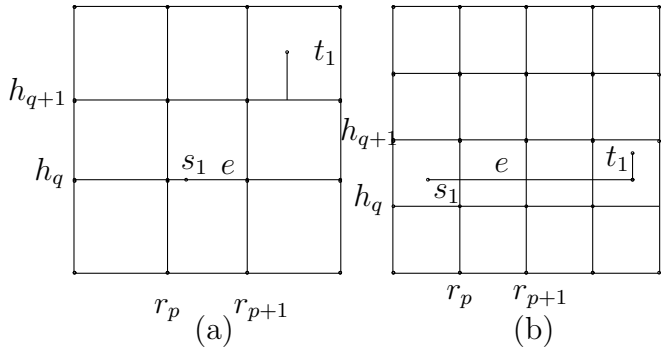


Figure 2.25: (a) Case 1 in the calculation of edge use distribution $r^1(e)$ when $r_p \leq s_{1x} \leq a$. (b) Case 2 in the calculation of edge use distribution $r^1(e)$ when $0 \leq s_{1x} < r_p$.

$e \notin P_1$. Thus,

$$\begin{aligned} r^1(e) &= \left[\frac{r_p(b+1)(m-a)(h_{q+1}-b)}{(m+1)^4}\right. \\ &\quad + \frac{r_p(m+1-b)(m-a)(b-h_{q-1})}{(m+1)^4} \\ &\quad \left. + \frac{(a+1-r_p)(m-a)(m+1)}{(m+1)^4}\right]. \end{aligned}$$

Case 2) $e = ((a,b),(a+1,b))$ and $e \notin E_1$. Since $e \notin E_1$, $r_p \leq a < r_{p+1}$ and $h_q < b < h_{q+1}$ for some $0 \leq p \leq k$ and $0 \leq q \leq t$, respectively. If $0 \leq s_{1x} < r_p$, then $e \in P_1$ if and only if $s_{1y} = b$, $a+1 \leq t_{1x} \leq m$, and $h_q \leq t_{1y} < h_{q+1}$, as shown in

Fig. 2.25(b). If $r_p \leq s_{1x} \leq a$, then $e \in P_1$ if and only if $s_{1y} = b$, $a+1 \leq t_{1x} \leq m$, and $0 \leq t_{1y} \leq m$. If $s_{1x} > a$, then $e \notin P_1$. Thus,

$$r^1(e) = \left[\frac{r_p(m-a)(h_{q+1}-h_q)}{(m+1)^4} + \frac{(a+1-r_p)(m-a)(m+1)}{(m+1)^4}\right].$$

Case 3) $e = ((a+1,b),(a,b))$. We use $X(e)$ to denote the set of (s_1,t_1) such that $e \in P_1$. We show that there is an one to one correspondence between $X(e)$ and $X(e^R)$. Choose $0 \leq c,d \leq k$ and $0 \leq f,g \leq t$ so that $r_c \leq s_{1x} < r_{c+1}$, $r_d \leq t_{1x} < r_{d+1}$, $h_f \leq s_{1y} < h_{f+1}$, and $h_g \leq t_{1y} < h_{g+1}$. When $c = d$ or $f = g$, $e \in P_1$ if and only if $e^R \in P_1'$ for $(s_1',t_1') = ((t_{1x},s_{1y}),(s_{1x},t_{1y}))$. When $c \neq d$ and $f \neq g$, by the symmetry of our path choice algorithm, $e \in P_1$ if and only if $e^R \in P_1'$ for $(s_1',t_1') = (t_1,s_1)$. Thus, there exists an one to one correspondence between $X(e)$ and $X(e^R)$ and thus $|X(e)| = |X(e^R)|$. Then we can calculate $r^1(e) = r^1(e^R)$ from cases 1 and 2.

Case 4) $e = ((a,b),(a,b+1))$, $e \in E_1$. Since $e \in E_1$, $a = r_p$ and $h_q \leq b < h_{q+1}$ for some $1 \leq p \leq k$ and $0 \leq q \leq t$, respectively. Choose $0 \leq c,d \leq k$ and $0 \leq f,g \leq t$ as we did in case 3. If $c = d$, then $e \in P_1$ if and only if $r_p \leq s_{1x} < r_{p+1}$, $0 \leq s_{1y} \leq b$, $t_{1x} = a$, and $b+1 \leq t_{1y} \leq m$. If $f = g$ and $c \neq d$, then $e \in P_1$ if and only if $0 \leq s_{1x} < r_p$ or $r_{p+1} \leq s_{1x} \leq m$, $h_q \leq s_{1y} \leq b$, $t_{1x} = a$, and $b+1 \leq t_{1y} < h_{q+1}$. If $f < g$ and $c < d$ and when $h_{q+1} \leq t_{1y} \leq m$, then $e \in P_1$ if and only if $r_{p-1} \leq s_{1x} < a$, $0 \leq s_{1y} \leq b$, and $a \leq t_{1x} \leq m$, as shown in Fig. 2.26(a). If $f < g$ and $c < d$ and when $b+1 \leq t_{1y} < h_{q+1}$, then $e \in P_1$ if and only if $0 \leq s_{1x} < a$, $0 \leq s_{1y} < h_q$, and $t_{1x} = a$, as shown in Fig. 2.26(b). If $f < g$ and $c > d$ and when $0 \leq s_{1y} < h_q$, then $e \in P_1$ if and only if $a \leq s_{1x} \leq m$, $r_{p-1} \leq t_{1x} < a$, and $b+1 \leq t_{1y} \leq m$, as shown in Fig. 2.27(a). If $f < g$ and $c > d$ and when $h_q \leq s_{1y} \leq b$, then $e \in P_1$ if and only if $s_{1x} = a$, $0 \leq t_{1x} < a$, and $h_{q+1} \leq t_{1y} \leq m$, as shown in Fig. 2.27(b). If $f > g$, then

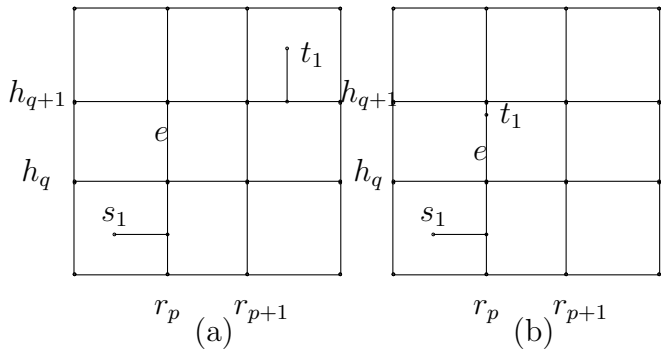


Figure 2.26: Case 4 in the calculation of edge use distribution $r^1(e)$ when $c < d$ and $f < g$. (a) $h_{q+1} \leq t_{1y} \leq m$ and (b) $b + 1 \leq t_{1y} < h_{q+1}$.

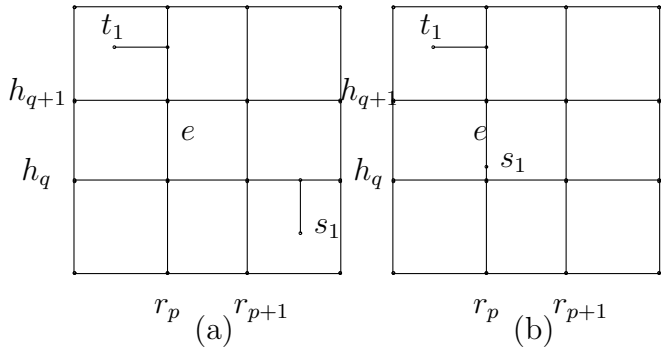


Figure 2.27: Case 4 in the calculation of edge use distribution $r^1(e)$ when $c > d$ and $f < g$. (a) $0 \leq s_{1y} < h_q$ and (b) $h_q \leq s_{1y} \leq b$.

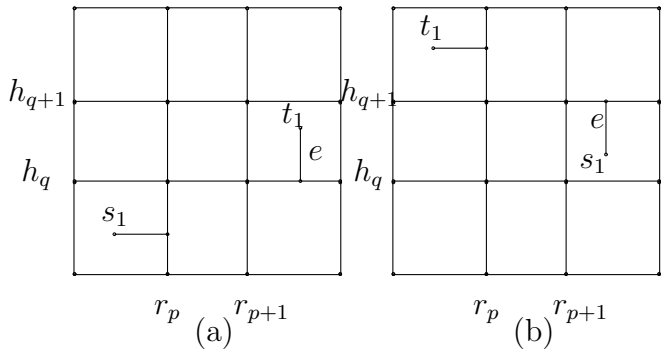


Figure 2.28: Case 5 in the calculation of edge use distribution $r^1(e)$ when $c \neq d$ and $f \neq g$ (a) $c < d$ and (b) $c > d$.

$e \notin P_1$. Thus,

$$\begin{aligned} r^1(e) \;=\; & \left[\frac{(r_{p+1}-r_p)(b+1)(m-b)}{(m+1)^4}\right. \\ & +\frac{(m+1-r_{p+1}+r_p)(b+1-h_q)(h_{q+1}-b-1)}{(m+1)^4} \\ & +\frac{(m+1-h_{q+1})(a-r_{p-1})(b+1)(m+1-a)}{(m+1)^4} \\ & +\frac{(h_{q+1}-b-1)ah_q}{(m+1)^4} \\ & +\frac{h_q(m+1-a)(a-r_{p-1})(m-b)}{(m+1)^4} \\ & \left.+\frac{(b+1-h_q)a(m+1-h_{q+1})}{(m+1)^4}\right]. \end{aligned}$$

Case 5) $e = ((a,b),(a,b+1))$ and $e \notin E_1$. Since $e \notin E_1$, $r_p < a < r_{p+1}$ and $h_q \leq b < h_{q+1}$ for some $0 \leq p \leq k$ and $0 \leq q \leq t$, respectively. If $c = d$, then $e \in P_1$ if and only if $r_p \leq s_{1x} < r_{p+1}$, $0 \leq s_{1y} \leq b$, $t_{1x} = a$, and $b+1 \leq t_{1y} \leq m$. If $f = g$ and $c \neq d$, then $e \in P_1$ if and only if $0 \leq s_{1x} < r_p$ or $r_{p+1} \leq s_{1x} \leq m$, $h_q \leq s_{1y} \leq b$, $t_{1x} = a$, $b+1 \leq t_{1x} < h_{q+1}$. If $c < d$ and $f \neq g$, then $e \in P_1$ if and only if $0 \leq s_{1x} < r_p$, $0 \leq s_{1y} < h_q$, $t_{1x} = a$, and $b+1 \leq t_{1y} < h_{q+1}$, as shown in Fig. 2.28(a). If $c > d$ and $f \neq g$, then $e \in P_1$ if and only if $s_{1x} = a$, $h_q \leq s_{1y} \leq b$, $0 \leq t_{1x} < r_p$, and

Table 2.3: Optimal row and column reverse carpooling lines placement on the G_8.

n	t^*	$(h_1^*, .., h_t^*)$	k^*	$(r_1^*, .., r_k^*)$
$n < 20$	3	(2,4,6)	2	(3,5)
$20 \leq n$	3	(2,4,6)	2	(2,5)

$h_{q+1} \leq t_{1y} \leq m$, as shown in Fig. 2.28(b). Thus,

$$\begin{aligned} r^1(e) &= \left[\frac{(r_{p+1} - r_p)(b+1)(m-b)}{(m+1)^4} \right. \\ &+ \frac{(m+1-r_{p+1}+r_p)(b+1-h_q)(h_{q+1}-b-1)}{(m+1)^4} \\ &+ \frac{r_p h_q (h_{q+1}-b-1)}{(m+1)^4} \\ &+ \left. \frac{(b+1-h_q) r_p (m+1-h_{q+1})}{(m+1)^4} \right]. \end{aligned}$$

Case 6) $e = ((a, b+1), (a, b))$. As in case 3, we show that there exists one to one correspondence between $X(e)$ and $X(e^R)$. When $c = d$ or $f = g$, $e \in P_1$ if and only if $e^R \in P_1'$ for $(s_1', t_1') = ((s_{1x}, t_{1y}), (t_{1x}, s_{1y}))$. When $c \neq d$ and $f \neq g$, $e \in P_1$ if and only if $e^R \in P_1'$ for $(s_1', t_1') = (t_1, s_1)$. Then $X(e) = X(e^R)$ and we can calculate $r^1(e) = r^1(e^R)$ from cases 4 and 5.

2.2.4.3 Results

Let n unicasts $U = \{(s_1, t_1), .., (s_n, t_n)\}$ be chosen uniformly at random on the wireless rectangular grid G_m. Table 2.3 shows the optimal number of row and column reverse carpooling lines (t^*) and (k^*), and their optimal locations $(h_1^*, .., h_{t^*}^*)$ and $(r_1^*, .., r_{k^*}^*)$ for $m = 8$. To obtain the optimal reverse carpooling line placement in this case, we search over all possible choices of $(h_1, .., h_t)$ and $(r_1, .., r_k)$ for $1 \leq t, k \leq m$ and choose the one that minimizes the expected network cost. Since we cannot decompose the optimization problem into independent subproblems in this case, we cannot apply the algorithm from previous section.

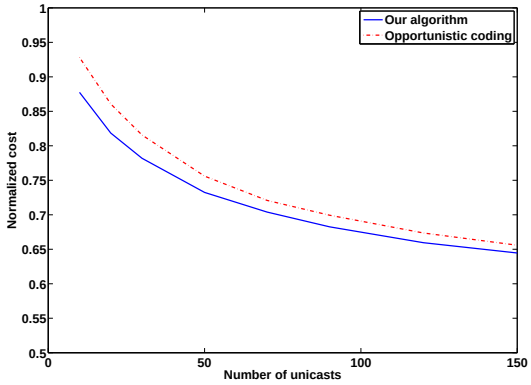


Figure 2.29: Normalized cost on G_8.

Fig. 2.29 plots the normalized cost as a function of the number of unicasts. The normalized cost decreases as the number of unicast sessions increases. Corresponding normalized cost when pure opportunistic coding is employed is also included. Our algorithm yields as much as 7% improvement over pure opportunistic coding when $n < 30$. Similar to the result presented in Sec 2.2.3, the percentage improvement over opportunistic coding decreases as n increases.

2.2.5 Conclusion

In this section, we have presented a distributed strategy for reducing the expected power consumption for multiple unicasts in a network coded wireless network. We use the rectangular grid as a simple network model and apply a simple network coding strategy called "reverse carpooling," which uses only XOR and forwarding operations. Our strategy is to attempt to increase the number of coding opportunities by designating "reverse carpooling routes." Each individual unicast chooses a route from its source to its destination independently but in a manner that maximizes the fraction of the paths spent on the reverse carpooling lines without increasing individual path lengths. Intermediate nodes apply reverse carpooling opportunistically along these routes. This approach increases the reverse carpooling opportunities of an opportunistic network code without requiring central coordination.

We have proposed distributed route choice algorithms for row model and row and

column model respectively, and analyzed the edge use distribution of our algorithms. Then we can optimize the reverse carpooling line placement to minimize the resulting expected cost. When all reverse carpooling lines are rows, we present a recursive algorithm that optimizes the line choice in time $O(qm^6 + n^2m^6)$ where m is the grid size, n is the number of unicasts, and $2^{q+1} - 2$ is the number of reverse carpooling lines. This algorithm yields as much as 7% improvement over pure opportunistic coding when $n < 40$. When reverse carpooling lines include both rows and columns, we optimize the line choice by brute force search and our strategy also yields 7% improvement over pure opportunistic coding when $n < 30$.

2.3 Network coding with periodic recomputations for minimum energy multicasting in MANETs

2.3.1 Introduction

In this section, we consider the problem of establishing minimum-energy multicast connections using network coding in mobile ad hoc networks (MANETs). In a static ad hoc network, this problem can be formulated as a linear program for linear and separable cost functions [12,17], unlike the case without coding which is NP-hard [46–51]. However, in a mobile scenario, where the locations of nodes in the network change over time, it may still be computationally unattractive to solve the linear optimization at every time slot.

We present a low-complexity approach, network coding with periodic recomputation, which recomputes an approximate solution at fixed time intervals, and uses this solution during each time interval. This approach comes from an intuition that when network topology changes slowly, small perturbations occur in the original optimization problem and the original solution remains relatively close to the new optimal solution. In the strategy, time is divided into equal intervals and a suboptimal solution is computed at the first time slot of each interval. As the network topology changes slowly, we use the resulting coding subgraph as a suboptimal solution during

each interval. We first derive a bound on the maximum percentage deviation from the optimal cost in terms of the percentage deviation in the cost vector coefficients.

For complexity analysis, we assume that barrier and interior-point method is used to solve a linear program at the first time slot of each interval. When we recompute a solution, we can use the suboptimal solution in the preceding interval as a good initial solution of the linear program at each fixed interval. By combining our cost bound with this warm start strategy, we obtain a bound on the complexity.

Finally, we derive a combined bound that minimizes the time complexity bound subject to the condition that suboptimal solution in the interval achieves a given optimality gap during the interval. By solving this optimization problem approximately, we have the qualitative insights and this matches the result obtained from example network scenario.

This section is organized as follows. Section 2.3.2 introduces the system model and formulates the problem. We describe our periodic recomputation approach in Section 2.3.3. In Section 2.3.4, we derive the theoretical cost bound and the time-complexity from interior-point method with a warm-start strategy. We also optimize the normalized complexity subject to the condition that suboptimal solution in the interval achieves a given optimality gap during the interval. We obtain the general qualitative insights by solving this optimization problem and confirm this with an example network scenario.

2.3.2 Problem formulation

Here we formulate the minimum-energy multicast problem with network coding in MANETs using linear programming. We adopt the framework from [17] which models a wireless network with a directed hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of hyperarcs. Each hyperarc (i, J) represents a loseless broadcast link from node i to nodes in nonempty set $J \subset \mathcal{N}$. z_{iJ} denotes the rate at which coded packets are injected into hyperarc (i, J). The rate vector z consisting of entries z_{iJ} defines a coding subgraph for the multicast connection. A linear and

separable cost function F maps valid rate vectors to real numbers. $d_{i,j}$ denotes the Euclidean distance between nodes i and j in the network and we assume for simplicity that $c_{iJ} = \max_{j \in J} d_{i,j}$. Then, network cost $F(z) = \sum_{(i,J) \in \mathcal{A}} c_{iJ} z_{iJ}$. The source node s transmits packets at a positive, real rate R to a nonempty set of sink nodes T.

We extend the above formulation to the problem in MANETs by introducing a discrete time dimension. As nodes move, the network topology and link costs change over time. We periodically update the hyperarc set, assuming that any two nodes i and j are connected if $d_{i,j} < D$, where D is a given threshold distance. We use $c_{iJ}^{(k)}$ and $\mathcal{A}^{(k)}$ to denote the cost of hyperarc (i, J) and the hyperarc set at the kth time slot, respectively. Then the optimization problem at the kth time slot in MANETs can be formulated as follows:

$$
\begin{aligned}
\min \quad & \sum_{\{(i,J) \in \mathcal{A}^{(k)}\}} c_{iJ}^{(k)} z_{iJ}^{(k)} \\
& z_{iJ}^{(k)} \geq \sum_{\{j \in J\}} x_{iJj}^{(t,k)}, \forall (i,J) \in \mathcal{A}^{(k)}, t \in T \\
& \sum_{\{J|(i,J) \in \mathcal{A}^{(k)}\}} \sum_{\{j \in J\}} x_{iJj}^{(t,k)} - \sum_{\{j|(j,I) \in \mathcal{A}^{(k)}, i \in I\}} x_{jIi}^{(t,k)} = \sigma_i^{(t)}, \\
& \forall i \in \mathcal{N}, t \in T. \\
& x_{iJj}^{(t,k)} \geq 0, \forall (i,J) \in \mathcal{A}^{(k)}, j \in J, t \in T
\end{aligned}
\tag{2.2}
$$

where

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

We use $L^{(k)}$ to denote this linear program (2.2) at the kth time slot. By solving $L^{(k)}$, we obtain the global optimal solution $(X^{(k)})^* = \{x_{iJj}^{(t,k)} | (i,J) \in \mathcal{A}, j \in J, t \in T\}$. Let $C^{(k)} = \{c_{iJ}^{(k)} | (i,J) \in \mathcal{A}\}$ and $(Z^{(k)})^* = \{z_{iJ}^{(k)} | (i,J) \in \mathcal{A}\}$ be the network cost vector at time k and the optimal rate vector corresponding to the optimal solution $(X^{(k)})^*$, respectively. An optimal solution for the problem in MANETs can be obtained by

solving (2.2) for every time slot, but it leads to high computational complexity.

2.3.3 Algorithm

Instead of solving (2.2) every time slot, we consider a suboptimal period recomputation strategy with lower computational complexity.

Algorithm 1 Algorithm for network coding with periodic recomputation

if $k \equiv 0 \bmod(pw)$
  Reconstruct a hyperarc set $\mathcal{A}^{(k)}$. Given $c_{iJ}^{(k)}$ and $\mathcal{A}^{(k)}$, solve $L^{(k)}$ with an optimality gap ϵ_k using interior-point method and obtain a suboptimal solution $(Z^{(k)}, X^{(k)})$.
else if $k \equiv 0 \bmod(p)$
  Given $c_{iJ}^{(k)}$ and $\mathcal{A}^{(k)}$, solve $L^{(k)}$ with an optimality gap ϵ_k using interior-point method. We use $(Z^{(k-p)}, X^{(k-p)})$ as a feasible warm start point of interior-point method and obtain a suboptimal solution $(Z^{(k)}, X^{(k)})$.
else
  $(Z^{(k)}, X^{(k)}) = (Z^{(k_1 \cdot p)}, X^{(k_1 \cdot p)})$ where $k_1 = \lfloor \frac{k}{p} \rfloor$.
end if

In the algorithm, time is divided into intervals where each interval contains p time slots. We recompute an approximate solution in the first time slot of each interval using interior point method and use the resulting coding subgraph as a suboptimal solution during each interval. We assume that the hyperarc set is reconstructed every w intervals, i.e., every pw time slots, and remains the same over w intervals. Reconstructing the hyperarc set periodically can cause loss of optimality, but it allows us to use the solution computed at the first time slot of the interval as a feasible suboptimal solution over the interval. When $k \equiv 0 \mod (pw)$, we recompute the suboptimal solution $(Z^{(k)}, X^{(k)})$ by solving $L^{(k)}$ using interior-point method until an optimality gap ϵ_k is achieved, as shown in (2.3).

$$\sum_{\{(i,J)\in\mathcal{A}\}} c_{iJ}^k z_{iJ}^k \leq \sum_{\{(i,J)\in\mathcal{A}\}} c_{iJ}^k (z_{iJ}^k)^* + \epsilon_k. \tag{2.3}$$

In this case, we use any feasible solution as a starting point of interior-point method.

When $k \equiv 0 \mod (p)$ and $k \not\equiv 0 \mod (pw)$, we solve $L^{(k)}$ with an optimality gap ϵ_k using interior-point method. Here we can use a suboptimal solution $(Z^{(k-p)}, X^{(k-p)})$ computed at the first time slot of the previous interval as a feasible warm start point of interior-point method since the hyperarc set remains the same. When $k \not\equiv 0 \mod (p)$, we use the suboptimal solution computed at the first time slot of the interval which contains the kth time slot as a feasible suboptimal solution. Since by assumption only coefficients of the network cost vector change during each interval, the set of feasible solutions remain the same in the interval. Therefore, we can use the coding subgraph obtained in the first time slot of the interval as a feasible suboptimal solution during each interval.

2.3.4 Analysis

2.3.4.1 Theoretical Cost Bound

We derive a theoretical bound on the performance gap between our suboptimal solution and the optimal solution. In our algorithm, a suboptimal solution is computed in the first slot of each interval, whose cost deviates from the optimal by at most a given optimality gap. We find a bound on the resulting loss when we use the same solution over the entire interval despite changes in the objective function coefficients.

First, we introduce a useful Lemma by Oguz [52]. This lemma upper bounds the maximum percentage deviation in the objective function from optimality in terms of the percentage deviation in the objective function coefficients when we stick to the same solution. We consider the following two instances of a general form optimization problem:

$$\min z_1 = \{C_1 X | X \in S\},$$

$$\min z_2 = \{C_2 X | X \in S\}.$$

where $C_k = (c_1^k, .., c_n^k) \in R_+^n$ is an objective coefficient vector for $k = 1, 2$ and S is an arbitrary closed and bounded, nonempty set in R_+^n. Let X_1^* and X_2^* be the optimal

solutions of the above two problems with $z_1 = C_1 X_1^*$ and $z_2 = C_2 X_2^*$, respectively. We assume that $c_i^1 = 0$ implies $c_i^2 = 0$.

Lemma 2.8 *If*

$$\frac{|c_i^1 - c_i^2|}{c_i^1} \leq \epsilon$$

for all i such that $c_i^1 \neq 0$, then

$$\frac{z_2 - z_3}{z_2} \leq \frac{2\epsilon}{1-\epsilon}.$$

where $z_3 = C^2 X_1^*$.

Proof. Please see [52]. ■

We extend the above Lemma 2.8 to the following result which states that small perturbations in the network cost vector during the interval leave the suboptimal solution computed at the start of the interval, relatively close to the optimal solutions in the interval.

Lemma 2.9 *If $C^{(mp)} Z^{(mp)} \leq C^{(mp)} (Z^{(mp)})^* + \epsilon_{mp}$ and*

$$\max_{(i,J)\in\mathcal{A}} \max_{0\leq j\leq p} \frac{|c_{iJ}^{(mp)} - c_{iJ}^{(mp+j)}|}{c_{iJ}^{(mp)}} = \delta_{mp}, \quad \textit{then}$$

$$C^{(mp+l)} Z^{(mp)} \leq \frac{1+\delta_{mp}}{1-\delta_{mp}} C^{(mp+l)} (Z^{(mp+l)})^* + (1+\delta_{mp})\epsilon_{mp},$$

for $\forall\ 0 \leq l \leq p$.

