Gossip and Random Walk Techniques for Network Coding

Ph.D. Thesis Proposal

Submitted by: **Mr. Michael Borokhovich**

Advisors: **Dr. Chen Avin, Dr. Zvi Lotker**

Department of Communication Systems Engineering

Faculty of Engineering

Ben-Gurion University of the Negev

April 9, 2010

Abstract

Ad hoc and sensor networks usually do not have a central entity for managing information spreading. Moreover, such wireless stations have limited energy and computational power. All this leads to a need for distributed and efficient algorithms for disseminating information across the network. *Network coding* in conjunction with *random walks* and *gossip* techniques proposes a local and distributed approach for compressing information and disseminating it across the network. In this work we study the stopping times of gossip algorithms for network coding. We analyze *algebraic gossip* (i.e., random linear coding) and consider three gossip algorithms for information spreading `PUSH`, `PULL`, and `EXCHANGE`. While traditionally algebraic gossip protocols used `PUSH` or `PULL` as their gossip algorithms, we prove that using the `EXCHANGE` algorithm can be unboundedly better. The stopping time of algebraic gossip is known to be linear for the complete graph; but the question of determining a tight upper bound for general graphs is still open. We take a major step in solving this question, and prove a linear tight upper bound for bounded degree graphs. Our proofs use a new method that relies on Jackson's queuing theorem to analyze the stopping time of network coding, and this technique is likely become useful for future research. In addition, we compare algebraic gossip to an average computation task via `EXCHANGE` and show that there are graphs for which algebraic gossip can be unboundedly faster.

In future work (Chapter 6) we will address many open questions that still remain. First, we are going to find tight upper and lower bounds for general graphs and find deterministic algorithm to compute these bounds as a function of a given graph. We would like to prove all bounds for the asynchronous time model and to give stopping time bounds with high probability along with the expected value results. We are also aiming to study gossip algorithms in general, and algebraic gossip (i.e., linear network coding over gossip algorithm) in particular, on *dynamic graphs* and *SINR* models ([1] and [2]). To the best of our knowledge, this, has not been addressed at all previously.

Contents

List of Figures

Chapter 1

Introduction

Consider the case of a connected network with n nodes, each holding a value it would like to share with the rest of the network. Motivated by wireless networks and limited resource sensor motes, researchers have studied in recent years the use of randomized gossip algorithms together with network coding for this task [18, 16]. Randomized gossip-based protocols are attractive due to their locality, simplicity, and structure-free nature, and been offered in the literature for various tasks [15, 5]. Network coding approaches for multicast (and other networking applications) have received growing attention in the networking community due to their ability to significantly increase the network capacity.

Algebraic gossip is a type of network coding known as random linear coding [11] that uses gossip algorithms. A gossip algorithm is a communication scheme in which at every timeslot a random pair of nodes communicates. It means that at every timeslot, a random node chooses a random neighbor and sends it a message (`PUSH`), or the chosen neighbor sends a message (`PULL`), or the nodes exchange messages (`EXCHANGE`). The gossip protocol (in our case – algebraic gossip) determines the content of messages sent according to the gossip algorithm. In random linear network coding the content of the outgoing message is the random linear combination of all messages stored in the database of the sending node.

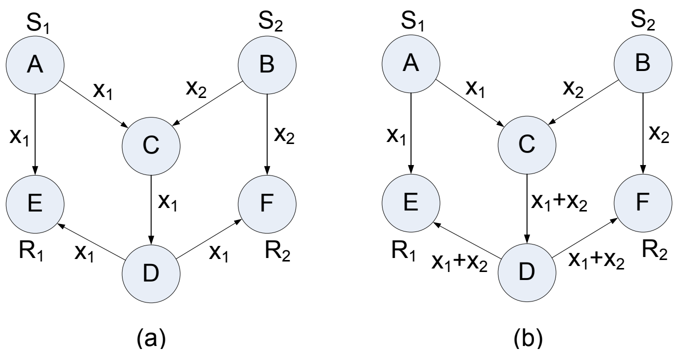

Figure 1.1: (a) – Without network coding, bottleneck at C brings the capacity to 1.5. (b) – With network coding, C transmits at each time unit information that is *helpful* to both receivers, thus the capacity is 2.

Once a node has received enough independent messages (independent linear equations) it can solve the system of linear equations and discover all the initial values of all other nodes. It has been proved [16] that network coding can improve throughput of the network by better sharing of the network resources. Let us look at the following example [10]. Consider a *Butterfly Network* (see Figure 1.1) with sources S_1 and S_2, each is wishing to multicast to both R_1 and R_2. All links have capacity 1. Without network coding, the maximum achievable source-destination rate (assuming both rates are equal) is 1.5, due to a bottleneck at node C. Using a simple linear network coding, we can "xor" the information coming from S_1 and S_2 at the node C. By doing so, each receiver will obtain two linear equations: R_1 will get: x_1 and $x_1 + x_2$, and R_2 will get: x_2 and $x_1 + x_2$. Now, each receiver is able to solve this simple linear system and discover x_1 and x_2. It is clear that the resulting throughput is now 2.

In this work we set to study the performance of algebraic gossip on arbitrary networks, where information is disseminated from all nodes in the network to all nodes, i.e., all-to-all dissemination. Gossip algorithms determine the way in which a protocol disseminates its information to the network. To study algebraic gossip we make use of three gossip algorithms:

PULL, PUSH, and EXCHANGE. In particular, we offer the use of algebraic gossip with EXCHANGE, and show that this can lead to significant improvements. Furthermore, beside network coding we consider two other gossip protocols (i.e., tasks that use gossip algorithms): i) *collect all information*, which is identical to the initial task but there are no restrictions on the message size and no obligation to use network coding, and ii) *average computation*, where all nodes need to learn the average value of all nodes initial value. We utilize these protocols to gain a better understating of algebraic gossip. Our main goal is to find the tight bounds of the stopping time of the algebraic gossip protocol (i.e., random linear coding over the gossip algorithm) in expectation and with high probability.

These and similar questions have been addressed in the past. Deb *et al.* [8] studied algebraic gossip using PULL and PUSH on the complete graph and showed a linear stopping time in expectation and with high probability.[1] Boyd *et al.* [5, 6] studied the average problem using the EXCHANGE algorithms and gave bound on symmetric networks that is based on the second largest eigenvalue of the transition matrix, or the mixing time of a random walk on the network. They actually showed that this measure captures the behavior of the protocol. Mosk-Aoyama and Shah [20] used a similar approach to [5, 6] to analyze network coding on symmetric networks and gave an upper bound for the PULL algorithm that is based on a measure of conductance of the network.

A recent, yet unpublished,[2] independent work by Vasudevan and Kudekar [25] also offered the use of EXCHANGE together with algebraic gossip. Moreover, they give a uniform strong bound on algebraic gossip for arbitrary networks: $O(n \log n)$ in expectation and $O(n \log^2 n)$ with high probability. It is not known if these bounds are tight but they significantly improve upon Mosk-Aoyama and Shahs bounds. Very recently we have found a graph that contradicts to the result of [25]. For further discussion see Chapter 6.

[1]In this work, we define *high probability* as a probability of at least $1 - O(\frac{1}{n})$.

[2]To the best of our knowledge at time of submission.

Graph	$\overline{R}_v$	$\overline{R}$	$\hat{R}$
Complete K_n	$\Theta(n)$[8]	$O(n \log n)$[20],[25]	$\Theta(n)$[8]
Ring R_n	$\Theta(n)$ [here]	$\Theta(n)$ [here]	$\Theta(n)$ [here]
Star S_n	$\Theta(n)$ [here]	$O(n \log n)$[25]	$\Theta(n)$ [here]
Bounded-Degree B_n	$\Theta(n)$ [here]	$\Theta(n)$ [here]	$\Theta(n)$ [here]
Any Graph G_n	$O(n \log n)$[25]	$O(n \log n)$[25]	$O(n \log^2 n)$[25]

Table 1.1: Summary of our and previous results. $\overline{R}_v$ - expected stopping time (in rounds) of a node v, $\overline{R}$ - expected stopping time (in rounds), $\hat{R}$ - stopping time (in rounds) with high probability. [8] - Deb et al., [20] - Mosk-Aoyama et al., [25] - Vasudevan et al.

Overview of our results. First, independently of [25], we promote the use of `EXCHANGE` with algebraic gossip. We show that there are graphs for which `EXCHANGE` can perform unboundedly better than `PULL` or `PUSH`. We provide new analytical bounds for the Star (Lemma 4.5) and the Ring (Theorem 3.1) graphs, where both bounds are better than any previous general bounds (i.e., [20, 25]). The technique used in the proof for the Ring graph involved a Jackson network of queues and is interesting in its own right. Then, we extended the result of the Ring graph to any bounded-degree graph (Theorem 3.3) thus providing our main result that the stopping time of the algebraic gossip on any bounded-degree graph is linear in expectation and with high probability. Our and previous theoretical results are summarized in Table 1.1.

In addition, we present a simulation study that supports our results for many networks such as the grid, hypercube, and the lollipop. Using our bounds we show that the spectral gap method that is used to analyze the average problem over `EXCHANGE` is not suitable for bounding network coding, since there are graphs for which network coding is faster and graphs for which average computation is faster.

The rest of the document is organized as follows. In Chapter 2 we present the communication and time models, define gossip algorithms, gossip protocols, and formally state the *gossip stopping problem*. In Chapter 3 we show our main theoretical result – linear bounds on any bounded-degree graph. In Chapter 4 we prove the unbounded difference between the `EXCHANGE` and the `PUSH/PULL` algorithms. Chapter 5 describes the simulation tool that we developed and the simulation results. Finally, we summarize all the results and present the future work towards my Ph.D. in Chapter 6.

Chapter 2

Preliminaries and Models

2.1 Network and Time Models

We model the communication network by a connected undirected graph $G(V,E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices and $E = \{e_{v,u} \mid e_{v,u} \in V \times V\}$ is the set of edges. Let $N(v) = \{u \mid e_{v,u} \in E\}$ be a set of neighbors of node v and $d_v = |N(v)|$ its degree. Any two nodes are able to communicate iff there is an edge between them. In the current work, we will mostly consider the following topologies:

- Complete graph – K_n: clique of size n.
- Star graph – S_n: where v_1 is the center of the star and other nodes are connected only to v_1.
- Ring graph – R_n: a connected cycle where each node has left and right neighbors.
- Bounded-degree graph – B_n: a connected graph in which the maximal degree d_m is constant (independent of number of vertices n).
- Symmetric graph – Sym_n: is a connected graph whose stochastic adjacency matrix is symmetric. All regular graphs are symmetric graphs.

In the simulation section we also consider the 2-D Grid graph, Lollipop graph, and a Hypercube graph, all of size n.

We consider the two following time models:

2.1.1 Synchronous time model

The time is assumed to be slotted and divided into *rounds*. Each round contains n timeslots. At every round, a random permutation of the nodes is selected uniformly and time slots in the round are taken by nodes according to their order in the permutation. At each timeslot, a single pair of nodes communicates. The model guarantees that during n consecutive timeslots each node will be selected exactly once.

2.1.2 Asynchronous time model

The time is assumed to be slotted. At every timeslot, a node selected independently and uniformly at random takes an action. As before, n consecutive timeslots are considered as one round. At each timeslot, a single pair of nodes communicates. In this model there is no guarantee that a node will be selected exactly once in a round; nodes can be selected several times or not at all.

2.2 Gossip Algorithms

Gossip algorithms define the way information is exchanged or spread in the network. At each time step (timeslot) a single node takes an information spreading action that is divided into two phases: (i) choosing a communication partner and (ii) spreading the information. First, a *communication partner* $u \in N(v)$ is chosen by node $v \in V$ with probability p_{vu}. Here, we will assume *uniform gossip algorithms*, i.e., $p_{vu} = \frac{1}{d(v)}$.

A *communication pair* at timeslot t is an ordered pair $\langle v, u \rangle_t$, where $v \in V$ is the caller node and $u \in N(v)$ is the called node.

Given a *communication pair* $\langle v, u \rangle_t$, we distinguish three **gossip algorithms** for information spreading between v and u, PUSH, PULL, and EXCHANGE:

- PUSH– one message is transmitted from the caller node v to the called node u.
- PULL– one message is transmitted from the called node u to the caller node v.
- EXCHANGE (Pull and Push) – one message is transmitted from the caller node v to the called node u and the second message is transmitted from the called node u to the caller node v.

For a graph G, a gossip algorithm run (or execution) σ_G (we usually omit G when its clear from the context) is a sequence of communication pairs that communicated during the algorithm. We define the set of all possible algorithm runs given a specific graph G, as $\Pi(G)$.

During a gossip algorithm run, information is passed in messages $m_{s,d}(t)$, where each message is indexed by its source node $s \in V$, the destination node $d \in N(s)$, and the timeslot t at which it was sent. Messages that are sent in timeslot t are received in timeslot t. Note that for a communication pair $\langle v, u \rangle_t$, the messages that will be sent in timeslot t are a function of the gossip algorithm that is being executed.

Given a gossip algorithm $\mathcal{A}$ and a run σ, an *information path* from node u to node v by the timeslot t is a sequence of messages that provides an information spreading path from node u to node v:

$$\begin{aligned} I^t_{u,v}(\sigma, \mathcal{A}) &= \{m_{s_1,d_1}(t_1), m_{s_2,d_2}(t_2), ..., m_{s_k,d_k}(t_k)\} \; s.t. \\ &\qquad s_1 = u, d_k = v, \forall_{0<i<k} : t_i < t_{i+1} \le t, d_i = s_{i+1} \end{aligned}$$

We say that if there exists at least one information path $I^t_{u,v}(\sigma, \mathcal{A})$ then nodes u and v are *connected* by timeslot t in the specific run σ and the gossip algorithm $\mathcal{A}$.

