Algebraic Gossip via EXCHANGE: Analytical and Simulation Results

Chen Avin Michael Borokhovich Zvi Lotker
Department of Communication Systems Engineering
Ben-Gurion University of the Negev, Israel
{avin, borokom, zvilo}@cse.bgu.ac.il

***Abstract*—In this paper we study gossip algorithms for network coding. We consider three uniform gossip protocols for information spreading PULL, PUSH and EXCHANGE and three different gossip protocols (tasks), i) collect all information, ii) average computation and iii) network coding using random linear coding (aka algebraic gossip). While traditionally algebraic gossip used PULL or PUSH as its gossip algorithms, we prove that using the EXCHANGE algorithm can be unboundedly better. Algebraic gossip is known to be linear for the complete graph, we conjecture here that algebraic gossip using EXCHANGE is linear for any graph and support the claim by proofs for several graphs and by extensive simulation. Our proofs uses Jackson queueing networks to analyze the stopping time and this new method is likely turn useful for future research. In addition, we compare algebraic gossip to average computation via EXCHANGE and show that there are graphs for which algebraic gossip can be unboundedly faster.**

I. 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 resources sensor motes, researcher had been studied in recent years the use of randomize gossip algorithms together with network coding for this task [1], [2]. Randomize gossip-based protocols are attractive due to their locality, simplicity and structure free nature and been offered in the literature for various tasks [3], [4]. Network coding approaches for multicast (and other networking applications) receive a 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 [5], that uses gossip algorithms. In this paper we set to study the performance of algebraic gossip on arbitrary networks. Gossip algorithms determine the way in which a protocol disseminate its information to the network. To study algebraic gossip we make use of three different gossip algorithms: PULL, PUSH and EXCHANGE. In particular we offer to use algebraic gossip with EXCHANGE, and show that this can lead to significant improvements. Furthermore, beside network coding we consider two other protocols (i.e., tasks): i) *collect all information*, which is identical to the initial task but there are no restriction 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 on algebraic gossip.

These and similar question have been addressed in the past. Deb *et al.* [6] 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.* [4], [7] 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 capture the behavior of the protocol.

Mosk-Aoyama and Shah [8] used similar approach to [4], [7] to analyze network coding on symmetric networks and gave upper bound for the PULL algorithm that is based on measure of conductance of the network.

A recent, yet unpublished[2], independent work by Vasudevan and Kudekar [9] 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 this bound are tight but they significantly improve upon Mosk-Aoyama and Shah bounds.

A. Overview of our results

First, independently to [9] we promote the use of EXCHANGE with algebraic gossip. We show that there are graphs for which EXCHANGE can preform unboundedly better than PULL or PUSH. Moreover, we conjecture that algebraic gossip using EXCHANGE is, in fact, linear in expectation on *any graph*. We support our claim by providing new analytical bounds for the Star and the Ring graphs, where both bounds are better than any previous general bounds (i.e., [8], [9]). The technique used in the proof for the Ring involved Jackson network of queues and is interesting by its own right. We believe this techniques could be extended in the future to prove our conjecture. In addition we present simulation study that supports our claim for other networks such as the grid, hypercube and the lollipop. Using our bounds we show that the spectral gap method that 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.

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

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

The paper is organized as follows: Section II presents preliminaries and models, Sections III and IV define the gossip algorithms and protocols respectively. Section V formally define the problem. Section VI then proves our analytical results (we omit some of the proof due to space limitations) and Section VII gives simulation results.

II. Preliminaries and Models

A. 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 paper, 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 start and other nodes are connected only to v_1.
- Ring graph - R_n - a connected cycle where each node has a left and right neighbor.

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:

1) Synchronous time model: The time is assumed to be slotted and divided in to rounds. Each round contains n time slots. 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. The model guarantees that during n consecutive timeslots each node will be selected exactly once.

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. In this model there is no guarantee that a node will be selected exactly once in around, nodes can be selected few times or none.

III. Gossip Algorithms

Gossip algorithms define the way information is exchanged or spread in the network. At each time step a single node takes an information spreading action which 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 on time 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:

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

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 $\mathcal{A}$.

IV. 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 [7] - 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[6]. Message size in this protocol is limited.

