

Algebraic Gossip: A Network Coding Approach to Optimal Multiple Rumor Mongering

Supratim Deb and Muriel Médard

Laboratory for Information & Decision Systems
Massachusetts Institute of Technology
Cambridge, MA-02139
{supratim,medard}@mit.edu

Abstract

We study the problem of simultaneously disseminating multiple messages in a large network in a decentralized and distributed manner. We consider a network with n nodes and k ($k = O(n)$) messages spread throughout the network to start with, but not all nodes have all the messages. Our communication model is such that the nodes communicate in discrete-time steps, and in every time-step, each node communicates with a *random* communication partner chosen uniformly from all the nodes (known as the *random phone call* model). The system is bandwidth limited and in each time-step, only one message can be transmitted. The goal is to disseminate rapidly all the messages among all the nodes. We study the time required for this dissemination to occur *with high probability*, and also in expectation.

We present a protocol based on *random linear coding* (RLC) that disseminates all the messages among all the nodes in $O(n)$ time, which is order optimal, if we ignore the small overhead associated with each transmission. The overhead does not depend on the size of the messages and is less than 1% for $k = 100$ and messages of size 100 KB. We also consider a *store and forward* mechanism without coding, which is a natural extension of *gossip-based* dissemination with one message in the network. We show that, such an uncoded scheme can do no better than a sequential approach (instead of doing it simultaneously) of disseminating the messages which takes $\Theta(n \ln(n))$ time, since disseminating a single message in a *gossip* network takes $\Theta(\ln(n))$ time.

1 Introduction

Of late, design of protocols to rapidly disseminate messages in large networks have gained a lot of attention. Much of the research on information dissemination started with updating messages in large distributed computer systems where computers can communicate with each other. More recently, the emergence of sensor networks has added a new paradigm to the problems of distributed message dissemination.

The use of *gossip based* protocols to disseminate message was first proposed in [4]. In *gossip* based protocols, nodes communicate with each other in communication steps called *rounds*, and the amount of information exchanged in each round between two communicating nodes is limited. Further, there is no centralized controller and every node in the network acts simply based on *state or information* of the node, and not that of the over all network. Thus, *gossip* based protocols are inherently distributed and easily implementable, and provides

powerful alternatives to *flooding* and *broadcast* based protocols for dissemination of messages. There is a wide literature on the practical aspects of gossip-based message dissemination [14]. A detailed analysis of a few gossip-based dissemination schemes were provided in [9]. A common performance measure in all the previous work is the time required to disseminate a single message to all the nodes in the network. In recent work, [11, 10] considered the scenario where there are multiple messages, each with a unique destination. The authors considered what they call *spatial gossip* where the nodes are embedded in a metric-space with some distance metric between the nodes. More recently, in [2], the authors have studied gossip-like distributed algorithms for computing averages at the nodes. In this paper, we envisage a different problem, in which the network seeks to simultaneously disseminate multiple messages to all the nodes in a distributed and decentralized manner.

As pointed out in [11], we note that, any *gossip* protocol has two layers of design aspects. One is the design of *gossip algorithm* by which, in every round, every node decides its communication partner, either in a deterministic, or in a randomized manner. The other important aspect is the design of *gossip-based protocol* by which any node, upon deciding the communication partner according to the *gossip* algorithm, decides the content of the message to send to the communication partner. The main contribution of our paper lies in proposing a *gossip based protocol* using the idea of *random linear coding*, which has previously been used in the context of communication theory for various purposes.

In this paper, we consider a scenario where there are multiple nodes in the network and also multiple messages, but not all messages are with all the nodes to start with. We are interested in designing mechanisms to ensure that all the nodes receive all the messages very fast. We restrict ourselves to *gossip protocols*, so that each node acts based on *local* information, without a centralized controller. Moreover, at each communication instant between nodes, only one *message* (we comment on this aspect later in the paper) can be transmitted. The *gossip algorithm* we consider in this paper is what is popularly known as the *random phone call* model or *rumor mongering* model [4]. In such a model, the system proceeds in rounds. In each round, every node u calls a communicating partner v chosen uniformly at random from from all the nodes. We propose *gossip-based protocol* using the idea of *random network coding* introduced by Ho et. al. in [6, 7], and compare the protocol with a naive one.

