Performance of Random Network Coding for Data Dissemination

by

Clifford Choute

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 3, 2005

Author .....
Department of Electrical Engineering and Computer Science
May 19, 2005

Certified by .....
Muriel Médard
Associate Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Performance of Random Network Coding for Data Dissemination

by

Clifford Choute

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2005, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Network coding is an alternative to traditional store-and-forward routing and is known to be necessary to achieve network capacity. It has also been shown randomized network coding is robust, and far outperforms store-and-forward for multicast. While much focus has been on the data rates achievable with coding, we focus on the time needed to broadcast a finite amount of data throughout networks using distributed randomized linear coding. We consider networks with increasingly complex graphs. We use analysis of the dissemination time using coding in the line network to discuss the performance of coding in networks with more complex topologies, such as the Manhattan grid network.

Thesis Supervisor: Muriel Médard
Title: Associate Professor

Acknowledgments

I would like to thank Dr. Supratim Deb, Professor Muriel Médard and her entire group at the Lab for Information and Decision Systems at M.I.T. (specifically Todd Coleman, Tracey Ho, and Anna Lee). I would also like to thank Professor Eytan Modiano, Professor Lizhong Zheng, Nikhila Deo, Carri Chan, Dian Chen, and all of my colleagues, friends, and family.

Contents

List of Figures

Introduction

Linear network coding, first introduced by Ahlswede et al. [1], is an alternative to traditional store-and-forward routing. In general, it is necessary to achieve network capacity [7]. It has been shown that randomized network coding is robust, and far outperforms store-and-forward for multicast [5, 6].

While much attention has been devoted to discussion of the data rates achievable with coding, this work focuses on the time to distribute a finite amount of data throughout a network. Building on previous work, we consider network graphs of increasing complexity, starting with a simple graph of nodes in series. Rather than capacity, we look at the time to disseminate all information in the network, and compare the performance of coding and store-and-forward to bounds on this time.

In [3, 4], Deb, Médard, and Choute analyze the performance of network coding for information distribution in fully connected networks. In this work we consider networks with more realistic topologies. As shown by Pereira in [8], this will involve more than a direct application of the technique used by Deb and Médard. Pereira specifically shows that this technique does not lead to an asymptotically tight bound on dissemination time (our key performance metric, as defined in Section 1.2) for the ring network. Consequently, new tools must be developed to properly characterize performance in general networks.

Our problem is the efficient dissemination of information throughout decentralized networks. Our task is to create tools to help us analyze the effectiveness of network

coding in addressing this problem, as well as to assess the utility of these tools.

To this end, Chapter 1 provides an introduction to network coding, with some examples and discussion of the merits and drawbacks of employing coding to address this problem. In Chapter 2 we get our first look at the performance of network coding as we review the analysis and simulation of both coding and routing in the fully connected network.

Chapter 3 contains the analysis of the simple line problem, which forms the basis of our analysis of general networks. In Chapter 4 we use results from the line problem to analyze the performance of coding in Manhattan grid network and to develop a model to estimate the performance of coding in general networks.

In Chapter 5 we develop a notion of equivalence, describing the effectiveness of network coding. We conclude with a few remarks on possible applications of network coding, as well as on implementation issues. We also consider future investigation into the performance of coding.

Chapter 1

Linear Network Coding

1.1 Overview

In this section we describe random linear network coding. By way of example we present the classic butterfly network. We then give a detailed description of the transmission protocol we operate under for this work. The transmission protocol governs the time and the recipient of each transmission. We also present the two information protocols we consider. The information protocol determines the content of each transmission. Random linear network coding is one of these protocols. Finally we define and discuss our main performance metric, the mean dissemination time.

1.1.1 The Butterfly Network

The classic network coding example is the butterfly network shown in Figure 1-1. Note that all the links in this network are unidirectional, so the network graph is directed. We assume, for this example only, that a member of the network may transmit to multiple neighbors at once. Observe that with store-and-forward routing (Figure 1-1a) the central link must be shared and therefor limits the capacity of the network. However, if we were to send the bitwise XOR of the two streams (Figure 1-1b), we can achieve capacity at the expense of some overhead and computation. This is deterministic coding because the coefficients of the linear combination are chosen deterministically. With random linear coding the coefficients are chosen according to a uniform distribution over a finite field and must be included with the encoded message as constant overhead. This is useful for networks where the overall topology and sources of information are not known to the members.

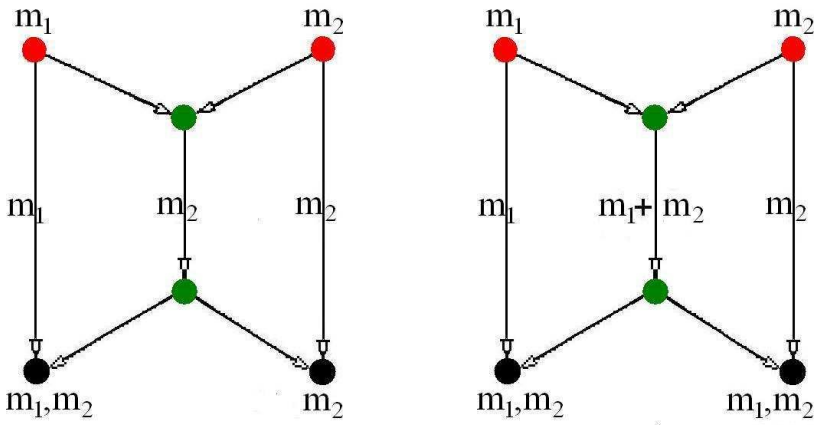


Figure 1-1: The Butterfly Network: Store-and-Forward vs. Coding: The sources are red nodes, the sinks are black nodes, and the relays are green nodes. All links are directed.