2.3 Gossip Protocols

A *gossip protocol* is a task that is being executed using gossip algorithms. Usually, a gossip protocol can be executed by any of the three gossip algorithms, `PULL`, `PUSH`, or `EXCHANGE` (but this is not always true). Initially, every node in the network has an initial value and in this work we study and compare three different gossip protocols (i.e., tasks) that compute a function of these values at every node:

CI Collect information from all – disseminating the n initial values to all n nodes. For this protocol, message size transmitted in the network is unlimited.

AVG Finding Average [6] – all n nodes have to evaluate the average of the initial values up to a predefined ϵ. Message size in this protocol is limited.

NC Network Coding – disseminating the n initial values to all n nodes using random linear coding (a.k.a. algebraic gossip)[8]. Message size in this protocol is limited.

For all three protocols we will use the following definitions:

- **Initial values vector** $\overline{x} = (x_1, x_2, ..., x_n)$ is a vector that consists of values that every node has initially (before starting the protocol). We assume that in all protocols $x_i \in \mathbb{Z}^+, \forall i \in [1..n]$.

- **Nodes database (DB).** Each node maintains a database D. For a node v, let $D_v(t)$ be the database of v at timeslot t. For different protocols the database can hold different information. If a node needs to send a message at time t, it constructs the message using a protocol-specific function $F^{\mathcal{P}}_{out}$, i.e., if node s needs to send a message to node d at time t, then $m_{s,d}(t) = F^{\mathcal{P}}_{out}(D_s(t))$.
  If a node has received a message during timeslot t, it will need to update its database at the end of this timeslot. Database update will be done using a protocol-specific

function $F_{in}^{\mathcal{P}}$ and this will result in the database of time $t+1$, i.e., if a message $m_{s,d}(t)$ received by node d from the node s at timeslot t, then $D_d(t+1) = F_{in}^{\mathcal{P}}(m_{s,d}(t), D_d(t))$.

- **Stopping condition.** A task is completed when every node has completed the task; therefore we have *local* and *global* stopping conditions, for a protocol $\mathcal{P}$:

  - *Local stopping condition*: $\mathbb{S}_v^{\mathcal{P}}(t)$
    is true iff node $v \in V$ has completed the protocol (task) by timeslot t.
  - *Global stopping condition*: $\mathbb{S}_V^{\mathcal{P}}(t)$
    is true iff all nodes have completed the protocol (task), i.e., whether the local stopping condition is true for all nodes.

We now describe the tasks presented above in more detail, i.e., messages, databases, and stopping conditions.

2.3.1 CI (Collect Information from All)

Initially, the DB of node v_i, $D_{v_i}(0)$, will contain only the self-initial value, i.e., $D_{v_i}(0) = \{x_i\}$. In CI, messages that nodes send ($m_{s,d}(t)$) can be of an unlimited size. Thus, a node will send all the data it has in its DB in every outgoing message, i.e., $F_{out}^{CI}(D_s(t)) := D_s(t)$. Database update is just extracting the initial values from the received message and appending the new values to the DB. $F_{in}^{CI}(m_{s,d}(t), D_d(t)) := \{D_d(t) \cup x_j \mid x_j \notin D_d(t), x_j \in m_{s,d}(t)\}$.
A node completes the CI task once it knows all the initial messages $\overline{x}$ in the graph, i.e., the *local stopping condition* is:

$$\mathbb{S}_v^{CI}(t) = \begin{cases} TRUE & \text{if } D_v(t) = \{\overline{x}\}, \\ FALSE & \text{otherwise} \end{cases}$$

2.3.2 AVG (Finding Average)

Initially, the DB of node v_i, $D_{v_i}(0)$, will contain only the self-initial value, i.e., $D_{v_i}(0) = x_i$. In AVG, messages that nodes send ($m_{s,d}(t)$) are a single value stored in the node's database, i.e., $F_{out}^{AVG}(D_s(t)) := D_s(t)$. Database update is just computing the average between the value stored in the node's DB and the value received in the message $m_{s,d}(t)$, i.e., $F_{in}^{AVG}(m_{s,d}(t), D_d(t)) := \frac{m_{s,d}(t)+D_d(t)}{2}$.
A node completes the AVG task once it has a value that is $\pm\epsilon$ ($\epsilon \in \mathbb{R}^+$) from the true average value of all nodes, i.e., the *local stopping condition* is:

$$\mathbb{S}_v^{AVG}(t) = \begin{cases} TRUE & \text{if } \left|D_v(t) - \frac{1}{n}\sum_{j=1}^{n} x_j\right| < \epsilon, \\ FALSE & \text{otherwise} \end{cases}$$

where we will usually consider $\epsilon = \frac{1}{n}$.

2.3.3 NC (Network Coding)

In this task we will use a random linear coding technique as described in [8]. As in the CI protocol, each node $v_i \in V$ has some initial value x_i that can be represented using $len(x_i) = \log_2(\max_{v_j \in V} x_j) = l$ bits, $\forall v_i \in V$.

But, unlike CI, where messages transmitted by nodes could be of an unlimited length, in the NC protocol all transmitted messages have a fixed length. Each transmitted message is comprised of a linear combination of all messages stored in a node's DB and all the random coefficients that built this linear combination. So, $len(m_{s,d}(t)) = l + len(n \times \log_2(q))$ bits, where q is the size of the field $\mathbb{F}_q$ from which coefficients are drawn. Every initial value x_i can be viewed as a number from a finite field $\mathbb{F}_q$, if $\forall v_i \in V, x_i < q$. If $\exists v_i \in V, x_i \geqq$, then the initial values x_i should be viewed as vectors over field $\mathbb{F}_q$, i.e., $x_i \in \mathbb{F}_q^r$ and $r = \left\lceil \log_q(x_i) \right\rceil$.

In the NC task, the node's DB should be able to hold n different messages $m_{s,d}(t)$. Each message represents a linear equation over $\mathbb{F}_q$. Variables of these equations are the initial values $x_i \in \mathbb{F}_q^r$. Once a node has n independent equations (messages) it is able to decode all the initial values $x_i \in \mathbb{F}_q^r$, and thus completes the protocol.

Initially, the DB of node v_i, $D_{v_i}(0)$, will contain only one linear equation that consists of only one variable corresponding to x_i multiplied by a coefficient 1 and equal to the value of x_i, i.e., the node knows only the self initial value.

The message that a node will send is: $F_{out}^{NC}(D_s(t)) := RLC(D_s(t)) \cup \{a_{x_i}\}_{i=1}^n$, where $RLC()$ is a random linear combination, in which random coefficients were chosen from $\mathbb{F}_q$, and $a_{x_i} \in \mathbb{F}_q$ is a coefficient of the x_i variable in the resulting random linear combination.

Database update in the NC protocol appends a message (which is a linear equation) to the database only if it is independent of all messages (linear equations) that are already stored in the node's DB.

$$F_{in}^{NC}(m_{s,d}(t), D_d(t)) := \begin{cases} D_d(t) \cup m_{s,d}(t) & \text{if } m_{s,d}(t) \text{ is lin. indep. with } D_d(t) \\ D_d(t) & \text{otherwise} \end{cases}$$

For a node v at timeslot t, let $S_v(t)$ be the subspace spanned by the linear equations (or vectors) in its DB (i.e., coordinates of each vector are coefficients that constitute the corresponding linear equation) at the beginning of timeslot t. Dimension (or rank) of a node is a dimension of its subspace, i.e., $dim(S_v(t))$ and it is equal to the number of independent linear equations stored in the node's DB.

A node completes the NC task once it is able to decode (by solving linear equations) the initial values of all other nodes. So, the *local stopping condition* is:

$$\mathbb{S}_v^{NC}(t) = \begin{cases} TRUE & \text{if } dim(S_v(t)) = n, \\ FALSE & \text{otherwise} \end{cases}$$

2.4 The Gossip Stopping Problem

Our goal is to compute bounds on time and number of messages needed to be sent in the network to complete various gossip protocols over various gossip algorithms. For this purpose we define the following:

Definition 2.1 (local stopping time) *Given a graph G (which we omit from notation when it is clear from the context), gossip algorithms $\mathcal{A}$, and a gossip protocol $\mathcal{P}$, the stopping time of a node v, $T_v^{\mathcal{A},\mathcal{P}}$ is defined as minimal number of timeslots by which node v completes the task:*

$$T_v^{\mathcal{A},\mathcal{P}} = \min_{t \in \mathbb{Z}^+} \Big[t \mid \mathbb{S}_v^{\mathcal{P}}(t) = TRUE \Big]$$

Definition 2.2 (global stopping time) *Given a graph G (which we omit from notation when it is clear from the context), gossip algorithms $\mathcal{A}$, and a gossip protocol $\mathcal{P}$, the stopping time $T^{\mathcal{A},\mathcal{P}}$ is defined as follows (minimal number of timeslots by which all nodes complete the task):*

$$T^{\mathcal{A},\mathcal{P}} = \max_{v} \Big\{ T_v^{\mathcal{A},\mathcal{P}} \Big\}$$

or:

$$T^{\mathcal{A},\mathcal{P}} = \min_{t \in \mathbb{Z}^+} \Big[t \mid \mathbb{S}_V^{\mathcal{P}}(t) = TRUE \Big]$$

Definition 2.3 (expected local stopping time) *The* expected local stopping time *of a node v, $\overline{T}_v^{\mathcal{A},\mathcal{P}}$ is defined as follows:*

$$\overline{T}_v^{\mathcal{A},\mathcal{P}} = \mathbb{E}\Big[T_v^{\mathcal{A},\mathcal{P}} \Big]$$

Definition 2.4 (expected global stopping time) *The* expected global stopping time $\overline{T}^{\mathcal{A},\mathcal{P}}$ *is defined as follows:*

$$\overline{T}^{\mathcal{A},\mathcal{P}} = \mathbb{E}\Big[T^{\mathcal{A},\mathcal{P}} \Big]$$

Definition 2.5 (high probability global stopping time) *The* high probability global stopping time $\hat{T}^{\mathcal{A},\mathcal{P}}$ *is defined as follows:*

$$\hat{T}^{\mathcal{A},\mathcal{P}} = \min_{t \in \mathbb{Z}} \Big[t \mid \Pr\Big(T^{\mathcal{A},\mathcal{P}} \le t \Big) \ge 1 - O\left(\frac{1}{n}\right) \Big]$$

Definition 2.6 (stopping time measured in rounds) *The stopping time can also be measured by* rounds, *where one round equals n consecutive timeslots. In particular, we define:*

$R_v^{\mathcal{A},\mathcal{P}} = T_v^{\mathcal{A},\mathcal{P}}/n$ – *local stopping time measured in rounds*

$R^{\mathcal{A},\mathcal{P}} = T^{\mathcal{A},\mathcal{P}}/n$ – *global stopping time measured in rounds*

$\overline{R}_v^{\mathcal{A},\mathcal{P}} = \overline{T}_v^{\mathcal{A},\mathcal{P}}/n$ – *expected number of rounds by which node v completes the task*

$\overline{R}^{\mathcal{A},\mathcal{P}} = \overline{T}^{\mathcal{A},\mathcal{P}}/n$ – *expected number of rounds by which all nodes complete the task*

$\hat{R}^{\mathcal{A},\mathcal{P}} = \hat{T}^{\mathcal{A},\mathcal{P}}/n$ – *number of rounds by which all nodes complete the task with high probability*

We can now express our research question formally:

Definition 2.7 (gossip stopping problem) *Given a graph G, a gossip algorithm $\mathcal{A}$, and a gossip protocol $\mathcal{P}$, the* gossip stopping problem *is to determine the expected stopping time and the stopping time with high probability.*

Chapter 3

`EXCHANGE` is Linear on Bounded-Degree Graphs

3.1 `EXCHANGE` is Linear on Ring Graph

In order to prove the main result of the chapter (NC task over `EXCHANGE` gossip is linear on bounded-degree graphs) we will first present a theorem that will be the basis for the more general result, i.e., for the bounded-degree graphs. This theorem proves that on a Ring graph R_n and using the `EXCHANGE` algorithm, the stopping time of the NC protocol is $O(n)$ in expectation and with high probability. The proof uses queuing networks ideas, which seem to contribute some insight to the problem.

Theorem 3.1 *For the synchronous time model and the Ring graph R_n, the gossip stopping time of the NC task is linear in expectation and with high probability:*

$$\overline{R}^{EX,NC} = O(n) \tag{3.1}$$

and

$$\hat{R}^{EX,NC} = O(n). \tag{3.2}$$

Proof of Theorem 3.1. The idea of the proof is to reduce the problem of network coding on the Ring graph R_n to a simple system of queues and then use Jackson's theorem for open networks.

To simplify our analysis, we cut the Ring in an arbitrary place and get a Path graph P_n (without loss of generality, we assume that the leftmost node in the Path is v_1 and the rightmost node is v_n). It is clear that the stopping time of the NC protocol will be larger in a Path graph than in a Ring graph. Another simplification that we will do, for the first part of the proof, is to consider only the messages that travel from left to right (i.e., other messages will be ignored).

We define a queuing system by assuming a queue at each node. Customers of our queuing network are the *helpful* messages, i.e., messages that increase the rank of a node they arrive at. This means that every customer arriving at some node increases its rank by 1. The queue length of node v_i at the beginning of timeslot t is defined as $Q_i(t)$. The size of the queue $Q_i(t)$, represents a measure of *helpfulness* of the node v_i to its right-hand neighbor v_{i+1}, $(i < n)$. For $i < n$, let $Q_i(t) = dim(S_{v_i}(t)) - dim(S_{v_{i+1}}(t)) + 1$. Notice that even when the ranks (dimensions) of v_i and v_{i+1} are equal, v_i is still *helpful* to v_{i+1}, since there is one message v_{i+1} knows that is unknown to v_i and thus there should be one message in v_i that is unknown to v_{i+1}. The last explains the $+1$ in the $Q_i(t)$ definition.

It is clear that during a single timeslot, Q_i can be increased by one if the rank of v_i was increased during this timeslot and the rank of v_{i+1} remained unchanged; Q_i can be decreased by one if the rank of v_i remained unchanged during this timeslot and the rank of v_{i+1} was increased, or, Q_i can remain unchanged if either none of the nodes increased their ranks or both of them increased their ranks by 1.