Proof. Since $\frac{|c_{iJ}^{(mp)} - c_{iJ}^{(mp+j)}|}{c_{iJ}^{(mp)}} \leq \delta_{mp}$ for $\forall (i,J) \in \mathcal{A}$ and $\forall\ 0 \leq j \leq p$ for which $c_{iJ}^{(mp)} \neq 0$,

$$(1-\delta_{mp}) C^{(mp)} \leq C^{(mp+l)} \leq (1+\delta_{mp}) C^{(mp)}, \quad \forall 0 \leq l \leq p.$$

By postmultiplying $(Z^{(mp+l)})^*$ and $Z^{(mp)}$ to the left and right inequalities respectively,

$$(1-\delta_{mp}) C^{(mp)} (Z^{(mp+l)})^* \leq C^{(mp+l)} (Z^{(mp+l)})^*, \tag{2.4}$$

$$C^{(mp+l)}Z^{(mp)} \leq (1+\delta_{mp})C^{(mp)}Z^{(mp)}.$$

Since $C^{(mp)}Z^{(mp)} \leq C^{(mp)}(Z^{(mp)})^* + \epsilon_{mp}$,

$$\begin{aligned} & C^{(mp+l)}Z^{(mp)} \\ \leq\ & (1+\delta_{mp})C^{(mp)}Z^{(mp)} \\ \leq\ & (1+\delta_{mp})(C^{(mp)}(Z^{(mp)})^* + \epsilon_{mp}) \\ \leq\ & (1+\delta_{mp})(C^{(mp)}(Z^{(mp+l)})^* + \epsilon_{mp}). \end{aligned}$$

The last inequality is obtained from the optimality of $(Z^{(mp)})^*$. Then, by (2.4),

$$C^{(mp+l)}Z^{(mp)} \leq \frac{1+\delta_{mp}}{1-\delta_{mp}}C^{(mp+l)}(Z^{(mp+l)})^* + (1+\delta_{mp})\epsilon_{mp},$$

for $\forall\ 0 \leq l \leq p$. ■

This bounds the optimality gap over the interval in terms of the maximum percentage deviation in the cost vector coefficients during the interval and the optimality gap of our suboptimal solution at the first time slot of the interval.

2.3.4.2 Complexity

When the suboptimal solution is recomputed at the first time slot of each interval, the linear program (2.2) can be solved using barrier and interior-point method as shown in [8,9]. Convergence analysis of barrier and interior-point method for the linear optimization problem is given in e.g. [8, Sec. 11.5], where computational complexity is defined in terms of the total number of Newton iterations. In the first time slot of each interval, the linear optimization problem is a slight perturbation of that of the previous time interval as network topology changes slowly. When the hyperarc set is not changed, we can use the suboptimal solution in the preceding interval as the feasible warm-start point for interior point method at the first time slot of the following interval. Combining our cost bound with a warm-start strategy using

interior point method gives a worst-case bound on the number of Newton iterations required to achieve a given optimality gap. $(Z^{(k)}(0), X^{(k)}(0))$ and $q^{(k)}$ are used to denote the feasible starting subgraph for $L^{(k)}$ and the optimal cost of $L^{(k)}$, respectively. From [8-9], an upper bound on the total number of Newton steps for $L^{(k)}$, $N^{(k)}$, is given as follows:

$$\begin{aligned} N^{(k)} &= G\left\lceil \sqrt{M}\log_2(\frac{(C^{(k)}Z^{(k)}(0) - C^{(k)}(Z^{(k)})^*)}{\epsilon_k})\right\rceil \\ &\leq G\left(\sqrt{M}\log_2(\frac{C^{(k)}Z^{(k)}(0) - q^{(k)}}{\epsilon_k}) + 1\right) \end{aligned} \tag{2.5}$$

where M is the number of inequalities in $L^{(k)}$, ϵ_k is the required optimality gap, and $G = 11.5$.

In our algorithm, the suboptimal solution $(Z^{(mp)}, X^{(mp)})$ is recomputed at time mp for $\forall m \geq 0$. Since the hyperarc set is reconstructed every w intervals, the feasible solution set is accordingly changed every w intervals. Thus, if $m \equiv 0 (\text{mod } w)$, we first find any feasible solution of $L^{(mp)}$ and start interior-point method from that point. Otherwise, $(Z^{(m-1)p}, X^{(m-1)p})$ is used as the suboptimal solution during the interval $((m-1)p, mp-1)$, and it is also used as a feasible starting point of interior-point method at time mp, i.e. $(Z^{(mp)}(0), X^{(mp)}(0)) = (Z^{(m-1)p}, X^{(m-1)p})$. Here we define the normalized complexity during the interval $(mp, (m+1)p-1)$ as the total number of Newton iterations at the first time slot of the interval divided by the interval length, i.e. $\frac{N^{(mp)}}{p}$. By combining Lemma 2.9 with (2.5), we can obtain an upper bound on the normalized complexity as follows.

Theorem 2.10 *For $m \not\equiv 0 \mod w$, the normalized complexity over interval $(mp, (m+1)p-1)$ is at most*

$$\frac{G}{p}\left(\sqrt{M}\log_2\left(f(\delta_{(m-1)p}, \epsilon_{(m-1)p}, \epsilon_{mp}, q^{(mp)})\right) + 1\right),$$

where

$$f(\delta_{(m-1)p}, \epsilon_{(m-1)p}, \epsilon_{mp}, q^{(mp)}) = \frac{\frac{2\delta_{(m-1)p}}{1-\delta_{(m-1)p}} q^{(mp)} + (1+\delta_{(m-1)p})\epsilon_{(m-1)p}}{\epsilon_{mp}}.$$

Proof. Since $m \not\equiv 0(\mod w)$, $(Z^{(mp)}(0), X^{(mp)}(0)) = (Z^{(m-1)p}, X^{(m-1)p})$. Then, from (2.5),

$$\begin{aligned} &\frac{N^{(mp)}}{p} \\ &\leq \frac{G}{p}\left(\sqrt{M}\log_2(\frac{C^{(mp)}Z^{(mp)}(0) - q^{(mp)}}{\epsilon_{mp}}) + 1\right) \\ &= \frac{G}{p}\left(\sqrt{M}\log_2(\frac{C^{(mp)}Z^{(m-1)p} - q^{(mp)}}{\epsilon_{mp}}) + 1\right). \end{aligned} \tag{2.6}$$

From Lemma 2.9,

$$\begin{aligned} &C^{(mp)}Z^{(m-1)p} - q^{(mp)} \\ &\leq \frac{1+\delta_{(m-1)p}}{1-\delta_{(m-1)p}} q^{(mp)} + (1+\delta_{(m-1)p})\epsilon_{(m-1)p} - q^{(mp)} \\ &= \frac{2\delta_{(m-1)p}}{1-\delta_{(m-1)p}} q^{(mp)} + (1+\delta_{(m-1)p})\epsilon_{(m-1)p} \\ &= \epsilon_{mp} f(\delta_{(m-1)p}, \epsilon_{(m-1)p}, \epsilon_{mp}, q^{(mp)}). \end{aligned} \tag{2.7}$$

Then,

$$\begin{aligned} &\frac{N^{(mp)}}{p} \\ &\leq \frac{G}{p}\left(\sqrt{M}\log_2\left(f(\delta_{(m-1)p}, \epsilon_{(m-1)p}, \epsilon_{mp}, q^{(mp)})\right) + 1\right). \end{aligned}$$

■

Corollary 2.11 *If we require a fractional performance gap* $\frac{\epsilon_{mp}}{q^{(mp)}} = r$ *for* $\forall m \geq 0$ *in the first time slot of each interval, then for each interval* $(mp, (m+1)p-1)$, $m \not\equiv 0 \mod w$, *the normalized complexity is at most*

$$\frac{G}{p}\left(\sqrt{M}\log_2(g(\delta_{(m-1)p}, r)) + 1\right),$$

where

$$g(\delta_{(m-1)p}, r) = \frac{2\delta_{(m-1)p} + (1+\delta_{(m-1)p})r}{(1-\delta_{(m-1)p})r}.$$

Proof. From Theorem 2.10,

$$\begin{aligned} & f(\delta_{(m-1)p}, \epsilon_{(m-1)p}, \epsilon_{mp}, q^{(mp)}) \\ = \; & \frac{2\delta_{(m-1)p}}{(1-\delta_{(m-1)p})r} + \frac{(1+\delta_{(m-1)p})\epsilon_{(m-1)p}}{\epsilon_{mp}}. \end{aligned}$$

By (2.4) in the proof of Lemma 2.9,

$$(1-\delta_{(m-1)p})C^{(m-1)p}(Z^{(mp)})^* \leq C^{(mp)}(Z^{(mp)})^*.$$

From the optimality of $(Z^{(m-1)p})^*$, we obtain

$$\begin{aligned} & (1-\delta_{(m-1)p})q^{(m-1)p} \\ = \; & (1-\delta_{(m-1)p})C^{(m-1)p}(Z^{(m-1)p})^* \\ \leq \; & (1-\delta_{(m-1)p})C^{(m-1)p}(Z^{(mp)})^* \\ \leq \; & C^{(mp)}(Z^{(mp)})^* = q^{(mp)}. \end{aligned}$$

Then,

$$\frac{q^{(m-1)p}}{q^{(mp)}} = \frac{\epsilon_{(m-1)p}}{\epsilon_{mp}} \leq \frac{1}{1-\delta_{(m-1)p}}. \tag{2.8}$$

and thus,

$$\begin{aligned} & f(\delta_{(m-1)p}, \epsilon_{(m-1)p}, \epsilon_{mp}, q^{(mp)}) \\ \leq & \frac{2\delta_{(m-1)p} + (1+\delta_{(m-1)p})r}{(1-\delta_{(m-1)p})r} \\ = & g(\delta_{(m-1)p}, \gamma). \end{aligned}$$

Therefore,

$$\begin{aligned} & \frac{N^{(mp)}}{p} \\ \leq & \frac{G}{p}\left(\sqrt{M}\log_2\left(f(\delta_{(m-1)p}, \epsilon_{(m-1)p}, \epsilon_{mp}, q^{(mp)})\right) + 1\right) \\ \leq & \frac{G}{p}\left(\sqrt{M}\log_2(g(\delta_{(m-1)p}, r)) + 1\right). \end{aligned}$$

■

Corollary 2.11 gives an upper bound on the normalized complexity that grows logarithmically with $\frac{1}{r}$. This result matches the intuition that the amount of computation increases as higher precision is required.

2.3.4.3 Combined optimization

In this section, we consider an optimization problem which minimizes the normalized complexity subject to the condition that suboptimal solution in the interval achieves a given optimality gap during the interval.

We assume that the rate of the hyperarc cost change is at most α for any hyperarc in the network. Then, we obtain

$$\max_{(i,J)\in\mathcal{A}} \max_{0\leq j\leq p} \frac{|c_{iJ}^{(mp)} - c_{iJ}^{(mp+j)}|}{c_{iJ}^{(mp)}} = \delta_{mp} \leq p\alpha = d, \tag{2.9}$$

where d denotes the upper bound on the percentage change of the hyperarc cost in

each interval. As in Corollary 2.11, a fractional performance gap at the beginning of each interval is $\frac{\epsilon_{mp}}{q^{(mp)}} = r$.

In this section, we obtain a combined bound by optimizing the complexity bound subject to the cost bound in terms of d and r.

We first express the upper bound on the optimality gap during the interval.

From Lemma 2.9,

$$\begin{aligned} C^{(mp+l)}Z^{(mp)} &\leq \frac{1+\delta_{mp}}{1-\delta_{mp}} C^{(mp+l)}(Z^{(mp+l)})^* + (1+\delta_{mp})\epsilon_{mp} \\ &\leq \frac{1+d}{1-d} C^{(mp+l)}(Z^{(mp+l)})^* + (1+d)\epsilon_{mp}. \end{aligned}$$

Then,

$$\begin{aligned} & \frac{C^{(mp+l)}Z^{(mp)} - C^{(mp+l)}(Z^{(mp+l)})^*}{C^{(mp+l)}(Z^{(mp+l)})^*} \\ \leq\ & \frac{2d}{1-d} + \frac{(1+d)\epsilon_{mp}}{C^{(mp+l)}(Z^{(mp+l)})^*}. \\ =\ & \frac{2d}{1-d} + \frac{(1+d)\epsilon_{mp}}{q^{(mp+l)}}. \end{aligned} \tag{2.10}$$

To compute $\frac{(1+d)\epsilon_{mp}}{q^{(mp+l)}}$, we derive following inequalities.

$$\begin{aligned} q^{(mp+l)} &= C^{(mp+l)}(Z^{(mp+l)})^* \\ &\geq (1-\delta_{mp})C^{(mp)}(Z^{(mp+l)})^* \\ &\geq (1-d)C^{(mp)}(Z^{(mp+l)})^* \\ &\geq (1-d)C^{(mp)}(Z^{(mp)})^* \\ &= (1-d)q^{(mp)}. \end{aligned} \tag{2.11}$$

First inequality comes from (2.4) in the proof of Lemma 2.9 and the second inequality is true since $\delta_{mp} \leq d$. The last inequality comes from the optimality of $(Z^{(mp)})^*$. Since $\frac{\epsilon_{mp}}{q^{(mp)}} = r$,

$$\frac{\epsilon_{mp}}{q^{(mp+l)}} = \frac{q^{mp}}{q^{(mp+l)}} \cdot \frac{\epsilon_{mp}}{q^{(mp)}} \leq \frac{r}{(1-d)}. \tag{2.12}$$

From (2.10) and (2.12), we obtain the upper bound on the optimality gap in terms of d and r as follows :

$$\frac{C^{(mp+l)}Z^{(mp)} - C^{(mp+l)}(Z^{(mp+l)})^*}{C^{(mp+l)}(Z^{(mp+l)})^*} \leq \frac{2d}{1-d} + \frac{(1+d)r}{(1-d)}. \tag{2.13}$$

Now we express the complexity bound. From (2.9) and Corollary 2.11,

$$\begin{aligned} g(\delta_{(m-1)p}, r) &= \frac{2\delta_{(m-1)p} + (1+\delta_{(m-1)p})r}{(1-\delta_{(m-1)p})r} \\ &\leq \frac{2d + (1+d)r}{(1-d)r}, \end{aligned} \tag{2.14}$$

and the normalized complexity is at most

$$\begin{aligned} &\frac{G}{p}\left(\sqrt{M}\log_2(g(\delta_{(m-1)p}, r)) + 1\right) \\ &\leq \frac{G}{p}\left(\sqrt{M}\log_2(\frac{2d + (1+d)r}{(1-d)r}) + 1\right) \\ &= \frac{\alpha G}{d}\left(\sqrt{M}\log_2(\frac{\frac{2d}{1-d} + \frac{(1+d)r}{1-d}}{r}) + 1\right), \end{aligned} \tag{2.15}$$

where the first inequality comes from (2.14).

Now we can formulate the combined bound by minimizing the normalized complexity subject to the condition that suboptimal solution in the interval achieves a given optimality gap R during the interval.

$$\begin{aligned} \min \quad & \frac{\alpha G}{d}\left(\sqrt{M}\log_2(\frac{\frac{2d}{1-d}+\frac{(1+d)r}{1-d}}{r})+1\right) \\ \text{s.t} \quad & \frac{2d}{1-d}+\frac{(1+d)r}{(1-d)} \leq R \\ & d, r \geq 0, \end{aligned} \tag{2.16}$$

where the objective function comes from (2.15) and the first constraint comes from (2.13).

2.3.4.4 Approximation for small optimality gap

Here we obtain approximate solutions of (2.16) which provide qualitative insights for small required optimality gap $R << 1$.

Since the normalized complexity decreases with d and r, while the cost bound increases with d and r, the inequality constraint is satisfied with equality in the optimal solution which is easy to obtain numerically. We substitute $d = \frac{R-r}{R+r+2}$ into the objective function in (2.16). Then we obtain a simple optimization problem

$$\begin{aligned} \min \quad & \frac{\alpha G(R+r+2)}{R-r}\left(\sqrt{M}\log_2(\frac{R}{r})+1\right) \\ \text{s.t} \quad & r \leq R \end{aligned} \tag{2.17}$$

To compute the optimal solution, we differentiate the objective function. Then,

$$(2R+2)\left(\sqrt{M}\log_2(\frac{R}{r})+1\right)-(R^2-r^2+2(R-r))\frac{\sqrt{M}}{r}=0. \tag{2.18}$$

Since $r \leq R$ and R is sufficiently small, we first ignore $R^2 - r^2$ and obtain

$$(R+1)\left(\sqrt{M}\log_2(\frac{R}{r})+1\right)-(R-r)\frac{\sqrt{M}}{r}=0. \tag{2.19}$$

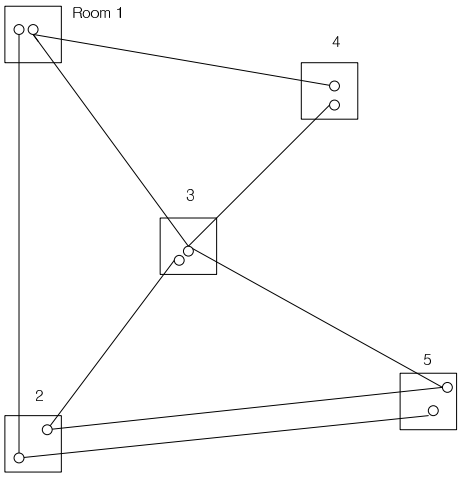


Figure 2.30: Scenario: 5 rooms are distributed over large area. Distance between any two rooms is sufficiently large. Each room contains two nodes and each node is connected to at least one node in a different room.

Since $R << 1$, above is simply approximated as follows :

$$\left(\log_2(\frac{R}{r}) + \frac{1}{\sqrt{M}}\right) - \frac{R-r}{r} = 0, \tag{2.20}$$

and we finally obtain

$$\frac{R}{r} - \log(\frac{R}{r}) = 1 + \frac{1}{\sqrt{M}}. \tag{2.21}$$

From (2.21), given M, d and r increase roughly linearly with R. Since $d = p\alpha$ from (2.9) and r is the fractional optimality gap at the beginning of the interval, as the optimality gap during the interval R decreases, p and r also decrease linearly and thus we should recompute the solution more often and more accurately.

Given R, from (2.21), as the number of inequalities in the optimization problem M increases, r increases and d decreases. Thus when the number of inequalities increases, we should recompute the solution more often with lower accuracy.

2.3.4.5 Example

In this section, we consider an example network scenario and solve the optimization problem (2.16) numerically. The results match our qualitative insights obtained in the previous section.

We consider a scenario in which m square rooms are distributed over same area, as shown in Fig. 2.30. Each room contains several nodes, total n nodes. Hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ is defined on this network. To communicate with nodes in different rooms, for any hyperarc (i, J), we assume that the set of destination nodes J contains at least one node contained in a different room from i. d_0 is used to denote the minimum distance between any two rooms. Then, by the definition of hyperarc cost, $c_{iJ} \geq d_0$ for $\forall (i, J) \in \mathcal{A}$. Here we use a mobility model based on a two-dimensional random walk model. The initial location of each node is given, and each node in each room moves as a random walker on a two-dimensional lattice. Each node has a probability of $\frac{1}{4}$ of moving to a position above, below, to the left, or to the right of its current position with step size β every time slot. When a node reaches a boundary of the room, it is reflected. Since any hyperarc has cost at least d_0 and each link cost can be changed at most 2β per each time slot,

$$\begin{aligned} \delta_{mp} &= \max_{(i,J)\in\mathcal{A}} \max_{0\leq s\leq p} \frac{|c_{iJ}^{(mp)} - c_{iJ}^{(mp+s)}|}{c_{iJ}^{(mp)}} \\ &\leq \frac{2\beta p}{d_0}. \end{aligned} \tag{2.22}$$

Thus we substitute $\frac{2\beta}{d_0}$ into α which is the maximum rate of hyperarc cost change in (2.9), and obtain the same optimization problem in (2.16).

As shown in Fig. 2.31, when $\frac{2\beta}{d_0} = \alpha = 10^{-3}$ and $M = 1000$, the optimal fractional performance gap at the start of the interval, r^*, grows almost linearly with the upper bound of fractional optimality gap during the interval, R. Given r^* and R, since $d = p\alpha = \frac{R-r}{R+r+2}$, the optimal length of the interval is $p^* = \frac{d}{\alpha} = \frac{R-r^*}{(R+r^*+2)\alpha}$. As shown in Fig. 2.32, p^*, also grows almost linearly with R. These results match our qualitative insights in Section 2.3.4.4.

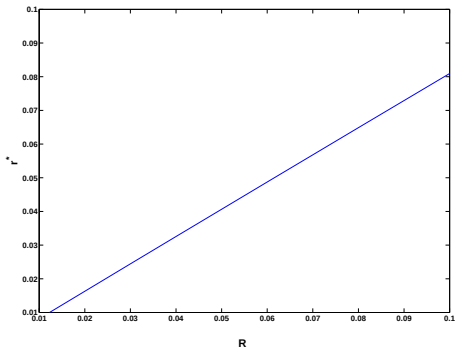


Figure 2.31: Optimal fractional performance gap at the start of the interval, r, versus R, an upper bound on the fractional optimality gap over each interval, when $\alpha = 10^{-3}$ and $M = 1000$.

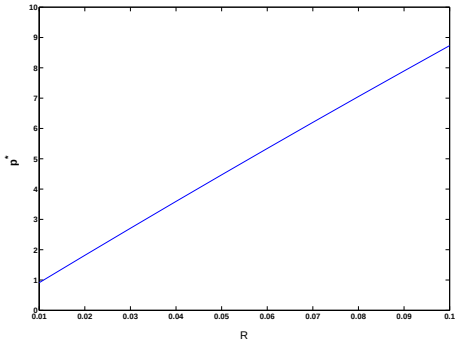


Figure 2.32: Optimal length of interval, p^*, versus R when $\alpha = 10^{-3}$ and $M = 1000$.

2.3.5 Conclusion

In this section, we have analyzed network coding with periodic recomputation for minimum energy multicasting in MANETs. In this approach, we recompute an approximate solution at fixed time intervals, and use this solution during each time interval although the network topology changes. We have obtained a simple theoretical cost bound on the gap between our solution and the optimal cost. We have combined our cost bound with warm-start strategy to obtain the bound on the complexity using interior-point method. Finally, we have derived a combined bound that minimizes the time complexity bound subject to the condition that suboptimal solution in the interval achieves a given optimality gap during the interval. By solving this optimization problem, the optimal length of the interval and the fractional optimality gap at the first time slot of the interval increase roughly linearly with the optimality gap during the interval. We have confirmed this qualitative insight with example network scenario.

2.4 Network optimization framework using back-pressure approach

2.4.1 Introduction

Most of the previous work in network optimization assumes a flow model for transmission with fixed input rates and link capacities. However, in real networks, traffic is usually bursty because either the sources generate traffic in bursts or the network nodes employ queuing and scheduling across multiple sessions. Then we have to consider not only routing and network coding but also scheduling of flows of different sessions.

In [18–25], several back-pressure type algorithms are proposed for the multiple unicasts problem without coding, in which nodes use the queue length information of neighboring nodes to make routing decisions. Packets are adaptively routed through-

out the network in response to congestion information. This approach is called back-pressure type since heavily loaded nodes downstream push back and slow down the flow coming down from nodes upstream. The back-pressure algorithm is resilient to link failures and topological changes. Such a back-pressure approach is generally optimal in the sense that it allows transmission at the maximum possible arrival rates into the network for which the queues at the various network nodes are still stable. Moreover, it is shown that the algorithms are distributed since decisions are made locally at each node based on feedback about the size of queues at the destination node of each link.

However, in the case without network coding, the algorithms are significantly more complex and harder to implement, even for wired networks. By combining network coding with back-pressure approach, several distributed polynomial-time algorithms for optimizing the network resources have been presented recently. In [27], dynamic-back-pressure algorithms for multicast routing, network coding, power allocation, session scheduling, and rate allocation across correlated sources, which achieve stability for rates within the capacity region, are presented. In [26], for wired and wireless networks, off-line and online back pressure algorithms for finding approximately throughput-optimal network codes within the class of network codes restricted to XOR coding between pairs of flows. In [29], another dynamic back-pressure routing-scheduling-coding strategy for inter-session network coding is introduced.

Observe that most of the previous works have a common frame in their stories. Given any optimization problem with a specified class of network codes, the authors first design virtual queues at each node in the network and implement generalized links between queues such that any routing and specified coding/decoding operation in the original network corresponds to the transmission between virtual queues over a generalized link. For instance, in [27], each node has just one virtual queue for each sink of each multicast session and there is a generalized link between two neighbor nodes which is designed for the transmission of packets destined to the same sink of the same session. Given a queue and generalized link design, a back-pressure type algorithm for routing, network coding, and scheduling is proposed. Finally, they have

shown that the algorithms stabilize the network for all input rates within the capacity region. This common frame in previous works motivates us to construct an general optimization framework with back-pressure approach.

Our main contribution is to propose a back-pressure based distributed optimization framework, which can be used for optimizing over any class of network codes, including pairwise XOR coding problem, and reverse carpooling and star coding problem. Our approach is to specify the class of coding operations by a set of generalized links, and to develop optimization tools that apply to any network composed of such generalized links. Our framework covers any optimization problem with class of network codes for which virtual queues and generalized links can be designed to satisfy following condition : each generalized link e is associated with a transmission set $\mathcal{P}_e$ of pairs $(\mathcal{O}, \mathcal{D})$ such that packets from each queue in the set $\mathcal{O}$ are transformed into an equivalent number of packets in each queue in the set $\mathcal{D}$ via e. We present a dynamic back-pressure type algorithm in which routing, network coding, and scheduling decisions are made locally by comparing, for each link, the difference in length of corresponding virtual queues. It is shown that our algorithm does not allow the length of any queue in the network to increase infinitely and achieve the stability for any input rates within the capacity region. In our stability proof, we can choose any degree of potential function while the square function of the queue length was used as potential function in previous works [26, 27]. We also minimize the upper bound on the queue length, and show that our minimized upper bound on the queue length improves the previous bound.

2.4.2 Preliminaries

2.4.2.1 queue and generalized link design

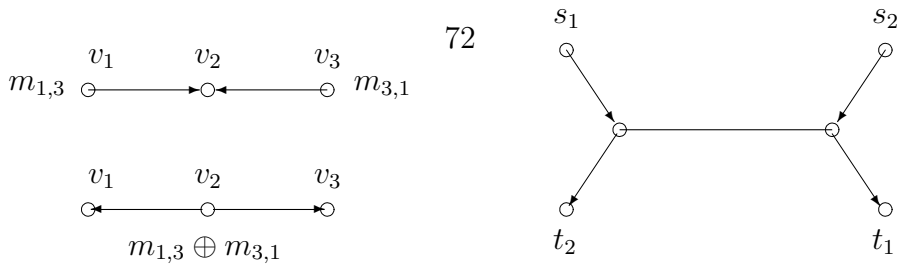
We consider a network as a directed graph, $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. Let $N = |\mathcal{N}|$ and $E = |\mathcal{E}|$. There is a set of communication sessions $\mathcal{C}$ sharing the network. We assume that each session $c \in \mathcal{C}$ is associated with a source node s_c and an arrival process of exogenous session c packets to be transmitted to each of a set $\mathcal{T}_c \subset \mathcal{N} - s_c$. To

apply back-pressure framework to network optimization problem, we have to design a set of conceptual queues and generalized links between queues appropriately. We consider a generalized network $\mathcal{G}'$ comprising a set $\mathcal{N}$ of nodes, with a set $\mathcal{E}'$ of generalized communication links. We assign the designed queues at each node in this generalized network and there is a virtual transmission between the sets of queues over generalized links. Each generalized link $e \in \mathcal{E}'$ is associated with a transmission set $\mathcal{P}_e$ of pairs $(\mathcal{O}, \mathcal{D})$ such that packets from each queue in the set $\mathcal{O}$ are transformed into an equivalent number of packets in each queue in the set $\mathcal{D}$ via e. In this paper, we consider a network optimization problem with class of network codes that satisfies the following condition.