As we show in the paper, information dissemination schemes based on the concepts of *network coding*, instead of a naive *store and forward* mechanisms, can provide substantial benefits in distributed environments at the cost of a small overhead (if the message sizes are reasonably large) associated with each packet. In networks with fixed communicating graphs, *network coding* can be viewed as a vast generalization of routing where each packet is treated as an algebraic entity that can be operated upon instead of simply storing and forwarding. Essentially, each node in the network sends to each output link any linear function of the data received at the input links. It was shown in [13] that, linear network coding can achieve the min-cut bound in networks with multicast flows. There is a significant recent work on network coding [8], especially on the algorithmic aspects of construction of linear network codes [1, 12]. In [6, 7], the authors proposed the novel idea of *random network coding*. Our protocol for message dissemination is inspired by this.

We present a *gossip based protocol* based on *random network coding* that can simultaneously disseminate $k = O(n)$ messages (where each message is initially distributed among $O(1)$ nodes) among n nodes in $O(n)$ time. This is clearly the optimal time to disseminate the messages, in an order sense. The key feature in our protocol which helps to attain this bound is an “algebraic mixing” of the messages using *random linear coding*. This is done by viewing each message as an element in a vector space with the scalars in a finite field of appropriate

size. We have also shown that a naive uncoded *store and forward* approach of disseminating the messages takes $\Omega(n \ln(n))$ time. It is worth noting that, a sequential approach of disseminating the messages one after the other would take a sum-total of $\Theta(n \ln(n))$ time, since the time to propagate a single message is $\Theta(\ln(n))$ [9].

We simply provide the key intuition behind the power of a coding based approach in the following. Suppose there are k distinct elements in a finite field of size q . Consider two approaches to store the elements in a database with k slots. Suppose each slot chooses an element at random without the knowledge of what the other slots are choosing. Then, the probability that all the elements are there in the database is very small. Now, consider a second approach in which each slot in the database stores a random linear combination of the messages. All the messages can be recovered from the database, only if the linear combination chosen by the slots are linearly independent. Now, there are $((q^k - 1)(q^k - q)(q^k - q^2) \dots (q^k - q^{k-1}))$ ways of choosing k linearly independent combination of the k elements in a finite field of size q . Thus, the probability that the elements can be recovered from the database is much higher in the latter scenario. This is the key idea which is at the heart of the *random linear coding* based protocol we present in this paper.

In Section 2, the model and the protocols considered are described along with a few preliminaries. We describe our main results in Section 3 and also provide heuristic arguments behind the results. We skip the detailed proof the results. Please refer to [3] for the detailed proofs. We conclude in Section 4.

2 Model, Protocols, and Preliminaries

2.1 Model and Protocols

Suppose there are n nodes and k ($k(n)$ to be more precise) messages. Initially, each node has one of the k messages indexed by the elements in the set $M = \{m_1, m_2, \dots, m_k\}$. The nodes are indexed by elements of the set $[n]$. Let V_m be the set of nodes that start out with the message $m \in M$. We also assume $|V_m| = \alpha, \forall m \in M$, i.e., each message is equally spread in the network to start with. We are interested in obtaining the time required to disseminate all the messages to all the nodes using a rumor mongering approach in the asymptote of large n such that n/k is kept fixed at $\alpha \geq 1$. We comment that the results and the derivations in this paper can be extended for the case when some nodes have more than one message and some have none, or when all the messages are there with one particular node to start with. The protocols described later may require minor modifications depending on the initial distribution of the messages.

Gossip Algorithms:

The system advances in *rounds* indexed by $t \in \mathbb{Z}^+$. The communication graph in round t , G_t , is obtained in a randomized manner as follows. In the beginning of each round, each node $u \in [n]$ calls a communication partner v chosen uniformly from $[n]$. We consider two variants of this *rumor mongering* model for message exchange as proposed in [4].

Pull: In this model, a message is transmitted from a *called* node to the *caller* node according to a suitable protocol we describe later. Thus, the communication process is initiated by the receiving node.

Push: Here, the message is transmitted from a *caller* node to the *called* node. Thus, the communication process is initiated by the transmitting node.

There can be other variants, for instance, by combining *push* and *pull* as considered in [9].

Gossip-Based Protocols:

Having described the model for communication graph in each round, we now describe two protocols or strategies for transmitting a message. The protocols will be adopted by the *caller* node in the *push* model and the *called* node in the *pull* model. Below we describe two protocols for message transmission we consider in this paper.