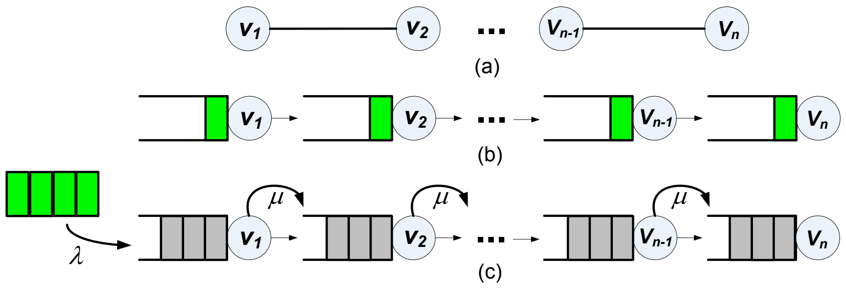

Figure 3.1: Modeling NC in a Path as a queuing network.

The last observation implies that customers at each node represent independent linear equations in the node's DB. Initially, there is a single customer at each node. The initial condition of our queuing network is illustrated in Fig. 3.1 (b). In the first part of the proof we find the average time needed for all customers in the network to arrive at the node v_n thus bringing its rank to n. In the second part of the proof we will find the average time needed for all customers located at the node v_n to pass through all other nodes (queues) towards v_1, thus bringing their ranks to n.

The service procedure at node v_i is a transmission of a *helpful* message (customer) from v_i to v_{i+1}. Clearly, if $Q_{i-1}(t) = 0$ the rank of node v_i cannot be increased during timeslot t, since the node v_{i-1} does not know more information than v_i, because all new customers (equations) are coming to v_i from v_{i-1}. If $Q_{v_{i-1}}(t) > 0$ it means that v_i is *helpful* to v_{i+1} in the timeslot t. So, from Lemma 4.6, the probability that a customers service will be finished at node v_i by the end of this timeslot is: $p \geq \frac{3}{4}\left(1 - \frac{1}{q}\right)$, where $\frac{3}{4}$ is the probability that in the `EXCHANGE` algorithm a message will be sent from v_i to v_{i+1}.

Thus, we can consider a service time in our queuing system that is geometrically distributed with parameter $p = \frac{3}{4}\left(1 - \frac{1}{q}\right)$. Our service time will be distributed over the set $\{0, 1, 2, 3, ...\}$, which means that a customer enters the queue at the moment the transmission from this queue occurs, i.e., with probability p, a customer entering an empty queue, can be serviced immediately. Since a customer cannot pass more than one node (queue) in a single

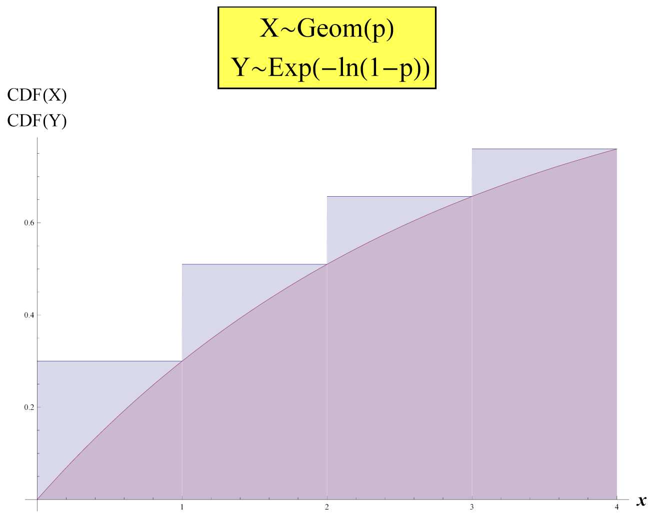

Figure 3.2: CDF of geometric random variable is bounded from below by CDF of the exponential random variable. In this example $p = 0.3$.

timeslot, we will define a transmission time as a one timeslot.

The following lemma shows that we can assume that the service rate can be bounded from below by an exponential random variable (see Figure 3.2).

Lemma 3.1 *Let X be a geometric random variable with a success probability p and supported on the set $\{0, 1, 2, 3, ...\}$, i.e., for $k \in \mathbb{Z}^+$: $\Pr(X = k) = (1-p)^k p$.*

Then, for all $x \in \mathbb{R}^+$

$$\Pr(X \leq x) \geq 1 - e^{\ln(1-p)x}$$

Proof. For a geometric random variable X with a success probability p and supported on the set $\{0, 1, 2, 3, ...\}$:

$\Pr(X > x) = (1-p)^{x+1}$, for $x \in \mathbb{Z}^+$

and

$\Pr(X > x) = (1-p)^{\lfloor x \rfloor + 1}$, for $x \in \mathbb{R}^+$.

Thus, for $x \in \mathbb{R}^+$,

$\Pr(X \leq x) = 1 - (1-p)^{\lfloor x \rfloor + 1} \geq 1 - (1-p)^x = 1 - e^{\ln(1-p)x}$

■

We can now assume that the service time is exponentially distributed with parameter

$$\mu = -\ln(1-p) = -\ln\left(1 - \frac{3}{4}\left(1 - \frac{1}{q}\right)\right) = -\ln\left(\frac{1}{4} + \frac{3}{4q}\right).$$

This assumption decreases the service rate of transmission of *helpful* messages, and therefore will only increase the stopping time.

We present here the Jackson's theorem from queuing theory. A proof of this theorem can be found in [7].

Theorem 3.2 (Jackson's Theorem [7]) *In an open Jackson network of n queues where the utilization ρ_i is less than 1 at every queue, the equilibrium state probability distribution exists, and for state $(k_1, k_2, \ldots, k_n)$ is given by the product of the individual queue equilibrium distributions*

$$\pi(k_1, k_2, ..., k_n) = \Pi_{i=1}^{n} \rho_i^{k_i}(1-\rho_i)$$

where $\rho_i = \frac{\lambda_i}{\mu_i}$.

Jackson's theorem requires a fixed arrival rate and equilibrium lengths of all queues. To achieve this, we will slightly change our queuing network to obtain these requirements.

The initial state of our system is that at every queue we have one customer (Fig. 3.1 (b)). So, first, we move all the n customers to the first queue and force them to cross all the n queues. Clearly, such a modification can only increase the stopping time.

Next, we will take all these customers out from the first queue and let them enter the system (via the first queue) with a predefined arrival rate. This modification, as well, increases the stopping time.

Now, we will define the customers' arrivals as a Poisson process with rate $\lambda = \frac{\mu}{2}$. So, $\rho_i = \frac{\lambda_i}{\mu_i} = \frac{1}{2} < 1$ for all queues ($i \in [1..n]$).

Our last step is to ensure that the lengths of all queues at time $t = 0$ are according to the equilibrium state probability distribution. We now add customers in all the queues according to the stationary distribution which we know exists from Theorem 3.2. By adding additional *dummy* customers (we call them *dummy* since their arrivals are not counted as a rank increment) to the system, we make the *real* customers wait longer in the queues, thus increasing the stopping time. Our queuing network with the above modifications is illustrated in Fig. 3.1 (c), where the *real* customers are green, and the *dummy* customers are grey.

Now, we are interested in finding the expected time it will take the nth *real* customer (nth customer that arrives after the time $t = 0$) to arrive at the rightmost node, i.e., at the node v_n. By that time, the rank of node v_n will become n and it will finish the NC protocol. Let us denote the expected time needed to the rank of node v_n become n as $\overline{T} = \overline{T}_1 + \overline{T}_2$, where $\overline{T}_1$ is the expected time needed for the n customer to arrive at the first queue, and $\overline{T}_2$ is the expected time needed for the nth customer to path through all the n queues in the system.

From Jackson's Theorem, it follows that the expected time of crossing all n queues ($\overline{T}_2$) equals the expected time to cross a single queue, multiplied by the total number of queues.

The next two lemmas help us to bound the time it takes to transfer all the information (all the *real* customers) to the node v_n on the Path.

Lemma 3.2 *The expected time it takes the n customers to arrive at the system is:*

$$\overline{T}_1 = \frac{n}{\lambda}$$

Proof. Since we assumed that the arrival process is a Poisson process with parameter λ, the inter-arrival time is exponentially distributed with parameter λ, and thus, the expected value of the inter-arrival time is $\frac{1}{\lambda}$. Expectation is linear, and thus we obtain the expected time until the n's arrival: $n \times \frac{1}{\lambda}$. ■

Lemma 3.3 *The expected time it takes to cross n $M/M/1$ queues in the equilibrium state (which is achievable for $\rho < 1$) is:*

$$\overline{T}_2 = \frac{n}{\mu - \lambda}$$

Proof. The expected time needed to cross one $M/M/1$ queue in the equilibrium state is $\frac{1}{\mu-\lambda}$ ([19], page 215). Since expectation is linear, the expected time needed to cross n $M/M/1$ queues is: $n \times \frac{1}{\mu-\lambda}$. ■

So,

$$\overline{T} = \overline{T}_1 + \overline{T}_2 = \frac{n}{\lambda} + \frac{n}{\mu - \lambda} = \frac{4n}{\mu}$$

Since,

$$\frac{4n}{\mu} = \frac{4n}{-\ln\left(\frac{1}{4} + \frac{3}{4q}\right)} = \frac{4n}{\ln\left(\frac{4q}{q+3}\right)} \leq \frac{4n}{\ln 2} \text{ (for } q > 3)$$

Hence,

$$\overline{T} = O(n).$$

Thus, after $\overline{T}$ rounds, the rank of node v_n becomes n and it can finish the NC protocol.

Now, let us assume that all nodes except the node v_n forget all the information they have. So, the rank of all nodes except v_n is 0. Let us now analyze the information flow from the rightmost node in the Path (v_n) to the leftmost node (v_1). In the same way, we will represent all the *helpful* messages that the node v_n will send as customers in our queuing system. In order to use Jackson's Theorem, we will again remove all the *real* customers from the system and will inject them to the queue of node v_n with a Poison rate $\lambda = \frac{\mu}{2}$. We also fill all the queues in the system with *dummy* customers in order to achieve queue lengths that correspond to the equilibrium state distribution.

Clearly, arrival of a *real* customer at some node v_i ($i \neq n$) will increase the rank of that node. So, after the last *real* customer arrives at the node v_1, the ranks of all nodes will be n, and the NC task will be finished. Using the same equilibrium state analysis as before, we obtain that the expected time it takes all *real* customers to arrive at the rightmost node

v_n, to cross all the n queues, and arrive at the node v_1 – is the same as $\overline{T}$ which we have already found.

Hence, after $2 \times \overline{T}$ rounds, the NC task will be finished. Since in all our assumptions we increased the stopping time, the first result of Theorem 3.1, i.e., the expectation bound (Equation (3.1)), follows:

$$\overline{R}^{EX,NC} \leq 2 \times \overline{T} = O(n)$$

Now, we will prove the second part of the Theorem 3.1, i.e., the high probability bound (Equation (3.2)).

In order to prove the high probability result we will need the following lemmas:

Lemma 3.4 ([22], section 4.3) *Time needed to cross one $M/M/1$ queue in the equilibrium state has an exponential distribution with parameter $\mu - \lambda$.*

Lemma 3.5 *Let Y be the sum of n independent and identically distributed exponential random variables with parameter $\delta \geq 0.5 \ln 2$, i.e., $Y = \sum_{i=1}^{n} X_i$, $f_X(x) = \delta e^{-\delta x}$, where X_is ($i \in [1..n]$) are distributed as X.*

Then,

$$\Pr(Y \geq 4n) \leq \left(\frac{\sqrt{2}}{1.5}\right)^n$$

Proof.

The generating function of X is given by:

$$G_X(s) = \mathrm{E}\left[e^{sX}\right] = \int_0^\infty e^{sx} f_X(x)dx$$

For any $s < \delta$:

$$G_X(s) = \frac{\delta}{\delta - s}$$

Now, we can obtain the generating function of Y (Y is a sum of n independent and identically distributed random variables):

$$G_Y(s) = \prod_{i=1}^{n} G_{X_i}(s) = (G_X(s))^n = \left(\frac{\delta}{\delta - s}\right)^n$$

Now, we will apply Markov inequality to obtain an upper bound on Y. For $s \geq 0$:

$$\Pr(Y \geq 4n) = \Pr\left(e^{sY} \geq e^{s\times 4n}\right) \leq \frac{\mathrm{E}\left[e^{sY}\right]}{e^{s\times 4n}} = \frac{G_Y(s)}{e^{s\times 4n}}$$

It is clear that $\Pr(Y \geq 4n)$ decreases when δ increases (faster server yields smaller waiting time). Thus, to obtain an upper bound, we will substitute δ with its minimal value: $0.5\ln 2$,

and by letting $s = \delta/4$ we get:

$$\Pr(Y \geq 4n) \leq \left(\frac{0.5\ln 2}{(0.5\ln 2 - 0.125\ln 2)e^{0.5\ln 2}}\right)^n = \left(\frac{\sqrt{2}}{1.5}\right)^n.$$

■

Let T_1 be the time needed for the nth customer to arrive at the system (from the side of node v_1), and T_2 – the time needed for the customer to cross n $M/M/1$ queues (from v_1 to v_n) in the equilibrium state. Also, let T_3 be the time needed for the nth customer to arrive at the system from another side (as we already mentioned, after node v_n finishes the protocol, we take all the *real* customers out of the system and inject them back through the v_n with the same rate $\lambda = \frac{\mu}{2}$). T_4 will denote the time needed for the customer to cross n $M/M/1$ queues (from v_n to v_1) in the equilibrium state. It is clear that T_3 has the same distribution as T_1, and T_4 has the same distribution as T_2.

Now, we will find the high probability bound on the total number of rounds T – needed to compete the NC task. As was argued earlier (in the proof of the expected time): $T = T_1 + T_2 + T_3 + T_4$.

Customers arrive at our system with a Poisson rate of $\lambda = \mu/2$. The minimal value of μ is $\ln 2$, since:

$\mu = -\ln\left(\frac{1}{4} - \frac{3}{4q}\right) = \ln\left(\frac{4q}{q+3}\right) \geq \ln 2$, for $q \geq 3$.