Condition 2.12 *there is a one to one correspondence between any sequence of events (routing, coding, decoding, and etc.) in the original coding network and the sequence of events through a generalized link on the generalized network.*

2.4.2.2 Model, approach, and notation

We consider the network optimization problem in time-varying networks. We assume that time is slotted with time slots of duration T. We also assume that the channel conditions are fixed over the duration of a slot, and known at the beginning of the slot. For each communication session c, we define a source queue $U_{s_c}^c$ at the source node s_c and a overflow queue $\bar{U}^c$. r_c denotes the input rates within the capacity region of session c. We use $\mathcal{Q}$ to denote the set of all queues in this generalized network apart from the overflow queues. For any queue $Q \in \mathcal{Q}$, we use $Q(t)$ to denote the length of queue Q at time t. We use x_Q and y_Q to denote the total allocated flow rate into and out of queue Q respectively. We assume that all generalized links in $\mathcal{E}'$ have capacity of at most μ. Let M denote the maximum possible increase or decrease rate of queue length in $\mathcal{Q}$ during each time unit, i.e, $x_Q(t), y_Q(t) \leq M$ for $Q \in \mathcal{Q}$, $\forall t$. (In [26], $M = 5\mu$. In Section 2.4.3.1, $M = \mu$.) V denotes the maximum length of source queue. A is used to denote the maximum length of any queue on the generalized network we will derive later. For simplicity, we assume that $|\mathcal{O}| \leq 2$



(a) (b)

Figure 2.33: Reverse carpooling : (a) Transmitting messages $m_{1,3}$ and $m_{3,1}$ from v_1 to v_3 and v_3 to v_1, respectively, requires three transmissions with network coding and four without. (b) Generalize reverse carpooling for two unicast sessions which overlap in opposite directions.

for any transmission set $(\mathcal{O}, \mathcal{D}) \in \mathcal{P}_e$ and $|\mathcal{O}| = 1$ when $e \in \mathcal{E}$ is a real link. Each queue $Q \in \mathcal{Q}$ has a potential $L_Q(t) = (Q(t))^{k+1}$ at time t , where k depends on the network optimization problem. Let $U^c(t)$ denote the length of overflow queue $\bar{U}^c$ for each session c at time t. The potential of overflow queue is defined as $(k+1)V^k \cdot U^c(t)$. Thus, the total potential is defined as follows:

$$L(t) = \sum_{Q \in \mathcal{Q}} Q(t)^{k+1} + (k+1)V^k \sum_c U^c(t).$$

2.4.3 Optimization problems

Here we introduce network optimization problems with a specified class of network codes that we mainly focus on in this paper. We first consider optimization problems using reverse carpooling and star coding strategies. We also consider optimization problems with pairwise XOR coding strategy. In this section, for each coding strategy, we design the queue and generalized links to apply our distributed optimization framework.

2.4.3.1 Reverse carpooling and star-coding

We consider simple network coding strategies reverse carpooling and star-coding shown in Chapter 1. As shown in [7, 10, 53, 54], these strategies can be used for power saving and increasing the throughput in wireless networks.

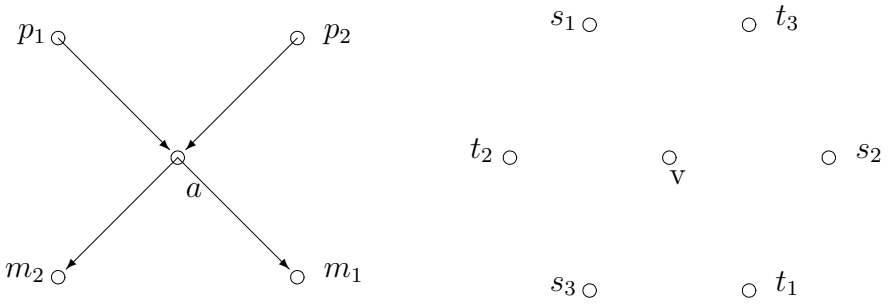


Figure 2.34: (a) 2-star coding : p_1 and p_2 want to transmit packets x_1 and x_2 to m_1 and m_2 respectively. Since m_2 overhears transmission from p_1 and m_1 overhears from p_2, v transmits $x_1 \oplus x_2$ to m_1 and m_2. This saves a single transmission. (b) 3-star coding: s_i wants to transmit packet x_i to t_i $(1 \leq i \leq 3)$ and t_j overhears from s_i $(j \neq i)$. Node v broadcasts $x_1 \oplus x_2 \oplus x_3$ to t_1, t_2, and t_3 and it gives a savings of two transmissions.

We present the design of queues and generalized links for reverse carpooling and star-coding strategies that satisfies condition 2.12. Wireless broadcast links are denoted (a, Z), where a is the originating node and Z is the set of destination nodes. We use $\mathcal{C}$ to denote the set of unicast sessions. Now we simply design virtual queues as following:

· $Q_i^{cN(a,Z)}$: uncoded session c packets stored at node i which are transmitted from hyperarc (a, Z) where $N(a, Z) = \{a\} \cup \{b : b \in Z\}$.

For each session c, we also define a source queue $U_{s_c}^c$ at the source node s_c and a overflow queue $\bar{U}^c$. When we apply reverse carpooling or star-coding strategy, a node make a decision on which of its queued packets to code and transmit, based on packets' next-hop nodes and which of these packets have been overheard by the next-hop nodes. Thus, for each packet in the queue, we have to keep track of its previous-hop node and hyperarc which was used for its previous transmission. Then our following queue design minimizes the number of queues on the network.

For any hyperarc (a, Z), the transmission set $P_{(a,Z)}$ is defined as follows:

· $P_{(a,z)} = \{(Q_a^{c_1 N(p_1,Z_1)}, Q_{m_1}^{c_1 N(a,Z)}) \cup (Q_a^{c_2 N(p_2,Z_2)}, Q_{m_2}^{c_2 N(a,Z)}) \ldots (Q_a^{c_k N(p_k,Z_k)}, Q_{m_k}^{c_k N(a,Z)})$: $c_i \in \mathcal{C}, M = \{m_1, .., m_k\} \subset Z, M - \{m_i\} \subset N(p_i, Z_i)\}$.

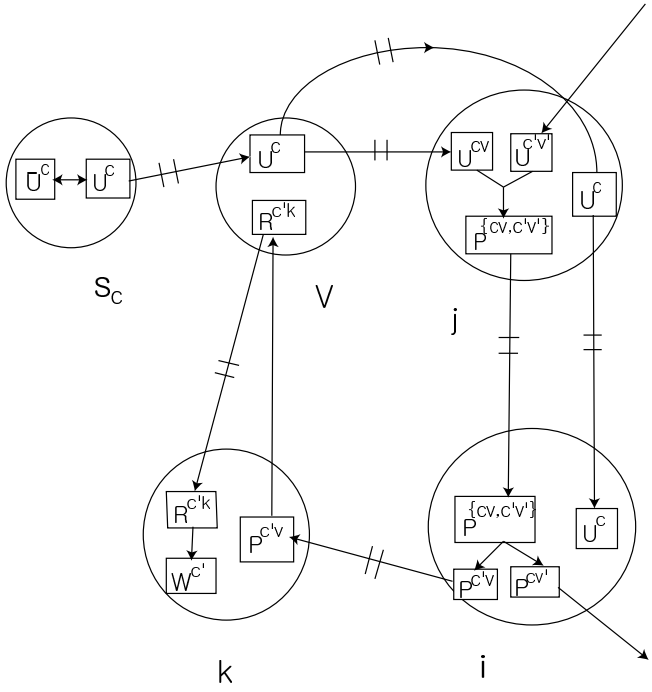


Figure 2.35: Pairwise XOR coding

We can check that the above queue and generalized link design satisfies the condition 2.12. Any reverse carpooling and star-coding event on the original network corresponds to some transmission set on the generalized network defined above. For instance, when we apply 2-star coding in Fig. 2.34(a), the transmission set assigned to hyperarc $(a, \{m_1, m_2\})$ is $P_{(a,Z)} = \{(Q_a^{c_1 N(s_1,Z_1)}, Q_{t_1}^{c_1 N(a,Z)}) \cup (Q_a^{c_2 N(s_2,Z_2)}, Q_{t_2}^{c_2 N(a,Z)})\}$ on the generalized network where $m_2 \in Z_1$, $m_1 \in Z_2$, and $Z = \{m_1, m_2\}$. Then there is a one to one correspondence between this 2-star coding in the original network and the transmissions through generalized link on the generalized network. For reverse carpooling in Fig. 2.33(a), the transmission set assigned to hyperarc (v_2, Z) where $(v_1, v_3) \in Z$ is $P_{(v_2,Z)} = \{(Q_{v_2}^{c_1 N(v_1,Z_1)}, Q_{v_3}^{c_1 N(v_2,Z)}) \cup (Q_{v_2}^{c_2 N(v_3,Z_1)}, Q_{v_1}^{c_2 N(v_2,Z)})\}$.

Based on this queue and generalized link design, we show that our optimization framework in Section 2.4.4 gives the optimal solution and achieves the stability.

2.4.3.2 Pairwise XOR coding

In this section, we consider network coding across multiple unicasts, using the class of pairwise XOR codes in [26, 55, 56]. In this class of codes, network coding is limited

to XOR coding between pairs of uncoded packets. Two uncoded packets of different sessions can be coded together to form a joint poison packet in order to share capacity on one or more hops. The joint poison packet is subsequently replicated to form two identical individual poison packets whose routes branch. Then these are met by corresponding remedy packets and decoded to form the original uncoded packets. In [26], they develop online back-pressure algorithm for finding approximately throughput-optimal network codes within the class of network codes restricted to XOR coding between pairs of flows, for wired and wireless networks. In a dynamic online setting, the instantaneous source arrival rates and link capacities/constraints may vary ergodically. In this problem, we assume that each session's elementary flow undergoes coding/decoding at most one time. We show that our back-pressure framework can be used to solve this pairwise XOR coding problem. Using our framework, it is able to stabilize all queues in the network for any stabilizable set of exogenous source rates.

Here we design queue and generalized link for pairwise XOR coding problem as follows:

· U_i^c : Uncoded session c packets at node i which did not undergo coding and decoding yet.

· U_i^{cv} : Uncoded session c packets at node i also stored at node v. These packets also did not undergo coding and decoding yet.

· W_i^c : Uncoded session c packets at node i which has undergone coding and decoding, and cannot be coded anymore.

· $P_i^{\{cv,c'v'\}}$: Joint poison packets at node i for session c and c' which are also stored at nodes v and v', respectively.

· P_i^{cv} : Individual poison packets at node i.

· R_k^{cv} : Remedy packets for session c at node k which was requested from individual poison packet at node v.

Based on above queue design, we can also define generalized link $\mathcal{E}'$ and corresponding transmission set. As shown in Fig. 2.35, for a coding link e at a node j, $P_e = \{(\{U_j^{cv}, U_j^{c'v'}\}, P_j^{\{cv,c'v'\}})\}$. For a branching link e at a node i, $P_e = \{(P_i^{\{cv,c'v'\}}, \{P_i^{cv'}, P_i^{c'v}\})\}$. For a requesting link e at a node k, $P_e = \{(P_k^{cv'}, R_{v'}^{ck})\}$. For a decoding link e at node

k, $P_e = \{(R_k^{ck}, W_k^c)\}$. For a real link $(a,b) \in \mathcal{E}$, $P_{(a,b)}$ consists of pairs (U_a^c, U_b^c), pairs (U_a^{cv}, U_b^{cv}), pairs (U_a^c, U_b^{ca}), pairs $(P_a^{\{cv,c'v'\}}, P_b^{\{cv,c'v'\}})$, pairs (P_a^{cj}, P_b^{cj}), pairs (W_a^c, W_b^c), and pairs (R_a^{cj}, R_b^{cj}). For a wireless link (a, Z), the transmission set contains all pairs in $P_{(a,b)}$, $b \in Z$ as well as pairs $(P_a^{\{cv,c'v'\}}, \{P_b^{cv'}, P_b^{\prime c'v}\})$ and $(U_a^c, U_b^{cb'})$ where $b, b' \in Z$.

The example of data flow for the above queue and generalized link is shown in Fig. 2.35. From [26], the above queue and generalized link design also satisfies the condition 2.12.

2.4.4 back-pressure framework

In this section, we propose a general back-pressure algorithm in which nodes maintain a queue for each session's packets at each node, and a route based on queue gradients that form by the addition of packets to sources and their removal from sinks. Then we prove the stability of our algorithm for any input rates within the capacity region.

2.4.4.1 back-pressure algorithm

We consider any transmission $(\mathcal{O}, \mathcal{D})$ over generalized link $e \in \mathcal{E}'$ where $\mathcal{O} = \{Q_o^1, ..., Q_o^m\}$ and $\mathcal{D} = \{Q_d^1, ..., Q_d^n\}$ $(m \leq 2)$. We define the weight of transmission $(\mathcal{O}, \mathcal{D}) \in P_e$ at time t as follows,

$$\begin{aligned} w_{(\mathcal{O},\mathcal{D})}^{(t)} &= \sum_{Q \in \mathcal{O}} L'_Q(t) - \sum_{Q \in \mathcal{D}} L'_Q(t) \\ &= (k+1)\{\sum_{Q \in \mathcal{O}} Q(t)^k - \sum_{Q \in \mathcal{D}} Q(t)^k\} \end{aligned}$$

In each time slot $(t, t+T]$, the following steps are carried out:

1) For each generalized link $e \in \mathcal{E}'$, choose the pair $(\mathcal{O}, \mathcal{D}) \in P_e$ that maximizes $w_{(\mathcal{O},\mathcal{D})}^{(t)}$. Then transfer across the chosen pair $(\mathcal{O}, \mathcal{D})$, at the instantaneous rate of link e.

2) Remove all packets from queues at sink nodes for each session.

3) Add r_c units to the source queues for each session.

4) After completing steps 1-3 at time $(t+T)^-$, for each session $c \in \mathcal{C}$, transfer packets between the source queue and the overflow queue $\bar{U}^c$ of each session c, so as to maximize $U^c_{s_c}((t+T)^+)$ subject to a maximum length constraint of V.

2.4.4.2 Proof of stability

Lemma 2.13 *Rebalancing policy in step 4 does not increase the potential $L(t)$.*

Proof. We first describe the detail of the rebalancing policy in step 4 and show that potential function is not increased in this step. At time $t = 0$, $U^c_{s_c} = \bar{U}^c = 0$. Let t^- and t^+ denote the time instant just before and after rebalancing, respectively. Let

$$W^c(t) = \min\{V - U^c_{s_c}(t^-), \bar{U}^c(t^-)\}$$

where V represents an upper bound on source queue length, enforced by policy. In this rebalancing policy, at time t^+, $W^c(t)$ packets are added to $U^c_{s_c}$ and subtracted from $\bar{U}^c(t^-)$, i.e.,

$$\begin{aligned} U^c_{s_c}(t^+) &= U^c_{s_c}(t^-) + W^c(t), \\ \bar{U}^c(t^+) &= \bar{U}^c(t^-) - W^c(t). \end{aligned}$$

Let $L^c(t) = (U^c_{s_c})^{k+1} + (k+1)V^k\bar{U}^c(t)$. From the above policy, the increased potential in step 4 is

$$\sum_c (L^c(t^+) - L^c(t^-)) = \sum_c ((U^c_{s_c}(t^-) + W^c(t))^{k+1} - (U^c_{s_c}(t^-))^{k+1} - (k+1)V^k W^c(t)).$$

If $V < U^c_{s_c}(t^-)$, $W^c(t) = V - U^c_{s_c}(t^-) < 0$. Thus,

$$\begin{aligned} L^c(t^+) - L^c(t^-) &= V^{k+1} - U^c_{s_c}(t^-)^{k+1} - (k+1)V^k(V - U^c_{s_c}(t^-)) \\ &= (V - U^c_{s_c}(t^-))(V^k + ... + U^c_{s_c}(t^-)^k - (k+1)V^k) \\ &\leq 0. \end{aligned}$$

Otherwise, $V \geq U^c_{s_c}(t^-)$ and $0 \leq W^c(t) \leq V - U^c_{s_c}(t^-)$. To prove that the potential is not increased, we use the following inequality :

$$(x+\delta)^{k+1} - x^{k+1} \leq \delta(k+1)(x+\delta)^k$$

for $\delta \geq 0$.

Since $W^c(t) \geq 0$ and $U^c_{s_c}(t^-) + W^c(t) \leq V$,

$$\begin{aligned} L^c(t^+) - L^c(t^-) &= (U^c_{s_c}(t^-) + W^c(t))^{k+1} - U^c_{s_c}(t^-)^{k+1} - (k+1)V^k W^c(t) \\ &= (k+1)W^c(t)(U^c_{s_c}(t^-) + W^c(t))^k - (k+1)V^k W^c(t) \\ &\leq 0. \end{aligned}$$

Therefore, $L^c(t^+) \leq L^c(t^-)$. ■

Before introducing the lemma which gives the maximum length of any queue, we first define the backtracking approach which will be used in the proof of lemma.

Consider any transmission $(\mathcal{O}, \mathcal{D})$ which happened between time $t - T$ and t over generalized link $e \in \mathcal{E}'$ where $\mathcal{O} = \{Q^1_o, ..., Q^m_o\}$ and $\mathcal{D} = \{Q^1_d, ..., Q^n_d\}$ $(m \leq 2)$. When we backtrack the flow from any queue $Q^j_d \in \mathcal{D}$ at time t such that data is transmitted from queues in $\mathcal{O}$, we choose $Q^i_o \in \mathcal{O}$ such that Q^i_o has the maximum queue length among queues in $\mathcal{O}$ at time $t - T$ before transmission occurs, i.e., $Q^i_o(t-T) = \max_{1 \leq p \leq m} Q^p_o(t-T)$, and move to chosen queue at time $t - T$.

Note that in our back-pressure policy, it transfers data in directions where the

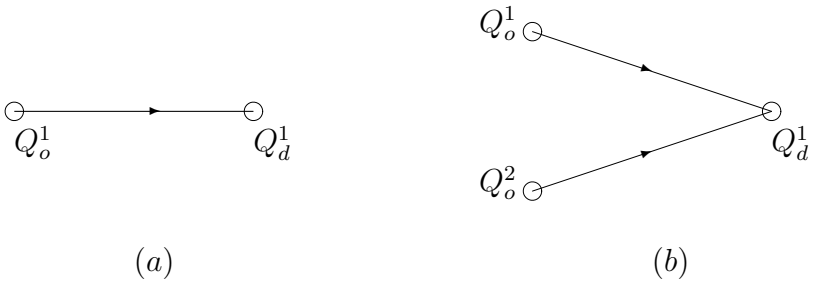


Figure 2.36: Backtracking approach: (a) Transmission pair $(\mathcal{O}, \mathcal{D})$ on the generalized link e where $|\mathcal{O}| = |\mathcal{D}| = 1$. Using the backtracking approach for this example, we move from $Q_d^1(t)$ to $Q_o^1(t-T)$. (b) Transmission pair $(\mathcal{O}, \mathcal{D})$ on the generalized link e where $|\mathcal{O}| = 2$ and $|\mathcal{D}| = 1$. Using the backtracking approach for this example, we move from $Q_d^1(t)$ to $\max(Q_o^1(t-T), Q_o^2(t-T))$.

forward differential backlog is nonnegative, i.e., $\sum_{1\le i\le m}(a_i)^k - \sum_{1\le j\le n}(b_j)^k \ge 0$. The length of any queue $Q \in \mathcal{Q}$ can be increased at most TM during each time slot.

Here we give examples for our backtracking procedure. When $|\mathcal{O}| = |\mathcal{D}| = 1$ as shown in Fig. 2.36(a), as in the queue and generalized link design for reverse carpooling and the star-coding problem, we move from $Q_d^1(t)$ to $Q_o^1(t-T)$ using our backtracking approach. Then,

$$Q_d^1(t) \le Q_d^1(t-T) + MT \le Q_o^1(t-T) + MT.$$

Second inequality comes from the fact that the forward differential backlog is nonnegative, $(Q_o^1(t-T))^k \ge (Q_d^1(t-T))^k$. Similarly, when $|\mathcal{O}| = 1$ and $|\mathcal{D}| = p \ge 1$, for any $Q_d^j \in \mathcal{D}$,

$$Q_d^j(t) \le Q_d^j(t-T) + MT \le Q_o^1(t-T) + MT. \tag{2.23}$$

since $(Q_o^1(t-T))^k \ge \sum_{i=1}^{p}(Q_d^i(t-T))^k \ge (Q_d^j(t-T))^k$.

When $|\mathcal{O}| = 2$ and $|\mathcal{D}| = 1$ as shown in Fig. 2.36(b), as in the coding link

transmission for pairwise XOR coding in Section 2.4.3.2, we move from $Q_d^1(t)$ to $\max(Q_o^1(t-T), Q_o^2(t-T))$ using the backtracking approach. Then,

$$\begin{aligned} Q_d^1(t) &\leq Q_d^1(t-T) + MT \\ &= (Q_d^1(t-T)^k)^{\frac{1}{k}} + MT \\ &\leq (Q_o^1(t-T)^k + Q_o^2(t-T)^k)^{\frac{1}{k}} + MT \\ &\leq (2 \cdot \max(Q_o^1(t-T)^k, Q_o^2(t-T))^k)^{\frac{1}{k}} + MT, \end{aligned}$$

which is equivalent to

$$Q_d^1(t) \leq 2^{\frac{1}{k}} \max(Q_o^1(t-T), Q_o^2(t-T)) + MT. \tag{2.24}$$

Lemma 2.14 *Under our algorithm, the length of any queue cannot exceed a constant which depends on the parameters of the network optimization problem.*

Proof.

First consider the network optimization problem where queue and generalized links are designed such that any data is transmitted from one-queue to one-queue over real link, as the reverse carpooling and star-coding problem in Section 2.4.3.1 and the dynamic multicast with intra-session network coding [27]. When we backtrack the data flow from any queue Q of the session c, since every data transmission is from one-queue to one-queue of the same session's, we reach the source queue $U_{s_c}^c$ after passing at most $N-1$ real links. Thus, applying (2.23) recursively $N-1$ times from our backtracking procedure gives

$$Q(t) \leq V + (N-1)MT. \tag{2.25}$$

Now we consider a pairwise XOR coding problem in Section 2.4.3.2. In this problem, each elementary flow undergoes coding/decoding at most one time from the source to the sink and every transmission is from one-queue to one-queue of the same

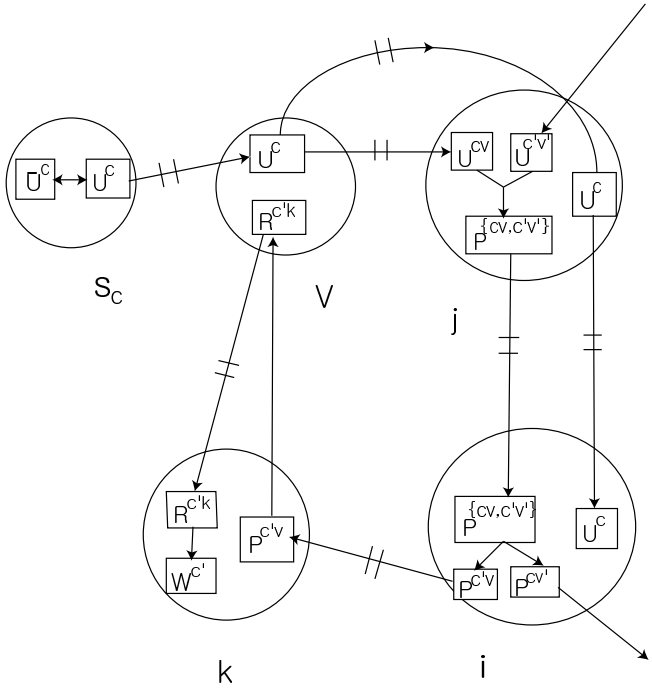


Figure 2.37: Example of case 2 in the proof of Lemma 2.14: we derive the maximum length of queue $W_k^{c'}$ using our backtracking approach by following transmissions in reverse order.

session's except the transmission over coding link from the queue and generalized link design. For any queue Q, we derive an upper bound on $Q(t)$ by using the backtracking approach recursively.

Case 1) When we backtrack the flow from Q of the session c, we do not pass the coding link and reach the source queue of the same session.

In this case, the flow we are backtracking does not undergo coding/decoding. Then every transmission is from one-queue to one-queue of the same session's over real link. We pass at most $N-1$ real links until we reach the source queue $U_{s_c}^c$. As in (2.25), $Q(t) \leq V + (N-1)MT$.

Case 2) When we backtrack the flow from the queue Q of the session c, we pass the coding and decoding link one time, respectively.

First consider the example in Fig. 2.37. Suppose that we want to derive the maximum length of queue $W_k^{c'}$ at time t using our backtracking approach. We apply backtracking procedure from $W_k^{c'}(t)$ until we reach the source queue of session c, $U_{s_c}^c$, by following transmissions shown in Fig. 2.37 in reverse order. Here time index for

queue length has been omitted for brevity.