1.2 Description

1.2.1 Transmission Protocol

We model networks as graphs, where nodes represent members of the network and edges represent communication links. We use the names *member* and *node* interchangeably throughout the paper as our discussion switches between networks and graphs. If there is a direct link between two nodes, we say they are neighbors.

For simplicity, time is partitioned into constant time steps, called rounds, where the duration of a round is assumed sufficient to accommodate any transmission. Each member sends at most one message to one of its neighbors per round and there are no restrictions on the number of messages received in a round. Because network topology is assumed to be unknown by the members, each member in general will not have prior information about the direction in which its information needs to be sent. Each member chooses which neighbor to transmit to with equal probability, independent of the member and round. We assume that no errors occur in transmission.

The networks of interest meet the following requirements. Each network has n members and k initial messages where k is $O(n)$. Each network is decentralized, but we assumed that each member knows k and the number of neighbors it has. The network graph must be connected and the majority, if not all, of the nodes must have the same degree D. Finally, each network is two-dimensional and each link is assumed to have negligible delay and sufficient capacity.

In effect, network coding involves linearly "mixing" (in an algebraic sense detailed in the next paragraph) the information that is to be transmitted, while keeping track of how it is mixed so that the information can be recovered. The basic tradeoff is that this "mixing" increases the average usefulness of each transmission but requires a small overhead per transmission and also some computation by members acting as sources, relays, and sinks (sinks must decode the information).

1.2.2 Information Protocols

The following protocols determine the content of each message sent in network. We consider two protocols: Random Message Selection (RMS) and Random Linear Network Coding (RLC).

Random Message Selection

The RMS protocol is simply random store-and-forward. Once a recipient is chosen among the neighbors, each member simply selects one of messages it already has, at random, and sends that message to its neighbor. The only overhead required is a message identifier.

Random Linear Network Coding

The RLC protocol is more complex. Each original message m_j is modeled as a vector of constant length r over a finite field, $\mathbb{F}_q$, of size q ($|\mathbb{F}_q| = q$) and is accompanied by an original code vector β_j of length k over $\mathbb{F}_q$. It should be noted that if the field size is chosen to be 2^u for some positive integer u, typical binary data can easily be split into segments of size q and the finite field operation of addition can be implemented by bitwise XOR. We implement multiplication through table lookup. The β_js must be linearly independent and are most practically chosen to be a canonical basis. At each round, the message and code vector that node i transmits are the same linear combination of the messages v_{ij} and code vectors α_{ij} already at the node at the start of the round. Thus, if node i transmits to node g, then the transmitted message vector is $\sum_j \gamma_j v_{ij}$ and the transmitted code vector is $\sum_j \gamma_j \alpha_{ij}$.

At this point, node g must decide whether to keep this message. We say an arriving message is *helpful* to a node if it provides new information. This corresponds to the message's code vector being outside of the space spanned by the code vectors at the recipient node. A *node* is helpful to another node if it is capable of sending a helpful message. This occurs when the code vector space of the sender is not spanned by the that of the receiver. The rank r_i of node i, for convenience, is the number of messages

v_{ij} with linearly independent code vectors α_{ij} at the node, so $j \leq r_i \leq k$. So r_i is the rank of the code vector matrix (the code vectors are the rows) at node i. If the new message is helpful to node g it becomes v_{gh} (and code vector α_{gh}) and is stored with the other messages at node g, whose rank increases by 1.

In randomized network coding, the coefficients γ_j of this linear combination are chosen randomly from the finite field with equal probability $\frac{1}{q}$. For this case, Ho et al. [5] show that, if a node receives a vector from a helpful node, then the probability that the vector is helpful is at least $(1 - \frac{1}{q})$. When the rank of a node reaches k, the information has been completely disseminated and the original messages can be recovered[1] by inverting the code matrix and multiplying by the received messages.

The Butterfly Network

Returning to the Butterfly network, we show coding and random coding in action[2].

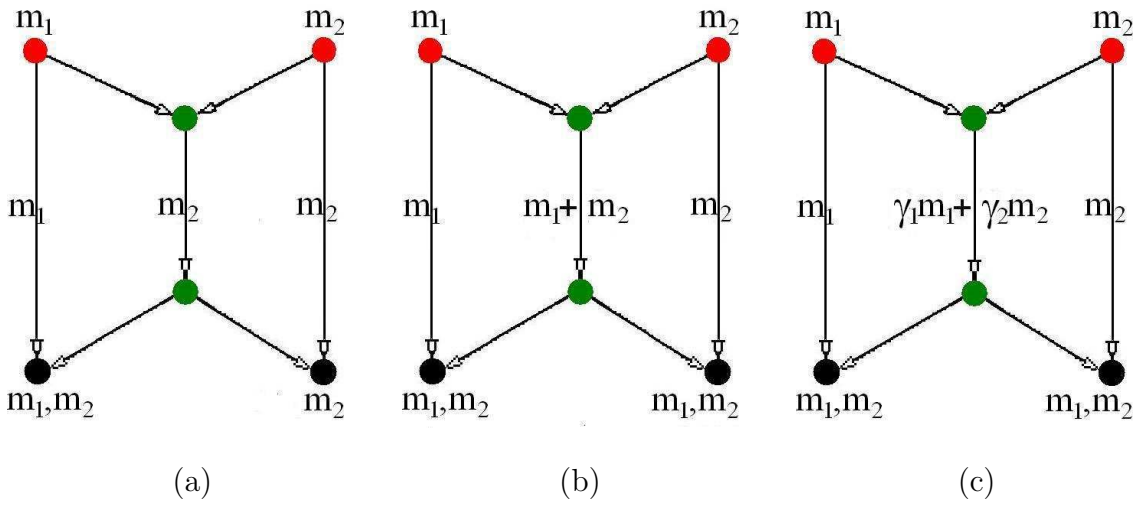


Figure 1-2: The Butterfly Network: (a) Store-and-Forward (b) Coding (c) Random Coding: The γ_i are chosen at random.