For all three protocol 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 it 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 had 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 $F^{\mathcal{P}}_{in}$ 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 $D_d(t+1) = F^{\mathcal{P}}_{in}(m_{s,d}(t), D_d(t))$.

- **Stopping condition.** A task is completed when every node completed the task, therefore we have *local* and *global* stopping conditions, for a protocol $\mathcal{P}$:
  - *Local stopping condition*: $\mathbb{S}^{\mathcal{P}}_v(t)$
    is true iff node $v \in V$ has completed the protocol (task) by timeslot t.
  - *Global stopping condition*: $\mathbb{S}^{\mathcal{P}}_V(t)$.
    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 different tasks in more details, i.e., messages, databases and stopping conditions:

A. 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., $F^{CI}_{out}(D_s(t))$:= $D_s(t)$. Database update is just extracting the initial values from the received message and append the new values to the DB. $F^{CI}_{in}(m_{s,d}(t), D_d(t)) := \{D_d(t) \cup x_j \mid x_j \notin D_d(t), x_j \in m_{s,d}(t)\}$
A node completes the CI task once it knows all the initial messages $\overline{x}$ in the graph, the *local stopping condition* is:

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

B. 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., $F^{AVG}_{out}(D_s(t))$:= $D_s(t)$. 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., $F^{AVG}_{in}(m_{s,d}(t), D_d(t))$:= $\frac{m_{s,d}(t)+D_d(t)}{2}$.
A node completes the AVG task once it has a value that is $\pm\epsilon$ ($\epsilon \in \mathbb{R}^+$) far away from the true average value of all nodes, the *local stopping condition* is:

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

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

C. NC (Network Coding)

In this task we will use a random linear coding technique as described in [6]. 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, $len(m_{s,d}(t)) = l + len(n \times \log(q))$ bits, where q is the sizeof the field $\mathbb{F}_q$ from which coefficients are drawn. Every initial value x_i can be viewd as a number from a finite field $\mathbb{F}_q$, if $\forall v_i \in V, x_i < q$. If $\exists v_i \in V, x_i \geq q$ then, the initial values x_i should be viewed as vectors over field $\mathbb{F}_q$, i.e., $x_i \in \mathbb{F}^r_q$ and $r = \lceil \log_q(x_i) \rceil$.

In the NC task, 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 $x_i \in \mathbb{F}^r_q$ Once a node has n independent equations (messages) it is able to decode all the initial values $x_i \in \mathbb{F}^r_q$ and thus completes the protocol.

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

Message that a node will send is: $F^{NC}_{out}(D_s(t)) := RLC(D_s(t)) \cup \{a_{x_i}\}^n_{i=1}$, 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 is appending to the database a message (which is a linear equation) only if it is independent with all messages (linear equations) that stored already in the node's DB.

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

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 constitutes 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 equation 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:

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

V. 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 1 (random 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 follow:

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

and for any node v we define:

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

Definition 2 (expected and high probability stopping time): The *expected stopping time* $\overline{T}_v^{\mathcal{A},\mathcal{P}}, \overline{T}^{\mathcal{A},\mathcal{P}}$ and *high probably stopping time* $\hat{T}^{\mathcal{A},\mathcal{P}}$ are define respectively:

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

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

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

The stopping time can also be measured by *rounds* where one round equals n consecutive timeslots. In particular, we define $\overline{R}_v^{\mathcal{A},\mathcal{P}} = \overline{T}_v^{\mathcal{A},\mathcal{P}}/n$, $\overline{R}^{\mathcal{A},\mathcal{P}} = \overline{T}^{\mathcal{A},\mathcal{P}}/n$ and $\hat{R}^{\mathcal{A},\mathcal{P}} = \hat{T}^{\mathcal{A},\mathcal{P}}/n$ as the expected number of rounds and number of rounds with high probability, respectively.

We can now express our research question formally: 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.