When flow is transmitted from $R_k^{c'k}$ to $W_k^{c'}$ over decoding link $\{(R_k^{c'k}, W_k^{c'})\}$, backtracking approach gives

$$W_k^{c'} \leq R_k^{c'k} + MT. \tag{2.26}$$

Similarly, we apply our backtracking approach sequentially to remedy packet transmission $(R_v^{c'k}, R_k^{c'k})$, requesting link $P_e = \{(P_k^{c'v}, R_v^{c'k})\}$, individual poison packet transmission $(P_i^{c'v}, P_k^{c'v})$, branching link $\{(P_i^{\{cv,c'v'\}}, \{P_i^{cv'}, P_i^{c'v}\})\}$, joint poison packet transmission $(P_j^{\{cv,c'v'\}}, P_i^{\{cv,c'v'\}})$, before reaching the coding link at node j. Since all of these links are from one-queue to one or two-queue, by applying (2.23) recursively, we obtain

$$R_k^{c'k} \leq P_j^{\{cv,c'v'\}} + 5MT. \tag{2.27}$$

From equations (2.26) and (2.27),

$$W_k^c \leq P_j^{\{cv,c'v'\}} + 6MT. \tag{2.28}$$

At coding link $\{(\{U_j^{cv}, U_j^{c'v'}\}, P_j^{\{cv,c'v'\}})\}$, assume that the length of queue U_j^{cv} is larger than that of $U_j^{c'v'}$, i.e., $U_j^{cv} \geq U_j^{c'v'}$. From our backtracking approach in Fig. 2.36(b), we move from $P_j^{\{cv,c'v'\}}$ to U_j^{cv}, and (2.24) gives

$$P_j^{\{cv,c'v'\}} \leq 2^{\frac{1}{k}} U_j^{cv} + MT. \tag{2.29}$$

We apply again our backtracking approach sequentially from U_j^{cv} to the source queue $U_{s_c}^c$ over real links transmissions (U_v^c, U_j^{cv}) and $(U_{s_c}^c, U_v^c)$. This gives

$$U_j^{cv} \leq U_{s_c}^c + 2MT. \tag{2.30}$$

From equations (2.28), (2.29), (2.30),

$$W_k^c \leq 2^{\frac{1}{k}}(U_{s_c}^c + 2MT) + 7MT. \tag{2.31}$$

Since $U_{s_c}^c \leq V$, we finally obtain

$$W_k^c \leq 2^{\frac{1}{k}}(V + 2MT) + 7MT. \tag{2.32}$$

Now we generalize above example to complete the proof in this case. Assume that the elementary flow of session c is coded with the elementary flow of session c' at coding link $X = (\{U_Y^{cv}, U_Y^{c'v'}\}, P_Y^{cv,c'v'})$ at node Y. We start from the queue Q of session c and backtrack the data flow to the queue of the same session's until we reach the coding link X. Before reaching the coding link, since each elementary flow undergoes coding/decoding at most one time, we pass at most $N-1$ real links, one branching link, one requesting link, and one decoding link (*). From our queue and generalized link design, all of this transmissions are from one-queue to one or two-queue. At coding link X, we apply backtracking approach as shown in Fig. 2.36(b). Then we backtrack from $P_Y^{cv,c'v'}$ to one of U_Y^{cv} and $U_Y^{c'v'}$ which has larger queue length. Since each elementary flow undergoes coding at most one time, after moving one of U_Y^{cv} and $U_Y^{c'v'}$, we backtrack the chosen session's flow until we reach the source queue by passing at most $N-1$ real links (**). For instance, if we move from $P_Y^{cv,c'v'}$ to $U_Y^{c'v'}$ in backtracking, we pass at most $N-1$ real links until we reach the source queue $U_{s_c'}^{c'}$.

Now we derive the upper bound on $Q(t)$ by computing the maximum queue length subsequently from the source queue using backtracking procedure described above. Let a denote the maximum length of queues U_Y^{cv} and $U_Y^{c'v'}$, and b denote the length of the queue $P_Y^{cv,c'v'}$ in our backtracking procedure. From (**), by applying equation (2.23) $N-1$ times recursively, the maximum length of queue between U_Y^{cv} and $U_Y^{c'v'}$ is less than or equal to the length of the source queue plus $(N-1)MT$, i.e.,

$$a \leq V + (N-1)MT. \tag{2.33}$$

At coding link, we apply (2.24) and obtain

$$b \leq 2^{\frac{1}{k}}a + MT. \tag{2.34}$$

From the queue $P_Y^{cv,c'v'}$ to the queue Q, we pass at most $N-1+3 = N+2$ generalized links by (*) such that each link transmits data from one-queue. Thus, applying (2.23) $N+2$ times recursively gives

$$Q(t) \leq b + (N+2)MT. \tag{2.35}$$

From equations (2.33), (2.34), (2.35), we obtain that

$$Q(t) \leq 2^{\frac{1}{k}}(V + (N-1)MT) + (N+3)MT. \tag{2.36}$$

This completes the proof.

■

Now we propose the main theorem in this work.

Theorem 2.15 *Given a queue and generalized link design, if input rates $(r^c + \epsilon)$ are achievable on the original network for some $\epsilon > 0$, then back-pressure policy stabilizes the system for rate (r^c).*

Proof. In Lemma 2.13, we have shown that step 4 does not increase $L(t)$. So we focus on the change in potential across steps 1-3. The queues evolve according to

$$Q(t+T) \leq \max\{Q(t) - Ty_Q(t), 0\} + Tx_Q(t). \tag{2.37}$$

We omit the time indexes t from $x_Q(t)$ and $y_Q(t)$ for brevity. Let $D_Q(t) = (k+1)T(y_Q - x_Q)Q(t)^k$.

Here is the outline of proof. We first show that

$$L_Q(t+T) - L_Q(t) = Q(t+T)^{k+1} - Q(t)^{k+1} \leq C - 2D_Q(t) \tag{2.38}$$

for $\forall Q \in \mathcal{Q}$, where C is a positive constant which will be defined later. Then,

$$\begin{aligned} E\{L(t+T) - L(t)\} &\leq \sum_{Q \in \mathcal{Q}} E\{L_Q(t+T) - L_Q(t)\} \\ &\leq \sum_{Q \in \mathcal{Q}} E\{C - 2D_Q(t)\} \\ &= |\mathcal{Q}|C - 2D(t), \end{aligned}$$

where $D(t) = E\{\sum_{Q \in \mathcal{Q}} D_Q(t)\} = E\{\sum_{Q \in \mathcal{Q}} (k+1)T(y_Q - x_Q)Q(t)^k\}$. Here $D(t)$ corresponds to drift terms in [26,27]. After proving (2.38), we show that back-pressure algorithm maximizes $D(t)$ and the stability of our back-pressure algorithm is shown by comparison with a randomized policy [26, 27, 57]. Finally we prove $E(L(t)) \leq |\mathcal{Q}||A|^{k+1}$ for $\forall t$, where A denotes the maximum queue length.

Case i) $Q(t) \geq Ty_Q$.

From (2.37),

$$L_Q(t+T) - L_Q(t) \leq (Q(t) + T(x_Q - y_Q))^{k+1} - (Q(t))^{k+1}.$$

Case i - a) $Q(t) > 2kMT$.

Let $\alpha = T(x_Q - y_Q)$. Since $Q(t) > 2kMT$,

$$\frac{|\alpha|}{Q(t)} = \frac{|T(x_Q - y_Q)|}{Q(t)} \leq \frac{T|\max(x_Q, y_Q)|}{Q(t)} \leq \frac{TM}{2kMT} = \frac{1}{2k}. \tag{2.39}$$

$$\begin{aligned} L_Q(t+T) &\leq (Q(t)+T(x_Q-y_Q))^{k+1} \\ &= Q(t)^{k+1}(1+\frac{T(x_Q-y_Q)}{Q(t)})^{k+1} \\ &= Q(t)^{k+1}(1+\frac{\alpha}{Q(t)})^{k+1} \\ &= Q(t)^{k+1}\left(\frac{1}{1-(\frac{\frac{\alpha}{Q(t)}}{1+\frac{\alpha}{Q(t)}})}\right)^{k+1}. \end{aligned}$$

From (2.39), $|\alpha|/Q(t) \leq 1/(2k)$ and thus

$$1-(k+1)(\frac{\frac{\alpha}{Q(t)}}{1+\frac{\alpha}{Q(t)}})>0. \tag{2.40}$$

Then,

$$\begin{aligned} L_Q(t+T) &\leq Q(t)^{k+1}\left(\frac{1}{1-(\frac{\frac{\alpha}{Q(t)}}{1+\frac{\alpha}{Q(t)}})}\right)^{k+1} \\ &\leq Q(t)^{k+1}\left(\frac{1}{1-(k+1)(\frac{\frac{\alpha}{Q(t)}}{1+\frac{\alpha}{Q(t)}})}\right) \\ &= Q(t)^{k+1}(\frac{Q(t)+\alpha}{Q(t)-k\alpha}) \\ &= Q(t)^{k+1}(1+\frac{(k+1)\alpha}{Q(t)-k\alpha}). \end{aligned}$$

Second inequality comes from $\frac{1}{(1-x)^t} \leq \frac{1}{1-tx}$ for $0 \leq x < \frac{1}{t}$ and (2.40).

From (2.39), $Q(t) - k\alpha \geq Q(t) - k|\alpha| \geq Q(t)/2$ and thus,

$$\begin{aligned} L_Q(t+T) &\leq Q(t)^{k+1}(1 + \frac{(k+1)\alpha}{Q(t) - k\alpha}) \\ &\leq Q(t)^{k+1} + 2(k+1)\alpha Q(t)^k \\ &= Q(t)^{k+1} + 2(k+1)T(x_Q - y_Q)Q(t)^k \\ &= Q(t)^{k+1} - 2D_Q(t). \end{aligned}$$

Therefore,

$$L_Q(t+T) - L_Q(t) \leq -2D_Q(t).$$

Case i - b) $Q(t) \leq 2kMT$.

Since $Q(t) \leq 2kMT$ and $|T(x_Q - y_Q)| \leq TM$,

$$|D_Q(t)| = |(k+1)T(y_Q - x_Q)Q(t)^k| \leq (k+1)(MT)^{k+1}(2k)^k. \tag{2.41}$$

$$\begin{aligned} L_Q(t+T) - L_Q(t) &\leq (Q(t) + T(x_Q - y_Q))^{k+1} - Q(t)^{k+1} \\ &= (k+1)T(x_Q - y_Q)Q(t)^k + \sum_{i=2}^{k+1} \binom{k+1}{i} (T(x_Q - y_Q))^i Q(t)^{k+1-i} \\ &\leq -D_Q(t) + \sum_{i=2}^{k+1} \binom{k+1}{i} (MT)^i (2kMT)^{k+1-i} \\ &= -D_Q(t) + (MT)^{k+1}((1+2k)^{k+1} - (2k)^{k+1} - (k+1)(2k)^k) \\ &\leq -2D_Q(t) + (MT)^{k+1}((1+2k)^{k+1} - (2k)^{k+1} - (k+1)(2k)^k) \\ &\quad +(k+1)(MT)^{k+1}(2k)^k \\ &= -2D_Q(t) + C, \end{aligned}$$

where $C = (MT)^{k+1}((1+2k)^{k+1} - (2k)^{k+1})$. First inequality is from (2.37) and second inequality comes from $|T(x_Q - y_Q)| \leq TM$ and $Q(t) \leq 2kMT$. Third inequality is

from (2.41).

Since $C > 0$, from cases i) - a) and i) - b),

$$L_Q(t+T) - L_Q(t) \leq -2D_Q(t) + C.$$

Case ii) $Q(t) < Ty_Q$.

Since $|T(y_Q - x_Q)| \leq TM$ and $Q(t) < Ty_Q \leq TM$,

$$|D_Q(t)| = |(k+1)T(x_Q - y_Q)Q(t)^k| \leq (k+1)TM(Ty_Q)^k \leq (k+1)(TM)^{k+1}. \quad (2.42)$$

$$\begin{aligned} L_Q(t+T) - L_Q(t) &\leq (Tx_Q)^{k+1} - (Q(t))^{k+1} \\ &\leq (MT)^{k+1} \\ &\leq -2D_Q(t) + (2k+3)(TM)^{k+1}. \end{aligned}$$

First and second inequalities are derived from (2.37) and $x_Q \leq M$, respectively. The last inequality is from (2.42).

Since $C = (MT)^{k+1}((1+2k)^{k+1} - (2k)^{k+1}) > (2k+3)(TM)^{k+1}$ for $k > 1$, in this case, we also derive

$$L_Q(t+T) - L_Q(t) \leq -2D_Q(t) + C.$$

From cases i) and ii), we have shown that

$$L_Q(t+T) - L_Q(t) \leq -2D_Q(t) + C$$

for $\forall Q \in \mathcal{Q}$.

Then,

$$\begin{aligned} E\{L(t+T) - L(t)\} &\leq \sum_{Q\in\mathcal{Q}} E\{L_Q(t+T) - L_Q(t)\} \\ &\leq \sum_{Q\in\mathcal{Q}} E\{C - 2D_Q(t)\} \\ &= |\mathcal{Q}|C - 2D(t), \end{aligned}$$

where

$$D(t) = E\{\sum_{Q\in\mathcal{Q}} D_Q(t)\} = E\{\sum_{Q\in\mathcal{Q}} (k+1)T(y_Q - x_Q)Q(t)^k\}. \tag{2.43}$$

Since input rates $(r_c + \epsilon)$ are feasible on the original network for some $\epsilon > 0$ and there is one to one correspondence between any sequence of events in the original coding network and the sequence of events through generalized link on the generalized network from condition 2.12, as in [26, 27, 57], there exists some value ζ of the vector of flow variables that satisfies

$$y_Q - x_Q = \begin{cases} \epsilon & \text{if } Q = U_{s_c}^c \text{ for some } c \\ 0 & \text{Otherwise.} \end{cases} \tag{2.44}$$

For a randomized policy which does power allocation and scheduling based on this solution vector ζ, analogously to the algorithm of [26, 27], from (2.43) we have

$$D_{rand}(t) = (k+1)T\epsilon \sum_c (U_{s_c}^c(t))^k. \tag{2.45}$$

Next, we consider our back-pressure algorithm. We can rewrite $D(t)$ as follows :

$$\begin{aligned} D(t) &= \sum_{Q\in\mathcal{Q}}(k+1)T(y_Q - x_Q)Q(t)^k \\ &= \sum_{e\in\mathcal{E}'}\sum_{(\mathcal{O},\mathcal{D})\in P_e} r^t_{(\mathcal{O},\mathcal{D})}(\sum_{Q\in\mathcal{O}} L'_Q(t) - \sum_{Q\in\mathcal{D}} L'_Q(t)) \\ &= \sum_{e\in\mathcal{E}'}\sum_{(\mathcal{O},\mathcal{D})\in P_e} r^t_{(\mathcal{O},\mathcal{D})} w^t_{(\mathcal{O},\mathcal{D})}, \end{aligned}$$

where $r^t_{(\mathcal{O},\mathcal{D})}$ denotes the transmission rate of $(\mathcal{O},\mathcal{D})$ at time t.

Since the back-pressure algorithm maximizes $D(t)$,

$$D_{back-pressure}(t) \geq D_{rand}(t). \tag{2.46}$$

Then from (2.45) and (2.46), we obtain

$$\begin{aligned} E\{L(t+T) - L(t)\} &\leq |\mathcal{Q}|C - 2D_{back-pressure}(t) \\ &\leq |\mathcal{Q}|C - (k+1)T\epsilon\sum_c (U^c_{s_c}(t))^k. \end{aligned}$$

If $U^c(t) = 0$ for all c, then

$$L(t) = \sum_{Q\in\mathcal{Q}} Q(t)^{k+1} \leq |\mathcal{Q}||A|^{k+1},$$

where A denotes the maximum queue length defined in Lemma 2.14.

Otherwise, if $U^c(t) > 0$ for some c, then $U^c_{s_c}(t) = V$, and

$$E\{L(t+T) - L(t)\} \leq |\mathcal{Q}|C - (k+1)TV^k\epsilon.$$

Setting

$$V = \left(\frac{|\mathcal{Q}|C}{(k+1)T\epsilon}\right)^{\frac{1}{k}} = \left(\frac{|\mathcal{Q}|(MT)^{k+1}((1+2k)^{k+1}-(2k)^{k+1})}{(k+1)T\epsilon}\right)^{\frac{1}{k}} \tag{2.47}$$

gives $E\{L(t+T)-L(t)\} \leq 0$. By induction on the number of time slots, we obtain

$$E\{L(t)\} \leq |\mathcal{Q}||A|^{k+1}$$

for all t and the queues are stable. ■

Before concluding this section, we consider the choice of $k+1$, which is the degree of potential function. Though our stability proof works for any integer k, we want to choose k that minimizes the upper bound on the queue length defined in the proof of Lemma 2.14. From (2.47), the maximum length of the source queue is

$$V < 2kMT(\frac{k|\mathcal{Q}|}{\epsilon})^{\frac{1}{k}} \leq 4kMT(\frac{|\mathcal{Q}|}{\epsilon})^{\frac{1}{k}}, \tag{2.48}$$

where the first inequality comes from $(1+2k)^{k+1}-(2k)^{k+1} < k(k+1)(2k)^k$ and the second inequality comes from $k^{\frac{1}{k}} \leq 2$ for any integer k.

For the reverse carpooling and star-coding problem, and the multicast with intra-session network coding problem where every transmission is from one-queue to one-queue over real link, from (2.25), the upper bound on any queue length is $A = V+(N-1)MT = 4kMT(\frac{|\mathcal{Q}|}{\epsilon})^{\frac{1}{k}}+(N-1)MT$. By differentiating the above equation, A is minimized when $k = \ln(\frac{|\mathcal{Q}|}{\epsilon})$. Thus we choose $k = \lceil \ln(\frac{|\mathcal{Q}|}{\epsilon}) \rceil$ or $\lfloor \ln(\frac{|\mathcal{Q}|}{\epsilon}) \rfloor$. In this case, when $T = 1$, the upper bound on any queue length is

$$A = 4\ln(\frac{|\mathcal{Q}|}{\epsilon})Me + (N-1)M. \tag{2.49}$$

We can compare this bound with the bound presented in [27] where $k = 1$ is chosen. From the proof of Theorem 8a in [27], when N is sufficiently large, the bound on the

queue length A' is given as follows :

$$A' = \frac{NM^2\tau_{\max}}{\epsilon} + (N-1)M \tag{2.50}$$

where $\tau_{\max}$ is the maximum number of sinks in any multicast session. From (2.49) and (2.50), $A \leq A'$ if $4\ln(\frac{|\mathcal{Q}|}{\epsilon})e < \frac{NM\tau_{\max}}{\epsilon}$. When each node can have at most C_1 queues, i.e., $|\mathcal{Q}| < C_1 N$, and $M > 1$, for sufficiently large N, we obtain $4\ln(\frac{C_1 N}{\epsilon})e < \frac{N}{\epsilon} < \frac{NM\tau_{\max}}{\epsilon}$ and thus $A \leq A'$.

For the pairwise XOR coding problem, from (2.36), the upper bound on the queue length is

$$\begin{aligned} A &= 2^{\frac{1}{k}}(V + (N-1)MT) + (N+3)MT \\ &= 2^{\frac{1}{k}}(4kMT(\frac{|\mathcal{Q}|}{\epsilon})^{\frac{1}{k}} + (N-1)MT) + (N+3)MT. \end{aligned} \tag{2.51}$$

We choose k that minimizes (2.51) by solving numerically. When $\frac{|\mathcal{Q}|}{\epsilon}$ is sufficiently large, $k = \ln(\frac{2|\mathcal{Q}|}{\epsilon})$. In this case,

$$A \leq 4\ln(\frac{2|\mathcal{Q}|}{\epsilon})Me + (3N-1)M. \tag{2.52}$$

We can also compare this bound (2.52) with the bound presented in [27] where $k = 1$ is chosen. From the proof of Theorem 3 in [26], when N is sufficiently large, the upper bound on the queue length is

$$A'' = \frac{18N^2M^2}{\epsilon} + 5M. \tag{2.53}$$

Therefore, $A \leq A''$ if

$$4\ln(\frac{2|\mathcal{Q}|}{\epsilon})e + (3N-6) < \frac{18N^2M}{\epsilon}. \tag{2.54}$$

When each node can have at most C_1 queues, and $M > 1$, for sufficiently large

N, we obtain

$$4\ln(\frac{2C_1N}{\epsilon})e + (3N-6) < \frac{2N}{\epsilon} + 3N - 6 < \frac{18N^2M}{\epsilon}. \tag{2.55}$$

Thus $A \leq A''$.

As shown above, our queue length bound is $O(\ln(\frac{N}{\epsilon}))$ while bound in [26] is $O(\frac{N^2}{\epsilon})$.

2.4.5 Conclusion

We have proposed a back-pressure based distributed optimization framework with network coding. Our framework can be used for optimizing over any class of network codes, including pairwise XOR coding problem, and reverse carpooling and star coding problem. Our approach is to specify the class of coding operations by a set of generalized links, and to develop optimization tools that apply to any network composed of such generalized links. We propose a dynamic back-pressure algorithm in which routing, network coding, and scheduling decisions are made locally by comparing the weights of transmission pairs on each generalized link. We first present the upper bound on the queue length for each optimization problem and prove that our algorithm achieves the stability for any input rates within the capacity region when there is a one to one correspondence between any sequence of events in the original coding network and the sequence of events through generalized link on the generalized network. In previous works, sum of the square of the queue length was used as potential function. In our framework, the stability proof works for any degree of potential function and we can choose the degree of potential function that minimizes the upper bound on the queue length. Moreover, our minimized upper bound on the queue length improves the previous bound.

Chapter 3

Network error correction with unequal link capacities

3.1 Introduction

In this chapter, we study the capacity of single-source single-sink noiseless networks under adversarial attack on no more than z edges. Unlike prior papers, which assume equal capacities on all links, we allow arbitrary link capacities. The unequal link capacity problem is substantially different from the problem studied by Yeung and Cai in [30,31] since the rate controlled by the adversary varies with his edge choice. For the equal link capacities case, coding only at the source and forwarding at intermediate nodes suffices to achieve the capacity for any single-source and single-sink network. In contrast, for networks with unequal link capacities, we show that network error correction is needed even for a single-source and single-sink network.

In Section 3.3, we propose a new cut-set upper bound which applies to general acyclic networks. For networks with unequal link capacities, this bound tightens the generalized Singleton bound given in our earlier work in [43] and independently in [42]. In Section 3.4, we consider a variety of linear and nonlinear coding strategies useful for achieving the capacity of the example networks. We present a method for upper bounding the linear coding capacity of an arbitrary network and prove the insufficiency of linear network codes to achieve the capacity. The proof provides an example for which the capacity achieved using nonlinear error detection at an inter-

mediate node is 50% greater than the linear coding capacity. A similar example for the problem with Byzantine attack on nodes rather than edges appears in [42]. We also give examples of single-source and single-sink networks for which intermediate nodes must perform coding, nonlinear error detection or error correction in order to achieve the network capacity. We describe a simple greedy algorithm for error correction at intermediate nodes. We then introduce a new coding strategy called "guess-and-forward." In this strategy, an intermediate node which receives some redundant information from multiple paths guesses which of its upstream links is controlled by the adversary. The intermediate node forwards its guess to the sink which tests the hypothesis of the guessing node. Section 3.5 investigates this strategy with a variety of example networks. We show that guess-and-forward achieves network capacity on the two-node network with multiple feedback links, as well as the proposed family of four-node acyclic networks. Finally, we apply guess-and-forward strategy to the zig-zag networks, deriving a general achievable rate region and presenting conditions under which that bound is tight.

3.2 Preliminaries

Consider any single-source and single-sink acyclic communication network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with unequal link capacities. Source node s transmits information to the sink node u. For each node $v \in \mathcal{V}$, we use $\gamma_+(v) = \{(c, v) : (c, v) \in \mathcal{E}\}$ and $\gamma_-(v) = \{(v, c) : (v, c) \in \mathcal{E}\}$ to denote the sets of incoming and outgoing edges for node v. Let $r(l)$ denote the capacity of edge $l \in \mathcal{E}$. We assume that the code alphabet $\mathcal{X}$ is equal to $GF(q)$ for some prime power q. An error vector on any link $l \in \mathcal{E}$ is a vector e_l containing $r(l)$ symbols in code alphabet $\mathcal{X}$. The output y_l of link l equals the modulo q sum of the input x_l to link l and the error e_l applied to link l. We say that τ error links occur in the network if $e_l \neq 0$ on τ links.

Definition 3.1 *A network code is z-error link-correcting if the source message can be recovered by the sink node provided that the adversary controls at most z links. Thus a z-error link-correcting network code can correct any τ adversarial links for*

$\tau \leq z$.

As in [30, 31], we can consider a linear network code V for a single-source and a single-sink network that assigns a linear subspace $L_v(a)$ to each node $a \in V$ and a set of $r(l)$ column vectors $\{v^\tau(l)_1, v^\tau(l)_2, .., v^\tau(l)_{r(l)}\}$ to each link $l \in \mathcal{E}$ in the network. Denote by G_a the matrix whose columns are the vectors assigned to the input links of node a. For any linear network code V, there exists a set of $r(l)$ column vectors $\{c^\tau(l)_1, c^\tau(l)_2, .., c^\tau(l)_{r(l)}\}$ such that $v^\tau(l)_i = G_a c^\tau(l)_i$. Then we can define a linear network code ϕ based on any linear network code V as in [31]. Let

$$\tilde{\phi}_l(w) = \{\langle w, v^\tau(l)_i \rangle : 1 \leq i \leq r(l)\}$$

denote the error-free output of link l when the network input is w. We again use vector e_l to denote the errors on link l and $e = (e_l : l \in \mathcal{E})$ to denote the entire network error. If an error vector e occurs, its components are added to the link inputs. Then the output of a link l is a function of both the network input w and the error vector e. We denote that output by $\psi_l(w, e)$. With this notation, a sink node u cannot distinguish between the case where w is the network input and error e occurs and the case where w' is the network input and error e' occurs if and only if

$$(\psi_l(w, e) : l \in \Gamma_+(u)) = (\psi_l(w', e') : l \in \Gamma_+(u)). \tag{3.1}$$

Let $N(e) = |\{l \in \mathcal{E} : e_l \neq 0\}|$ denote the number of links in which an error occurs. We say that any pair of input vectors w and w' are z links separable at sink node u if (3.1) does not hold for any pair of error vectors e and e' such that $N(e) \leq z$ and $N(e') \leq z$. [31, Lemma 1] establishes the linear properties of $\psi_l(w, e)$ for networks with unit link capacities. This results extends directly to networks with arbitrary link capacities.

Lemma 3.2 *For all $l \in \mathcal{E}$, all network inputs w and w', error vectors e and e', and $\mu \in GF(q)$,*

$$\psi_l(w + w', e + e') = \psi_l(w, e) + \psi_l(w', e')$$

and

$$\psi_l(\mu w) = \mu \psi_l(w).$$

From Lemma 3.2,

$$\psi_l(w, e) = \psi_l(w, 0) + \psi_l(0, e) = \tilde{\phi}_l(w) + \theta_l(e).$$

Thus $\psi_l(w, e)$ can be written as the sum of a linear function of w and a linear function of e.

Let (A, B) be a partition of $\mathcal{V}$, and define the cut for the partition (A, B) by

$$cut(A, B) = \{(a, b) \in \mathcal{E} : a \in A, b \in B\}.$$

The quantity $m(A) = \sum_{(a,b) \in cut(A,B)} r(a, b)$ is called the volume of $cut(A, B)$. The cut $cut(A, B)$ separates nodes a and b if $a \in A$ and $b \in B$. We use $CS(a, b)$ to denote the set of cuts between a and b and $c(a, b)$ to denote the minimum volume of a cut between a and b.

3.3 Upper bound

First, we state the generalized Singleton upper bound which is presented in [43]. A similar upper bound for the problem of adversarial attack on nodes rather than edges appears independently in [42]. We then propose a new cut-set upper bound, which can be applied for general acyclic networks. For networks with unequal link capacities, this bound tightens the generalized Singleton bound. As examples, we analyze these bounds for both the four-node acyclic network and the zig-zag network.