[1]If, for some reason, a node never reaches full rank k, the network member may still be able to recover part of the data, if its code vector matrix has a sub-matrix of full rank.

[2]It should be noted that we are operating under a different transmission protocol in our butterfly network example. A member may transmit to more than one recipient at once. Moreover we are primarily concerned with information flow and network capacity (as opposed to messages and dissemination time) in this example.

As illustrated in Figure 1-2c, the sender over the central link sends a linear combination, $v_2 = \gamma_1 m_1 + \gamma_2 m_2$ of the streams he is receiving. The coefficients γ_1 and γ_2 must be sent with the encoded data because they are chosen at random. Thus, the sender (the higher center node) is operating the RLC protocol. The lower left node, having received both v_2 and v_1, now has a matrix of code vectors with full rank, namely $\left[\begin{smallmatrix} 1 & 0 \\ \gamma_1 & \gamma_2 \end{smallmatrix}\right]$. This code matrix can be inverted and multiplied by the received data v_i to recover the original data m_i as in equation (1.1b). Equation (1.1b) shows the same process for the deterministic coding example shown in Figure 1-2b [3].

$$\begin{bmatrix} 1 & 0 \\ \gamma_1 & \gamma_2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} v_1 = m_1 \\ v_2 = \gamma_1 m_1 + \gamma_2 m_2 \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} \tag{1.1a}$$

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} v_1 = m_1 \\ v_2 = m_1 + m_2 \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} \tag{1.1b}$$

1.3 Performance

We are interested in multiple source, multicast networks. Specifically, we want to observe the amount of time it takes for all sources to distribute all their data to every other source, in other words the dissemination time, T_d. We will use this key performance metric to compare the two information protocols described in Section 1.2.2. Of course, both protocols have random input so what we will focus on asymptotic bounds of the expected value, $\overline{T}_d$.

One motivation for characterization of dissemination time and use of it as a performance metric is that network operation under network coding is not continuous. Each member needs to know how many messages are in the network initially so that it knows when it has all the data. Also, T_d must be used to determine some timeout,

[3] Though of course in this simple example information could be recovered with another bitwise XOR, significantly reducing computation

T_r, when all the network members may stop forwarding linear combinations to its neighbors and the algorithm can restart. T_r should be greater than T_d with some high probability to ensure that most or all of the messages are completely disseminated with high probability. The choice of T_r in turn affects average data rates. In particular, we see that the choice of T_r introduces a tradeoff between maintaining low average idle time, $T_r - \overline{T}_d$, and reducing the number of necessary retransmissions.

Chapter 2

Fully Connected Networks

2.1 The Problem

We now describe the gossip problem: a fully connected network in which each member requires all of the information starting in the network. Each member has a direct link with every other member, so the network graph forms a complete graph. We restrict the number of messages, k, to be $\mathrm{O}(n)$. From previous work, we know that $T_d = \Omega(n)$ even with full knowledge and control over the network. Without this knowledge and control, coding achieves this order bound, whereas store-and-forward routing takes $\Omega(n \ln n)$ time [3]. Coding also outperforms store-and-forward when $k \neq \mathrm{O}(n)$ [4]. In this chapter, we present simulation results and discuss results for the fully connected network.

2.2 Simulation

This section outlines the basics of the simulation. In Section 2.2.2, we explain and interpret the results of the simulation, which are displayed graphically in Figure 2-1.

2.2.1 Overview

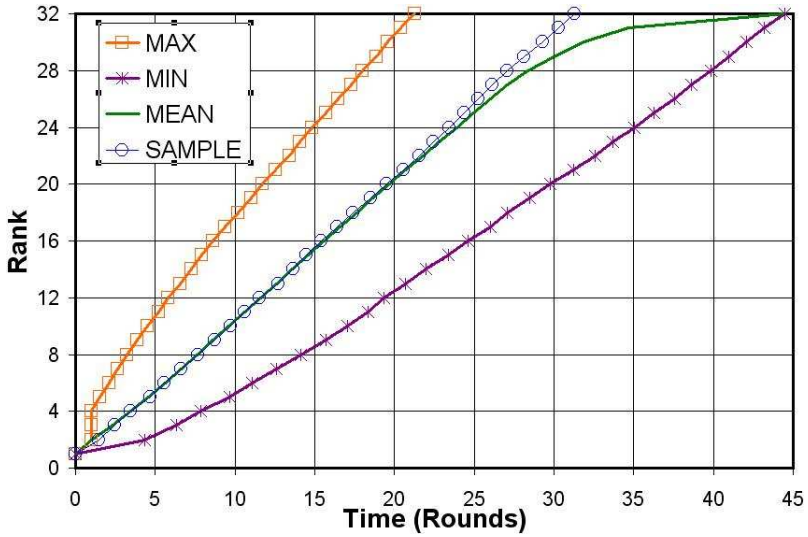
Simulations have been conducted for networks with 4, 8, 16, or 32 members. The number of messages, k, is taken to be 2,4, $\frac{n}{2}$, or n. The field size q is set to k. When $k < n$, only k members start with a message so that initially there is exactly one copy of each message in the network. The network is initialized accordingly, so each member starts with 1 or 0 messages. 100 iterations are run for each scenario.