So, the minimal value of λ is $0.5 \ln 2$.

Thus, we can use Lemma 3.5 to conclude that:

$$\Pr\left(T_1 \geq 4n\right) \leq \left(\frac{\sqrt{2}}{1.5}\right)^n .$$

In order to find T_2 we will use Lemma 3.4, which states that the time needed to cross one $M/M/1$ queue is exponentially distributed with parameter $\mu - \lambda$, and Theorem 3.2 from which we conclude that waiting times in the queues are independent (waiting times depend on the number of customers in each queue, which is independent due to the Jackson's Theorem). Then we will use Lemma 3.5 to conclude that:

$$\Pr\left(T_2 \geq 4n\right) \leq \left(\frac{\sqrt{2}}{1.5}\right)^n$$

(the minimal value of $\mu - \lambda$ is $0.5 \ln 2$).

So (using union bound),

$$\Pr\left(T_1 \geq 4n \cup T_2 \geq 4n \cup T_3 \geq 4n \cup T_4 \geq 4n\right) \leq 4 \times \left(\frac{\sqrt{2}}{1.5}\right)^n$$

and thus:

$$\Pr\left(T < 16n\right) \geq \Pr\left(T_1 < 4n \cap T_2 < 4n \cap T_3 < 4n \cap T_4 < 4n\right) =$$

$$= 1 - \Pr\left(T_1 \geq 4n \cup T_2 \geq 4n \cup T_3 \geq 4n \cup T_4 \geq 4n\right) \geq 1 - 4 \times \left(\frac{\sqrt{2}}{1.5}\right)^n \geq 1 - O\left(\frac{1}{n}\right).$$

The result of the second part of the Theorem 3.1 (Equation (3.2)) is then follows:

$$\hat{R}^{EX,NC} = O(n).$$

■ [End of proof Theorem 3.1.]

3.2 `EXCHANGE` is Linear on Bounded-Degree Graphs

Now, we are ready to prove our main result. The following theorem states that the stopping time of the NC protocol over the `EXCHANGE` gossip algorithm is linear for bounded-degree graphs, both in expectation and with high probability.

Theorem 3.3 *For the synchronous time model and any bounded-degree graph B_n with a constant maximum degree d, the gossip stopping time of the NC task over the* ***`EXCHANGE`*** *gossip algorithm is linear in expectation and with high probability:*

$$\overline{R}^{EX,NC} = O(d^3 n) = O(n) \tag{3.3}$$

and

$$\hat{R}^{EX,NC} = O(d^3 n) = O(n). \tag{3.4}$$

The proof is based on Theorem 3.1, and the fact that in G^3 (third power of a graph) exists a Hamiltonian path. First, we will present the following definition and lemma and then the proof will follow.

Definition 3.1 (power of a graph) *The k-th power of a graph G, denoted as G^k, is a graph with the same set of vertices as G and an edge between two vertices iff there is a path of length at most k between them in G ([23], p. 229).*

The following lemma is a known result ([13]).

Lemma 3.6 ([13]) *For any connected graph G, G^3 contains a Hamiltonian path.*

Proof of Theorem 3.3. Consider a bounded-degree graph G with maximal degree d. Let us create a cube of the given graph – G^3. Using Lemma 3.6, we obtain a Hamiltonian path in the G^3 graph. Each edge of the G^3 graph has length of at most 3 edges in the original graph G. Thus, in the original graph G we have a virtual-Hamiltonian path that consists of *super-edges*, and each *super-edge* consists of at most 3 original edges.

Next, let us concentrate on the information flow on this virtual-Hamiltonian path. We will virtually remove all the original edges that are not included in this virtual-Hamiltonian path. Messages received via the virtually removed edges will be ignored. This assumption can only increase the stopping time since additional messages can only increase the probability that future messages will be *helpful*.

Now, we can use arguments used in Theorem 3.1. We model the virtual-Hamiltonian path as a queuing system with a queue at every node, and every two adjacent queues are connected by a *super-edge*. The *super-edges* between nodes consist of at most 3 original edges, so the probability that a *helpful* message will be delivered from a *helpful* node to its neighbor in the virtual-Hamiltonian path, in 3 rounds, is at least:

$$p = \left[\left(1 - \left(\frac{d-1}{d}\right)^2\right)\left(1 - \frac{1}{q}\right)\right]^3$$

where $\left(1 - \left(\frac{d-1}{d}\right)^2\right)$ is the probability that a message will be sent over the required original edge in a single round in the `EXCHANGE` gossip algorithm.

Using the same arguments as in the proof of Theorem 3.1, we model the service time in the queuing system as an exponential random variable with parameter (service rate) μ, where:

$\mu = -\ln(1-p) = -\ln\left(1 - \left[\left(1 - \left(\frac{d-1}{d}\right)^2\right)\left(1 - \frac{1}{q}\right)\right]^3\right)$

Note, that the *time unit* now is 3 rounds.

As was already found in Lemmas 3.2 and 3.3:

$$\overline{T} = \frac{4n}{\mu}$$

so:

$$\overline{T} = \frac{4n}{-\ln\left(1 - \left[\left(1 - \left(\frac{d-1}{d}\right)^2\right)\left(1 - \frac{1}{q}\right)\right]^3\right)}$$

where $-\ln\left(1 - \left[\left(1 - \left(\frac{d-1}{d}\right)^2\right)\left(1 - \frac{1}{q}\right)\right]^3\right)$ is bounded by a constant for any constant d and $q \geq 2$.

Moreover, we can use the following fact:

$$\ln x \leq (x-1) \Rightarrow -\ln x \geq (1-x)$$

and then:

$$\overline{T} = \frac{4n}{-\ln\left(1 - \left[\left(1 - \left(\frac{d-1}{d}\right)^2\right)\left(1 - \frac{1}{q}\right)\right]^3\right)} \leq \frac{4n}{\left[\left(1 - \left(\frac{d-1}{d}\right)^2\right)\left(1 - \frac{1}{q}\right)\right]^3}.$$

We can see that:

$$\left[\left(1 - \left(\frac{d-1}{d}\right)^2\right)\left(1 - \frac{1}{q}\right)\right]^3 \geq \left(\left(1 - \left(\frac{d-1}{d}\right)^2\right) \cdot \frac{1}{2}\right)^3 = \left(\left(\frac{2}{d} - \frac{1}{d^2}\right) \cdot \frac{1}{2}\right)^3 \geq \left(\frac{1}{d} \cdot \frac{1}{2}\right)^3 = \frac{1}{8d^3}$$

where the first inequality holds since for $q \geq 2$: $1 - \frac{1}{q} \geq \frac{1}{2}$, and the second, since for $d \geq 1$: $\frac{1}{d} \geq \frac{1}{d^2}$.

Thus,

$$\overline{T} \leq \frac{4n}{1/8d^3} = 32d^3n.$$

Following the same arguments as in the proof of Theorem 3.1, after $2 \times \overline{T}$ *time units* (which is now equal to 3 rounds), the NC task will be finished.

Thus:

$$\overline{R} \leq 2 \times \overline{T} \times 3 = O(d^3n) = O(n)$$

i.e., the expected stopping time of the NC protocol over the `EXCHANGE` gossip algorithm is linear for bounded-degree graphs.

To prove the high probability result we will use the following Lemma:

Lemma 3.7 *Let Y be the sum of n independent and identically distributed exponential random variables with parameter $\delta \geq \frac{4}{27d^3}$, where $d \geq 2$, i.e., $Y = \sum_{i=1}^{n} X_i$, $f_X(x) = \delta e^{-\delta x}$, where X_is ($i \in [1..n]$) are distributed as X.*

Then,

$$\Pr\left(Y \geq 8nd^3\right) \leq (0.992)^n .$$

Proof.

The generating function of X is given by:

$$G_X(s) = \mathrm{E}\left[e^{sX}\right] = \int_0^\infty e^{sx} f_X(x)dx$$

For any $s < \delta$:

$$G_X(s) = \frac{\delta}{\delta - s}$$

Now, we can obtain the generating function of Y:

$$G_Y(s) = (G_X(s))^n = \left(\frac{\delta}{\delta - s}\right)^n$$

Now, we will apply Markov inequality to obtain an upper bound on Y. For $s \geq 0$:

$$\Pr\left(Y \geq 8nd^3\right) = \Pr\left(e^{sY} \geq e^{s\times 8nd^3}\right) \leq \frac{\mathrm{E}\left[e^{sY}\right]}{e^{s\times 8nd^3}} = \frac{G_Y(s)}{e^{s\times 8nd^3}}$$

It is clear that $\Pr\left(Y \geq 8nd^3\right)$ decreases when δ increases (faster server yields smaller waiting time). Thus, to obtain an upper bound, we will substitute δ with its minimal value: $\frac{4}{27d^3}$, And by letting $s = \delta/4$ we get:

$$\Pr\left(Y \geq 8nd^3\right) \leq \left(\frac{\frac{4}{27d^3}}{(\frac{4}{27d^3} - \frac{1}{27d^3})e^{\frac{8d^3}{27d^3}}}\right)^n < (0.992)^n$$

■

The time needed to cross one queue in our system is distributed exponentially with parameter $\delta = \mu - \lambda$, where $\lambda = \frac{\mu}{2}$. Thus, the minimal value of δ is:

$$\delta = \mu - \lambda = 0.5\mu = -\frac{1}{2}\ln\left(1 - \left[\left(1 - \left(\frac{d-1}{d}\right)^2\right)\left(1 - \frac{1}{q}\right)\right]^3\right)$$

for $q \geq 3$, $1 - \frac{1}{q} \geq \frac{2}{3}$, so:

$$\delta \geq -\frac{1}{2}\ln\left(1 - \left[\left(1 - \left(\frac{d-1}{d}\right)^2\right)\left(\frac{2}{3}\right)\right]^3\right) = -\frac{1}{2}\ln\left(1 - \left(\frac{4}{3d} - \frac{2}{3d^2}\right)^3\right) \geq -\frac{1}{2}\ln\left(1 - \left(\frac{2}{3d}\right)^3\right)$$

Since $\ln x < x - 1$ and thus: $-\ln x > 1 - x$, we get:

$$\delta \geq \frac{4}{27d^3}$$

now, it is also clear that: $\lambda = \frac{\mu}{2} \geq \frac{4}{27d^3}$ and $\mu \geq \frac{8}{27d^3}$.

Using Lemma 3.7, we obtain: $\Pr\left(T_1 \geq 8nd^3\right) \leq (0.992)^n$ and $\Pr\left(T_2 \geq 8nd^3\right) \leq (0.992)^n$, and using the same argument as in the proof of the high probability result in the Ring graph (Theorem 3.3), we obtain the high probability result for any bounded-degre graph B_n:

$$\hat{R}^{EX,NC} = O(nd^3) = O(n).$$

■ [End of proof Theorem 3.3.]

3.3 NC with `EXCHANGE` Can be Faster Than AVG

The following theorem is a mix of known and new results. The goal of the theorem is to show that previous analytical results obtained for the AVG protocol and `EXCHANGE` algorithm cannot be used as a lower bound for NC. Such a conjecture was tempting to state since the average problem may seem simpler than NC and the AVG protocol was known to be much faster than NC on the complete graph.

Theorem 3.4

i) For **`EXCHANGE`** *the NC protocol is faster than the AVG protocol on the ring graph* R_n*:*

$$\overline{R}^{EX,NC} \leq \overline{R}^{EX,AVG} \qquad \text{and} \qquad \hat{R}^{EX,NC} \leq \hat{R}^{EX,AVG}$$

ii) For **`EXCHANGE`** *the AVG protocol is faster than the NC protocol on the complete graph* K_n*:*

$$\overline{R}^{EX,AVG} \leq \overline{R}^{EX,NC} \qquad \text{and} \qquad \hat{R}^{EX,AVG} \leq \hat{R}^{EX,NC}$$

Proof of Theorem 3.4. The first claim of the theorem then follows using the results of Boys *et al.* [6]. They proved that the number of rounds needed to complete the AVG task is $\Theta(\log n + T_{mix})$ with high probability (where T_{mix} is the mixing time of a simple random walk on the given graph). The mixing time of the simple random walk on a Ring graph is $\Omega(n^2)$ [17] and thus, the number of rounds needed to complete the AVG task is $\hat{R}^{EX,AVG} = \Omega(n^2)$ with high probability. From this result it follows that the expectation of the averaging time is at least of the same order as the high probability time. Hence: $\overline{R}^{EX,AVG} = \Omega(n^2)$. In Theorem 3.4 we showed that $\hat{R}^{EX,NC} = O(n \log n)$ and $\overline{R}^{EX,NC} = O(n)$. Thus, the first part of the theorem holds.

The second part of the theorem is a restatement of known results. In [15], the authors show that the number of rounds (in a complete graph) for the AVG is $\hat{R}^{EX,AVG} = O(\log n)$

w.h.p., while in [14] they give a lower bound of $\hat{R}^{EX,AVG} = \Omega(\log n)$. For NC, in [8] the authors show that the number of rounds needed for the NC protocol in the complete graph is $O(n)$ in expectation and with high probability. Thus the second part of the theorem holds.
■

Chapter 4

`EXCHANGE` vs. `PUSH` and `PULL`

Our second result is about the power of using `EXCHANGE` instead of `PUSH` and `PULL` algorithms.