Lemma 3.3 *(Generalized Singleton bound) Consider any z-error correcting network code with source alphabet X in an acyclic network $\mathcal{G}$. Consider any set S consisting of $2z$ links on a source-sink cut Q such that none of the remaining links on Q are*

downstream of any link in S. Let M be the total capacity of the remaining links. Then

$$\log |X| \leq M \cdot \log q.$$

Proof. We assume that $|X| > q^M$, and show that this leads to a contradiction.

For brevity, let $Q = \{l_1, .., l_{K(Q)}\}$ where $S = \{l_{K(Q)-2z+1}, ...l_{K(Q)}\}$ and links in S are in the coding order of the given network code. Since $|X| > q^M$, from the definition of M, there exist two distinct symbols $x, x' \in X$ such that $\phi'_{l_i}(x) = \phi'_{l_i}(x')$ $\forall i = 1, .., K(Q) - 2z$. So we can write

$$O(x) = \{y_1, .., y_{K(Q)-2z}, u_1, .., u_z, w_1, .., w_z\},$$

$$O(x') = \{y_1, .., y_{K(Q)-2z}, u'_1, .., u'_z, w'_1, .., w'_z\}.$$

We will show that it is possible for the adversary to produce exactly the same outputs at all the channels on Q when errors are occurred at most z links on Q.

Assume the input of the network is x. The adversary will inject errors on z links $l_{K(Q)-2z+1}, \ldots, l_{K(Q)-z}$ in this order as follows. First the adversary applies an error on link $l_{K(Q)-2z+1}$ to change the output from u_1 to u'_1. Then the output of links $(l_{K(Q)-2z+2}, .., l_{K(Q)})$ may be affected, but not the outputs of links $(l_1, .., l_{K(Q)-2z})$. Let $u'_i(j)$ and $w'_i(j)$ denote the outputs of links $l_{K(Q)-2z+i}$ and $l_{K(Q)-z+i}$, respectively after the adversary has injected errors on link $l_{K(Q)-2z+j}$, where $j = 1, 2, .., t$ with $u'_1(1) = u'_1$. Then the adversary injects errors on link $l_{K(Q)-2z+2}$ to change its output from u'_1 to u_2. This process continues until the adversary finishes injecting errors on z links $l_{K(Q)-2z+1}, .., l_{K(Q)-z}$ and the output of this channel changes from $O(x)$ to $\{y_1, .., y_{K(Q)-2z}, u'_1, .., u'_z, w'_1(t), .., w'_z(t)\}$. Now suppose the input is x'. We can follow a similar procedure by injecting errors on z links $l_{K(Q)-z+1}, .., l_{K(Q)}$. Then the adversary can produce the outputs

$$\{y_1, .., y_{K(Q)-2z}, u'_1, .., u'_z, w'_1(t), .., w'_z(t)\}.$$

Thus, sink node u cannot reliably distinguish between the source symbol x and x', which gives a contradiction. ■

Any set of links S on the cut Q is said to satisfy the *downstream condition* if none of the remaining links on the cut Q are downstream of any link in S.

Given a $Q = cut(P, \mathcal{V} - P)$, let Q^R denote the set of feedback links of the cut. Given a set of $m \leq z$ feedback links $W \subset Q^R$ and a set of $k \leq z - m$ forward links $F \subset Q$, we use $N^{F,W}_{z,m,k}(Q)$ to denote the upper bound obtained from Lemma 3.3 with $z - m - k$ adversarial links on the cut Q after erasing W and F from the graph $\mathcal{G}$. Let

$$N_{z,k,m}(Q) = \min_{\{F \subset Q, |F|=k \leq z-m\}} \min_{\{W \subset Q^R, |W|=m \leq z\}} N^{F,W}_{z,k,m}(Q).$$

Then we define $N_z(Q)$ as follows.

$$N_z(Q) = \min_{0 \leq k \leq z-m} \min_{0 \leq m \leq z} N_{z,k,m}(Q).$$

Lemma 3.4 *Consider any z-error correcting network code with source alphabet X in an acyclic network.*

$$\log |X| \leq \min_{Q \in CS(s,u)} \{N_z(Q)\} \cdot \log q$$

Proof. For any cut $Q \in CS(s, u)$, the adversary can choose to erase a set $W \subset Q^R$ feedback links and a set $F \subset Q$ of forward links where $|W| = m \leq z$ and $|F| = k \leq z - m$. Applying Lemma 3.3 on Q after erasing W and F gives the upper bound $N^{F,W}_{z,k,m}(Q)$. By taking the minimum over all such cuts, we obtain the above bound. ■

The following examples illustrate how above upper bound tightens the generalized Singleton bound. We first consider a four-node acyclic network as shown in Fig. 3.1. In each example, unbounded reliable communication is allowed from source S to its neighbor B on one side of the cut and from node A to sink U on the other side of the cut, respectively. There are feedback links with arbitrary capacities from A to B.

When we compute the generalized Singleton bound, for any cut Q, we choose and erase $2z$ links in the cut such that none of the remaining links in the cut are downstream of chosen $2z$ links. Then we sum the remaining link capacities and take

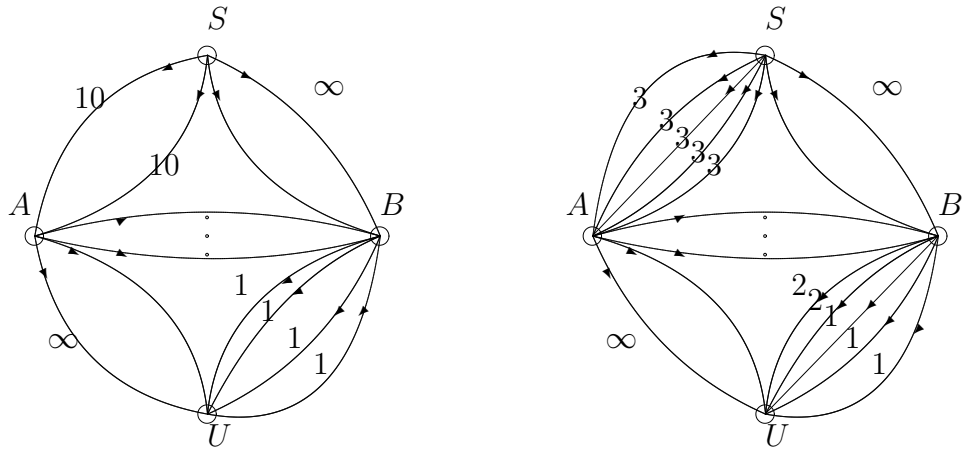


Figure 3.1: Four-node acyclic network: unbounded reliable communication is allowed from source S to its neighbor B on one side of the cut and from node A to sink U on the other side of the cut, respectively. (a) There are 2 links of the capacity 10 from S to A and 4 unit-capacity links from B to U. (b) There are 5 links of the capacity 3 from S to A. There are 2 links of the capacity 2 and 3 links of the capacity 1 from B to U.

the minimum over all cuts. Because of the downstream condition, when the link capacities between S and A are much larger than the link capacities between B and U as shown in Fig. 3.1 (a), Singleton bound may not be tight. When $z = 2$, the generalized Singleton bound gives upper bound 20. However, when the adversary declares that he will use two forward links between S and A, we obtain the erasure bound 4.

We consider the network in Fig. 3.1 (b). Suppose that $z = 2$. Applying the generalized Singleton bound gives upper bound 16. If the adversary erases one of the forward links between S and A and we apply the generalized Singleton bound on the remaining network, then our upper bound is improved to 15. The intuition behind this example is that when the adversary erases $p \leq z$ large capacity links which do not satisfy the downstream condition, applying the generalized Singleton bound on remaining network with $(z - p)$ adversarial links can give tighter bound.

For the 2-layer zig-zag network in Fig. 3.2, when $z = 4$, min-cut is 37 and the generalized Singleton bound gives upper bound 27. Suppose that the adversary declares

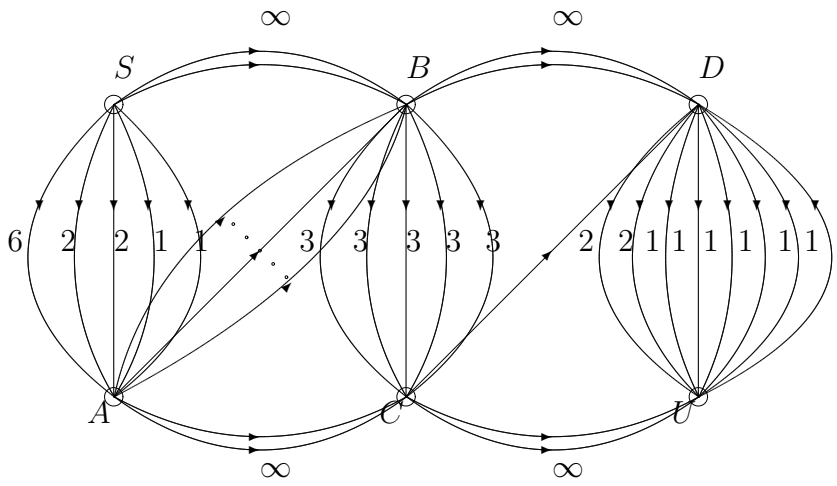


Figure 3.2: 2-layer zig-zag network: unbounded reliable communication is allowed from S to B, from B to D, from A to C, and from C to U respectively. There is a sufficiently large number of feedback links from A to B. There is one feedback link from C to D.

that he will use feedback link between C and D, and forward link with capacity 6 between S and A. By applying the generalized Singleton bound on remaining network with two adversarial links, we obtain 37-6-(3+3+3+3)=19. The intuition behind this example is that the links between B and C and the links between D and U have the same topological order by erasing the single feedback link between C and D. Since the generalized Singleton bound is obtained by erasing $2z$ links on the cut such that none of the remaining links on the cut are downstream of any erased links, by erasing the single feedback link between C and D, we can have tighter Singleton bound even with a fewer number of adversarial links. Moreover, before applying Singleton bound, we first erase the link with capacity 6 which is the largest between S and A as we did in example in Fig. 3.1(b).

Now we introduce another cut-set upper bound. For any cut $Q = (P, \mathcal{V} - P)$ and a set of nodes $A \subseteq \mathcal{V} - P$, let $F_A(Q) = \{(a, b) \in \mathcal{E} : a \in P, b \in A\} \subset Q$ and $W_A(Q) = \{(a, b) \in \mathcal{E} : a \in A, b \in P\} \subset Q^R$ to denote the set of all forward and feedback links incident to nodes in A, respectively. Let $|W_A(Q)| = m_A(Q)$.

Suppose for a cut Q, there exists a set of nodes $A \subseteq \mathcal{V} - P$ such that $m_A(Q) \leq z$. For any $k \leq z - m_A(Q)$, choose any set of k links $P_A(Q) \subseteq F_A(Q)$. Then choose any set of $z-k-m_A(Q)$ links $R_A(Q) \subset Q-P_A(Q)$ that satisfies the downstream condition on Q. Let $Z_A(Q) = P_A(Q) \cup R_A(Q)$. Similarly, for any set of nodes $B \subseteq \mathcal{V} - P - A$ such that $m_B(Q) \leq z$, choose any set of $p \leq z - m_B(Q)$ links $P_B(Q) \subseteq F_B(Q)$ and a set of $z-p-m_B(Q)$ links $R_B(Q) \subseteq Q-Z_A(Q)-P_B(Q)$ that satisfies the downstream condition on Q. $Z_B(Q) = P_B(Q) \cup R_B(Q)$.

Lemma 3.5 *Let M denote the total capacity of the remaining links on $Q - Z_A(Q) - Z_B(Q)$. Then,*

$$\log |X| \leq M \cdot \log q.$$

Proof. We assume that $|X| > q^M$, and show that this leads to a contradiction. Let $K(Q)$ denote the number of links on the cut Q. Since $|X| > q^M$, from the definition of M, there exist two distinct codewords $x, x' \in X$ such that error-free outputs on the links in $Q - Z_A(Q) - Z_B(Q)$ are the same. Let $a = |Z_A(Q)|$ and $b = |Z_B(Q)|$. So we can write

$$O(x) = \{y_1, .., y_{K(Q)-a-b}, u_1, .., u_a, w_1, .., w_b\},$$

$$O(x') = \{y_1, .., y_{K(Q)-a-b}, u'_1, .., u'_a, w'_1, .., w'_b\},$$

where $(y_1, .., y_{K(Q)-a-b})$ denotes the error-free outputs on the links in $Q - Z_A(Q) - Z_B(Q)$, $(u_1, .., u_k)$ and $(u'_1, .., u'_k)$ denote the error-free outputs on the links in $P_A(Q)$ for x and x' respectively, and $(u_{k+1}, .., u_a)$ and $(u'_{k+1}, .., u'_a)$ denote the error-free outputs on the links in $R_A(Q)$ for x and x' respectively. Similarly, let $(w_1, .., w_p)$ and $(w'_1, .., w'_p)$ denote the error-free outputs on the links in $P_B(Q)$ for x and x' respectively, and let $(w_{p+1}, .., w_b)$ and $(w'_{p+1}, .., w'_b)$ denote the error-free outputs on the links in $R_B(Q)$ for x and x' respectively. We will show that it is possible for the adversary to produce exactly the same outputs at all the channels on Q when errors occur on at most z links. When codeword x is sent, we use $B_l(x)$ to denote the error-free output on feedback link l.

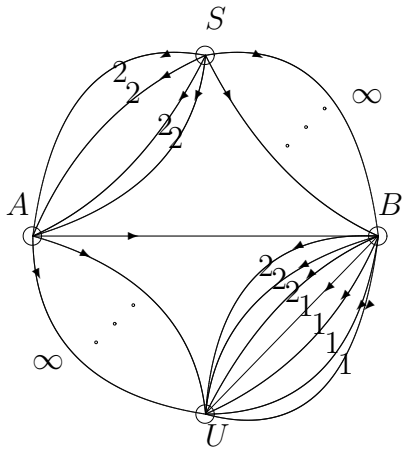


Figure 3.3: Four node acyclic network: There are 4 links of the capacity 2 from S to A. There are 3 links of the capacity 2 and 1 link of the capacity 4 from B to U.

Assume the input of network is x. The adversary chooses feedback links set $W_A(Q)$ and forward links set $Z_A(Q)$ as z adversarial links. First the adversary applies errors on $P_A(Q)$ to change the output from u_i to u'_i for $\forall 1 \leq i \leq k$ and causes each feedback link $l \in W_A(Q)$ to transmit $B_l(x)$. Since all feedback links in $W_A(Q)$ transmit the error-free output, when the output u_i on any link in $P_A(Q)$ is changed, the outputs of downstream links of it are not affected. Thus $(u_1, .., u_k)$ is changed to $(u'_1, .., u'_k)$ without affecting the outputs of any other links. Then the adversary applies errors on $R_A(Q)$ to change the output from $(u_{k+1}, .., u_a)$ to $(u'_{k+1}, .., u'_a)$. Since the links in $R_A(Q)$ satisfy downstream condition, the outputs of any other links are not affected. Sink finally observes $\{y_1, .., y_{K(Q)-a-b}, u'_1, .., u'_a, w_1, .., w_b\}$.

When codeword x' is transmitted, the adversary chooses feedback links set $W_B(Q)$ and forward links set $Z_B(Q)$ as z adversarial links. The adversary applies errors on them to change $(w_1, .., w_b)$ to $(w'_1, .., w'_b)$ without affecting the outputs on other links as shown above. Then output is changed from $O(x')$ to $\{y_1, .., y_{K(Q)-a-b}, u'_1, .., u'_a, w_1, .., w_b\}$. Thus, the sink node u cannot reliably distinguish between the codewords x and x', which gives a contradiction. ■

Given a cut Q, we consider all possible sets $(Z_A(Q), Z_B(Q))$ on the Q satisfying the condition on Lemma 3.5. We choose sets $(Z_A(Q)^*, Z_B(Q)^*)$ among them that have the maximum total link capacities and define $M_z(Q)$ to be the sum of the capacities of the links on Q which are not in $(Z_A(Q)^*, Z_B(Q)^*)$. This gives the upper bound

$$\log |X| \leq \min_{Q \in cut(s,u)} M_z(Q) \cdot \log q.$$

The following example shows that we can obtain tighter upper bound using Lemma 3.5. For example network in Fig. 3.3, when $z = 3$, Lemma 3.4 gives upper bound 9. However, Lemma 3.5 gives tighter upper bound 8 when $Z_A(Q)^*$ is composed of 2 forward links of the capacity 2 and feedback link from A to B, and $Z_B(Q)^*$ is composed of 3 forward links of the capacity 2.

Now we derive the generalized cut-set upper bound that unifies Lemma 3.4 and Lemma 3.5. Given a cut Q, we choose the set of $m \leq z$ feedback links $W \subset Q^R$ and the set of $k \leq z - m$ forward links $F \subset Q$. We use $C^{F,W}_{z,m,k}(Q)$ to denote the upper bound obtained from Lemma 3.5 with $z - m - k$ adversarial links on the cut Q after erasing W and F from original graph $\mathcal{G}$. Let

$$C_{z,k,m}(Q) = \min_{\{F \subset Q, |F| = k \leq z-m\}} \min_{\{W \subset Q^R, |W| = m \leq z\}} C^{F,W}_{z,k,m}(Q).$$

Then we define $C_z(Q)$ as follows.

$$C_z(Q) = \min_{0 \leq k \leq z-m} \min_{0 \leq m \leq z} C_{z,k,m}(Q).$$

This gives the following upper bound.

Theorem 3.6 *(Generalized cut-set upper bound) Consider any z-error correcting network code with source alphabet X in an acyclic network.*

$$\log |X| \leq \min_{Q \in CS(s,u)} C_z(Q) \cdot \log q$$

3.4 Coding strategies

We consider a variety of linear and nonlinear coding strategies useful for achieving the capacity of the example networks. We show the insufficiency of linear network codes for achieving the capacity in general, by providing a method for upper bounding the linear coding capacities of an arbitrary network. We also demonstrate examples of networks with a single source and a single sink where, unlike the equal link capacity case, it is necessary for intermediate nodes to do coding, nonlinear error detection or error correction in order to achieve the capacity. We then introduce a new coding strategy, guess-and-forward, and show the optimality of this scheme in some examples.

3.4.1 Insufficiency of linear network code

Here we show that there exists a network where the capacity is 50% greater than the best rate that can be achieved with linear coding. We consider the single source and the single sink network in Fig. 3.4, where source s aims to transmit the information to a sink node u. We index the links and assume the capacities of links as shown in Fig. 3.4. For a single adversarial link, our upper bound from Theorem 3.6 is 2.

Lemma 3.7 *Given a network in Fig. 3.4, for a single adversarial link, rate 2 is asymptotically achievable with nonlinear error detection strategy, whereas scalar linear network code achieves at most 4/3.*

Proof. We first illustrate the nonlinear error detection strategy as follows. Source wants to transmit two packets (X, Y). We send them in n channel uses, but each packet has only $n-1$ bits. We use one bit as a signaling bit. We send (X, Y) down all links in the top layer. In the middle layer, we do the following operations:

(1) Send the linear combination of X and Y, $aX + bY$, down link l_4.

(2) Send X down both links l_5 and l_6.

(3) Send Y down both links l_7 and l_8.

(4) Send a different linear combination of X and Y, $cX + dY$, down link l_9.

At the bottom layer, we do the following operations:

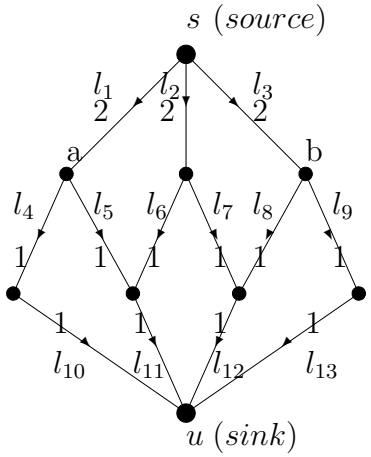


Figure 3.4: A single source and a single sink network: all links on the top layer have capacity 2. All links on the middle and bottom layer have capacity 1. When $z = 1$, the capacity of this network is 2 while linear network codes achieve at most 4/3.

(1) Forward the received packet on link l_{10}.

(2) Send a 1 followed by X on link l_{11} if the two copies of X match, send a 0 otherwise.

(3) Send a 1 followed by Y on link l_{12} if the two copies of Y match, send a 0 otherwise.

(4) Forward the received packet on link l_{13}.

We can show that the above nonlinear error detection strategy allows a sink node to decode (X, Y). Suppose that (a, b) and (c, d) are independent. Then coding vectors on any two links on the bottom layer are independent and they satisfy with MDS (maximum distance separable) properties. If nothing was sent down both l_{11} and l_{12}, the decoder can recover (X, Y) from the information received on links l_{10} and l_{13}. If nothing was sent down only on l_{11}, then the outputs of l_{12} and l_{13} should not be corrupted and the decoder can recover (X, Y). Similarly, the decoder can decode correctly when nothing was sent down only on l_{12}. If all the links in the bottom layer received symbols, there is at most one erroneous link on the bottom layer, which has MDS code. Thus we can achieve rate $2 - \frac{2}{n}$ with error detection strategy.

Now we show that scalar linear network code can achieve at most rate 4/3. Suppose that we want to achieve linear coding capacity k/n by transmitting k symbols reliably by using scalar linear network code ϕ during n time slots. To show the insufficiency of linear coding for achieving this capacity, from (3.1), it is sufficient to prove that there exist pairs (w, e) and (w', e') for linear network code ϕ such that

$$(\psi_l(w, e) : l \in \Gamma_+(u)) = (\psi_l(w', e') : l \in \Gamma_+(u)),$$

$N(e), N(e') \leq 1$. Since the above equation is equivalent to

$$(\tilde{\phi}_l(w - w') : l \in \Gamma_+(u)) = (\theta_l(-e + e') : l \in \Gamma_+(u)),$$

by linearity, it is enough to find x and e'' such that $x \in \mathcal{X}$, $N(e'') \leq 2$, and

$$(\tilde{\phi}_l(x) : l \in \Gamma_+(u)) = (\theta_l(e'') : l \in \Gamma_+(u)), \tag{3.2}$$

where $\mathcal{X}$ denotes a source alphabet and $|\mathcal{X}| = q^k$. We will show that there exists (x, e'') satisfying the above equation when errors occur on the links l_1 and l_3 in error vector e''.

Let M_1 and M_2 denote transfer matrices between a and u, and between b and u during n time slots respectively. To transmit k symbols reliably in this network, both M_1 and M_2 should have rank at least k, i.e., $rank(M_1) \geq k$ and $rank(M_2) \geq k$. Otherwise, when the adversarial link is on the top layer, the maximum achievable rate is at most $\min\{rank(M_1), rank(M_2)\}$ from data processing inequality and it gives us a contradiction.

Let e_1 and e_2 denote the errors occurring on links l_1 and l_3, respectively. Errors on e_1 propagates to l_{10} and l_{11}, and errors on e_2 propagate to l_{12} and l_{13}. We use $4n \times k$ matrix G_u to denote the transfer matrix between s and u during n time slots. Its columns are global coding vectors assigned on l_{10}, l_{11}, l_{12}, and l_{13}.

From (3.7), we have following set of equations

$$G_u x = \begin{pmatrix} M_1 & 0 \\ 0 & M_2 \end{pmatrix} (e_1, e_2)^\tau = M \cdot e''.$$

If $rank(G_u) < k$, then there exists some $x_1 \neq 0$ such that $G_u x_1 = 0$. Then $(x, e_1, e_2) = (x_1, 0, 0)$ satisfies above equation. Actually, this network code is a bad code itself since we cannot distinguish any pair of source messages w and w' such that $w - w' = x_1$ even when there are no error links in the network.

Otherwise, $rank(G_u) = k$. Since $rank(M_1) \geq k$ and $rank(M_2) \geq k$, $rank(M) \geq 2k$. Then $A = \{G_u x : x \in \mathcal{X}\}$ and $B = \{Me'' : e'' \in GF^{4n}(q)\}$ are both linear subspaces of $GF^{4n}(q)$, and $\dim(A) = k$ and $\dim(B) \geq 2k$.

Let $\{x_1, .., x_k\}$ denote the basis of $\mathcal{X}$. Then $\{G_u x_1, .., G_u x_k\}$ is the basis of A. Similarly, since $rank(M) \geq 2k$, there exist $2k$ vectors $\{y_1, .., y_{2k}\}$ such that $\{My_1, .., My_{2k}\}$ is a subset of basis of B.

If $3k > 4n$, since both A and B are linear subspaces of $GF^{4n}(q)$, there exists $(a_1, .., a_k, b_1, .., b_{2k}) \neq (0, ..., 0)$ such that

$$\sum_{i=1}^{k} a_i (G_u x_i) + \sum_{j=1}^{2k} b_i (My_i) = 0.$$

If $(a_1, .., a_k) = (0, ..., 0)$ or $(b_1, .., b_{2k}) = (0, ..., 0)$, then it contradicts the linear independence of basis. Thus, $(a_1, .., a_k) \neq (0, ..., 0)$ and $(b_1, .., b_{2k}) \neq 0$. Then,

$$\begin{aligned} & \sum_{i=1}^{k} a_i (G_u x_i) + \sum_{j=1}^{2k} b_i (My_i) \\ = & \sum_{i=1}^{k} G_u (a_i x_i) + \sum_{j=1}^{2k} M(b_i y_i) \\ = & \sum_{i=1}^{k} G_u (a_i x_i) - \sum_{j=1}^{2k} M(-b_i y_i) \\ = & \ 0. \end{aligned}$$

Therefore, we have found nonzero $x = \sum_{i=1}^{k} a_i x_i$ and $(e_1, e_2)^\tau = -\sum_{j=1}^{2k}(-b_j y_j)$ such that $G_u x = Me'$.

In [42], it is shown that rate 4/3 is achievable asymptotically using simple linear codes. It completes the proof.

■

Corollary 3.8 *Given a network in Fig. 3.4, for a single adversarial link, vector linear network code can achieve at most 4/3.*

Proof. For a network code using vector transmission, the outgoing edges of each node carries vectors of alphabet symbols which are a function of the vectors carried on the incoming edges to the node. We consider a vector linear code that groups m symbols into a vector. As in Theorem 2, we define $(4n)m \times km$ generator matrix G_u between s and u. Transfer matrices M_1 and M_2 are also defined in the same way, and $rank(M_1) \geq km$ and $rank(M_2) \geq km$. As in the proof of Theorem 1, when $k > \frac{4n}{3}$, we can show that there exist vectors (x, e_1, e_2) $(x \neq 0)$ satisfying

$$G_u x = (M_1 \cdot e_1, M_2 \cdot e_2).$$

■

3.4.1.1 Upper bound on the linear coding capacity

We propose a method that gives an upper bound on the linear coding capacity for arbitrary networks. Suppose that we want to transmit k symbols reliably by using scalar linear network code ϕ during n time slots. To show the insufficiency of linear coding for achieving the rate k/n, it is sufficient to find $x \in \mathcal{X}$ and e'' such that $|\mathcal{X}| = q^k$, $N(e'') \leq 2z$ and

$$(\phi_l(x) : l \in \Gamma_+(u)) = (\theta_l(e'') : l \in \Gamma_+(u)). \tag{3.3}$$

Let $\Omega_u = cut\{\mathcal{V} - \{u\}, u\}$ denote the cut between the sink node u and all other nodes. $C_1 = \sum_{l \in \Omega_u} r(l)$ denotes the volume of Ω_u. Suppose that there exists a cut Ω

which contains $p \geq 2z$ links and there are m disjoint sets of links $(L_1, \ldots, L_m)$ such that $2z \geq m(p-2z)$, $L_i \subset \Omega$, $|L_i| = p-2z$, $L_i \cap L_j = \emptyset$, and $\Omega(L_1) \cup \ldots \cup \Omega(L_m) = \Omega_u$ where $\Omega(L_i)$ denotes the set of links in Ω_u such that symbols on L_i can be propagated. We prove that $C_1/(m+1)$ is an upper bound of linear coding capacity by showing that there is (x, e'') that satisfies (3.3) when error vector e'' consists of error links in $(L_1, \ldots, L_m)$.