In each round of the simulation, each of the n nodes randomly chooses one of the other $n - 1$ nodes and sends a linear combination of the messages it has previously received along with the coefficients of that combination. The recipient concatenates the new code vector to its code matrix[1] and reports whether the rank of the matrix has increased, keeping the message only if it is helpful. Thus, we can follow the growth of the rank dynamically. The simulation ends when each node's code matrix has rank k. It then verifies that the messages are decoded successfully.

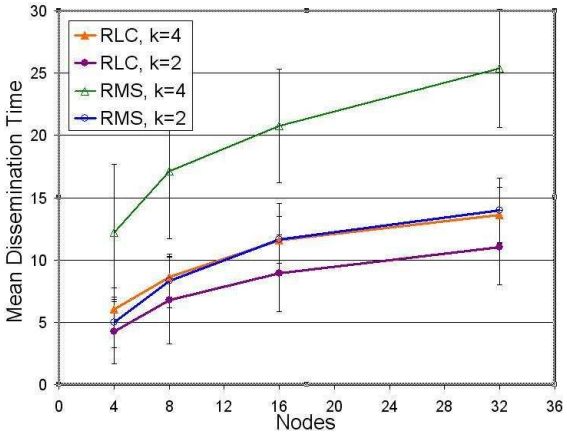
2.2.2 Results

We are interested in the time (in rounds) it takes to disseminate the original messages. Each round, the simulation notes the rank of each matrix. After all the iterations are complete the simulation uses the recorded values to find the average number of rounds for the maximum, minimum, and average rank to reach r for $1 \leq r \leq k$. It also looks at one sample node and reports the evolution of its rank. Figure 2-1a is a graph of this data for $n = k = q = 32$. Finally, the simulation computes the mean and the variance of the total dissemination time. This is graphed for various values of n and k in Figure 2-1b,c.

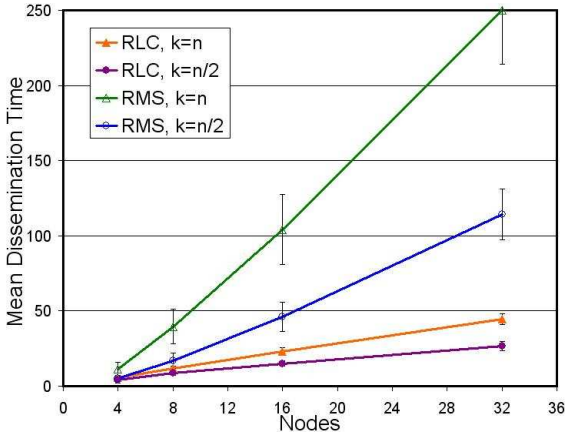
[1]Code vector matrix. This is a growing matrix with the received code vectors as rows.



(a)



(b)



(c)

Figure 2-1: The Fully Connected Network: (a) Rank vs. time for $n = k = q = 32$; (b) $\overline{T}_d$ vs. the number of nodes (members) with $k = 2$, 4; (c) $\overline{T}_d$ vs. the number of nodes with $k = \frac{n}{2}$, n.

The data indicate that, when $k = \Theta(n)$, the information is indeed disseminated in $\Theta(n)$ time under the RLC protocol and in $\Theta(n \ln n)$ time under the RMS protocol. We also note that the average rank in the network grows linearly with time under RLC.

Chapter 3

Line Networks

3.1 The Problem

Two simple networks to be considered are the line network and ring network. Simulations indicate that dissemination time for both networks under the RLC protocol is $\Theta(n)$ if $k = \Theta(n)$, regardless of initial location of the information in the network. We consider a number initial scenarios, including one where each member starts with one message, so $k = n$. Characterization of the performance of coding for these simple networks gives insight into the problem of dissemination over more general topologies. For example, in our analysis of the Manhattan grid network in Chapter 4, we use results from analysis of the line network to find a bound on dissemination time. We give a preview of the general utility of this analysis in Section 3.3.

3.2 Analysis

Figure 3-1: The Line Network with One Source: The source is the leftmost node (shaded red), and all members are sinks.

3.2.1 The Line Network with $k = 1$

We consider the performance of the RLC protocol in the line network with one source and one message. The dissemination time is (neglecting edge effects) a negative binomial random variable with parameters $n-1$ and $\frac{1}{2}(1-\frac{1}{q})$, or $T_d \sim NB(n-1, \frac{1}{2}(1-\frac{1}{q}))$. Thus,

$$\overline{T}_d = \frac{n-1}{\frac{1}{2}(1-\frac{1}{q})} = \Theta(n) \tag{3.1}$$

3.2.2 The Line Network with $k > 1$

If we consider the same scenario but with $k > 1$, the analysis is not so simple. We utilize queuing theory to show that the mean dissemination time for the line network (shown in Figure 3.2) is $\overline{T}_d = \Theta(n+k)$. The network consists of n nodes (members) in series with k messages originating at the leftmost node, node 1.