Random Message Selection (RMS): This is a naive strategy, where the transmitting node simply looks at the messages it has received and picks any of the messages with equal probability to transmit to the receiving node. Thus, if M_v are the set of messages with node v , then v transmits a “random” message e to its communicating partner, where

$$\Pr(e = m) = \frac{I_{m \in M_v}}{|M_v|}.$$

Random Linear Coding (RLC): Suppose the messages are vectors over the finite field, \mathbb{F}_q of size $q \geq k$. If the message size is m bits, this can be done by viewing each message as a $r = \lceil m / \log_2(q) \rceil$ dimensional vector over \mathbb{F}_q (instead of viewing each message as a m dimensional vector over the binary field). To this end, let $m_i \in \mathbb{F}_q^r$ ($m_i, i = 1, 2, \dots, k$, are the messages) for some integer r . Thus the messages are over a vector space with the scalars in \mathbb{F}_q . All the additions and the multiplications in the following description are assumed to be over \mathbb{F}_q . In the RLC protocol, the nodes start collecting several linear combinations of the messages in M . Once there are k independent linear combinations with a node, it can recover all the messages successfully. Let S_v denote the set of all the coded messages (each coded message is a linear combination of the messages in M) with node v at the beginning of a round. More precisely, if $f_l \in S_v$, where $l = 1, 2, \dots, |S_v|$, then $f_l \in \mathbb{F}_q^r$ has the form

$$f_l = \sum_{i=1}^k a_{li} m_i, \quad a_{li} \in \mathbb{F}_q,$$

and further the protocol ensures that a_{li} 's are known to v . This can be done with a minimal overhead with each packet in a manner described soon.

Now, if the node v has to transmit a message to u , then v transmits a “random” coded message with content $e \in \mathbb{F}_q^r$ to u , where

$$e = \sum_{f_l \in S_v} \beta_l f_l, \quad \beta_l \in \mathbb{F}_q \quad (1)$$

and

$$\Pr(\beta_l = \beta) = \frac{1}{q}, \quad \forall \beta \in \mathbb{F}_q. \quad (2)$$

For decoding purposes, the transmitting nodes also send the “random coding vectors” as overhead with each packet. This can be achieved by padding an additional $k \log_2 q$ bits with each message. To see the precise structure of the overhead associated with a

packet, note that the payload part of the transmitted message e in (1) can be represented as follows:

$$\begin{aligned}
e &= \sum_{f_l \in S_v} \beta_l f_l \\
&= \sum_{f_l \in S_v} \beta_l \sum_{i=1}^k a_{li} m_i \quad (\text{where } a_{li} \in \mathbb{F}_q) \\
&= \sum_{i=1}^k \theta_i m_i \quad (\text{where, } \theta_i = \sum_{f_l \in S_v} \beta_l a_{li} \in \mathbb{F}_q)
\end{aligned}$$

It is the θ_i 's that are sent as overhead with the transmitted messages. Thus, once the β_l 's are decided in randomized manner according to (2), the transmitting nodes can precisely obtain the values of θ_i 's ($i = 1, 2 \dots k$) and send as overhead. This overhead would clearly require additional $k \log_2(q)$ bits. We also call the overhead $(\theta_1, \theta_2, \dots, \theta_k) \in \mathbb{F}_q^k$, the transmitted ‘‘code-vector.’’ We simply comment that, if the message size $m \gg k \log_2(q)$, then the overhead required with the protocol is minimal. Note that the overload scales with the number of messages being disseminated simultaneously. However, in the set-up we consider in this paper, $k = O(n)$ and $q \approx k$, and thus the size of the overhead is around $n \log_2(n)$.

The decoding of the messages is not hard to see. In RLC approach, the nodes start collecting the ‘‘code vectors’’ as the system progresses. Once the dimension of the subspace spanned by the received ‘‘code vectors’’ with a node becomes k , then the node can recover all the messages.

We are interested in the finding the expected time (rounds) required for all the nodes to receive (decode) all the messages, and also the time required to receive all the messages with high probability for four cases: RLC with *pull*, RLC with *push*, RMS with *pull*, RMS with *push*.

We comment that the underlying probability space has a probability measure determined by the random communication graphs in each round and the random transmitted messages. By a natural abuse of terminology, in our analysis and discussion of the RLC protocol, we also refer to ‘‘dimension of the subspace spanned by the code-vectors received by the node’’ as ‘‘dimension of a node.’’ Throughout this paper, we also use the terms ‘‘round’’ and ‘‘time’’ interchangeably.

2.1.1 A useful result on the RLC protocol

We now state (see [3] for proof) a useful, but simple and intuitive result which is key to demonstrating the benefits of the RLC protocol. In the following, we assume that a coded message is transmitted from node v to node u . It is implicit that, with a ‘‘pull’’ mechanism u is the caller node, and with a ‘‘push’’ mechanism v is the caller node.

Lemma 2.1. *Suppose node v transmits a coded message to node u in a particular round using the RLC protocol. Let S_u^- and S_v^- denote the subspaces spanned by the code-vectors with u and v respectively at the beginning of the round. Let S_u^+ denote the subspace spanned by u at the end of the round, i.e., after receiving a coded message from v according to the scheme described by the RLC protocol. Then,*

$$\Pr(\dim(S_u^+) > \dim(S_u^-) | S_v^- \not\subseteq S_u^-) \geq 1 - \frac{1}{q},$$

where q is the size of the field.