We use e_i to denote an error vector on L_i. Let $\theta^i(e_i) = (\theta_l(e_i) : l \in \Omega(L_i))$ denote the output on $\Omega(L_i) \subseteq \Omega_u$ given e_i. Given a linear network code ϕ, let M_i denote a transfer matrix between L_i and $\Omega(L_i)$. i.e., $\theta^i(e_i) = M_i \cdot e_i$. To transmit k symbols reliably in this network, M_i should have rank at least k, i.e., $rank(M_i) \geq k$ for $1 \leq i \leq m$. Given an error vector $e'' = (e_1, \ldots, e_m)$ on the cut Ω, since $\theta_l(e) = \sum_{\{j: l \in \Omega(L_j)\}} \theta_l(e_j)$ for $l \in \Omega_u$, we obtain following equation

$$(\theta_l(e'') : l \in \Omega_u) = A \cdot (\theta^1(e_1), \ldots, \theta^m(e_m))^\tau \tag{3.4}$$

which is equivalent to

$$\begin{aligned} & (\theta_l(e'') : l \in \Omega_u) \\ = \; & A \begin{pmatrix} M_1 & .. & .. & 0 \\ 0 & M_2 & .. & .. \\ .. & .. & .. & .. \\ 0 & .. & 0 & M_m \end{pmatrix} (e_1, .., e_m)^\tau \\ = \; & A \cdot M \cdot (e'')^\tau. \end{aligned}$$

Here a matrix A depends on the graph topology. For instance, when $L_1 = l_1$ and $L_2 = l_3$ in Fig. 3.4, $\Omega(L_1) \cup \Omega(L_2) = \Omega_u$ and $\Omega(L_1) \cap \Omega(L_2) = \emptyset$. Since $M \cdot (e'')^\tau = (\theta_{l_{10}}(e_1), \theta_{l_{11}}(e_1), \theta_{l_{12}}(e_2), \theta_{l_{13}}(e_2))$,and $\theta_l(e'') = \theta_l(e_1)$ for $l \in \{l_{10}, l_{11}\}$ and $\theta_l(e'') = \theta_l(e_2)$ for $l \in \{l_{12}, l_{13}\}$, $A = I_{4n}$. Since we assume that errors on $(L_1, \ldots, L_m)$ can be propagated to any link in Ω_u, i.e., $\Omega(L_1) \cup \ldots \cup \Omega(L_m) = \Omega_u$, A has always

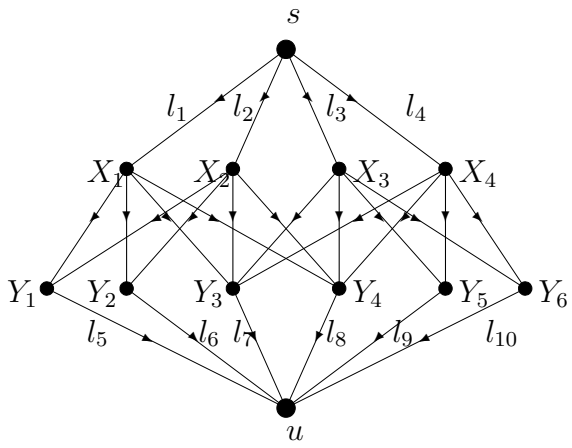


Figure 3.5: A single source and a single sink network: the link capacity in this network is as follows: $r(l_1) = r(l_2) = r(l_3) = r(l_4) = 4, r(l_5) = ... = r(l_{10}) = 2$. All the links in the middle layer have capacity 1. Error correction at Y_3 and Y_4 is necessary for achieving the capacity.

full rank.

We use G_u to denote the generator matrix between s and u. Then (3.3) is equivalent to

$$G_u x = A \cdot M \cdot (e'')^\tau. \tag{3.5}$$

Since $rank(M) = \sum_{i=1}^{m} rank(M_i) \geq km$ and A has full rank, $rank(AM) = rank(M) \geq km$.

If $rank(G_u) < k$, then there exists some $x_1 \neq 0$ such that $G_u x_1 = 0$. Then $(x, e'') = (x_1, 0)$ satisfies (3.3) and this network code is a bad code itself.

When $rank(G_u) = k$, since $rank(M) \geq mk$, we can always find (x, e'') satisfying (3.3) when $k + mk > C_1$. Thus, the upper bound on the achievable linear coding capacity is $C_1/(m + 1)$.

3.4.2 Error correction at intermediate nodes

We give an example in which error correction at intermediate nodes is used for achieving the capacity. The intuition behind our approach is that error correction at in-

termediate nodes can reduce the error propagation to the links in the bottom layer and MDS code assigned on the bottom layer gives the correct output. We consider a single source-destination network in Fig. 3.5. For a single adversarial link, upper bound from Theorem 3.6 is 8. From Sec. IV-A, the upper bound on the linear coding capacity is $\sum_{i=5}^{10} r(l_i)/(m+1) = 6$.

Lemma 3.9 *Given the network in Fig. 3.5, for a single adversarial link, rate 8 is achievable using error correction at intermediate nodes.*

Proof. Without loss of generality, all nodes forward the received information except Y_3 and Y_4. We first assign $(12, 8)$ MDS code $(a, b, \ldots, l)$ on the bottom layer links and apply (4,2) MDS code at each decision node, e.g., assign $(e, f, e+f, e+2f)$ and $(g, h, g+h, g+2h)$ on incoming links to Y_3 and Y_4 respectively. Then we can assign codewords on all links in the network since all nodes except Y_3 and Y_4 are forwarding. If the adversarial link is on the middle or bottom layer, at most two errors are propagated to the sink node and MDS code assigned on the bottom layer gives the correct output. If adversarial link is on the top layer, at most two errors are propagated to the sink node through forwarding nodes Y_1, Y_2, Y_5, and Y_6. Since at most one error is incoming to Y_3 and Y_4 respectively, (4,2) MDS code applied at each decision node gives error-free output (e, f) and (g, h). Therefore, when the adversarial link is on the top layer, at most two errors are propagated to the sink and (12,8) MDS code returns the correct output.

■

Now we generalize the approach with error correction at intermediate nodes. Given an acyclic network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we use $c_{\mathcal{G}}(s, i)$ and $c_{\mathcal{G}}(i, u)$ denote the min-cut between the source s and i, and the min-cut between i and the sink u in $\mathcal{G}$, respectively. We assume that there is a fixed set of nodes $N \subset \mathcal{V}$ such that $c_{\mathcal{G}}(s, i) \geq c_{\mathcal{G}}(i, u)$ for $\forall i \in N$ and error correction can be applied only at nodes in N. For instance, in Fig. 3.5, $\mathcal{N} = \{Y_3, Y_4\}$. Let $d(\mathcal{G}, i) = c_{\mathcal{G}}(s, i) - c_{\mathcal{G}}(i, u)$ denote the difference between the max-flow from s to i and the max-flow from i to u.

Algorithm 2 Algorithm for error correction at intermediate nodes

$M \leftarrow N$
$CS \leftarrow \emptyset$
$\mathcal{G}' = \mathcal{G}$
  While $|M| \geq 1$
    $M = M - I(\mathcal{G}', M)$
    $CS = CS \cup S_{I(\mathcal{G}',M)}$
    $\mathcal{G}' = \mathcal{G}' - S_{I(\mathcal{G}',M)}$
  endwhile
 If $|M| = 0$
    return CS
 elseif $d(\mathcal{G}', i) = 0$ for all $i \in M$
    return CS
  endif

The selection function $I(\mathcal{G}, N)$ chooses a node $i \in N$ on $\mathcal{G}$ which maximizes $d(\mathcal{G}, i)$. Precisely,

$$I(\mathcal{G}, N) = \arg\max_{i \in N}\{d(\mathcal{G}, i)\}.$$

Here is the outline of our greedy algorithm with error correction at intermediate nodes. Given an acyclic network $\mathcal{G}$ and the set of error correction nodes N, we choose a node $i = I(\mathcal{G}, N)$ that maximizes $d(\mathcal{G}, i)$ on $\mathcal{G}$. Since $c_{\mathcal{G}}(s, i)$ is the max-flow from s to i, we can find $c_{\mathcal{G}}(s, i)$ paths so that each path carries one symbol from s to i. Likewise, we also find $c_{\mathcal{G}}(i, u)$ paths from i to u. Let $S_{I(\mathcal{G},M)}$ denote the subgraph composed of the above paths. We assign a $(c_{\mathcal{G}}(s, i), c_{\mathcal{G}}(i, u))$ MDS code on $S_{I(\mathcal{G},M)}$. We subtract $S_{I(\mathcal{G}',M)}$ from $\mathcal{G}$ and add it to CS which denotes that the union of subgraph codewords are already assigned. We also subtract i from N and repeat above procedure until $N = \emptyset$ or there is no node $i \in N$ such that $d(\mathcal{G}, i) > 0$. Precise description of the algorithm is shown in algorithm 2. Since max-flow can be computed in polynomial-time, this algorithm is a polynomial-time greedy algorithm.

3.4.3 Coding at intermediate nodes

Here we give an example of a single source single sink network, shown in Fig. 3.6, for which linear coding at intermediate nodes rather than error correction or detection is used for achieving the capacity. For a single adversarial link, the upper bound from

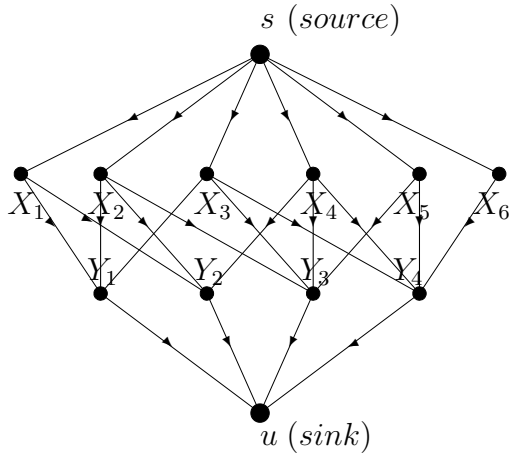


Figure 3.6: A single source and a single sink network: all links on the top or middle layer have capacity 1. All links on the bottom layer have capacity 2. In this network, coding at intermediate nodes is necessary for achieving the capacity but not error-detection and correction.

Theorem 3.6 is 4.

Lemma 3.10 *Given the network in Fig. 3.6, for a single adversarial link, coding at intermediate nodes achieves the rate 4.*

Proof. To achieve rate 4, any four links on the top layer should carry 4 independent packets. Otherwise, when adversarial link is on the top layer, source cannot transmit 4 packets reliably. Then data processing inequality gives us contradiction. Similarly, any two links on the bottom layer should carry 4 independent packets. Since Y_i is connected to at most four different nodes among $(X_1, .., X_6)$ for $\forall 1 \leq i \leq 4$ and all links in the middle layer have capacity 1, each of Y_1, Y_2, Y_3, and Y_4 receives all independent information. Thus we cannot apply simple error-detection or correction at Y_1, Y_2, Y_3, and Y_4. Suppose that only forwarding strategy is used on this network. Then we show that rate 4 is not achievable. There are six symbols on the top layer. Since we use only forwarding, these are forwarded to the bottom layer. Since bottom layer links have total capacity 8, there are at least two same symbols on the bottom layer links. This contradicts that any two links on the bottom layer should

carry 4 independent pieces of information to achieve rate 4. Therefore forwarding is insufficient for achieving the rate 4 in this network.

Now we show that a generic linear network code, where intermediate nodes do coding achieve rate 4. From [58, Ch 19], generic network code can be constructed with high probability by randomly choosing the global encoding kernels provided that the base field is much larger than sufficient. So when we apply random linear network code on this network, it is generic with high probability when q is very large. If adversarial link is on the top or middle layer, then each capacity 2 on the bottom layer is equivalent to two unit capacity links. Then all links in the network have capacity one and this problem is reduced to the equal link capacities problem. From [31], rate $6 - 2 \times 1 = 4$ is achievable. From [58, Theorem 19.32], since the min-cut between s and (Y_i, Y_j) is at least 4 for $\forall 1 \leq i \neq j \leq 4$, in a generic network code the global encoding kernels on any two links on the bottom layer are linearly independent and they satisfy with MDS property. Thus an error on the last layer can be corrected. ■

3.4.4 Guess-and-forward

Here we introduce a new achievable coding strategy guess-and-forward. In this strategy, an intermediate node which receives some redundant information from multiple paths guesses which of its upstream links is controlled by the adversary. The intermediate node forward its guess to the sink which tests the hypothesis of the guessing node. Here we show the optimality of this strategy in following example.

Consider the four-node acyclic network shown in Fig. 3.7. From Theorem 3.6, when $z = 2$, upper bound is 7. Now we show that rate 7 is achievable in this network. Let W and $\hat{W}$ denote the symbols sent by S and received by A respectively. W is sent reliably from S to B and $\hat{W}$ is sent reliably from A to U. Guess-and-forward strategy for this example network is as follows.

At each time step, S and B together send a $(11, 7)$ MDS code to A and U across the cut $cut(\{S, B\}, \{A, U\})$. Since feedback link has capacity 6, A sends its code-

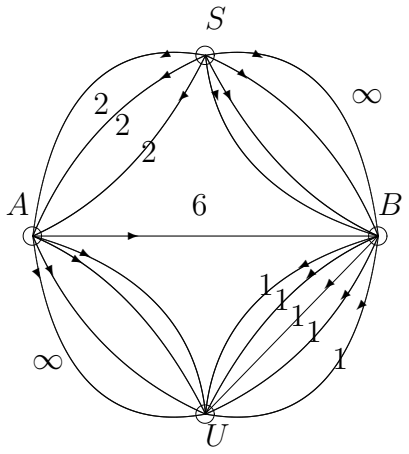


Figure 3.7: Four node acyclic networks: this network consists of 3 links of capacity 2 from S to A, 5 links of capacity 1 from B to U, 1 link of capacity 6 from A to B. Given the cut $(\{S, B\}, \{A, U\})$, unbounded reliable communication is allowed from source S to its neighbor B on one side of the cut and from node A to sink U on the other side of the cut, respectively.

word symbols $\hat{W}$ to B along feedback link l. For feedback link l, let P_l denote the information received by B on l. B compares P_l with W which is received from S. If $P_l \neq W$, then B obtains a guess X_l identifying the locations of adversarial links between S and A assuming P_l is reliable. B sends the claim (X_l, P_l) to U along each link between B and U using repetition code. If $P_l = W$, B does not send anything. The above strategy is applied at each time step. B sends claims only when it guesses at least one adversarial forward link which is different from forward links guessed at previous time steps.

Lemma 3.11 *Given the network in Fig. 3.7, rate 7 is achievable.*

Proof. Since there are $5 \geq 2z + 1$ links from B to U, any claim (X_l, P_l) can be sent reliably from B to U using a repetition code.

Case 1) sink receives some claim (X_l, P_l).

The sink compares P_l with $\hat{W}$ which is received from A reliably. If $P_l \neq \hat{W}$, then feedback link transmitting P_l is adversarial and the sink ignores it. Otherwise, P_l is reliable. Since the claim is sent, the sink knows that $P_l = \hat{W} \neq W$ and that guess X_l

is correct. Thus the sink identifies at least one forward adversarial link from S to A, which is subsequently ignored.

Case 2) no claims are sent.

In this case, we show that the correct output is achieved without using any claims. B does not sent any claim when it receives W from each feedback link. There are two possibilities.

a) All links between S and A and feedback link are uncorrupted.

b) Some links between S and A are corrupted and feedback link is corrupted such that feedback link transmits error-free output.

In a), feedback link transmits W to B. In b), A sends $\hat{W} \neq W$ but the feedback link changes it to W so that B does not send any claims. We first consider all sets of 7 forward links on the cut. There are $\binom{8}{7} = 8$ such sets of links. Each set has total capacity of at least 9. For each such set L, the sink check the consistency of the output of rate 7 obtained from L. We also consider all sets of 6 links such that each set includes all 3 links between S and A and any 3 links between B and U. There are $\binom{5}{3}$ such sets. The sink also checks the consistency of the output of rate 7 for each set.

Case 2 - a) there is no set of 7 links giving consistent output.

In this case, there are more than 1 forward adversarial links on the cut. Since $z = 2$, all two adversarial links are forward links and thus possibility b) cannot hold. Then a) is true and there are at most two forward adversarial links with capacity 1 on the cut. We obtain correct answer from our (11,7) MDS code.

Case 2 - b) there is no set of 6 links that include all 3 links from S to A and give consistent output.

In this case, possibility b) is true. Then there is at most one forward adversarial link on the cut. We obtain a correct answer from our (11,7) MDS code.

Case 2 - c) There exist both 7 links set L_1 giving consistent output and 6 links set L_2 that include all 3 links between S and A and give consistent output.

It is clear that $\sum_{l_1 \in L_1 \cap L_2} r(l_1) \geq 7$ for any L_1 and L_2. Thus L_1 and L_2 give the same consistent output. Since at least one of a) and b) is true, this output is correct.

From cases 1-2, since $z = 2$, B needs to send claim at most 2 times to obtain the correct output. ■

3.5 Example networks

In this section, we employ the guess-and-forward strategy on a sequence of networks designed to provide increasingly complex generalizations of the cut-set bounds. The first is a two-node network with multiple feedback links. The second is a four-node acyclic network. The third is a family of 'zig-zag' networks. In the first two cases, the guess-and-forward strategy achieves the capacity. For zig-zag networks, we derive the achievable rate of guess-and-forward strategy and present conditions under which this bound is tight.

3.5.1 Two-node network

We characterize the error-correction capacity of a two-node network with multiple feedback links by using guess-and-forward strategy. A two-node network shown in Fig. 3.8 is composed of n forward links with arbitrary capacity and m feedback links with arbitrary capacity. In Lemma 3.12, we first characterize the capacity of this network when each forward link has capacity 1. We extend this result to Theorem 3.13 when each forward link has arbitrary capacity.

Lemma 3.12 *Consider the two-node network shown in Fig. 3.8 such that each forward link has capacity 1. Let C denote the error-correction capacity with z adversarial links. If $n \leq 2z$, $C = 0$. Otherwise, $C = \min\{n - z, n - 2(z - m)\}$.*

Proof.

Case 1) $n \leq 2z$.

Suppose that all z adversarial links are among the forward links and $C > 0$. We show a contradiction. Since $C > 0$, there are two codewords $X = (x_1, .., x_n)$ and $Y = (y_1, .., y_n)$ that can be sent reliably. When X is sent along forward links and leftmost z links are adversarial, the adversary changes X to $X' = (y_1, .., y_{\lfloor n/2 \rfloor}, x_{\lfloor n/2 \rfloor + 1}, .., x_n)$

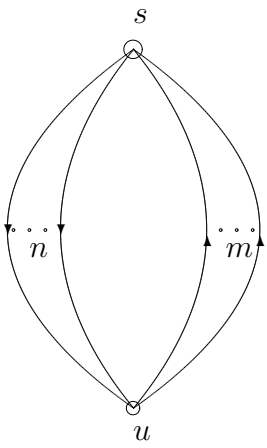


Figure 3.8: two-node network $\mathcal{G}$ with n forward links and m feedback links.

so that first $\lfloor n/2 \rfloor$ bits of X' are the same as that of Y. Similarly, when Y is sent along forward links and rightmost z links are adversarial, the adversary changes Y to $Y' = (y_1, .., y_{\lfloor n/2 \rfloor}, x_{\lfloor n/2 \rfloor + 1}, .., x_n)$ so that the last $\lceil n/2 \rceil$ bits of Y' are the same as that of X. Then sink receives the same observations for the two codewords. Since information on feedback links is determined by what the sink receives, source also cannot get any different information from feedback links. Thus the two codewords cannot be distinguished and this contradicts $C > 0$.

Case 2) $n > 2z$.

We first show the converse. If z adversarial links are all forward links, then the capacity is less than or equal to $n - z$. If all $m \leq z$ feedback links are adversarial, the remaining network is a two-node network composed of n unit capacity forward links with $z - m$ adversarial links whose capacity is $n - 2(z - m)$ from [31]. Thus, the upper bound is $C = \min\{n - z, n - 2(z - m)\}$.

Now we apply our guess-and-forward strategy to achieve rate C as follows. We consider two cases.

Case 2 - a) $m \leq \frac{z}{2}$.

Step 1) In each time slot, the source s sends an $(n, n - 2(z - m))$ MDS code on the n forward links. Since $m \leq z/2$, $n - 2(z - m) \leq n - z$. Thus for any received

n signals, there exist $n-2(z-m)$ uncorrupted signals. If all $\binom{n}{n-2(z-m)}$ subsets of received symbols decode to the same message, this message is correct. Otherwise, the sink sends the n received signals to the source s on each feedback link using a repetition code.

Step 2) Based on the received information on each feedback link, the source tries to identify the bad forward links. Thus, for each feedback link, the source obtains a claim regarding the location of forward adversarial links which is correct if that feedback link is not adversarial.

Step 3) This step consists of m rounds, each composed of a finite number of time slots. In the ith round, the source sends the claim obtained from the ith feedback link together with what it received on that feedback link to the sink. This information can be sent reliably to the sink using a repetition code because $n-2z>0$. If what the source received matches what the sink sent, the ith feedback link was not corrupted and the associated claim is correct. Using this claim, the sink can decode the message as well as identify at least one of the forward adversarial links. If all m feedback links were corrupted, the sink knows that there are only $z-m$ forward adversarial links and since we are using a $(n, n-2(z-m))$ MDS code the message is correctly decodable at the sink.

Note that we only need to use above scheme the first $2m$ times the sink sees inconsistency at step 1. The reason is that from steps 1-3, the sink either figures out that all feedback links are adversarial or identifies at least one forward adversarial link. If all feedback links are bad, they are ignored and the $(n, n-2(z-m))$ MDS code gives us the correct output. If there are $k \leq 2m$ forward adversarial links, after the first k times the sink sees inconsistency at step 1, all forward adversarial links are identified subsequently and no further inconsistency is seen among the remaining forward links. Otherwise, when there are more than $2m$ adversarial links, the sink finds $2m$ forward adversarial links and ignores them. Then from [31], the rate $n-2m-2(z-2m) = n-2(z-m)$ can be achieved using the remaining forward links only.

Case 2 - b) $m > \frac{z}{2}$.

In each time slot, the source s sends an $(n, n-z)$ MDS code on the n forward links. For any received n signals, there exist $n-z$ uncorrupted signals. If all $\binom{n}{n-z}$ subsets of received symbols decode to the same message, this message is correct. As in the case 2-a, from steps 2-3, the sink either concludes that all feedback links are adversarial or identifies at least one forward adversarial link. If all m feedback links were corrupted, there are only $z-m < z/2$ bad forward links and subsequently only the forward links are used to achieve the rate $n-z$. Otherwise, the above scheme is used at most z times inconsistency is seen at step 1, after which the sink has identified all bad forward links and the remaining forward links suffice to achieve rate $n-z$. ■

Now we generalize the above result to the general case when each forward link has also arbitrary capacity.

Theorem 3.13 *Consider the two-node network shown in Fig. 3.8 with arbitrary link capacities. Let D_p denote the sum of the p smallest forward link capacities. The error-correction capacity is*

$$C = \begin{cases} 0 & \text{if } n \leq 2z \\ \min\{D_{n-z}, D_{n-2(z-m)^+}\} & \text{if } n > 2z. \end{cases}$$

Proof. We use the similar proof in Lemma 3.12 for the case $n \leq 2z$. Suppose that $C > 0$ and we show a contradiction. Since $C > 0$, there are two codewords X and Y that can be sent reliably. When X is sent along forward links and leftmost z links are adversarial, the adversary changes X to X' so that the outputs of leftmost $\lfloor n/2 \rfloor$ links of X' are the same as that of Y. Similarly, when Y is sent along forward links and rightmost z links are adversarial, the adversary changes Y to Y' so that the outputs of rightmost $\lceil n/2 \rceil$ links of Y' are the same as that of X. Then the two codewords cannot be distinguished and contradicts $C > 0$.

Consider the case $n \geq 2z$. We first show the converse. When sink knows z adversarial links are the z largest capacities forward links, the maximum achievable capacity is D_{n-z}. When $m \leq z$ and all m feedback links are adversarial, there are $z-m$ adversarial forward links such that locations are unknown. In this scenario, the

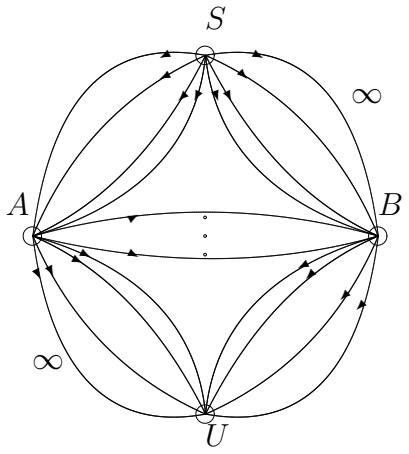


Figure 3.9: Four node acyclic networks: unbounded reliable communication is allowed from source S to its neighbor B and from node A to sink U, respectively. This network consists of a links of arbitrary capacity from S to A, b links of arbitrary capacity from B to U. From A to B, there are m feedback links and each feedback link has the minimum capacity.

best achievable rate is $D_{n-2(z-m)^+}$, which is the sum of $n-2(z-m)$ smallest forward link capacities [43, Theorem 1].

We use D to denote the sum of all n forward link capacities. For achievability, when $m \leq z/2$, source sends (D, D_{n-z}) MDS code to sink. When $m > z/2$, source sends $(D, D_{n-2(z-m)^+})$ MDS code to sink. By using the same strategy in the proof of Lemma 3.12, we can achieve the rate C.