VI. Theoretical Results

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

Theorem 1: For the Star graph S_n Algebraic gossip (i.e., NC) using EXCHANGE is unboundedly better than using PULL or PUSH algorithms, formally: for $\mathcal{A} \in \{\texttt{PULL}, \texttt{PUSH}\}$

$$\lim_{n\to\infty} \frac{\hat{R}^{\mathcal{A},NC}}{\hat{R}^{EX,NC}} \to \infty \tag{1}$$

and for any v

$$\lim_{n\to\infty} \frac{\overline{R}_v^{\mathcal{A},NC}}{\overline{R}_v^{EX,NC}} \to \infty \tag{2}$$

We give here a short proof overview, the full details are bellow. We first show in Lemma 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 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 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.

The second theorem proves that on the Ring R_n and using the EXCHANGE algorithm, the expected time for the NC protocol in $O(n)$ and with high probability $O(n\log n)$. The proof of the expected value uses queueing networks ideas, which seems to contribute some insight to the problem.

Theorem 2: For the synchronous time model and the ring graph, R_n:

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

and

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

The third 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 seems simpler than NC and the AVG protocol was known to be much faster than NC on the complete graph.

Theorem 3:

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

$$\overline{R}^{EX,NC} \le \overline{R}^{EX,AVG} \quad \text{and} \quad \hat{R}^{EX,NC} \le \hat{R}^{EX,AVG}$$

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

$$\overline{R}^{EX,AVG} \le \overline{R}^{EX,NC} \quad \text{and} \quad \hat{R}^{EX,AVG} \le \hat{R}^{EX,NC}$$

A. Proof Of Theorem 1

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

Definition 3 (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:

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

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

Definition 4 (t-subset of algorithm runs): A subset of algorithm runs that the stopping time for them is t:

$$\Pi_t^{\mathcal{A},\mathcal{P}}(G) = \left\{\sigma \mid \sigma \in \Pi(G), T_\sigma^{\mathcal{A},\mathcal{P}} = t\right\}$$

Definition 5 (t^+-subset of algorithm runs): A subset of algorithm runs that the stopping time for them is at least t:

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

Definition 6 (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 lineary independent with all equations (messages) stored in the u's DB.

Definition 7 (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 1: For any graph G and for gossip algorithm $\mathcal{A} \in \{\texttt{PUSH}, \texttt{PULL}, \texttt{EXCHANGE}\}$:

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

To prove lemma 1, first, we will show that for any specific algorithm $\texttt{run}\ \sigma \in \Pi(G)$ the stopping time of the CI protocol is less than or equals to the stopping time of the NC protocol.

Lemma 2: For any graph $G(V,E)$, gossip algorithm $\mathcal{A} \in \{\texttt{PUSH}, \texttt{PULL}, \texttt{EXCHANGE}\}$ and any algorithm $\texttt{run}\ \sigma \in \Pi(G)$:

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

To prove lemma 2 we need the following two claims.

Claim 1: For CI protocol: If and only if nodes u and v are *connected*[3] by the timeslot t, $D_u(0) \subset D_v(t)$.

Claim 2: If nodes u and v are *not connected* by the timeslot t then $S_V^{NC}(t) = FALSE$.

Now we can prove Lemma 2:

Proof of Lemma 2: The proof will be by contradiction, let's 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 1, we can conclude that nodes v_a and v_b are *not connected*. Hence, from Claim 2, $S_V^{NC}(t) = FALSE$ which contrdicts 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 1:

Proof of Lemma 1: From Lemma 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:

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

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 1 (we therefore omit the proof).

Lemma 3: For any graph $G(V,E)$, and gossip algorithm $\mathcal{A} \in \{\texttt{PUSH}, \texttt{PULL}, \texttt{EXCHANGE}\}$:

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

and

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

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

Lemma 4: For the synchronous time model, the Star graph S_n and $\mathcal{A} \in \{\texttt{PUSH}, \texttt{PULL}\}$:

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* [10].

Claim 3: Let X be the r.w for the number of coupon needed to obtain n distinct coupon, then:

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

Proof of Lemma 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) 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 protocol 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 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 5 we will use the following two claims.

Claim 4: Let X_i be independent geometric random variables with parameter p, and let $X = \sum_{i=1}^n X_i$. For $p \geq \frac{1}{2}$:

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

The following lemma is a part of Lemma 2.1 in [6].

Lemma 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 5.

Proof of Lemma 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 6, after $\log_q n$ rounds (or $n \times log_q n$ timeslots), the first phase will be ended with high probability.