Theorem 4.1 *For the Star graph S_n, algebraic gossip (i.e., NC) using* `EXCHANGE` *is unboundedly better than using* `PUSH` *or* `PULL` *algorithms, formally: for $\mathcal{A} \in \{$*`PUSH`*, *`PULL`*$\}$*

$$\lim_{n\to\infty} \frac{\hat{R}^{\mathcal{A},NC}}{\hat{R}^{EX,NC}} \to \infty \tag{4.1}$$

and for any node v

$$\lim_{n\to\infty} \frac{\overline{R}_v^{\mathcal{A},NC}}{\overline{R}_v^{EX,NC}} \to \infty. \tag{4.2}$$

We give here a short proof overview; the full details are below. We first show in Lemma 4.1 that for $\mathcal{A} \in \{\texttt{PUSH}, \texttt{PULL}\}$ the stopping time of the CI protocol is a lower bound for the stopping time of the NC protocol. Next, we prove in Lemma 4.4 that for the Star graph S_n, we have $\overline{R}_v^{\mathcal{A},CI} = \Omega(n \log n)$ and $\hat{R}^{\mathcal{A},CI} = \Omega(n \log n)$. Therefore, we conclude that the NC on the Star is $\Omega(n \log n)$, both in expectation per node and with high probability. We then prove in Lemma 4.5 that for a Star graph, if the NC protocol uses `EXCHANGE`, then $\overline{R}_v^{EX,NC} = O(n)$ and $\hat{R}^{EX,NC} = O(n)$. Thus, the theorem follows.

We first give some definitions that will be used in the proof:

Definition 4.1 (deterministic stopping time) *Given a graph G, specific* **`run`** *on this graph ($\sigma \in \Pi(G)$), protocol $\mathcal{P}$, and algorithm $\mathcal{A}$, the* deterministic stopping time *of gossip protocol* **`run`** *is the minimal number of timeslots by which the protocol (or task) is completed:*

$$T_{\sigma}^{\mathcal{A},\mathcal{P}} = \min_{t \in \mathbb{Z}^{+}} \Big[t \mid \mathbb{S}_{V}^{\mathcal{P}}(t) = TRUE \Big].$$

Similarly, we define $T_{v,\sigma}^{\mathcal{A},\mathcal{P}}$ for a node v.

Definition 4.2 (t-subset of algorithm `runs`) *A subset of algorithm* **`runs`** *where the stopping time for them is t:*

$$\Pi_{t}^{\mathcal{A},\mathcal{P}}(G) = \Big\{ \sigma \mid \sigma \in \Pi(G), T_{\sigma}^{\mathcal{A},\mathcal{P}} = t \Big\}$$

Definition 4.3 (t^{+}-subset of algorithm `runs`) *A subset of algorithm* **`runs`** *where the stopping time for them is at least t:*

$$\Pi_{t^{+}}^{\mathcal{A},\mathcal{P}}(G) = \Big\{ \sigma \mid \sigma \in \Pi(G), T_{\sigma}^{\mathcal{A},\mathcal{P}} \geq t \Big\}$$

Definition 4.4 (*Helpful* node in the NC protocol) *A node v is* helpful *to node u at the timeslot t in the NC protocol, if and only if $S_v(t) \not\subset S_u(t)$, i.e, iff a random linear combination constructed by v can be linearly independent with all equations (messages) stored in the u's DB.*

Definition 4.5 (*Evidence* in the NC protocol) *For a variable x_i, which represents the initial value of node v_i, we define an* evidence *of x_i as a linear equation containing x_i.*

Our first claim is that for all three algorithms, CI protocol is faster than NC, formally:

Lemma 4.1 *For any graph G and for gossip algorithm $\mathcal{A} \in \{$**`PUSH`**$,$ **`PULL`**$,$ **`EXCHANGE`**$\}$:*

$$\Pr\Big(R^{\mathcal{A},CI} \geq t\Big) \leq \Pr\Big(R^{\mathcal{A},NC} \geq t\Big)$$

To prove Lemma 4.1, first we will show that for any specific algorithm `run` $\sigma \in \Pi(G)$ the stopping time of the CI protocol is less than or equal to the stopping time of the NC protocol.

Lemma 4.2 *For any graph $G(V,E)$, gossip algorithm $\mathcal{A} \in \{$PUSH, PULL, EXCHANGE$\}$ and any algorithm **run** $\sigma \in \Pi(G)$:*

$$T_{\sigma}^{\mathcal{A},CI} \leq T_{\sigma}^{\mathcal{A},NC}$$

To prove Lemma 4.2 we need the following two claims.

Claim 4.1 *For CI protocol: If and only if nodes u and v are* connected[1] *by the timeslot t, $D_u(0) \subset D_v(t)$.*

Proof. Since, in the CI protocol, every transmitted message carries all the values seen so far by the sending node, existence of the *information path* directly implies that the information known to the node at the beginning of the *path* (u) is known to the node at the end of the *path* (v). On the other hand, if there is no *information path* between the nodes, then node v has not seen any message containing the initial value of node u. ■

Claim 4.2 *If nodes u and v are* not connected *by the timeslot t then $\mathbb{S}_V^{NC}(t) = FALSE$.*

Proof. The goal of the NC task is that every node will be able to decode all the initial values in the network, i.e., will be able to solve n linear equations with n variables. If an *information path* between v_i and v_j by the timeslot t **does not exist**, then v_j did not receive any *evidence* regarding the initial value of v_i (x_i) by the end of timeslot t which means that the database of node v_j does not include any linear equation with x_i and thus, v_j cannot decode all the initial values and finish the protocol. ■

Now we can prove Lemma 4.2:

[1]As mentioned, two nodes are *connected* iff there exists at least one *information path* between them.

Proof of Lemma 4.2. The proof will be by contradiction, let us assume that by the timeslot $t = T_{\sigma}^{\mathcal{A},NC}$, NC protocol finished, but the CI protocol still did not. This implies that, in CI protocol, there is at least one node (let it be node v_b) that still does not know all the initial values ($\overline{x}$) in the network. Thus, there exists at least one node (let it be node v_a) such that: $D_{v_a}(0) \not\subset D_{v_b}(t)$. Now, from Claim 4.1, we can conclude that nodes v_a and v_b are *not connected*. Hence, from Claim 4.2, $S_V^{NC}(t) = FALSE$ which contradicts the assumption that NC protocol had finished, i.e., $T_{\sigma}^{\mathcal{A},CI} \leq T_{\sigma}^{\mathcal{A},NC}$. ■

Now we can continue with the proof of Lemma 4.1:

Proof of Lemma 4.1. From Lemma 4.2: $\forall \sigma \in \Pi(G)$

$$T_{\sigma}^{\mathcal{A},NC} \geq T_{\sigma}^{\mathcal{A},CI}$$

thus:

$$\Pi_{t^+}^{\mathcal{A},CI}(G) \subseteq \Pi_{t^+}^{\mathcal{A},NC}(G)$$

Let, $\sigma_1 \in \Pi_{t^+}^{\mathcal{A},NC}(G)$ and $\sigma_2 \in \Pi_{t^+}^{\mathcal{A},CI}(G)$ then:

$$\begin{aligned}
\Pr\left(T^{\mathcal{A},NC} \geq t\right) &= \sum_{t_1 \geq t} \Pr\left(T^{\mathcal{A},NC} = t_1\right) \\
&= \sum_{\sigma_1} \Pr(\sigma_1) \geq \sum_{\sigma_2} \Pr(\sigma_2) \\
&= \Pr\left(T^{\mathcal{A},CI} \geq t\right).
\end{aligned}$$

And since: $R^{\mathcal{A},\mathcal{P}} = n \times T^{\mathcal{A},\mathcal{P}}$ we obtain:

$$\Pr\left(R^{\mathcal{A},CI} \geq t\right) \leq \Pr\left(R^{\mathcal{A},NC} \geq t\right)$$

■

The next lemma is an immediate result of Lemma 4.1.

Lemma 4.3 *For any graph $G(V, E)$, and gossip algorithm $\mathcal{A} \in \{$PUSH, PULL, EXCHANGE$\}$:*

$$\overline{R}^{\mathcal{A},CI} \leq \overline{R}^{\mathcal{A},NC}$$

and

$$\hat{R}^{\mathcal{A},CI} \leq \hat{R}^{\mathcal{A},NC}$$

Proof. For positive integer random variable X [19]:

$$\mathrm{E}\left[X\right] = \sum_{i=1}^{\infty} \Pr\left(X \geq i\right)$$

Thus,

$$\mathrm{E}\left[R^{\mathcal{A},CI}\right] = \sum_{t=1}^{\infty} \Pr\left(R^{\mathcal{A},CI} \geq t\right)$$

Using Lemma 4.1:

$$\sum_{t=1}^{\infty} \Pr\left(R^{\mathcal{A},CI} \geq t\right) \leq \sum_{t=1}^{\infty} \Pr\left(R^{\mathcal{A},NC} \geq t\right) = \mathrm{E}\left[R^{\mathcal{A},NC}\right]$$

And thus: $\overline{R}^{\mathcal{A},CI} \leq \overline{R}^{\mathcal{A},NC}$.

Since

$$\hat{R}^{\mathcal{A},\mathcal{P}} = \min_{t\in\mathbb{Z}} \left[t \mid \Pr\left(R^{\mathcal{A},\mathcal{P}} \leq t\right) \geq 1 - O\left(\frac{1}{n}\right)\right]$$

which implies that:

$$\hat{R}^{\mathcal{A},\mathcal{P}} = \min_{t\in\mathbb{Z}} \left[t \mid \Pr\left(R^{\mathcal{A},\mathcal{P}} \geq t\right) \leq O\left(\frac{1}{n}\right)\right].$$

We can conclude that $\hat{R}^{\mathcal{A},CI}$ is the minimal number of rounds for which: $\Pr\left(R^{\mathcal{A},CI} \geq \hat{R}^{\mathcal{A},CI}\right) \leq O(\frac{1}{n})$ and $\hat{R}^{\mathcal{A},NC}$ is the minimal number of rounds for which: $\Pr\left(R^{\mathcal{A},NC} \geq \hat{R}^{\mathcal{A},NC}\right) \leq O(\frac{1}{n})$.

Now, let us assume that $\hat{R}^{\mathcal{A},CI} > \hat{R}^{\mathcal{A},NC}$.

From Lemma 4.1:

$$\Pr\left(R^{\mathcal{A},NC} \geq \hat{R}^{\mathcal{A},NC}\right) \geq \Pr\left(R^{\mathcal{A},CI} \geq \hat{R}^{\mathcal{A},NC}\right)$$

and since

$$O\left(\frac{1}{n}\right) \geq \Pr\left(R^{\mathcal{A},NC} \geq \hat{R}^{\mathcal{A},NC}\right) \geq \Pr\left(R^{\mathcal{A},CI} \geq \hat{R}^{\mathcal{A},NC}\right)$$

it follows that:

$$\Pr\left(R^{\mathcal{A},CI} \geq \hat{R}^{\mathcal{A},NC}\right) \leq O\left(\frac{1}{n}\right).$$

Now, we have got a contradiction to the fact that $\hat{R}^{\mathcal{A},CI}$ is the minimal value for which $\Pr\left(R^{\mathcal{A},CI} \geq \hat{R}^{\mathcal{A},CI}\right) \leq O(\frac{1}{n})$, since we assumed that $\hat{R}^{\mathcal{A},CI} > \hat{R}^{\mathcal{A},NC}$. Hence, our assumption is wrong and thus: $\hat{R}^{\mathcal{A},CI} \leq \hat{R}^{\mathcal{A},NC}$. ■

Next, we provide a lower bound for the NC protocol over `PUSH`, `PULL` by bounds on the CI protocol.

Lemma 4.4 *For the synchronous time model, the Star graph S_n and $\mathcal{A} \in \{$PUSH, PULL$\}$:*

i) $\overline{R}^{\mathcal{A},CI} = \Theta(n \log n)$, *and*

ii) $\hat{R}^{\mathcal{A},CI} = \Theta(n \log n)$.

The proof is based on bounds for the *coupon collector problem* [19].

Claim 4.3 *Let X be the r.v. for the number of coupons needed to obtain n distinct coupons, then:*

$$E[X] = \Theta(n \log n) \quad \textit{and w.h.p.} \quad X = \Theta(n \log n).$$

Proof. The first results and the upper bound w.h.p. are well known; see for example [21, 19]. We have not found a direct reference for the lower bound, namely that w.h.p. $X = \Omega(n \log n)$, so we give an outline here. Let $\mathcal{E}_x$ denote the event that all n different coupons have been collected after X steps. Let $X = \sum_{i=1}^{n} X_i$ where X_i is an r.v. that denotes the number of coupons of type i collected. Clearly X_i's are dependent. To overcome this difficulty we will use Poisson approximation of the binomial random variable X_i [19]. Consider n Poisson independent random variables Y_i ($i \in [1...n]$) with mean $\lambda = \frac{X}{n}$. Each variable represents the number of coupons of type i. Thus, the expected total number of coupons collected is X. Let $\mathcal{E}_y$ denote the Poisson version of the event $\mathcal{E}_x$, i.e., that after collecting the different types of coupons independently with Poisson distribution with λ, we have at least one type of each coupon. Since Y_i's are i.i.d., we have $\Pr(\mathcal{E}_y) = (\Pr(Y_i \geq 1))^n$. It is clear that both $\Pr(\mathcal{E}_x)$ and $\Pr(\mathcal{E}_y)$ are monotonically increasing with X; therefore we can use the Poisson approximation that states that $\Pr(\mathcal{E}_x) \leq 2 \times \Pr(\mathcal{E}_y)$ ([19], Corollary 5.11). Now, assume $X = n \ln n - n \ln\ln n$ and we have:

$$\begin{aligned}\lim_{n\to\infty} \Pr(\mathcal{E}_y) &= \left(1 - e^{-(\ln n - \ln\ln n)}\right)^n \\ &= \left(\frac{1}{e}\right)^{\ln n} = \frac{1}{n}.\end{aligned}$$

Thus: $\lim_{n\to\infty} \Pr(\mathcal{E}_x) \leq \frac{2}{n}$ and

$$\lim_{n\to\infty} \Pr\{X \geq n\ln n - n\ln\ln n\} = 1 - \frac{2}{n}.$$

■

Proof of Lemma 4.4. We divide the task into two phases, the first phase is the time T_1 until the center node v_1 learns all the values of $\overline{x}$. The second phase is the time T_2 it takes v_1 to distribute the information to all the nodes. In `PUSH`, the center node v_1 will know all the initial values $\overline{x}$ in the network after exactly n timeslots, so $T_1 = n$. Now, to distribute the $\overline{x}$ to all nodes, the center node should reach (`PUSH`) to all the other nodes one by one. In synchronous time model, v_1 will transmit once in n consecutive timeslots (or once in one round). This is exactly the coupon collector problem (or balls into bins), so T_2 is $\Theta(n^2 \log n)$ both in expectation and w.h.p. and the result follows for `PUSH`.