■

3.5.2 Four-node acyclic network

In this section, we investigate our guess-and-forward strategy with a four-node acyclic network. In the two-node network, this cut-set abstraction is insufficient to fully capture the effect of network topology relative to the cut since it assumes that all feedback is available to the source node and all information crossing the cut in the forward direction is available to the sink. We therefore introduce the four-node acyclic network shown in Fig. 3.9 as step towards generalizing the cut-set too. In this acyclic network,

source node S and its neighbor node B lie on one side of a cut that separates them from sink node U an its neighbor A. As in the cut-set model, we allow unbounded reliable communication from source S to its neighbor B on one side of the cut and from node A to sink U on the other side of the cut; this differs from the cut-set assumption only in that the communication goes one way only.

This network is composed of a set of a forward links $\{l_1, .., l_a\}$ with arbitrary capacities from S to A, a set of b forward links $\{l_{a+1}, .., l_{a+b}\}$ with arbitrary capacities from B to sink U, a set of m feedback links from A to B for which each feedback link has capacity h which will be derived in following Section 3.5.2.1. $C_1 = \sum_{l\in(l_1,..,l_a)} r(l)$ and $C_2 = \sum_{l\in(l_{a+1},..,l_{a+b})} r(l)$ denotes the sum of forward link capacities from S to A, and from B to U, respectively. Let $C = C_1 + C_2$. C_z is the upper bound on this network obtained from Theorem 3.6.

In [44], we have shown that rate C_z is asymptotically achievable on this four-node acyclic network when each feedback link has capacity of at least C_1. Here we show that rate C_z is achievable even when each feedback link has smaller capacity than C_1. In Section 3.5.2.1, we first consider a coding strategy at node A and formulate the linear optimization problem which gives the minimum capacity of each feedback link. Then we show that rate C_z is asymptotically achievable with smaller feedback link capacity than C_1 using our guess-and-forward strategy in Section 3.5.2.2.

3.5.2.1 Coding strategy at node A

Suppose that (S, B) sends (C, C_z) MDS code across the cut to (A, U). We consider the encoding strategy at node A and derive the minimum capacity of each feedback link. Suppose that node A receives the vector of symbols $\hat{W} = (\hat{W}_{l_1}, \ldots, \hat{W}_{l_a})$ from S where $\hat{W}_l = (p_l^1, \ldots, p_l^{r(l)})$ denotes the codewords on link $l \in \{l_1, \ldots, l_a\}$. We first assume that node A transmits, on each feedback link to B, the same set of codewords each of which is a linear combination of codewords received on a single link from S to A. Precisely, for any forward link l_i, node A transmits on each feedback link $g(\hat{W}_{l_i}) = (g_{l_i}^1(\hat{W}_{l_i}), \ldots, g_{l_i}^{k_i}(\hat{W}_{l_i}))$ where $g_{l_i}^j(\hat{W}_{l_i})$ is a single linear combination of $\hat{W}_{l_i} = (p_{l_i}^1, \ldots, p_{l_i}^{r(l_i)})$. For instance, given a network in Fig. 3.10(a), A transmits

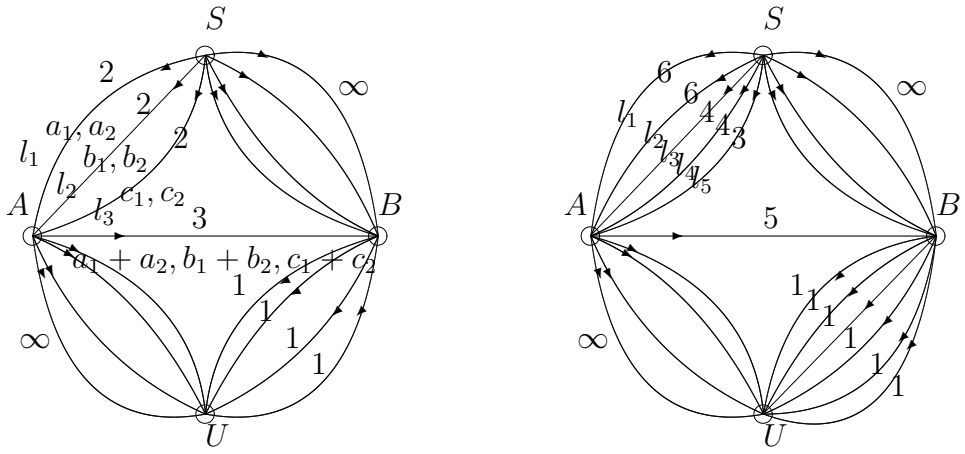


Figure 3.10: Four node acyclic networks: (a) $z = 2$ and feedback link transmits $(a_1 + a_2, b_1 + b_2, c_1 + c_2)$. (b) $z = 3$. Assume that $(a_1, .., a_6)$, $(b_1, .., b_6)$, $(c_1, .., c_4)$, $(d_1, .., d_4)$, and (e_1, e_2, e_3) are transmitted on forward links $(l_1, .., l_5)$ from S to A, respectively. Feedback link also transmits the sum of codewords on each forward link.

$g(\hat{W}) = (g(\hat{W}_{l_1}), g(\hat{W}_{l_2}), g(\hat{W}_{l_3}))$ where $g(\hat{W}_{l_1}) = a_1 + a_2$, $g(\hat{W}_{l_2}) = b_1 + b_2$, and $g(\hat{W}_{l_3}) = c_1 + c_2$.

Here, we define the degree of freedom of forward link l between S and A as follows.

Definition 3.14 *Consider the vector of symbols $\hat{W}_l$ received on forward link l from S to A and assume that node A transmits k linear combinations of $\hat{W}_l$, $g(\hat{W}_l) = (g_l^1(\hat{W}_l), \ldots, g_l^k(\hat{W}_l))$. Let M_l denote the $r(l) \times k$ encoding matrix at A for link l such that $\hat{W}_l \cdot M_l = g(\hat{W}_l)$. Then the degree of freedom of link l, $f(l)$, is defined as the capacity of link l minus the rank of the matrix M_l, i.e., $f(l) = r(l) - rank(M_l)$. For any forward link l between B and U, we simply define the degree of freedom $f(l)$ as the link capacity, i.e., $f(l) = r(l)$.*

For example, in Fig. 3.10(a), since feedback link transmits $(a_1 + a_2, b_1 + b_2, c_1 + c_2)$, $f(l) = 1$ for all forward links from S to A. In Fig. 3.10(b), since feedback link transmits $(\sum_{i=1}^{6} a_i, \sum_{i=1}^{6} b_i, \sum_{i=1}^{4} c_i, \sum_{i=1}^{4} d_i, \sum_{i=1}^{3} e_i)$, $f(l_1) = f(l_2) = 5$, $f(l_3) = f(l_4) = 3$, and $f(l_5) = 2$.

From the definition of degree of freedom, node A sends

$$h = \sum_{l \in \{l_1,..,l_a\}} (r(l) - f(l)) = C_1 - \sum_{l \in \{l_1,..,l_a\}} f(l) \tag{3.6}$$

codewords to B along each feedback link.

Now we introduce our coding strategy at node A as follows.

Node A is allowed to choose any $g(\hat{W})$ which satisfies two following conditions on the degree of freedom of links.

Condition 3.15 *Given any set A_1 composed of $2z$ forward links, $\sum_{l \in A_1} f(l) \leq C - C_z$.*

Condition 3.16 *Given any set A_2 composed of z forward links and A_3 composed of $z - m$ forward links such that $A_2 \cap A_3 = \emptyset$, $\sum_{l \in A_2} f(l) + \sum_{l \in A_3} r(l) \leq C - C_z$.*

Condition 3.15 means that the sum of the degree of freedom of any $2z$ forward links are less than or equal to $C - C_z$. Condition 3.16 means that the sum of the degree of freedom of any z links plus the sum of any $z - m$ link capacities is less than or equal to $C - C_z$. In the proof of Lemma 3.19 and 3.20, we show that these two conditions are necessary to prove the tightness of our upper bound in Theorem 3.6. For example network in Fig. 3.10(a), $z = 2$ and the upper bound $C_z = 6$. 3 codewords sent by A satisfies the above two conditions, and feedback capacity 3 is sufficient. Likewise, when $z = 3$ and the upper bound $C_z = 9$ in the network Fig. 3.10(b), 5 codewords sent by A also satisfies the above two conditions. In [44], the minimum required capacity for each feedback link to achieve rate C_z is the sum of all forward link capacities between S and A, which is 6 and 23 for the networks in Fig. 3.10(a) and (b), respectively.

Finally, we formulate a linear optimization problem which gives the minimum capacity of each feedback link, based on conditions 3.15 and 3.16.

$$
\begin{aligned}
\min \quad & h = C_1 - \sum_{i=1}^{a} f(l_i) \\
& f(l_i) \le r(l_i), \quad \forall 1 \le i \le a+b \\
& \sum_{l \in M} f(l) \le C - C_z, \quad M \subset \mathcal{E}, |M| \le 2z \\
& \sum_{l \in N_1} r(l) + \sum_{l \in N_2} f(l) \le C - C_z, \\
& \quad N_1, N_2 \subset \mathcal{E}, |N_1| \le z - m, |N_2| \le z, N_1 \cap N_2 = \emptyset
\end{aligned} \tag{3.7}
$$

Objective function h is defined in equation (3.6). First inequality constraint is the link capacity constraint. Second and third constraints come from condition 3.15 and 3.16, respectively. We can check that solving above optimization problem for the networks in Fig. 3.10(a) and (b) gives the feedback link capacity 3 and 5, respectively.

3.5.2.2 Guess-and-forward strategy

We first prove the following useful lemma.

Lemma 3.17 *Given the four-node acyclic network in Fig. 3.9, let t denote the sum of $2z$ largest degree of freedom of links in the network. When the adversary introduces error on z forward links subject to the constraint that codewords on feedback links are unchanged, there exists $(C, C-t)$ generic linear code that corrects these z error links.*

Proof. See the appendix. ■

Since the sum of $2z$ largest degree of freedom is at most $C - C_z$ from the condition 3.15, we obtain $C - t \ge C_z$.

Now we introduce a guess-and-forward strategy for the four-node acyclic network shown in Fig. 3.9 that is used for achieving the rate C_z.

At each time step, S and B together send a (C, C_z) MDS code, obtained from Lemma 3.17, to A and U across the cut $cut(\{S, B\}, \{A, U\})$. Let W and $\hat{W}$ denote the codewords S sends to A, and A received from S, respectively. Using the coding strategy in Section 3.5.2.1, A sends $g(\hat{W})$ to B along each feedback link using a

repetition code. For each feedback link l, let P_l denote the information received by B on l. B compares P_l with $g(W)$. If $P_l \neq g(W)$, then B obtains a guess X_l identifying the locations of adversarial links between S and A assuming P_l is reliable. B sends the claim (X_l, P_l) to U along each link between B and U using repetition code. If $P_l = g(W)$, B does not send anything. The above strategy is applied at each time step. B sends claims only when X_l guesses at least one forward adversarial link which is different from forward links guessed from l at previous time steps.

We show that this strategy achieves rate C_z asymptotically by proving following Lemma 3.19 and 3.20.

We first introduce a definition which is used in the proof of lemmas.

Definition 3.18 *Given a set of any k forward links $L = \{l_1, \ldots, l_k\}$ in the four-node acyclic network, we say L gives consistent output if $\sum_{l \in L} r(l) \geq C_z$ and the decoded output from any C_z code symbols on L is the same.*

Lemma 3.19 *Given the four-node network in Fig. 3.9 such that $b \geq 2z + 1$, rate C_z is achievable.*

Proof. Since $b \geq 2z + 1$, any claim (X_l, P_l) can be sent reliably from B to U using a repetition code. The details of proof are presented in the appendix. ■

Lemma 3.20 *Given the four-node network in Fig. 3.9 such that $b \leq 2z$, rate C_z is achievable.*

Proof. When $b \leq 2z$, reliable transmission of claims from B to U is not guaranteed. Thus we cannot use the same technique used in the proof of Lemma 3.19. The proof is presented in the appendix. ■

3.5.3 Zig-zag network

In this section, we consider a more general family of zig-zag networks which capture the behavior of any cut with k feedback links more accurately. We present conditions under which our upper bound is tight and derive a general achievable bound.

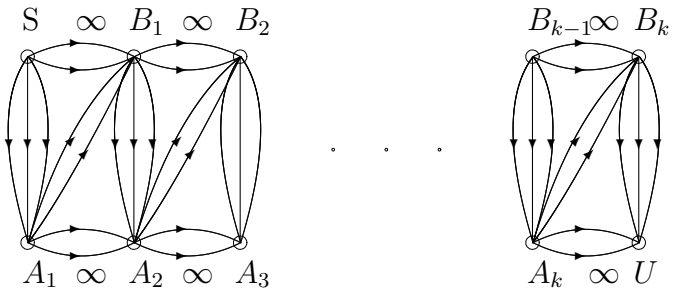


Figure 3.11: k-layer zig-zag network: given the cut $cut(\{S, B_1, .., B_k\}, \{A_1, .., A_k, U\})$, A_i and B_i can communicate reliably with unbounded rate to A_{i+1} and B_{i+1}, respectively. ($S = B_0$, $U = A_{k+1}$). The links from A_i to B_i represent feedback across the cut. This model more accurately captures the behavior of any cut with k feedback links across the cut.

We call the network shown in Fig. 3.11 a k-layer zig-zag network. A_i and B_i can communicate reliably with unbounded rate to A_{i+1} and B_{i+1}, respectivley. ($S = B_0$, $U = A_{k+1}$). Thus, reliable transmission with unbounded rate is possible from A_i to A_j, and from B_i to B_j for $\forall i < j$. We use F_i and W_i to denote the set of forward links and feedback links from B_{i-1} to A_i, and from A_i to B_i, respectively. Let $|F_i| = b_i$ and $|W_i| = m_i$. In this network, we assume that each feedback link from A_i to B_i has a sufficient capacity to forward the information A_i received from B_{i-1} to B_i. It is clear that the four-node network is 1-layer zig-zag network. Given a k-layer zig-zag network $\mathcal{G}$, we use C_z to denote the upper bound on $\mathcal{G}$ obtained from Theorem 3.6.

Now we consider following strategy which is similar to that for a four-node network. We use C to denote the sum of all forward link capacities.

At each time step, S and $(B_1, .., B_k)$ together send a (C, C_z) MDS code to $(A_1, .., A_k)$ and U across the cut $cut(\{S, B_1, .., B_k\}, \{A_1, .., A_k, U\})$. For $1 \leq i \leq k$, A_i sends its codeword symbols $\hat{W}$ to B_i along each feedback link using a repetition code. For each feedback link l, let P_l denote the information received by B_i along on l. B_i compares

P_l with W which is received from S. If $P_l \neq W$, then B_i obtains a guess X_l identifying the locations of adversarial links between B_{i-1} and A_i assuming P_l is reliable. B_i sends claim (X_l, P_l) to A_{i+1} along each link using repetition code. If $P_l = W$, B_i does not send anything. For all $2 \leq j \leq k$, A_j sends any received claim from B_{j-1} to the sink reliably. The above strategy is applied at each time step. B_i sends claims only when X_l guesses at least one adversarial forward link which is different from forward links guessed from l at previous time steps.

For a four-node acyclic network in Fig. 3.9, Lemma 3.19 shows that our bound is tight when claim is sent reliably from node B to the sink U, i.e., $b \geq 2z + 1$. Using our strategy, we simply extend this result for the zig-zag network as follows.

Lemma 3.21 *Given a family of k-layers zig-zag networks such that $b_i \geq 2z + 1$ for $2 \leq i \leq k + 1$, rate C_z is achievable.*

Proof. Since $b_i \geq 2z + 1$ for $2 \leq i \leq k + 1$, any claim (X_l, P_l) can be sent reliably from B_{i-1} to A_i using repetition code. Then A_i sends this claim reliably to sink U. As in the proof of Lemma 3.19, we first show that at least one adversarial link is removed whenever sink receives some claim, in case 1. We also show that correct output is always achievable when no claims are sent in case 2.

Case 1) sink receives some claim (X_l, P_l).

Assume that feedback link l is between A_j and B_j, and B_j sends this claim to A_{j+1}. In this case, we use the same strategy as in the case 1 in Lemma 3.19. Then we show that the sink removes at least one bad link whenever it receives claim.

Case 2) no claims are sent to the sink.

Here we extend the case 2 in the proof of Lemma 3.19. There are following possibilities for zig-zag network such that no claims are sent:

a) All forward links in (F_1,..,F_k) and feedback links in (W_1,..,W_k) are not corrupted.

b) For some $\{i_1, .., i_p\} \subseteq \{1, 2, .., k\}$ such that $m_{i_1} + .. + m_{i_p} \leq z$, all feedback links in $(W_{i_1}, .., W_{i_p})$ are corrupted and some forward links in $(F_{i_1}, .., F_{i_p})$ are corrupted. The furthest downstream forward links in F_{k+1} can be also corrupted. For $\forall j \notin$

$\{i_1, .., i_p, k+1\}$, links in F_j and W_j are not corrupted.

Let $N = \{(i_1, .., i_p) | 1 \leq i_1 < .., < i_p \leq k, m_{i_1} + .. + m_{i_p} \leq z\} \cup \{\emptyset\}$. (Note that $\{\emptyset\}$ corresponds to the possibility in a)). From a) and b), there are total $|N|$ possibilities. Exactly only one of them is true. Now we describe how a correct solution with rate C_z can be obtained. We check the consistency of the output for each possibility. For each $(i_1, .., i_p) \in N$, we check that there are $K - (z - (m_{i_1} + .. + m_{i_p}))$ forward links giving consistent output such that removed $(z - (m_{i_1} + .. + m_{i_p}))$ forward links are elements of $F_{i_1} \cup .. \cup F_{i_p} \cup F_{k+1}$. We use $G(i_1, .., i_p)$ to denote the the set of such forward links giving consistency. If there are no such $K - (z - (m_{i_1} + .. + m_{i_p}))$ forward links giving consistency, we remove $(i_1, .., i_p)$ from N and ignore corresponding possibilities.

Now we show that only tuples $(i_1, .., i_p)$ such that $G(i_1, .., i_p)$ gives the correct consistent output remain in N. Since at least one remaining tuple gives the correct output, it is sufficient to prove that for any remaining $(i_1, .., i_p) \in N$ and $(j_1, .., j_r) \in N$, $G(i_1, .., i_p)$ and $G(j_1, .., j_r)$ gives the same output. This is equivalent to show that the sum of capacities of forward links which are contained in both $G(i_1, .., i_p)$ and $G(j_1, .., j_r)$ are at least C_z, i.e.,

$$\sum_{l \in G(i_1,..,i_p) \cap G(j_1,..,j_r)} r(l) \geq C_z.$$

$G(i_1, .., i_p)$ gives $K - (z - (m_{i_1} + .. + m_{i_p}))$ forward links giving consistent output such that removed $(z - (m_{i_1} + .. + m_{i_p}))$ forward links are in $F_{i_1} \cup .. \cup F_{i_p} \cup F_{k+1}$. Similarly, $G(j_1, .., j_r)$ gives $K - (z - (m_{j_1} + .. + m_{j_r}))$ forward links giving consistent output such that removed $(z - (m_{j_1} + .. + m_{j_r}))$ forward links are in $F_{j_1} \cup .. \cup F_{j_r} \cup F_{k+1}$. In this case, from the definition of cut-set upper bound in Lemma 3.5, sum of the capacity of forward links assumed to be correct by both $G(i_1, .., i_p)$ and $G(j_1, .., j_r)$ are at least C_z. Since each guess gives consistent output, these two guesses gives the same output. Since any two remaining guesses in N give the same consistent output, all remaining guesses give the same output.

■

We derive another condition under which our bound is tight.

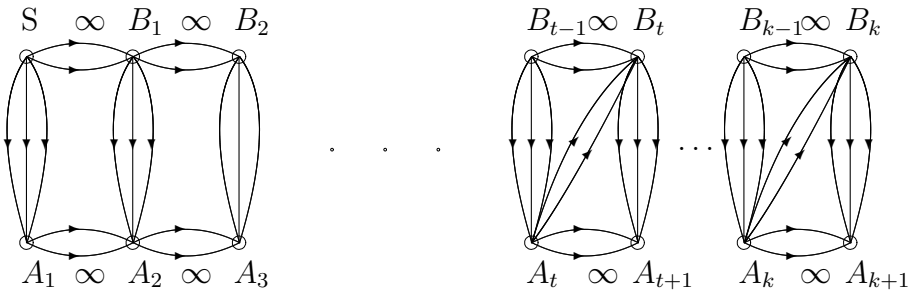


Figure 3.12: Reduced zig-zag network $\mathcal{G}'$ such that $m_t > z$ and $b_{t+1} \geq 2z + 1, \ldots, b_{k+1} \geq 2z + 1$. This graph is obtained from $\mathcal{G}$ by erasing all feedback links in $W_1 \cup W_2 \ldots \cup W_{t-1}$.

Lemma 3.22 *Given a family of k-layers zig-zag networks such that $m_t > z$ and $b_j \geq 2z + 1$ for $\forall j \geq t + 1$ for any $1 \leq t \leq k$, rate C_z is achievable.*

Proof. We consider a reduced zig-zag network $\mathcal{G}'$ shown in Fig. 3.12 which is obtained from given a k-layer zig-zag network by erasing $m_1 + .. + m_{t-1}$ feedback links $W_1 \cup .. \cup W_{t-1}$. We use C'_z to denote the upper bound on $\mathcal{G}'$ from Theorem 3.6. Since $\mathcal{G}'$ is weaker than $\mathcal{G}$, it is sufficient to show that $C'_z \geq C_z$ and C'_z is achievable on $\mathcal{G}'$.

Step 1) We show that $C_z \leq C'_z$.

We first compute C'_z on $\mathcal{G}'$ from Theorem 3.6. Suppose that C'_z is obtained by choosing and $A^* = \{A_{i_1}, .., A_{i_p}\} \subseteq \{A_{t+1}.., A_k\}$ and $B^* = \{A_{j_1}, .., A_{j_r}\} \subseteq \{A_{t+1}, .., A_k\} - A^*$ and applying Lemma 3.5 after erasing k forward links set F^*, m feedback links set W^*. It is sufficient to prove that choosing the same F^*, W^*, A^*, and B^* on original graph $\mathcal{G}$ gives the same upper bound C'_z.

Since $m_t > z$, $A_t \notin A^*$ and $A_t \notin B^*$ from the definition of upper bound in Lemma 3.5. Then $P_{A^*} \subseteq F_{A_{i_1}} \cup .. \cup F_{A_{i_p}} \subset F_{t+1} \cup \ldots \cup F_k$, $P_{B^*} \subseteq F_{A_{j_1}} \cup .. \cup F_{A_{j_r}} \subset F_{t+1} \cup \ldots \cup F_k$.

Since $m_t > z$ and $b_{k+1} > z$, no matter what W^* is erased on $\mathcal{G}'$, chosen downstream links R_{A^*} and R_{B^*} are in $F_{t+1} \cup .. \cup F_k$, i.e., $R_{A^*}, R_{B^*} \in F_{t+1} \cup \ldots \cup F_k$.

Thus, $Z_{A^*} = P_{A^*} \cup R_{A^*} \subset F_{t+1} \cup .. \cup F_k$ and $Z_{B^*} = P_{B^*} \cup R_{B^*} \subset F_{t+1} \cup .. \cup F_k$.

Since all erased forward links in $Z_{A^*} \cup Z_{B^*}$ are in $F_{t+1} \cup .. \cup F_k$ for $\mathcal{G}'$, erasing the same F^*, W^*, Z_{A^*}, and Z_{B^*} on original graph $\mathcal{G}$ also gives the same upper bound C'_z for $\mathcal{G}$. Since C_z is the minimal upper bound for $\mathcal{G}$, $C_z \leq C'_z$.

Step 2) We show that rate C_z is achievable.

From Lemma 3.21, since $b_{t+1} \geq 2z+1, \ldots, b_{k+1} \geq 2z+1$ and $C_z \leq C'_z$, rate C_z is achievable on $\mathcal{G}'$. Thus, given a zig-zag network $\mathcal{G}$, we first ignore all feedback links between A_i and B_i $(1 \leq i \leq t-1)$ and apply the same achievable strategy for $\mathcal{G}'$.

From steps 1) and 2), we complete the proof. ■

Now we derive an achievable rate of guess-and-forward strategy for any zig-zag network.

We use $\mathcal{G}_I$ to denote the zig-zag network obtained from original $\mathcal{G}$ by erasing all feedback links in W_i such that $i \notin I$. Let $b(i,j) = \sum_{t=i+1}^{j} b_i$ denote the number of forward links between i th layer and jth layer. Supersets P, Q, and R are defined as follows.

$$
\begin{aligned}
P &= \{\{i\} | 1 \leq i \leq k\}, \\
Q &= \{\{i_1, .., i_t\} | \{i_1, .., i_t\} \subset \{1, .., k\}, b(1, i_1) \geq 2z+1, \\
&\quad\ b(i_1, i_2) \geq 2z+1, .., b(i_t, k+1) \geq 2z+1\}, \\
R &= \{\{i_1, .., i_t\} | \{i_1, .., i_t\} \subset \{1, .., k\}, m_{i_1} > z, \\
&\quad\ b(i_1, i_2) \geq 2z+1, .., b(i_t, k+1) \geq 2z+1\}.
\end{aligned}
$$

Lemma 3.23 *Given the network in Fig. 3.11, rate* $\max_{I \in P \cup Q \cup R} C_I$ *is achievable.*

Proof. We first show that rate $C_{\{i\}}$ is achievable for $1 \leq i \leq k$. We ignore all feedback links except the feedback links in W_i. Then applying the same achievability

strategy for four-node acyclic network gives the rate $C_{\{i\}}$ from Lemma 3.19 and 3.20.

For any subset $I \in Q$, we ignore all feedback links except the feedback links in W_i such that $i \in I$. Then from Lemma 3.21, rate C_I is achievable. Similarly, for any subset $I \in R$, rate C_I is achievable from Lemma 3.22. This completes the proof.

■

Chapter 4

Conclusion and Future Work

In the first part of this thesis, we demonstrate the benefits of network coding for optimizing the use of various network resources. We first study the problem of minimizing the power consumption for wireless multiple unicasts. We consider a simple XOR-based coding strategy, reverse carpooling, which can be used to reduce the number of transmissions and the corresponding power consumption. We investigate the use of this scheme on a wireless triangular grid network. We propose a centralized algorithm that approximately minimizes the number of transmissions for two unicasts and extended this to obtain a polynomial time greedy algorithm for the general problem with multiple unicasts.

We also present a distributed strategy for reducing the power consumption in a network coded wireless network with multiple unicasts. Our strategy attempts to increase network coding opportunities without the overhead required for centralized design or coordination. The proposed technique designates "reverse carpooling lines" such that intermediate nodes apply reverse carpooling opportunistically along these routes. We show that our optimization algorithm chooses the reverse carpooling lines in a manner that maximizes the expected power savings in polynomial time.