To simplify our discussion, we define the rank and space of node i to be the rank and space of the matrix of code vectors that describe the messages at that node, respectively. We also say that node i is "helpful" to node j if node j's space does not span node i's space. That is, there exists some message at node i whose code vector is outside the code vector space of node j. Also any message whose corresponding code vector is not in the space of node j is also said to be "helpful" to node j.

In the case of the line network, the code vectors at node $i+1$ always span the space of the code vectors at i. All transmission against the direction of propagation is useless and we are only interested in the case where i is useful to $i+1$. When this

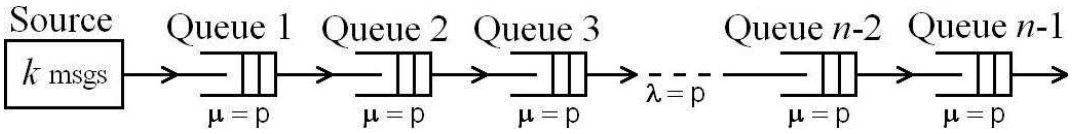


Figure 3-2: A Series of $Geo/Geo/1$ queues

is the case, the probability, p, that node i sends a message that is helpful to the next node $i+1$ (thereby increasing node $i+1$'s rank) is at least

$$p = \frac{1}{2}\left(1 - \frac{1}{q}\right) \tag{3.2}$$

where q is the size of the finite field we are operating over [5].

Our description thus far is reminiscent of the discrete time version of a first-come first-serve $M/M/1$ queue with service rate of p clients per round. We shift focus to a series of $n-1$ such queues as illustrated above in Figure 3-2. Each node in our network corresponds to the waiting line in front of a queue and each link in the network corresponds to the queue. Thus client service corresponds to transmission of helpful data in the direction of propagation. The 3rd client is merely the 3rd independent (helpful) message and thus does not refer to any particular message.

k clients begin in the wait line of queue 1. The average rate of service at the first queue is p clients per round. Also, the output of this first queue is a Bernoulli process with parameter p, so the remaining $n-2$ queues are $Geo/Geo/1$ queues [9]. Thus the average rate of client propagation through the series of queues is p queues per round.

In order to determine the dissemination time we measure the number of discrete-time steps (which we continue to call rounds) it takes for the last client to clear the last queue. The last client has to wait for the $k-1$ clients ahead of it to be served by queue 1 (at rate p clients/round) before it may enter into service at queue 1. Then the last client must to travel through $n-1$ queues (at rate p queues/round), resulting

in a mean dissemination time of

$$\overline{T}_d \leq \frac{k-1}{p} + \frac{n-1}{p} \tag{3.3}$$

Also, in the best case,

$$T_d \geq k - 1 + n - 1 \tag{3.4}$$

so

$$\overline{T}_d = \Theta(n+k) \tag{3.5}$$

3.2.3 The Ring Network with $k = n$

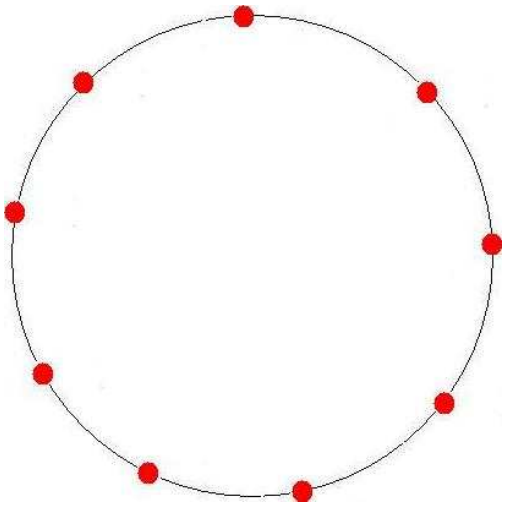


Figure 3-3: The Ring Network: Every member starts with one message.

We consider the ring network of n members (nodes), each with one message before round 1, so $k = n$. Message m_i and its code vector β_i originate at node i for each i. Let A_{ij} denote the number of rounds it takes for node i to acquire a message whose code vector contains a nonzero component in the direction of the original code vector β_j[1]. Clearly $A_{ij} = 0$ if $i = j$. Otherwise, we model each A_{ij} as a negative binomial

[1]This is equivalent to the dot product between one of the node's code vectors, α_{il}, and the original code vector, β_j, being non-zero, $\alpha_{il} \cdot \beta_j \neq 0$ for some l.

random variable with parameters $r = \frac{n-1}{2}$ and $p = \frac{1}{2}(1 - \frac{1}{q})$.

Let A_i denote the number of rounds it takes for node i to acquire enough vectors such that for each original code vector β_j, there is some message at node i whose code vector has a nonzero component in the direction of β_j. Then, $A_i = \max_j A_{ij}$. Let A be the number of rounds for all nodes to satisfy this condition, so that $A = \max_i A_i = \max_{i,j} A_{ij}$. This is a necessary but insufficient condition for complete dissemination, so A is statistically less than T_d and $\overline{A} \leq \overline{T}_d$.

Now we use results about the line network to find an upper bound for T_d. Consider a line network of length n in which the member at one end initially has n packets and all the other members have none. Let B denote the dissemination time for this network. We posit that $\overline{A} + \overline{B}$ is an upper bound for $\overline{T}_d$. We know that $\overline{A}$ is $\Omega(n)$ and we know from the analysis in Section 3.2.2 that $\overline{B}$ is $\Theta(n)$, so we expect $\overline{T}_d$ to be $\Theta(n)$.