3 Main Results and Discussion

We now describe the main results of the paper. Our first result is on the performance of RLC with *pull* mechanism.

Theorem 3.1. *Let \bar{T}_{RLC}^{pull} be the random variable denoting the time required by all the nodes to get all the messages using an RLC approach with pull mechanism. Then,*

$$\bar{T}_{RLC}^{pull} = O(n), \text{ w.p. } 1 - O\left(\frac{1}{n}\right)$$

Further, if T_{RLC}^{pull} is the time required for a particular node to get all the messages, then

$$\mathbb{E}[T_{RLC}^{pull}] = O(n).$$

We also have a similar result with a *push* based mechanism.

Theorem 3.2. *Let \bar{T}_{RLC}^{push} be the random variable denoting the time required for all the nodes to get all the messages using an RLC protocol with push mechanism. Then,*

$$\bar{T}_{RLC}^{push} = O(n), \text{ w.p. } 1 - O\left(\frac{1}{n}\right)$$

If T_{RLC}^{push} is the time required for a particular node to get all the messages with RLC based push, then

$$\mathbb{E}[T_{RLC}^{push}] = O(n),$$

Remark: The following extensions of the results are routine.

1. If there is only one copy of each of the k messages with k different nodes initially, so that there are some nodes with no messages, then Theorem 3.1 and 3.2 easily go through.
2. Suppose initially there are k different messages at a single node. Then, the result in Theorem 3.1 and 3.2 again go through with minor additional considerations in the analysis provided later.
3. A more careful look at our derivation later in the paper suggests that, the size of the finite field could be restricted to roughly \sqrt{k} and the results in Theorem 3.1 and Theorem 3.2 would go through. This suggests that the overhead with each packet can be reduced to $0.5k \log_2 k$ instead.

The power of a coding based approach comes from the fact that packets are treated as algebraic entities which can be operated upon. The next natural question is, what if the nodes do not manipulate the packets and simply *store* and *forward* the packets? We show that, in one such protocol as we have described in the paper, which we call RMS or ‘‘Random Message Selection,’’ one can do no better than the case when the messages are disseminated in the network sequentially one after the other.

Theorem 3.3. *Let $T_k^{RMSpull}$ be the time required for all the nodes to get all the k messages using an RMS protocol with pull mechanism. Then, we have*

$$\mathbb{E}T_k^{RMSpull} = \Omega(n \ln n)$$

and

$$\lim_{k \rightarrow \infty} \Pr \left(T_k^{RMSpull} = \Omega(k \ln(k)) \right) = 1$$

We also have a very similar result for RMS with a *push* based mechanism.

Theorem 3.4. *Let $T_k^{RMSpush}$ be the time required for all the nodes to get all the k messages using an RMS protocol with pull mechanism. Then, we have*

$$\mathbb{E}T_k^{RMSpush} = \Omega(n \ln n)$$

and

$$\lim_{k \rightarrow \infty} \Pr \left(T_k^{RMSpush} = \Omega(k \ln(k)) \right) = 1$$

Remark: We comment the following based on the results described.