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

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 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's define X_i^u as the number of rounds needed to 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 (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 round to reach rank n is linear since the center node is always *helpful*. ∎

Proof of Theorem 1: The proof of Equation (1) is an immidiate result of Lemmas 3, 4 and 5. Lemma 3 states that the stopping time of the CI protocol is a lower bound for the stopping time of the NC protocol. Lemma 4 shows that the stopping time of the CI protocol over PUSH and PULL gossip algorithms in the Star graph S_n is of order of $n \log n$ with high probability. Combining Lemmas 3, 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 5 we proved that the stopping time of the NC protocol over the EXCHANGE gossip algorithm in the Star graph S_n is of 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 (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 5, $\overline{R}_v^{EX,NC} = O(n)$. ∎

B. Proof Of Theorems 2 and 3

First, we will prove the Therom 2 and then will use it to prove the Theorem 3.

Proof of Theorem 2: 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 to. 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 that v_{i+1} knows 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.

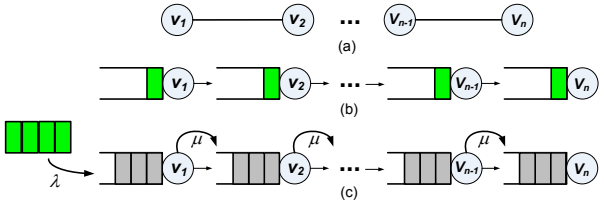

Fig. 1. Modeling NC in a Path as a queueing network.

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 incereased thier ranks or both of them increased their ranks by 1.

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. Fig. 1 (b) illustrates the initial condition of our queuing network. In the first part of the proof we find the average time needed to 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 to 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 can not be increased during the timeslot t, since the node v_{i-1} does not know more information than v_i, because all new customers (equations) are comming 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 6, probability that a customer 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 $p = \frac{3}{4}\left(1 - \frac{1}{q}\right)$. The following lemma shows that we can assume that the service time is exponentially distributed.

Lemma 7: Let X be a geometric random variable with a

success probability p. Then, for all $x \in \mathbb{R}^+$

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

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

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

We present here the Jackson's theorem from queueing theory. A proof of this theorem can by found in [11].

Theorem 4: 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 $\rho_i = \frac{\lambda_i}{\mu_i}$.

The 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. 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 will 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 queues.

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 exist from Theorem 4. By adding additional *dummy* customers (since their arrivals are not counted as a rank increment) to the system, we make the *real* customers to wait more in the queues, thus, increasing the stopping time. Fig. 1 (c) illustrates our queuing network with the above modifications. Green, are the *real* customers, and grey are the *dummy* customers.

Now, we are interested to find the expected time that will take to the n's *real* customer (n's 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's denote this time as $\overline{T} = \overline{T}_1 + \overline{T}_2$, where $\overline{T}_1$ is the time needed to the n's customer to arrive at the first queue, and $\overline{T}_2$ is the time needed to the n's 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 to the expected time crossing 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 n on the Path.

Lemma 8: The expected time it takes to the n customers to arrive to the system is:

$$\frac{n}{\lambda}$$

And thus, $\overline{T}_1 = \frac{n}{\lambda}$.

Lemma 9: The expected time it takes to cross one queue in the equilibrium state for $\rho = \frac{1}{2}$ is

$$\frac{1}{\mu - \lambda}$$

And thus, $\overline{T}_1 = \frac{n}{\mu - \lambda}$.

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, lets 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's now analyse 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 the 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 fills all the queue in the system with *dummy* customers in order to achieve queues lengths that correspond to the equilibrium state distribution.

Clearly, that an 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 will arrive 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 this time will be the same as $\overline{T}$ that we 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 result follows:

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

Now, we will prove the high probability bound.

Since the $R^{EX,NC}$ random variable is positive, we can use Markov inequality to conclude that: $\Pr\left(R^{EX,NC} \geq 2\overline{R}^{EX,NC}\right) \leq \frac{1}{2}$. Now, lets find the probability of not completing the NC task after $2\overline{R}^{EX,NC} \times \log_2 n$ rounds. We can represent this probability as a probability of not completing the task in any of the $\log_2 n$ experiments, each one of duration $2\overline{R}^{EX,NC}$ rounds. So, $\Pr\left(R^{EX,NC} \geq 2\overline{R}^{EX,NC} \times \log_2 n\right) \leq \frac{1}{2}^{\log_2 n} = \frac{1}{n}$.