This is far from rigorous but the result is verified by simulation (see Figure 3-4). Simulations also suggest that dissemination time in the ring network under the RMS protocol is $\Theta(n^2)$ or $\Theta(kn)$ in general.

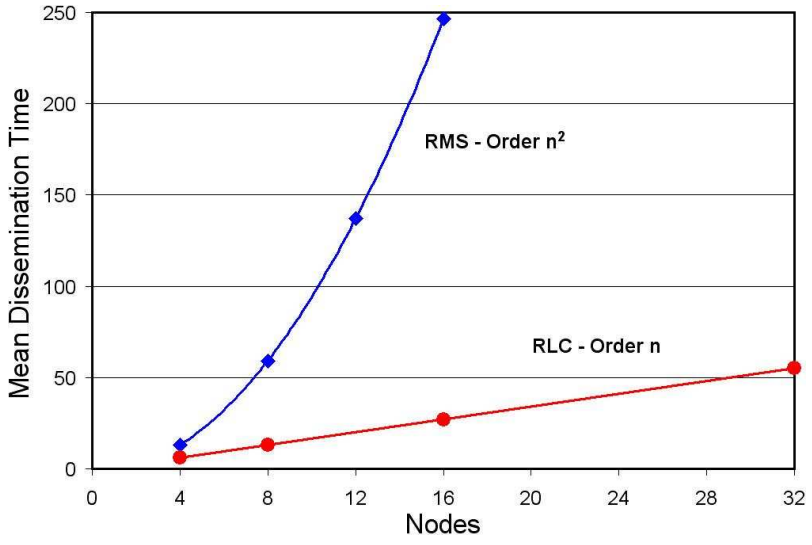


Figure 3-4: The Ring Network: Simulation

3.3 Extension

Part of our motivation for analyzing the line network was to gain insight into how coding solves the problem of efficient information dissemination in general networks. As we will see in the next chapter, general networks are difficult to analyze. We seek a way to provide an estimate of expected performance time based on certain network parameters.

Though in general information spreads in all directions under the RLC protocol, we choose to look at this problem as one of information *propagation*. In Section 3.3.1 we liken certain general networks to the line networks using the longest path.

We recognize that as information propagates in a general network, it can spread to multiple paths. Consequently, we must take other network parameters into account when estimating dissemination time. This is the subject of Section 3.3.2.

3.3.1 Diameter

We first note that, if the degree of our network is relatively uniform and we have have a realistic, low-density two-dimensional network, then we can simplify the network to a line. The idea is to take the diameter of the network graph and analyze a line graph of that length (and hence diameter). In this way we account for the longest path that information must travel in the network. Due to the nature of network coding, multiple flows of independent information do not hinder each other's progress, so the longest path will dominate the dissemination time.

3.3.2 Other Parameters

The model described in the previous section is of course an imperfect one. This is in part due to the effects of spreading, multiple flows over longest paths, and edge effects. We'd also like to develop a better idea of which networks are well captured

by this model. We address these issues in Chapter 4 where we further develop our model.

Chapter 4

Grid Networks

4.1 The Problem

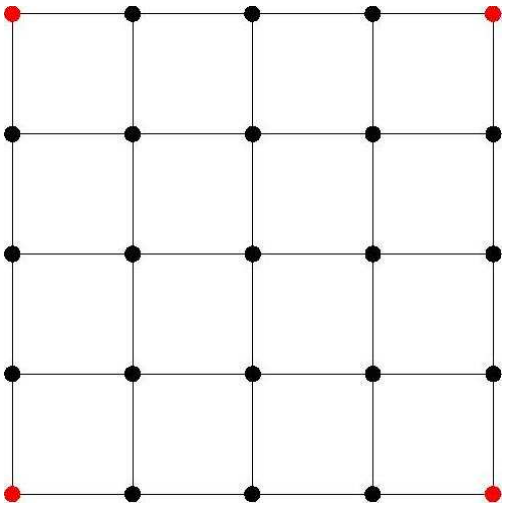


Figure 4-1: The Manhattan Grid Network with Corner Sources

The Manhattan grid, shown in Figure 4-1, is a perfect example of a realistic two-dimensional, network topology with low, and relatively uniform, degree nodes. We study this network graph as we develop a method of predicting the approximate performance of the RLC protocol in general graphs. We consider a scenario where only the corner members are sources.

We would like to use our analysis of this case to form a method of approximating T_d for networks. This method would entail the consideration of relevant parameters such as k, n, q, D and the longest path. Choosing how to measure the longest path requires some thought, as well. We choose the diameter, d, of the graph because this is the worst case. We also seek to define clearly under what conditions this approximation is reasonable. For example, we have developed an approximation for the grid case that indicates that the dissemination time is sensitive to d when $k = \mathrm{O}(d)$. We discuss both the performance of the RLC protocol and the accuracy of our model in Section 4.3.

4.2 Analysis: Lessons from the Line Network

As noted earlier, for networks with certain graph properties, we we can simplify our analysis without losing much accuracy. We do this by analyzing one or more line graphs with the same diameter of the graph of the network in question. For the $w \times w$ Manhattan grid, for example, we analyze a line of length $2w - 1$.

The basic idea is that, each round, any given member will choose a neighbor in the direction of desired information propagation about half the time. For the grid, we see that the degree, D, of each internal node is 4 and that for any internal node and any direction of propagation, *there is one neighbor in the direction of propagation for every neighbor in the opposite direction.* This is also the case with the line network, and, as with the line, each transmission in the direction of propagation from a helpful node successfully propagates information with high probability, at least $(1 - \frac{1}{q})$ [5].