We study the problem of minimum-energy multicast using network coding in mobile ad hoc networks (MANETs). Instead of solving a linear program every time slot, we present a low-complexity approach, network coding with periodic recomputation, which recomputes an approximate solution at fixed time intervals, and uses this solution during each time interval. We obtain a simple theoretical cost bound

on the gap between our solution and the optimal cost, and analyze the complexity using a warm-start method. It is shown how our results can be applied to trade off performance and complexity in a given network scenario.

We further develop a back-pressure based distributed optimization framework, which can be used for optimizing over any class of network codes, including pairwise XOR coding, reverse carpooling, and star-coding. Our approach is to specify the class of coding operations by a set of generalized links, and to develop optimization tools that apply to any network composed of such generalized links. We show that our algorithm achieves the stability for any input rates within the capacity region.

Lastly, we study the capacity of single-source single-sink noiseless networks under adversarial attack on no more than z edges. In this work, we allow arbitrary link capacities, unlike prior papers. We propose a new cut-set upper bound for the error-correction capacity for general acyclic networks. This bound tightens previous cut-set upper bounds. For example networks where the bounds are tight, we employ both linear and nonlinear coding strategies to achieve the capacity. We present a method for upper bounding the linear coding capacity of an arbitrary network and prove the insufficiency of linear network codes to achieve the capacity in general. We also show by example that there exist single-source and single-sink networks for which intermediate nodes must perform coding, nonlinear error detection or error correction in order to achieve the network capacity. This is unlike the equal link capacity case, where coding only at the source suffices to achieve the capacity of any single-source and single-sink network. A new strategy, guess-and-forward is then introduced. We first find the capacity of the two-node network by employing guess-and-forward. Guess-and-forward is also applied to the proposed family of four-node acyclic networks, showing it achieves the network capacity. Finally, for a class of so called zig-zag networks, we derive achievable rate of guess-and-forward and present conditions under which that bound is tight.

With the increasing demand for wireless multimedia services and high-speed Internet access, we expect to see increasing interest in exploiting network coding in network design, which offers both theoretical and practical benefits. The study in

this thesis only scratches the tip of the iceberg and many important problems remain to be answered.

In Section 2.1 and Section 2.2, we have developed the centralized and distributed algorithms for low-power multiple wireless unicasts. Since our proposed algorithms are defined on specific network topologies, we hope to develop a general algorithm which applies for any network topology. When a lot of unicast sessions share the wireless grid network, some links can be shared by the optimal paths solution of different sessions and thus network congestion may occur. Thus, one open problem is to formulate a new optimization framework which jointly optimizes the expected power saving using reverse carpooling and the network congestion cost.

We have developed a new low-complexity approach, network coding with periodic recomputation, in Section 2.3. We can apply our approach to other network optimization problems in MANETs to reduce the computational complexity. If the set of feasible solutions does not change every time slot, we can directly use our periodic recomputation approach. It is interesting further work to extend our results to the case where this does not hold, and to analyze the performance-complexity tradeoff of other algorithms. We can also consider another method to solve a linear program instead of interior-point method and analyze its complexity with our approach.

In Chapter 3, we have studied the problem of error correction for network codes with unequal link capacities. Further work includes characterizing the capacity region of a four-node acyclic network when the capacity of feedback links is small. It would also be interesting to study more conditions in zig-zag network under which our upper bound is tight. This may give us new achievability results. Though our new bound tightens the previous one, the cut-set bound is still not achievable in general for an example network given in [59]. Investigating the network for which there exists a gap between achievable capacity and the upper bound from our result can give us a new insight for developing tighter cut-set upper bound or another achievable strategy. It is worth thinking to apply our approach for high probability error correction with a causal adversary model in [41].

Chapter 5

Appendix

Proof of Lemma 3.17: Since the adversary controls forward links such that codewords on feedback links are unchanged, from the definition 3.14, the degree of freedom of errors that the adversary can control for any forward link l is at most $f(l)$. We prove this lemma by simply extending [31, Theorem 4] which is for the equal link capacities case to the unequal link capacities case.

Let M denote the transfer matrix whose columns are the coding vectors assigned to links. Then, the difference set is

$$\begin{aligned}
&\Delta(V,z)\\
&= \{(\theta_l(e)-\theta_l(e'))\cdot M^{-1} : l\in\Gamma_+(u), N(e)\le z, N(e')\le z\}\\
&= \{\theta_l(e-e')\cdot M^{-1} : l\in\Gamma_+(u), N(e)\le z, N(e')\le z\}\\
&= \{\theta_l(d)\cdot M^{-1} : l\in\Gamma_+(u), N(d)\le 2z\},
\end{aligned}$$

where $N(e)$ denotes the number of links error e occurs and $\theta_l(e)$ denotes the output of error vector e at the sink with zero-input.

Last equality comes from $\{e-e' : N(e)\le z, N(e')\le z\} = \{d : N(d)\le 2z\}$.

We use p to denote the maximum number of different error vectors when the

adversary controls $2z$ links. Since $\Delta(V,z) = \{\theta_l(d) \cdot M^{-1} : l \in \Gamma_+(u), N(d) \leq 2z\}$,

$$|\Delta(V,z)| \leq p \cdot \sum_{i=0}^{2z} \binom{a+b}{2z}.$$

Since t is the sum of $2z$ largest degree of freedom, $p \leq (q-1)^t$. Thus,

$$|\Delta(V,z)| \quad \leq \quad (q-1)^t \sum_{i=0}^{2z} \binom{a+b}{2z}.$$

After computing the size of the difference set $\Delta(V,z)$, we can apply exactly the same argument as in [31, Theorem 4] and complete the proof.

Proof of Lemma 3.19:

Since $b \geq 2z+1$, any claim (X_l, P_l) can be sent reliably from B to U using a repetition code. In case 1, we first show that at least one adversarial link is removed whenever sink receives some claim. From our strategy, the case that no claims are sent from B occurs only when codewords received on each feedback link are equal to $g(W)$ where W is an uncorrupted codeword sent by S to B. In case 2, we show that rate C_z is achievable even when no claims are sent from B.

Case 1) sink receives some claim (X_i, P_i).

The sink compares P_i with $g(\hat{W})$ which is received from A reliably. If $P_i \neq g(\hat{W})$, then feedback link transmitting P_i is adversarial and the sink ignores it. Otherwise, P_i is reliable. Since the claim is sent, the sink knows that $P_i = g(\hat{W}) \neq g(W)$ and that guess X_i is correct. Thus the sink identifies at least one adversarial link between S and A, which is subsequently ignored.

Therefore, in this case, the sink removes one bad link whenever B sends claims.

Case 2) no claims are sent.

From our strategy, the case that no claims are sent from B occurs only when codewords received on each feedback link are equal to $g(W)$ where W is uncorrupted codeword sent by S to B. Since A transmits $g(\hat{W})$ to B, there are three following possibilities in this case.

a) All links between S and A and all feedback links are uncorrupted. Then $\hat{W} = W$ and $g(W)$ is reliably transmitted.

b) Some links between S and A are corrupted so that A receives $g(\hat{W}) \neq g(W)$ from S, but the adversary controls all feedback links such that each feedback link changes $g(\hat{W})$ to $g(W)$.

c) Some links between S and A are corrupted such that codewords A sends along each feedback link are unchanged, i.e., $\hat{W} \neq W$ and $g(\hat{W}) = g(W)$. All feedback links are reliable and B receives $g(W)$.

If a) is true, all links between S and A and all feedback links are uncorrupted. Then there exists a set of $(a+b-z)$ forward links on the cut such that this set includes all a links between S and A and some $b-z$ links between B and U, and gives consistent output with rate C_z. (Note that the sum of capacities of any $(a+b-z)$ forward links is larger than or equal to C_z from the definition of our bound in Theorem 3.6.) If b) is true, all m feedback links are corrupted. Then there exist a set of $(a+b-z+m)$ forward links on the cut that gives consistent output with rate C_z. If c) is true, then we obtain the correct output from (C, C_z) MDS code in Lemma 3.17.

Based on the above analysis, we give following simple decoding algorithm. We prove the correctness of this algorithm as follows.

Algorithm 3 Decoding algorithm for achieving rate C_z when no claims are sent.

IF there is a set L_1 which is composed of $a+b-z+m$ forward links and gives consistent output,
  THEN the output with rate C_z from L_1 is correct.
ELSE IF there is a set L_2 which is composed of all a forward links from S to A and some $b-z$ forward links from B to U, and gives consistent output,
  THEN the output with rate C_z from L_2 is correct.
ELSE the output with rate C_z obtained from (C, C_z) MDS code is correct.
END IF

Case 2 - a) there is a set L_1 composed of $(a+b-z+m)$ forward links giving consistent output.

In this case, we show that output with rate C_z obtained from L_1 is correct. First we prove that output from L_1 is correct when b) or c) is true. If b) is true, all m feedback links are corrupted and thus L_1 contains at least $(a+b-z+m)-(z-m) =$

$a + b - 2(z - m)$ uncorrupted links. From the definition of our upper bound in Lemma 3.4, the sum of capacities of any $a+b-2(z-m)$ forward links is larger than or equal to C_z. Since L_1 gives consistent output, the output is correct. If c) is true, L_1 contains at most z corrupted links. From the condition 3.16, for any set $A_1 \subset L_1$ composed of z links, the sum of degree of freedom of z links in A_1 plus the sum of capacities of $z - m$ forward links not included in L_1 is less than or equal to C_z, i.e., $\sum_{l\in A_1} f(l)+\sum_{l\in\mathcal{E}-L_1} r(l) \leq C-C_z$. Thus, L_1 contains at least rate $C-(C-C_z) = C_z$ uncorrupted output and the output is correct.

Case 2 - a - i) there is no set of $(a + b - z)$ links that includes all a forward links from S to A and gives consistent output.

In this case, a) cannot hold and thus b) or c) is true. Thus output from L_1 is correct.

Case 2 - a - ii) there exists a set L_2 composed of $(a + b - z)$ forward links that includes all a links from S to A and gives consistent output.

We first show that L_1 and L_2 gives the same consistent output. $L_1 \cap L_2$ is obtained from the cut by erasing z forward links from B to U that L_2 does not include and $z-m$ forward links L_1 does not include. From the definition of our bound in Lemma 3.5, $\sum_{l\in L_1\cap L_2} r(l) \geq C_z$. Thus L_1 and L_2 give the same consistent output. Since L_2 gives the correct output when a) is true, and L_1 and L_2 give the same consistent output in this case, output from L_1 is correct when a) is true. Moreover, we have already shown that L_1 gives the correct output if b) or c) is true. Therefore, L_1 always gives correct output.

Case 2 - b) there is no set of $(a+b-z+m)$ forward links giving consistent output.

In this case, there are more than $z-m$ adversarial forward links on the cut. Thus b) cannot hold and a) or c) is true. If there is no set of $(a+b-z)$ forward links that includes all a links from S to A and gives consistent output, then c) is true. From Lemma 3.17, output obtained from (C, C_z) MDS code is correct. Otherwise, suppose that there exists a set L_2 composed of $(a + b - z)$ forward links that includes all a links from S to A and gives consistent output. We show that output obtained from L_2 is correct.

If a) is true, since all links between S and A and all feedback links are uncorrupted, L_2 contains at most z corrupted forward links between B and U. From the definition of the Singleton bound, the sum of capacities of a links between S and A plus sum of any $b-2z$ forward links between B and U is larger than or equal to C_z. Thus, from uncorrupted a links between S and A and some $b-2z$ links between B and U which are not corrupted, we obtain the correct output rate C_z. Since L_2 gives consistent output, the decoded output is correct.

If c) is true, the adversary controls some forward links from S to A such that each feedback link transmits $g(W)$, and L_2 contains at most z unknown corrupted links. From condition 3.15, the sum of any $2z$ degree of freedom of links are less than or equal to $C-C_z$. Since degree of freedom of any forward link from B to U is equal to the link capacity, the sum of degree of freedom of z truly corrupted links in L_2 and the sum of z forward links between B and U which are not included in L_2 is less than or equal to $C-C_z$. Therefore, L_2 contains at least $C-(C-C_z)=C_z$ uncorrupted symbols. Since L_2 gives consistent output, the decoded output from L_2 is correct.

Therefore, either a) or c) is true, L_2 gives the correct output.

Proof of Lemma 3.20:

Since $b \leq 2z$, a claim (X_l, P_l) for any feedback link l is not reliably transmitted to the sink and adversarial links between B and U can corrupt this claim arbitrarily. Thus, the sink can receive different claims on different incoming links. Let $G(l)$ be the set of distinct claims $\{(X_{l1}, P_{l1}), .., (X_{lk}, P_{lk}), Y\}$ where Y denotes that no claims are received. Here is the outline of the proof. We first show that at least one adversarial link is removed except when $b > z$ and the sink receives no claim on all b links for all feedback links. When $b > z$ and the sink receives no claim on all b links, since all b links cannot be corrupted at the same time, the sink knows that B does not send any claim. This case exactly corresponds to the case 2 in the proof of Lemma 3.19 and we achieve the correct output. This completes the proof. Note that the same guess-and-forward strategy in Section 3.5.2.2 is used.

First we show that any uncorrupted $(a+b-2z)$ forward links between S and A give the correct decoded output with rate C_z. From the definition of Singleton bound,

after erasing $b \leq 2z$ links between B and U and any set of $2z - b$ links between S and A, the sum of the remaining link capacities are larger than or equal to C_z. Thus any uncorrupted $(a + b - 2z)$ links between S and A give the correct message.

Now we assume that (X_{li}, P_{li}) is received on n_i links and Y is received on n_{k+1} links $(n_1 + .. + n_{k+1} = b)$. First we ignore any (X_{li}, P_{li}) claiming that there are more than $z - (b - n_i)$ adversarial links between S and A. Since X_{li} is shown on n_i links, believing X_{li} implies more than z adversarial links on the cut which is a contradiction. Thus, each of remaining claim (X_{lj}, P_{lj}) specifies a set L_j which is composed of at least $(a - (z - (b - n_i))) = a + b - z - n_i$ links between S and A claimed to be correct by (X_{lj}, P_{lj}). For each such claim, we check the consistency of the decoded outputs of L_j. We show that if there exist two different claims (X_{li}, P_{li}) and (X_{lj}, P_{lj}) both corresponding to consistent outputs, then those two outputs should be the same. Since $|L_i| = a + b - z - n_i$, $|L_j| = a + b - z - n_j$, and $|L_i \cup L_j| \leq a$,

$$\begin{aligned} |L_i \cap L_j| &\geq (a + b - z - n_i) + (a + b - z - n_j) - a \\ &\geq a + b - 2z. \end{aligned}$$

As we mentioned at the beginning of the proof, the sum of capacities of any $(a+b-2z)$ link between S and A is larger than or equal to C_z. Therefore L_i and L_j give the same consistent output.

Suppose that we have figured out that a set of links L gives the correct consistent decoded output. In this case, we add remaining links not included in L sequentially to L, and check the consistency of any decoded output with rate C_z. If outputs are no more consistent, the added link is adversarial (*).

Now we show that at least one adversarial link is removed except when $b > z$ and the sink receives no claim on all b links for all feedback links.

Case 1) all claims are ignored or none of the remaining claims give consistent output or all claims (X_{li}, P_{li}) that give consistent output satisfy that $P_{li} \neq \hat{W}$

In this case, there are only two possibilities.

a) Feedback link l is adversarial.

b) Feedback link l is reliable and all b links between B and U are adversarial.

If $b > z$, then b) cannot hold and feedback link l is adversarial. We remove it. If $b \leq z$, the sink checks the consistency of outputs from each set of $(a + b - z)$ links between S and A. If no $(a+b-z)$ links set give consistency, then there are more than $z - b$ adversarial links between S and A. Thus a) is true and we remove feedback link l. Otherwise, there exists a set L of $(a + b - z)$ links giving consistency. Since this set contains at most z corrupted links, and thus includes at least $(a + b - 2z)$ uncorrupted links between S and A. Then the sum of capacities of uncorrupted links are larger than or equal to C_z. Thus L gives correct output rate C_z. From (*), we can detect forward adversarial links in this case.

Case 2) there exists a claim (X_{li}, P_{li}) giving consistent output and $P_{li} = \hat{W}$.

We show that output obtained from claim (X_{li}, P_{li}) should be correct. If there is at least one uncorrupted link showing (X_{li}, P_{li}), then feedback link l is also not corrupted since $P_{li} = \hat{W}$, and this claim gives correct output rate C_z. Otherwise, if all n_i links showing this claim are adversarial, in which case, there are at most $z - n_i$ adversarial links between S and A. Then L_i includes at least $(a+b-z-n_i)-(z-n_i) = a+b-2z$ uncorrupted links, and thus give correct consistent output. From (*), we can also detect adversarial links in this case.

Case 3) only Y gives consistent output and $b < z$.

In this case, a set of all a forward links from S to A gives consistent output. a links between S and A include at least $a - z \geq a + b - 2z$ uncorrupted links since $b < z$. Thus we obtain correct consistent output from a links and detect adversarial links from (*).

Case 4) only Y gives consistent output, $b > z$, and at least one of b links between B and U show claim different from Y, i.e., $n_{k+1} < b$.

Case 4 - a) $n_{k+1} < b - z$.

If feedback link l is reliable, the links showing claims different from Y are adversarial. Thus there are more than $b - n_{k+1} > z$ adversarial links and it contradicts. Thus feedback link l is adversarial and we remove it.

Case 4 - b) $b - z \leq n_{k+1} \leq z$.

Y is shown on $n_{k+1} \geq b - z$ links and $b - n_{k+1} \geq b - z$ links show claims different from Y. Thus there are at least $b - z$ adversarial links between B and U. Then there are at most $2z - b$ adversarial links between S and A and at least $a + b - 2z$ uncorrupted links. Thus we also obtain correct output from a links and use (*) to detect adversarial links.

Case 4 - c) $z < n_{k+1} < b$.

Since $n_{k+1} > z$, feedback link l transmits $g(\hat{W}) = g(W)$ to B and B does not send any claim. Thus, the links showing claims different from Y are all adversarial.

For cases 1-4, we have shown that at least one adversarial link is removed when $b > z$ and the sink receives some claim different from Y for any feedback link.

To complete the proof it is now sufficient to show that correct output can be achieved when $b > z$ and the sink receives no claim for all feedback links l. Since $b > z$, at least one link between B and U is uncorrupted. Since all b links show Y, this means that each feedback link transmits $g(\hat{W}) = g(W)$ and B does not send any claim. This case corresponds to the case 2 in Lemma 3.19. Therefore, we can obtain the correct output.

Bibliography

[1] R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[2] S. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[3] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 5, pp. 782–795, 2003.

[4] T. Ho, M. Médard, J. Shi, M. Effros, and D. Karger, "On randomized network coding," in *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, vol. 41, no. 1. Citeseer, 2003, pp. 11–20.

[5] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

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

[7] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: practical wireless network coding," *IEEE/ACM Transactions on Networking (TON)*, vol. 16, no. 3, pp. 497–510, 2008.

[8] J. Park, D. Lun, F. Soldo, M. Gerla, and M. Médard, "Performance of network coding in ad hoc networks," in *Proc. IEEE Milcom*. Citeseer, 2006.

[9] Z. Li, B. Li, D. Jiang, and L. Lau, "On achieving optimal throughput with network coding," in *IEEE INFOCOM*, vol. 3. Citeseer, 2005, p. 2184.

[10] M. Effros, T. Ho, and S. Kim, "A tiling approach to network code design for wireless networks," in *IEEE Information Theory Workshop*, 2006, pp. 62–66.

[11] Y. Wu, P. Chou, and S. Kung, "Information exchange in wireless networks with network coding and physical-layer broadcast," in *Proc. of CISS*. Citeseer, 2005.

[12] ——, "Minimum-energy multicast in mobile ad hoc networks using network coding," *IEEE Transactions on communications*, vol. 53, no. 11, pp. 1906–1918, 2005.

[13] Y. Sagduyu and A. Ephremides, "Some optimization trade-offs in wireless network coding," in *Proc. of CISS*. Citeseer, 2006.

[14] ——, "Cross-layer optimization of MAC and network coding in wireless queueing tandem networks," *IEEE Transactions on Information Theory*, vol. 54, no. 2, pp. 554–571, 2008.

[15] T. Ho and D. Lun, *Network coding: an introduction*. Cambridge University Press, 2008.

[16] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Medard, "The importance of being opportunistic: Practical network coding for wireless environments," in *Proc. 43rd Annual Allerton Conference on Communication, Control, and Computing*, 2005.

[17] D. Lun, N. Ratnakar, M. Médard, R. Koetter, D. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, p. 2623, 2006.

[18] B. Awerbuch, A. Brinkmann, and C. Scheideler, "Anycasting and multicasting in adversarial systems," *Unpublished manuscript, March*, 2002.

[19] T. Klein and H. Viswanathan, "Centralized power control and routing policies for multihop wireless networks," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 849–866, 2006.

[20] M. Neely, "Energy optimal control for time-varying wireless networks," *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 2915–2934, 2006.

[21] M. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 16, no. 2, pp. 396–409, 2008.

[22] M. Neely, E. Modiano, and C. Rohrs, "Packet routing over parallel time-varying queues with application to satellite and wireless networks," in *PROCEEDINGS OF THE ANNUAL ALLERTON CONFERENCE ON COMMUNICATION CONTROL AND COMPUTING*, vol. 39, no. 2. The University; 1998, 2001, pp. 1110–1111.

[23] ——, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, pp. 89–103, 2005.

[24] H. Viswanathan and K. Kumaran, "Rate scheduling in multiple antenna downlink wireless systems," *IEEE Transactions on Communications*, vol. 53, no. 4, pp. 645–655, 2005.

[25] E. Yeh and A. Cohen, "Throughput and delay optimal resource allocation in multiaccess fading channels," in *IEEE INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY*, 2003, pp. 245–245.

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

[27] T. Ho and H. Viswanathan, "Dynamic Algorithms for Multicast with Intra-session Network Coding," *Information Theory, IEEE Transactions on*, vol. 11, no. 5, pp. 782–795, 2009.

[28] A. Khreishah, C. Wang, and N. Shroff, "Optimization based rate control for communication networks with inter-session network coding," in *Proc. IEEE Infocom*, 2007.

[29] A. Eryilmaz and D. Lun, "Control for inter-session network coding," in *Proc. Workshop on Network Coding, Theory & Applications*. Citeseer, 2007.

[30] R. Yeung and N. Cai, "Network error correction, part I: Basic concepts and upper bounds," *Communications in Information and Systems*, vol. 6, no. 1, pp. 19–36, 2006.

[31] N. Cai and R. Yeung, "Network error correction, part II: Lower bounds," *Communications in Information and Systems*, vol. 6, no. 1, pp. 37–54, 2006.

[32] T. Ho, B. Leong, R. Koetter, M. Médard, M. Effros, and D. Karger, "Byzantine modification detection in multicast networks using randomized network coding," in *IEEE International Symposium on Information Theory*, 2004, pp. 144–144.

[33] Z. Zhang, "Linear network error correction codes in packet networks," *IEEE Transactions on Information Theory*, vol. 54, no. 1, pp. 209–218, 2008.

[34] C. Gkantsidis and P. Rodriguez, "Cooperative security for network coding file distribution," in *IEEE INFOCOM*. Citeseer, 2006, pp. 2004–137.

[35] M. Krohn, M. Freedman, and D. Mazieres, "On-the-fly verification of rateless erasure codes for efficient content distribution," in *IEEE Symposium on Security and Privacy*. Citeseer, 2004, pp. 226–240.

[36] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard, "Resilient network coding in the presence of byzantine adversaries," *benefits*, vol. 16, p. 6.

[37] G. Liang, R. Agarwal, and N. Vaidya, "Non-Linear Network Coding against Byzantine Adversary: Part I."

[38] ——, "When watchdog meets coding."

[39] R. Koetter and F. Kschischang, "Coding for errors and erasures in random network coding," *Arxiv preprint cs/0703061*, 2007.

[40] D. Silva, F. Kschischang, and R. Koetter, "A rank-metric approach to error control in random network coding," *IEEE Transactions on Information Theory*, vol. 54, no. 9, pp. 3951–3967, 2008.

[41] M. Langberg, S. Jaggi, and B. Dey, "Binary Causal-Adversary Channels," *Imprint*, 2009.

[42] O. Kosut, L. Tong, and D. Tse, "Nonlinear network coding is necessary to combat general byzantine attacks," in *47th Annual Allerton Conference on Communication, Control, and Computing*, 2009.

[43] S. Kim, T. Ho, M. Effros, and S. Avestimehr, "Network error correction with unequal link capacities," in *Communication, Control, and Computing, 2009 47th Annual Allerton Conference on*.

[44] ——, "New results on network error correction : capacities and upper bounds," *ITA 2010, San Diego*.

[45] J. Widmer, C. Fragouli, and J. Le Boudec, "Low-complexity energy-efficient broadcasting in wireless ad-hoc networks using network coding," in *Proc. Workshop on Network Coding, Theory, and Applications*, 2005.

[46] K. Bharath-Kumar and J. Jaffe, "Routing to multiple destinations in computer networks," *Communications, IEEE Transactions on [legacy, pre-1988]*, vol. 31, no. 3, pp. 343–351, 1983.

[47] B. Waxman, "Routing of multiple connections," *IEEE Journal on selected areas in Communications*, vol. 6, no. 9, pp. 1617–1622, 1988.

[48] P. Winter, "Steiner problem in networks: a survey," *Networks*, vol. 17, no. 2, pp. 129–167, 1987.

[49] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li, "Approximation algorithms for directed Steiner problems," *Journal of Algorithms*, vol. 33, no. 1, pp. 73–91, 1999.

[50] L. Zosin and S. Khuller, "On directed Steiner trees," in *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2002, pp. 59–63.

[51] S. Ramanathan, "Multicast tree generation in networks with asymmetric links," *IEEE/ACM Transactions on Networking (TON)*, vol. 4, no. 4, p. 568, 1996.

[52] O. Oguz, "Bounds on the opportunity cost of neglecting reoptimization in mathematical programming," *Management Science*, vol. 46, no. 7, pp. 1009–1012, 2000.

[53] S. Kim, M. Effros, and T. Ho, "On Low-Power Multiple Unicast Network Coding Over a Wireless Triangular Grid," in *Forty-Fifth Annual Allerton Conference*. Citeseer, 2007.

[54] ——, "Distributed design of network codes for wireless multiple unicasts," in *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, 2008, pp. 324–331.

[55] N. Ratnakar, R. Koetter, and T. Ho, "Linear flow equations for network coding in the multiple unicast case," in *DIMACS workshop on network coding*, 2005.

[56] N. Ratnakar, D. Traskov, and R. Koetter, "Approaches to network coding for multiple unicasts," in *Communications, 2006 International Zurich Seminar on*, pp. 70–73.

[57] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," in *IEEE INFOCOM 2003. Twenty-Second*

Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1.

[58] R. Yeung, *Information theory and network coding*. Springer, 2008.

[59] O. Kosut, L. Tong, and D. Tse, "Polytope Codes Against Adversaries in Networks."