For the `PULL` algorithm, the arguments are similar but T_1 is $\Theta(n^2 \log n)$ both in expectation and w.h.p., and $T_2 = n$. ■

Lemma 4.5 *For the synchronous time model and the Star graph S_n:*

i) $\hat{R}^{EX,NC} = O(n)$

ii) $\overline{R}_v^{EX,NC} = O(n)$

To prove Lemma 4.5 we will use the following two claims.

Claim 4.4 *Let X_i be independent geometric random variables with parameter p, and let $X = \sum_{i=1}^{n} X_i$. For $p \geq \frac{1}{2}$:*

$$\Pr(X \geq 4n) \leq \left(\frac{2}{3}\right)^n$$

Proof. In order to obtain this upper bound on the sum of n independent geometric random variables we will use a Chernoff bound. The generating function of a geometric random variable X_i is given by:

$$G_{X_i}(t) = \mathrm{E}\left[e^{tX_i}\right] = \frac{pe^t}{1-(1-p)e^t}$$

where $t < -\ln(1-p)$.

The generating function of the sum of independent random variables is a multiplication of their generating functions. Thus:

$$G_X(t) = \mathrm{E}\left[e^{t\sum_{i=1}^{n} X_i}\right] = \mathrm{E}\left[e^{ntX_i}\right] = \left(\frac{pe^t}{1-(1-p)e^t}\right)^n.$$

Now, we will apply Markov inequality to obtain an upper bound on X. For $t \geq 0$:

$$\Pr(X \geq 4n) = \Pr\left(e^{tX} \geq e^{t\times 4n}\right) \leq \frac{\mathrm{E}\left[e^{tX}\right]}{e^{t\times 4n}} = \frac{G_X(t)}{e^{t\times 4n}}.$$

By letting $t = -0.5\ln(1-p)$ we get:

$$\Pr(X \geq 4n) \leq \left(\frac{(1-p)^{1.5}p}{1-(1-p)^{0.5}}\right)^n.$$

It is clear that $\Pr(X \geq 4n)$ decreases when p increases. Thus, to obtain an upper bound, we will substitute p with its minimal value, i.e., $\frac{1}{2}$, and we get the result:

$$\Pr(X \geq 4n) \leq \left(\frac{(1-0.5)^{1.5}0.5}{1-(1-0.5)^{0.5}}\right)^n \leq \left(\frac{2}{3}\right)^n$$

■

The following lemma is a part of Lemma 2.1 in [8].

Lemma 4.6 *Suppose that the node v is* helpful *to the node u at the beginning of the timeslot t. If v transmits a message to u at the timeslot t,*

$$\Pr\left(dim(S_u(t+1)) > dim(S_u(t))\right) \geq 1 - \frac{1}{q}.$$

We can now proceed with the proof of Lemma 4.5.

Proof of Lemma 4.5. We will prove the result with high probability and omit the proof of the result about the expectation, which is similar. As before, we will split the task into two phases, the first phase is the time T_1 until the center node v_1 learns all the values of $\overline{x}$, i.e., $dim(S_{v_1}(t)) = n$. The second phase is the time T_2 it takes v_1 to distribute the information to all the nodes. Since every node $u \in V \setminus \{v_1\}$ is *helpful* to v_1, by Lemma 4.6, after $\log_q n$ rounds (or $n \times log_q n$ timeslots), the first phase will end with high probability.

Now, from the beginning of phase two, and since $dim(S_{v_1}) = n$, node v_1 will be *helpful* to every other node until the rank of that node becomes n. From Lemma 4.6, a message transmitted to some node from a node *helpful* to it, will increase its dimension with probability $p \geq 1 - \frac{1}{q}$.

Let us define X_i^u as the number of rounds needed for v_1 to increase the rank of some node $u \in V \backslash \{v_1\}$. It is clear that X_i has a geometric distribution with parameter p. We are interested to find $X^u = \sum_{i=1}^{n} X_i^u$, which represents the number of rounds by which the rank of node u will become n. Using Claim 4.4 (and the fact that for $q > 2$, $p = 1 - \frac{1}{q} > \frac{1}{2}$), we obtain that $X^u < 4n$ with probability $1 - (\frac{2}{3})^n$.

Using union bound, we obtain the probability that ranks of all nodes will become n after $4n$ rounds: $\Pr\left(\cap_{u \in V \backslash \{v_1\}} X^u < 4n\right) \geq 1 - n\left(\frac{2}{3}\right)^n$. Thus, $T_2 = 4n^2$ w.h.p. Part ii) can be proved similarly by showing that for each node v the expected number of rounds to reach rank n is linear since the center node is always *helpful*. ■

Proof of Theorem 4.1. The proof of Equation (4.1) is an immediate result of Lemmas 4.3, 4.4, and 4.5. Lemma 4.3 states that the stopping time of the CI protocol is a lower bound for the stopping time of the NC protocol. Lemma 4.4 shows that the stopping time of the CI protocol over PUSH and PULL gossip algorithms in the Star graph S_n is of the order of $n \log n$ with high probability. Combining Lemmas 4.3 and 4.4 we obtain a lower bound on the stopping time of the NC protocol over PUSH and PULL gossip algorithms, i.e., $\hat{R}^{\texttt{PULL},NC} = \Omega(n \log n)$. In Lemma 4.5 we proved that the stopping time of the NC protocol over the EXCHANGE gossip algorithm in the Star graph S_n is of the order $O(n)$ with high probability, i.e., $\hat{R}^{EX,NC} = O(n)$. Thus, for $\mathcal{A} \in \{\texttt{PUSH}, \texttt{PULL}\}$:

$$\lim_{n\to\infty} \frac{\hat{R}^{\mathcal{A},NC}}{\hat{R}^{EX,NC}} = \frac{\Omega(n \log n)}{O(n)} \to \infty.$$

The proof of Equation (4.2) is similar since using the coupon collector results we can show that for $\mathcal{A} \in \{\texttt{PUSH}, \texttt{PULL}\}$, $\overline{R}_v^{\mathcal{A},NC} = \Omega(n \log n)$ and from Lemma 4.5, $\overline{R}_v^{EX,NC} = O(n)$. ■

Chapter 5

Simulations

5.1 Simulator Features

For evaluating and strengthening our conjectures regarding stopping times of presented above algorithms and protocols, we developed a simulation tool with the following main features:

- Written in C.
- Convenient command line interface.
- Can be compiled and executed on Linux or Windows.
- Results are automatically logged into text files and plotted on graphs.
- Various topologies are supported: line, ring, star, clique, lollipop, grid, torus.
- Performs real ”Network Coding” simulation.

The program consists of the following main modules:

- Topology Module – responsible for creating adjacency matrices of desired graphs.

- Node Module – implements basic node functions (functionality common to all nodes in all protocols).
- Simulation Module – defines the simulation scenarios.
- Task-specific modules – implements node functions required by specific protocols:
  - "Find Maximum" module.
  - "Collect Information from All" module.
  - "Find Average" module.
  - "Network Coding" module.

5.2 Simulations Setup

We simulated various network topologies represented by the following graphs: complete, ring, star, 2-dimensional grid, lollipop (a graph that consists of a complete graph with $\frac{n}{2}$ vertices, attached by a line graph of length $\frac{n}{2}$), and hypercube (a graph with $n = 2^k$ vertices that can be represented as k-dimensional vectors over $\{0, 1\}$ and there exists an edge between two vertices iff the Hamming distance between the vectors is 1).

Both time models presented in the work (synchronous and asynchronous) were tested. We simulated all presented above gossip algorithms $\mathcal{A} \in \{\texttt{PUSH}, \texttt{PULL}, \texttt{EXCHANGE}\}$, and over these algorithms we tested the gossip protocols $\mathcal{P} \in \{CI, AVG, NC\}$. The parameter that was tested in the simulation is the average stopping time $\overline{R}^{\mathcal{A},\mathcal{P}}$ (each experiment was repeated 30 times) of the corresponding simulation setup as a function of n, which is the number of nodes in a given graph.

In our results, the maximal number of nodes for all graphs is about 300 and this limitation is dictated by a relatively complex NC protocol. During this protocol, each node, after receiving a message, has to decide whether it is *helpful* or not, i.e, it should determine whether the received message is linearly independent of all other messages stored in the node's DB. This operation requires matrix triangulation, which is relatively heavy. In all simulations of the NC protocol we assumed a finite field $\mathbb{F}_q$ with $q = 128$.

5.3 Results

In this section we present the most interesting and even surprising results.

5.3.1 `EXCHANGE` is faster than `PUSH`/`PULL`

It is clear that using the `EXCHANGE` algorithm instead of `PUSH` or `PULL` can only speed up the protocol, since at each timeslot two messages are sent instead of one. But what is surprising, is that the speedup achieved by `EXCHANGE` is not just a constant factor, but sometimes it is even unbounded. In Figure 5.1 we can see the difference in stopping times of the NC protocol in a Star graph when different gossip algorithms are used. We can see that the `EXCHANGE` algorithm yields much smaller stopping times, and its benefit grows with n. In a Star graph, `PUSH` or `PULL` takes, on average $\overline{R}_v^{\mathcal{A},NC} = \Omega(n \log n)$ rounds for a particular node to finish the NC protocol, and using the `EXCHANGE`, it takes in average $\overline{R}_v^{EX,NC} = O(n)$ rounds for a particular node. Thus, it is clear that using `EXCHANGE` we will also use many fewer (even unboundedly fewer) messages than using `PUSH` or `PULL`, since the number of messages per each round differ only by a factor of 2 (`EXCHANGE` uses two messages per communication action, and `PUSH` or `PULL` use only one message). In the complete graph, there is no unbounded difference between `EXCHANGE` and `PUSH` or `PULL` algorithms in the NC protocol. All algorithms achieve a linear ($O(n)$) stopping time. But also in this case, where the `EXCHANGE` is better than `PUSH` or `PULL` only by a constant factor, the number of messages used by `EXCHANGE` is much smaller than the number of messages used by `PUSH` or `PULL`. We can see this in Figure 5.2.

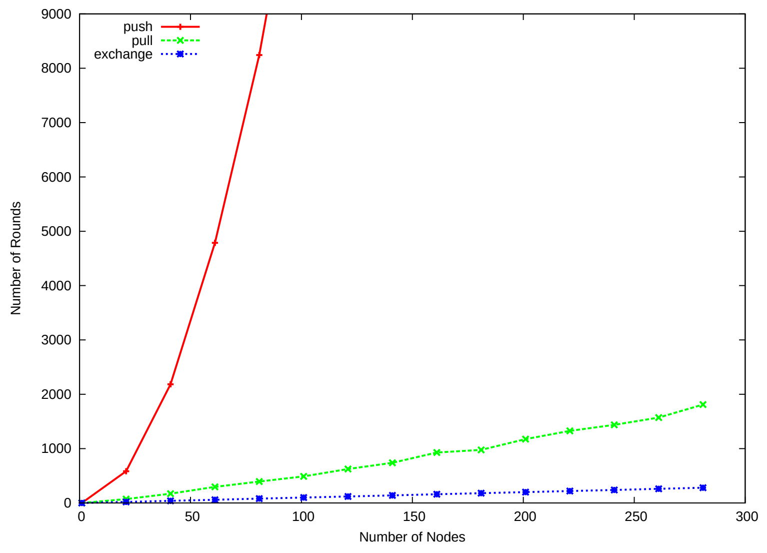

Figure 5.1: Stopping time (in rounds). Synchronous time model, NC protocol, Star graph, `PUSH`, `PULL`, `EXCHANGE`.

In Figure 5.3 we can see the number of messages used by the NC protocol in star graph, when various gossip algorithms are used. It is clear that the `EXCHANGE` protocol uses many fewer messages to complete the task.

5.3.2 NC over `EXCHANGE` is linear on many graphs

The NC protocol over the `EXCHANGE` gossip algorithm is linear for many graphs. In Figure 5.4 we can see the stopping times of the NC protocol over `EXCHANGE` in various network topologies and the synchronous time model. We can see that for all tested graphs, the stopping time is linear with the number of nodes in the network. We also performed a similar simulation for the asynchronous time model and the results were almost the same, with a small difference that in the synchronous model all stopping times are slightly smaller than in the asynchronous case.

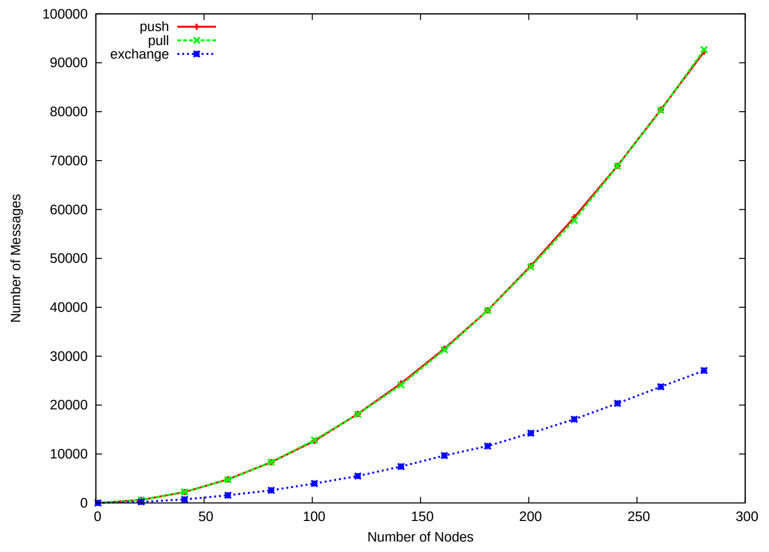

Figure 5.2: Number of messages sent in the network until NC is finished. Asynchronous time model, NC protocol, Complete graph, `PUSH`, `PULL`, `EXCHANGE`.