Hence, $R^{EX,NC} < 2\overline{R}^{EX,NC} \times \log_2 n$ with probability of at least $1 - \frac{1}{n}$. ■

Proof of Theorem 3: The first claim of the theorem then follows using the results of Boys *et al.* [7]. 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 $\Omega(n^2)$ [12] and thus, number of rounds needed to complete the AVG task is $\hat{R}^{EX,AVG} = \Omega(n^2)$ with high probability. From this result 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 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 [3] the authors shows that the number of rounds for the AVG is $\hat{R}^{EX,AVG} = O(\log n)$ w.h.p while in [13] they give a lower bound of $\hat{R}^{EX,AVG} = \Omega(\log n)$. For NC, in [6] 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. ■

VII. Simulation Results

A. Simulation Setup

For evaluating and strengthen our conjectures regarding stopping times of, presented above, algorithms and protocols, we developed a simulation tool using C. In this section we present the most interesting and even surprising results.

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 paper (synchronous and asynchronous) were tested. We simulated all presented above gossip algoritmhs $\mathcal{A} \in \{\texttt{PUSH}, \texttt{PULL}, \texttt{EXCHANGE}\}$, and over these algorithm we tested the gossip protocols $\mathcal{P} \in \{CI, AVG, NC\}$. 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* to it or not. I.e, it should determine whether the received message is liniarly independent with 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 $q = 128$.

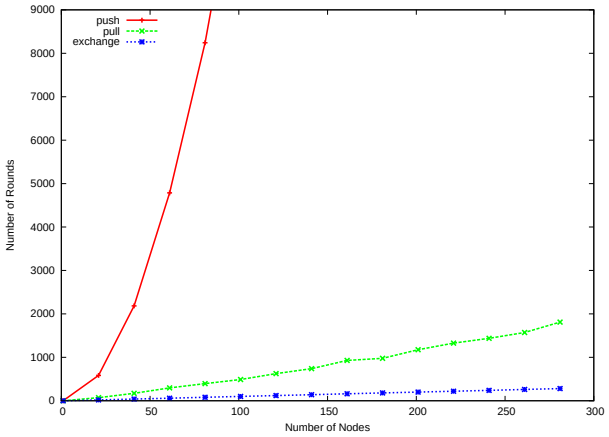

Fig. 2. Stopping time (in rounds). Synchronous time model, NC protocol, Star graph, PUSH, PULL, EXCHANGE.

B. Results

EXCHANGE is faster than PUSH/PULL . It is clear that using EXCHANGE algorithm instead of PUSH or PULL can only speedup 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 Fig. 2 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 yelds much smaller stopping times, and its benefit grows with n. In a Star graph, PUSH or PULL takes in 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 much less (even unboundedly less) messages than using PUSH or PULL, since the difference in number of messages per each round is only by a factor of 2 (EXCHANGE uses two messages per communication action, and PUSH or PULL use only one message). In 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 it in Fig. 3.

NC over EXCHANGE is linear. We have a conjecture that the NC protocol over the EXCHANGE gossip algorithm is linear for all graphs. In Fig. 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

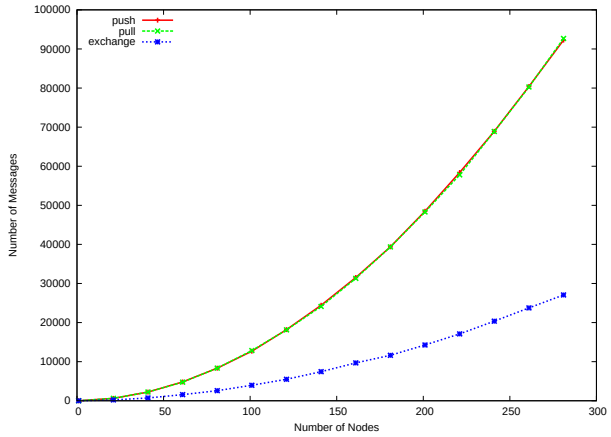

Fig. 3. # of messages sent in the network until NC is finished. Asynchronous time model, NC protocol, Complete graph, PUSH, PULL, EXCHANGE.