For networks in which this simplification is valid, we can use a network coding protocol to achieve optimal dissemination time in an order sense. This is due to the fact that the diameter of a network will always be a bound on dissemination time and, in these cases, coding leads to a dissemination time that is a multiple of the dissemination time. This supports the theory of equivalence that we develop in Section 5.2.

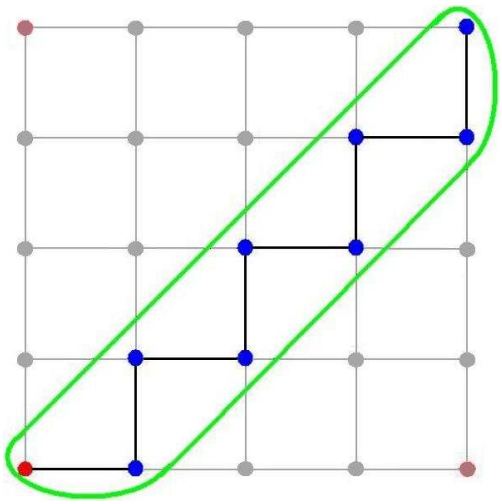


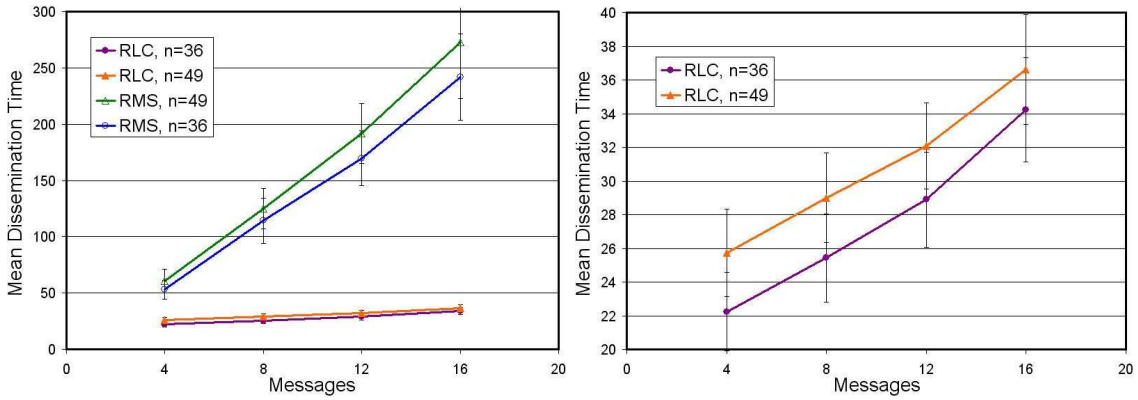
Figure 4-2: The Diameter of the Manhattan Grid: One of the longest paths is circled with members on the path (other than the red source) represented by blue nodes.

Next, we approximate the effect of multiple information flows over longest paths: the fact that multiple messages may have to travel along the longest paths. One way to do this is to model the network as multiple line networks and take the maximum dissemination time as our prediction. Another way is to divide the number of messages by the number of longest paths. This number may be hard to quantify and further research would be required to compare these two approaches.

Finally, we consider the effects of *spreading*: the fact information may travel several paths, speeding up the propagation. We know that degree, min-cut (or other notion of "thickness"), and other graph parameters affect the ability of coding to take advantage of spreading. However, even without spreading the model achieves the order bound for the networks we have considered. We leave the inclusion of this effect in the model for future consideration.

4.3 Simulation

4.3.1 Performance vs. RMS



(a) (b)

Figure 4-3: The Manhattan Grid Network: (a) RLC vs. RMS, (b) Blowup of RLC

Simulations indicate that for a $w \times w$ grid with constant k, T_d is $\Theta(w)$ under the RLC protocol. Because k is constant, T_d is also $\Theta(w)$ under the RMS protocol. This is because the message we want to propagate is chosen to be transmitted a constant fraction of the time. Even so, RLC clearly outperforms RMS (see Figure 4-3). When k is larger, RMS performance erodes, indicating that random store-and-forward routing is not optimal, even in an order sense.

4.3.2 Performance vs. Model

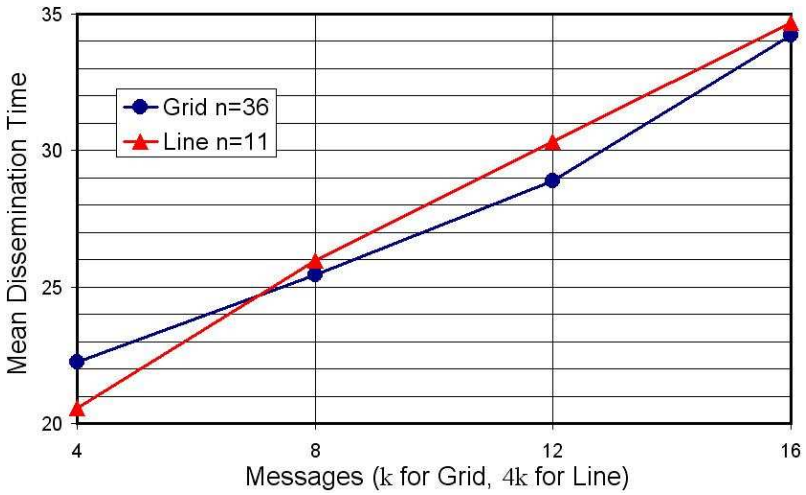


Figure 4-4: The Grid Network: Simulation vs. Model

Figure 4-4 is a comparison of the grid network and the corresponding line. Because there are clearly 4 longest paths in this example we take the number of messages in the line to be $\frac{k}{4}$. This gives a good estimate of the performance of the RLC protocol in the grid network with significantly less computation. Furthermore, if we analyze the line with queuing theory, as in Section 3.2.2, we can estimate performance analytically.