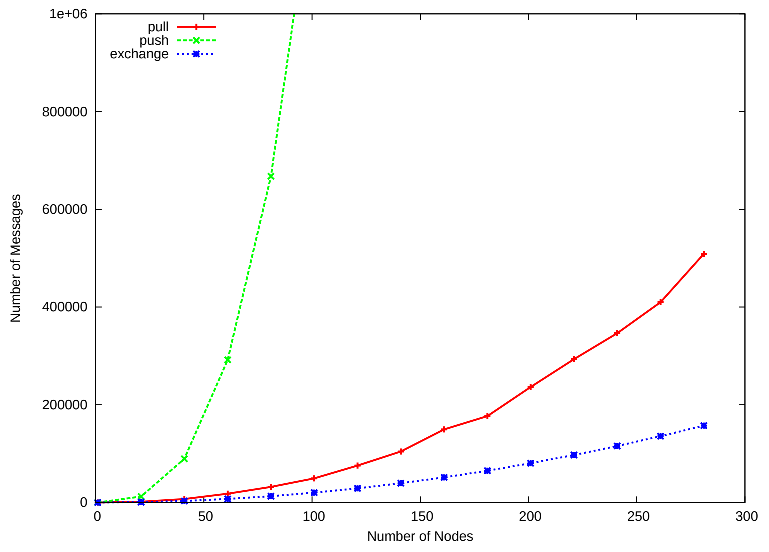

Figure 5.3: Number of messages sent in the network until NC is finished. Synchronous time model, NC protocol, Star graph, `PUSH`, `PULL`, `EXCHANGE`.

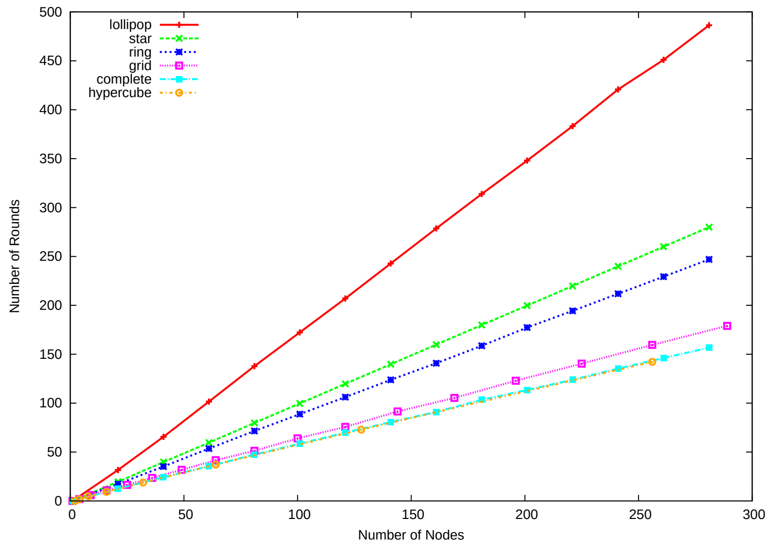

Figure 5.4: Synchronous time model, NC protocol, `EXCHANGE` algorithm, various topologies. Stopping time is linear.

In Figure 5.5 we can see the results of almost the same simulation setup, but this time we used the asynchronous time model. As we can see, both time models achieve very similar results.

5.3.3 NC over `EXCHANGE` can be faster than AVG

Since the task of the NC protocol is that every node will know the initial value of every other node once the NC task is completed, every node can determine the exact average of all initial values in the network. Thus, one can conclude that the AVG task should take less time than the NC task, since by completing NC we clearly discover the average value. But this is not true. It was already shown theoretically, that for a Ring graph it is faster to execute the NC protocol than the AVG. In Figure 5.6 we can see the ratio between the AVG and NC stopping times and this ratio grows linearly with n for the both time models.

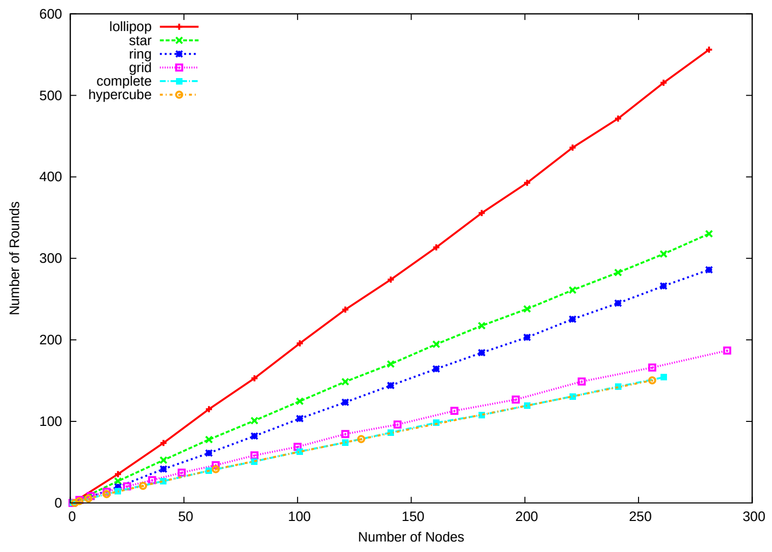

Figure 5.5: Asynchronous time model, NC protocol, EXCHANGE algorithm, various topologies. Stopping time is linear.

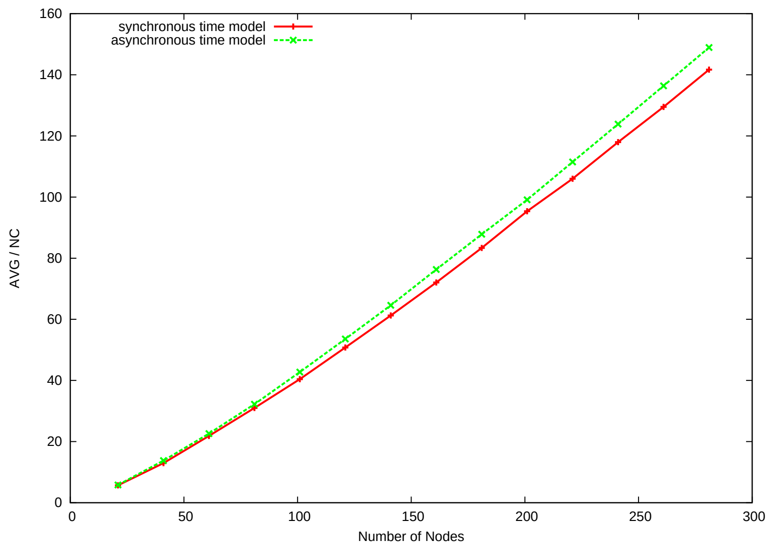

Figure 5.6: Synchronous and asynchronous time models, EXCHANGE algorithm, $\frac{AVG}{NC}$ in Ring graph. NC is unboundedly faster in Ring graph.

Chapter 6

Conclusions and Future Work Towards Ph.D.

6.1 Conclusions

We promote the use of the `EXCHANGE` gossip algorithm for the network coding (NC) protocol. This is a new approach that was recently and independently proposed by [25]. In this work we prove tight bounds on the stopping time of the network coding (NC) protocol over the `EXCHANGE` gossip algorithm. We provide tight linear bounds for the Star and Ring topologies and for any bounded-degree graph as well.

It is important to note that all previous results for general topologies ([20] and [25]) give non-linear bounds. While in [20] a spectral-gap analysis approach is used and in [25] graph theory is used, we provide a completely different approach (in the proofs of the bounds for Ring graph – Theorem 3.1 and for any bounded-degree graph – Theorem 3.3). We relate the network coding problem to queuing theory and particularly to Jackson's networks, and we believe that this approach will lead us to new general results.

The simulation tool that we have developed allows us to simulate large networks (up to 350 nodes) where each node performs a real network coding task (which involves matrix triangulation procedure over a finite field after every communication action). We did not encounter such tools (or simulation results with the same order of nodes) in the literature. Simulation results presented in [8] use networks up to 32 nodes and a single, complete graph topology. Our tool is capable of simulating many varied topologies (see section 5.1) and both time models: synchronous and asynchronous.

Our theoretical results were proved for the synchronous time model, where every node acts exactly once in one round (round is n consecutive timeslots). Simulation results show (e.g., Figures 5.4 and 5.5) that there almost no difference between the synchronous and the asynchronous models. This observation is quite expected, because when n is relatively large, and the number of rounds needed to accomplish the NC task is at least $\overline{R} = O(n)$, both time models are almost the same since, on average, each node will act once in a round, i.e., $\overline{R}$ is the expected number of times that a node acts during $\overline{R}$ rounds. In other words, there are formal proofs or intuitive conjectures that both models give almost the same results. In [8] there is a synchronous time model used, while in [20] and [25] the asynchronous model is used. In [5] the authors showed that in both time models, the stopping time of the AVG (Find Average) task is the same.

Recently, we have found a graph – a Barbell graph (Figure 6.1) for which the expected stopping time of the network coding protocol is $\overline{R} = \Omega(n^2)$ and $\overline{R} = O(n^2)$ thus giving: $\overline{R} = \Theta(n^2)$. Moreover, we believe that the Barbell graph is the worst case topology for the algebraic gossip and thus, we are aiming to prove the following conjecture:

Conjecture 6.1 (Conjecture we are aiming to prove) *The expected stopping time of the algebraic gossip protocol (i.e., linear network coding over gossip algorithm) on any connected graph G_n is: $\overline{R} = O(n^2)$, and there exists a graph for which $\overline{R} = \Theta(n^2)$.*

If our conjecture is true, the result for general graphs presented in [25] is incorrect since it states that for any graph: $\overline{R} = O(n \log n)$. This fact gives even more power to our result

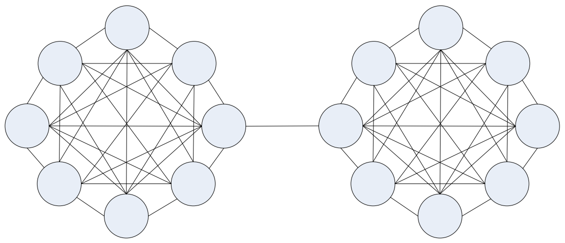

Figure 6.1: Barbell graph. Two cliques of size $\frac{n}{2}$ connected with a single edge.

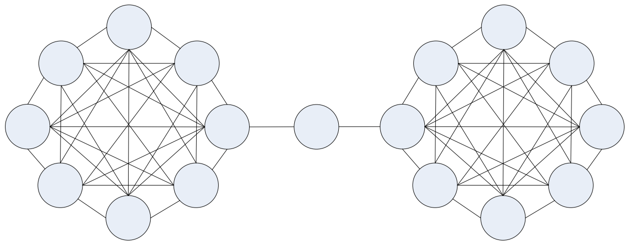

Figure 6.2: Extended-Barbell graph. Two cliques of size $\frac{n-1}{2}$ connected via a single node.

– linear bound for any bounded-degree graph (Theorem 3.3).

An interesting observation is the fact that on the Extended-Barbell graph (Figure 6.2) the expected stopping time of algebraic gossip seems to be linear. So, adding a single node to the graph changes the bound by an order of magnitude! This leads us to a need for an algorithm that will compute the stopping time bound given a specific topology.

6.2 Open Questions and Future Work Towards Ph.D.

In the Ph.D. work I will continue to investigate the problem of efficient information dissemination and network utilization using the network coding approach, which is a new research area in networking and information theory. Moreover, I will explore the gossip algorithms in other network models that are more suitable for sensors and wireless networks (e.g., *dynamic graphs* ([2]) and *SINR* ([1]) models). As well, other open questions related to network capacity and optimal power allocation will be addressed (Zander's [26] approach for optimal power allocation). Following are the specific research directions that I plan to address in my Ph.D. research:

6.2.1 Algebraic Gossip

1. Extensions to the results presented in [4].

    - The general bound for algebraic gossip that we proved in [4] is a function of Δ - maximum degree of a graph. This bound is not tight for specific topologies, e.g., for complete graph. Thus, the maximum degree does not fully capture the behavior of the algebraic gossip. A different approach used by Mosk-Aoyama and Shah in [20], is based on a measure of conductance of the network. As the authors mentioned, the offered bound is not tight, which indicates that the conductance-based measure does not reflect the behavior of the protocol. Hence, we would like to develop a deterministic algorithm that will compute the stopping time of algebraic gossip for any given topology by determining the graph properties that affect the stopping time.
    - Extend our results to many-to-many and many-to-all dissemination models. Until now, we have talked about information dissemination from all nodes in the network to all nodes (a.k.a. all-to-all dissemination). The many-to-all dissemination model has been investigated in [8], but only for a complete graph.
    - When is network coding better than store-and-forward? When is algebraic gossip is the most efficient way to disseminate information? In order to answer these questions we should analyze the bit complexity of the algebraic gossip algorithm and not only the stopping time. For example, in [9] the authors propose a message dissemination store-and-forward epidemic algorithm that achieves the same

bound as algebraic gossip. We want to investigate various aspects of algebraic gossip performance: bit complexity, running time, direction of communication (push/pull/exchange), local knowledge required by nodes, etc. We would like to find scenarios in which algorithm with network coding is definitely better than any store-and-forward approach.

- Study the network coding and gossip problems on dynamic graphs, based on the dynamic graphs models presented in [2], and on the SINR model presented in [1]. To the best of our knowledge, these problems have not yet been reported in the literature.

2. Terminals with Side Information.

Consider a network with a source X that wishes to multicast information to a group of terminals $T = \{t_1, t_2, ..., t_k\}$. Each terminal t_i has some side information Y_i regarding the source X. From the Slepian-Wolf problem [24] it follows that the sufficient information rate required by terminal t_i in order to decode X is $H(X|Y_i)$. Such rates can be obtained using a 'random binning' coding scheme. Consequently, the 'bins rate' for each terminal will be $2^{nH(x|Y_i)}$. Now, we want to utilize the gossip communication scheme for efficient dissemination of the bin index. Let's take for example (see Fig. 6.3) $T = \{t_1, t_2\}$, $nH(X|Y_1) = 8$, $nH(X|Y_2) = 4$. In order to decode the message sent from X, t_1 needs to receive 3 bits that identify the bin of the message, while t_2 needs only 2 bits.