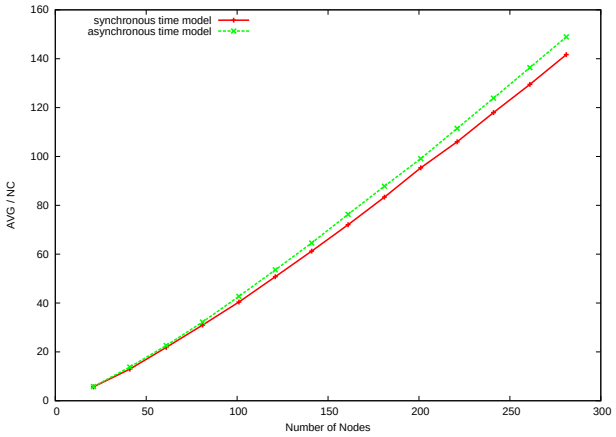

Fig. 5. Synchronous and asynchronous time models, EXCHANGE algorithm, $\frac{AVG}{NC}$ in Ring graph. NC is unboundedly faster in Ring graph.

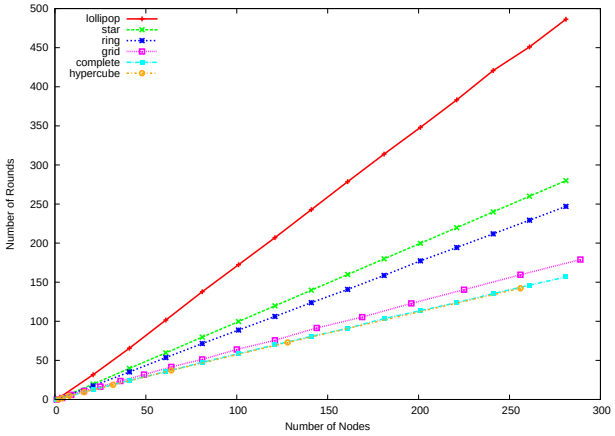

Fig. 4. Synchronous time model, NC protocol, EXCHANGE algorithm, various topologies. Stopping time is linear.

all stopping times are slightly smaller than in the asynchronous case.

NC over EXCHANGE can be faster then 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 Fig. 5 we can see the ratio between the AVG and NC stopping times and this ratio grows linearly with n for the synchronous time model and also for the asynchronous.

REFERENCES

[1] M. Médard and R. Koetter, "Beyond routing: An algebraic approach to network coding," in *INFOCOM*, 2002. [Online]. Available: http://www.ieee-infocom.org/2002/papers/751.pdf

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

[3] 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*, Oct. 2003, pp. 482–491.

[4] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2508–2530, June 2006.

[5] 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*, 2003, pp. 442+. [Online]. Available: http://dx.doi.org/10.1109/ISIT.2003.1228459

[6] S. Deb, M. Médard, and C. Choute, "Algebraic gossip: a network coding approach to optimal multiple rumor mongering," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2486–2507, 2006.

[7] S. P. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: design, analysis and applications," in *INFOCOM*. IEEE, 2005, pp. 1653–1664. [Online]. Available: http://dx.doi.org/10.1109/INFCOM.2005.1498447

[8] D. Mosk-Aoyama and D. Shah, "Information dissemination via network coding," in *Information Theory, 2006 IEEE International Symposium on*, July 2006, pp. 1748–1752.

[9] D. Vasudevan and S. Kudekar, "Algebraic gossip on arbitrary networks," 2009. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:0901.1444

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

[11] H. Chen and D. Yao, *Fundamentals of Queueing Networks: Performance, Asymptotics, and Optimization*, 1st ed., ser. Applications of Mathematics. New York: Springer-Verlag, 2001, vol. 46.

[12] L. Lovász, "Random walks on graphs: A survey," in *Combinatorics, Paul Erdos is eighty, Vol. 2 (Keszthely, 1993)*, ser. Bolyai Soc. Math. Stud. Budapest: János Bolyai Math. Soc., 1996, vol. 2, pp. 353–397.

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