Chapter 5

Equivalence

5.1 A Notion of Equivalence

One should note from our discussion in Chapter 2 that, if the members of the fully connected network were aware of the topology of the network, they could trivially achieve optimal dissemination time. The fully connected case shows that, thanks to random coding, this knowledge is unnecessary. All a network member needs to know is who its neighbors are and the network can operate in an efficient and distributed fashion. In [3], Deb and Médard prove that dissemination time for the fully connected case is also optimal in an order sense.

One of the key observations of this (and previous) work is that the use coding is often equivalent to knowledge of network topology and control over the network, given a reasonable transmission protocol. Clearly this is the case for the fully connected network. In the following sections we develop this idea of equivalence and extend it to other networks of interest, and to general networks.

5.2 A Theory of Equivalence

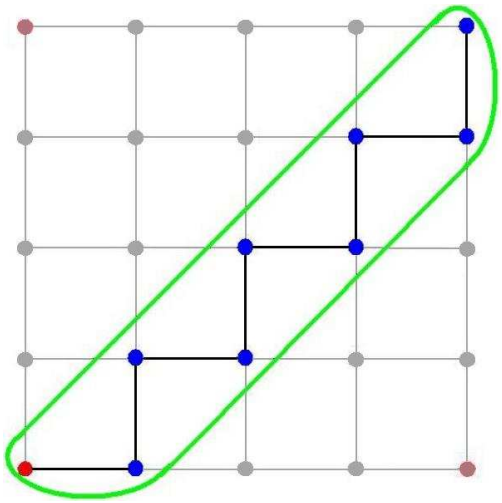


Figure 5-1: The Diameter of the Manhattan Grid: One of the longest paths is circled with members on the path (other than the red source) represented by blue nodes.

We propose a theory of equivalence for networks with graphs where the degree of each node is much lower than the diameter of the graph. These networks lend themselves to propagation-based models. The Manhattan grid network is a great example.

If we have a Manhattan grid network where each corner member starts with one message, as in Chapter 4, then the optimal dissemination time is equal to the diameter, $d = 2w - 1$ (see Figure 5-1). We can do no better than the diameter, in general as well as in an order sense. Again, for certain realistic networks we can achieve this order bound. By our argument in Section 4.2, information propagating in a specific direction is successfully transmitted in that direction almost 50% of the time, suggesting a dissemination time in the vicinity of $2d$. Thus, the use of RLC makes up for the fact that any given member is unaware of the network topology and direction of intended information propagation.

5.3 Extension

We have also found that our theory may hold for networks where there is no concept of an overall direction of propagation. Since the fully connected case represents maximum density (maximum degree and minimum diameter), it is the other end of the spectrum, and our propagation model does not apply. In this case, coding takes advantage of the maximum "thickness" of the network and the ability to spread information efficiently helps coding achieve the known order bound.

Thus we infer that, for networks in which it is possible to define a deterministically achievable bound on dissemination time, random linear network coding is sufficient to achieve this bound in an order sense. Furthermore, we know by counterexample that this does not hold for store-and-forward routing under a randomized transmission protocol. This has been shown with the RMS protocol in Chapters 3 and 4 as well as in [3] by Deb and Médard.

We envision that coding may be used to find a metric for information flow and network capacity for networks for which there is not a clear method of measurement.

Conclusion

From our analysis of the line, ring, grid, and fully connected networks, we have found that the random linear network coding protocol significantly outperforms the random message selection protocol. Furthermore, for certain networks of interest, network coding is equivalent to knowledge of the network topology, so this information need not be stored and maintained, allowing for efficient information dissemination in decentralized networks.

Relevant extensions of this work include analysis of networks with fewer restrictions on topology, networks with different transmission protocols (such as point-to-multipoint transmission), and networks where the probability of error in each transmission is nonzero. Particular cases of immediate interest are networks with mobile members and erasure channels.

Bibliography

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, vol. 46(no. 4):1204–1216, July 2000.

[2] D. P. Bertsekas and R. G. Gallager. *Data Networks*. Prentice Hall, Saddle River, New Jersey, 2nd edition, 1992.

[3] S. Deb and M. Médard. Algebraic gossip: A network coding approach to optimal multiple rumor mongering. *Allerton Conference on Communication, Control, and Computing*, September 2004.

[4] S. Deb, M. Médard, and C. Choute. On random network coding based information dissemination. *IEEE International Symposium on Information Theory*, 2005.

[5] T. Ho, R. Koetter, M. Médard, D. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. *IEEE Symposium on Information Theory*, 2003.

[6] T. Ho, M. Médard, M. Effros, and D. Karger. On randomized network coding. *Allerton Conference on Communication, Control, and Computing*, October 2003.

[7] R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, October 2003.

[8] S. Pereira. Random linear coding in ad-hoc networks. Stanford University Course Project, June 2004.

[9] M. E. Woodward. *Communication and Computer Networks: Modeling with Discrete-time Queues*. IEEE Computer Society Press, Los Alamitos, California, 1994.