The source X sends out a (random) linear combination of 3 bits to a random neighbor, which in turn does the same (i.e., gossip communication scheme). Once t_1 receives 3 independent equations, it can identify the bin of the original message. It can be shown that t_2 needs to receive only 2 independent equations in order to be able to identify the bin (note, the bins for t_1 and t_2 are not the same). Clearly, after decoding the original message, t_2 knows all the 3 bits needed to t_1 and thus it can continue to participate in the algorithm and help t_1 to finish.

We would like to analyze the stopping time of such multicast algorithm, i.e., after how many communication steps, all k terminals will know the original message of X. We will consider various network topologies, and, possibly, will give bounds for general graphs and the worst case source location.

3. Correlated Sources.

   In the real world scenarios, especially in the sensors networks, the data collected by nodes is not completely independent. Thus, we can you the correlation to achieve better network utilization by jointly compressing the sources using the network coding approach.

   Consider a network where each source, from a set of sources $S = \{X_1, X_2, ..., X_m\}$ wishes to multicast to all terminals $T = \{t_1, t_2, ..., t_k\}$ (see Fig. 6.4). Network coding with correlated sources has studied before (see for example [12]), but here we want to use it along with the gossip communication scheme. In order to use correlation we propose to use the 'random binning' coding scheme, so each source X_i performs random binning at rate $2^{nH(X_i}$. From this moment we can use the regular algebraic gossip to disseminate the bins indices of each source to all terminals. Such 'coded algebraic gossip' should be faster than the regular one. For example, if we have only two sources (X_1 and X_2), in regular algebraic gossip each terminal should receive $nH(X_1) + nH(X_2)$ equations on bits in order to decode the messages of X_1 and X_2. Using the Slepian-Wolf principle we conjecture that in the 'coded algebraic gossip' it is enough for a terminal to receive $nH(X_1|X_2)$ equations on bits of the bin index of X_1, $nH(X_2|X_1)$ equations on bits of the bin index of X_2, and if the total number of received equations is at least $nH(X_2, X_1)$ the terminal can decode the messages. Since $nH(X_2|X_1) + nH(X_1|X_2) \leq nH(X_1) + nH(X_2)$ and $nH(X_2, X_1) \leq nH(X_1) + nH(X_2)$

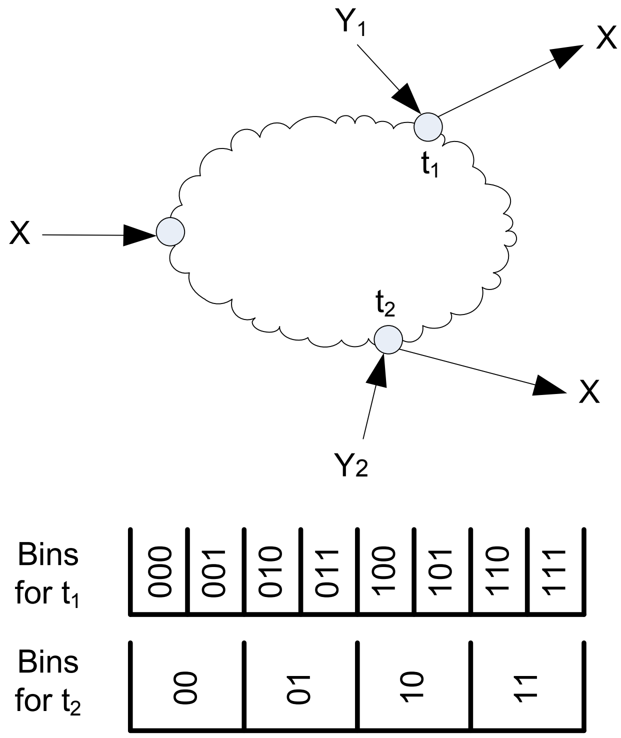

Figure 6.3: Terminals with side information and bins required for each terminal.

we can expect a better stopping time than in the regular algebraic gossip.

6.2.2 Adaptive/Non-Uniform Gossip

Consider the algebraic gossip analyzed in [4], which is a uniform gossip. In uniform gossip the communication partner is chosen uniformly among neighbors. On some graphs such a uniform selection causes 'bottlenecks' since some useful neighbors accessed very rarely. We conjecture that adding to the nodes the ability to choose neighbors can significantly decrease the stopping time. For example, in [3] authors show that a random walk that is allowed to inspect some set of neighbors before making a move, has a faster cover time. So, we will investigate the non-uniform and even adaptive gossip. Such an approach should help the

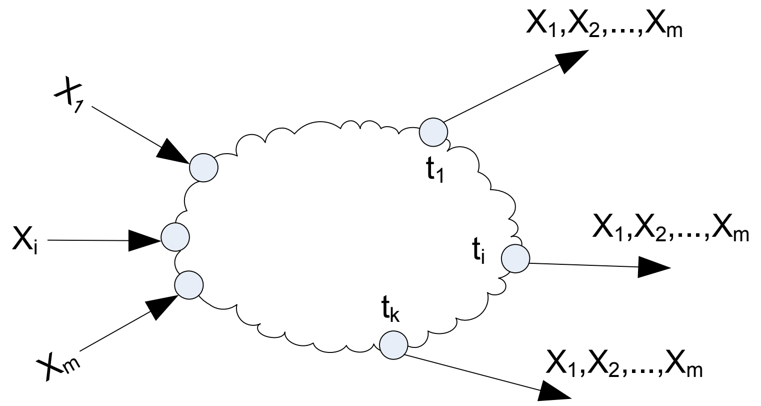

Figure 6.4: Algebraic gossip with correlated sources.

gossip algorithm to overcome 'bottlenecks' in the networks for which algebraic gossip is slow (e.g. barbell) by adapting the probability of choosing a specific neighbor according to (e.g.) the rate of *helpful messages* coming from a specific link.

6.2.3 SINR and Power Allocation

1. Receivers with Blocking Capabilities.

   Consider a set of n transmitter-receiver pairs located arbitrarily on a plane. Each transmitter wishes to transmit information to its dedicated receiver. Clearly, simultaneous transmission from all transmitters causes interference at each receiver, i.e., besides the signal of the desired transmitter, each receiver hears all other (undesired) transmitters. Using an approach presented in [26] we can find a maximum common C/I (carrier to interference ratio) value that can be achieved and also the power allocation for each transmitter, such that this maximum C/I is achieved. The technique proposed by Zander[26] is as following. Consider a square $n \times n$ matrix $Z_{ij} = \frac{G_{ij}}{G_{ii}}$, where G_{ii} is the gain of the transmitter i at the receiver i (the desired gain), and G_{ij} is the gain of the transmitter j at the receiver i (the undesired gain - interference). I.e., each element i, j in the matrix indicates how much the transmitter j disturbs to the receiver i to receive transmission of its dedicated transmitter i. Now, we have to find the largest eigenvalue λ^* of the matrix Z_{ij}, and the eigenvector $\boldsymbol{P}$ that corresponds to λ^*. Zander shows that the maximum achievable common C/I is $\gamma^* = \frac{1}{\lambda^* - 1}$, and the power vector achieving γ^* is $\boldsymbol{P}$.

Now, suppose that each receiver has a shield with which it can "defend" from a single transmitter. Clearly, this will improve the common C/I value, but the question is: 'What is the optimal strategy of using such shields (blocks) at each receiver?'. I.e., from which transmitter should each receiver to defend? We have a conjecture (based on simulation) that a simple "greedy" strategy (to defend from the most disturbing transmitter) is not optimal. So, finding an optimal strategy is an interesting question.

2. Power Allocation When a Group of Transmitters Transmit to the Same Receiver.

   Let us look at a setting where each receiver is wishing to receive simultaneous transmissions of a group of transmitters. Each transmitter transmits exactly the same information and we assume that they are perfectly synchronized. I.e., we have here some type of MIMO (multiple-input and multiple-output) at each receiver. Again, our task is to find the maximum SINR value and the power allocation that achieves it. This problem can be defined as a linear programming problem for a given SINR - β. But since β is unknown, the optimization problem is difficult. To overcome this difficulty we can try to perform a binary search on β, by solving at each step a simpler optimization problem. Formally, we can define:

$$\frac{\sum_{j \in T_i} E_{ij}}{\sum_{j \notin T_i} E_{ij} + N} \geq \beta \text{ , } \forall i \in [1, \ldots n].$$

   Where: T_i is the set of transmitters dedicated to the receiver i, $E_{ij} = \frac{P_j}{d(i,j)^\alpha}$ is the energy of transmitter j seen at the receiver i, N is an additive Gaussian noise, $d(i, j)$

is the distance between the transmitter j and the receiver i, and α is the path loss constant.

Now, given: α, $d(i,j)$, and the SINR β, we can define the following problem:

Find $\boldsymbol{P} = (P_1, P_2, \ldots, P_n)$, s.t.:

$$\forall i \in [1, \ldots n] : \frac{\sum_{j \in T_i} \frac{P_j}{d(i,j)^\alpha}}{\sum_{j \notin T_i} \frac{P_j}{d(i,j)^\alpha} + N} \geq \beta \ , \ P_i \geq 0.$$

We can start with a simpler problem by assuming $N = 0$ and the groups of transmitters are all of size 2. I.e., for each receiver there are exactly two dedicated transmitters. In this setting we can start with a minimal $\beta = \beta_0$ where β_0 is the C/I value obtained using a Zander approach (a function of the largest eigenvalue of the Zander's matrix). We can do so since the β_0 that satisfies: $\frac{E_i}{\sum_{j \neq i} E_{ij}} \geq \beta_0$ will also satisfy: $\frac{\sum_{j \in T_i} E_{ij}}{\sum_{j \notin T_i} E_{ij}} \geq \beta_0$. We have a preliminary conjecture that only one transmitter, closest to the receiver will be active, and all other transmitters will not transmit.

3. Optimal Powers Allocation for a given SINR threshold β.

   As we have already mentioned, Zander's approach gives an optimal (maximal) SINR that can be achieved and the corresponding powers. But what if we don't need such a high SINR value? Can we obtain in this case more 'convenient' power allocation (e.g. less total power, smaller differences in powers, etc.)

Bibliography

[1] C. Avin, Y. Emek, E. Kantor, Z. Lotker, D. Peleg, and L. Roditty. Sinr diagrams: towards algorithmically usable sinr models of wireless networks. In *PODC*, pages 200–209, 2009.

[2] C. Avin, M. Koucký, and Z. Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *ICALP (1)*, pages 121–132, 2008.

[3] C. Avin and B. Krishnamachari. The power of choice in random walks: an empirical study. In *In MSWiM 06: Proceedings of the 9th ACM international*, pages 219–228. ACM Press, 2006.

[4] M. Borokhovich, C. Avin, and Z. Lotker. Tight bounds for algebraic gossip on graphs, 2010.

[5] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *Information Theory, IEEE Transactions on*, 52(6):2508–2530, June 2006.

[6] S. P. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: design, analysis and applications. In *INFOCOM*, pages 1653–1664. IEEE, 2005.

[7] H. Chen and D. Yao. *Fundamentals of Queueing Networks: Performance, Asymptotics, and Optimization*, volume 46 of *Applications of Mathematics*. Springer-Verlag, New York, first edition, 2001.

[8] S. Deb, M. Médard, and C. Choute. Algebraic gossip: a network coding approach to optimal multiple rumor mongering. *IEEE Transactions on Information Theory*, 52(6):2486–2507, 2006.

[9] Y. Fernandess and D. Malkhi. On collaborative content distribution using multi-message gossip. *J. Parallel Distrib. Comput.*, 67(12):1232–1239, 2007.

[10] C. Fragouli, J.-Y. L. Boudec, and J. Widmer. Network coding: an instant primer. *Computer Communication Review*, 36(1):63–68, 2006.

[11] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *Information Theory, 2003. Proceedings. IEEE International Symposium on*, pages 442+, 2003.

[12] T. Ho, M. Me'dard, M. Effros, and R. Koetter. Network coding for correlated sources. In *in CISS*, 2004.

[13] J. J. Karaganis. On the cube of a graph. *Canadian Mathematical Bulletin 11*, pages 295–296, 1969.

[14] Karp, Schindelhauert, Shenkert, and Vocking. Randomized rumor spreading. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 2000.

[15] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 482–491, Oct. 2003.

[16] Li, Yeung, and Cai. Linear network coding. *IEEETIT: IEEE Transactions on Information Theory*, 49, 2003.

[17] L. Lovász. Random walks on graphs: A survey. In *Combinatorics, Paul Erdos is eighty, Vol. 2 (Keszthely, 1993)*, volume 2 of *Bolyai Soc. Math. Stud.*, pages 353–397. János Bolyai Math. Soc., Budapest, 1996.

[18] M. Médard and R. Koetter. Beyond routing: An algebraic approach to network coding. In *INFOCOM*, 2002.

[19] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, New York, NY, USA, 2005.

[20] D. Mosk-Aoyama and D. Shah. Information dissemination via network coding. In *Information Theory, 2006 IEEE International Symposium on*, pages 1748–1752, July 2006.

[21] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[22] C.-H. Ng and S. Boon-Hee. *Queueing Modelling Fundamentals: With Applications in Communication Networks.* Wiley Publishing, 2008.

[23] S. S. Skiena. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, with programs by Steven Skiena and Anil Bhansali.* Addison-Wesley, Reading, MA, USA, 1990.

[24] D. Slepian and J. Wolf. Noiseless coding of correlated information sources. *IEEE Transactions in Information Theory*, 25:471–480, 1973.

[25] D. Vasudevan and S. Kudekar. Algebraic gossip on arbitrary networks, 2009.

[26] J. Zander. Performance of optimum transmitter power control in cellular radio systems. *IEEE Transactions Vehicular Technology*, 41(1):57–62, Feb. 1992.