1. If no overheads are allowed, i.e., if the size of the transmission between any two communicating nodes is strictly limited to m (m is the size of a message), then clearly, it will take at least $O(n)$ rounds for complete dissemination to occur when $k = O(n)$. Any protocol which achieves this dissemination time is clearly order optimal. The RLC protocol achieves this optimal dissemination time at the cost of a small overhead for reasonably large message sizes. Since, an implementation of RLC mechanism needs a padding of additional $k \log_2(k)$ bits with each coded message that is transmitted. If the size of each message, m , is such that $m \gg k \log_2(k)$ then, with a minimal overload with each transmission, the RLC protocol can overcome the lack of knowledge that a transmitting node has about the contents of the receiving node. For example, with $k = 100$, the overhead is 1% for $m = 100$ KB and it is 0.1% for $m = 1$ MB. We simply note that the overhead does not grow with the size of the messages or available bandwidth and simply depends on the number of messages that are to be disseminated simultaneously.
2. The inherent advantage of RLC comes from ‘‘coding.’’ The RMS scheme cannot do as well even if packets were chopped up into multiple parts, or multiple packets were combined into large packets. To see this, suppose each packet of size m is chopped into r mini-packets of size m/r each. There are kr packets in the system. Suppose, there are r mini-rounds within each round for the transmission of these min-packets. The new RMS scheme will take $\Omega(kr \ln(kr))$ mini-rounds or equivalently time worth $\Omega(k \ln(kr))$ rounds in the original scheme. Also, a very similar modification for RMS scheme can be done with combining a fixed number of packets. Hence, splitting or combining packets cannot help the non-coding nature of RMS scheme to achieve the optimal order attained by a coding based scheme.
3. In gossip based communication with one message, it takes $\Theta(\ln(n))$ time for the complete dissemination to occur with high probability. Thus, if the k messages are disseminated sequentially one after the other, it will take $\Theta(n \ln(n))$ time to disseminate all the messages when $k = O(n)$. The result in Theorem 3.4 shows that the uncoded RMS protocol can do no better than when the messages are disseminated one after the other.
4. Note that, if there is no bandwidth constraint (i.e., if a transmitting node can transmit its entire database) between two communicating nodes, the dissemination time is simply $O(\ln(n))$ for any k . This is since the system behaves as if there is only one message for which the dissemination time is $O(\ln(n))$ [9].
5. An interesting quantity is the total amount of information that is exchanged. If each message is of size m , the total amount of information exchanged in the RLC protocol is $O(n^2(m + k \log_2(k)))$. In the case of RMS, this quantity is at least $\Omega(n^2 \ln(n)(m + \log_2(k)))$ (additional $\log_2(k)$ bits for identifying each message). Further, any protocol will require at least $\Omega(n^2 m)$ bits of transmission.

3.1 Key idea behind the results using a mean-field approach

In the rest of this section, we provide an intuition behind our results, and also comment on the analysis approach of the protocols. The argument in this subsection is not rigorous and far from formal, and is only to provide a heuristic behind the optimal order attained by RLC mechanism. For formal and detailed proofs of the results, please refer to [3].

First consider the RMS protocol and let us concentrate on any particular node, u . Since u starts with one message at round zero, in the initial rounds, any communication from some other node is very likely to provide u with a new message. However, as u gathers more and more messages, any new message is more and more likely to be something u already has (recall the famous *coupon collector* problem [5]). Indeed, our proof of the result with RMS protocol shows that, the system takes $\Omega(n \ln(n))$ rounds just to receive the last $k/2$ messages. Thus, the performance of the RMS protocol deteriorates once a node already has roughly half the total messages.

Now consider the RLC protocol with *push* mechanism (a similar intuition can be given for the *pull* model). As before concentrate on a particular node u . The node u keeps receiving *code vectors* and decodes all the messages once the dimension of node u is k . Suppose the dimension of node u is i . We are interested in finding an expression for the number of rounds for which u has dimension i . First, let's classify the nodes as "helpful" and "unhelpful" as follows. We call a node "helpful" to u , if the subspace spanned by its *code vectors* do not lie in that of u . Otherwise, a node is "unhelpful" to u . The first point to note is that, if u is pushed by a helpful node, the conditional probability of node u increasing its dimension to $i + 1$ is at least $1 - 1/q$ by Lemma 2.1. This is true for any unhelpful node as well, i.e., if any node that is unhelpful to u is pushed by a node that is helpful to u , the unhelpful node increases its dimension (and thus becomes a helpful node, provided u is yet to increase its dimension) with probability at least $1 - 1/q$. Let F be the fraction of helpful nodes when node u has dimension i for the first time. It is not hard to argue that $F \geq (k - i)/k$, with equality corresponding to the case when node u has recovered i messages. This is because, if u has recovered i messages, even then there are $(k - i)/k$ fraction of nodes that started with the remaining $k - i$ messages. Now, consider a worst case scenario. Suppose, u just refrains from any communication and observes till F becomes at least $1/2$, after which u participates in the process again. Now, the number of unhelpful nodes pushed by a helpful node is roughly proportional to the number of helpful nodes, which is nF to start with (i.e., first time u has dimension i). We will take the proportionality factor as one (it will "mostly" be less than one since multiple helpful nodes may push to the same unhelpful node) simply for the ease of heuristic computation we intend to show here. Thus on an average, roughly $nF(1 - 1/q)$ unhelpful nodes become helpful nodes after one more round of message exchange. Thus, we have after one additional round of message exchange

$$F \leftarrow F + F(1 - 1/q) = F(2 - 1/q) .$$

It follows that, the updated value of F after an additional round of message exchange satisfies $F \geq \frac{k-i}{k}(2 - \frac{1}{q})$. Thus, after r rounds of message exchanges F becomes at least $((k - i)/k)(2 - 1/q)^r$. A simple calculation shows that, after roughly $\ln(k/(2(k - i)))/\ln(2 - 1/q)$ rounds, the fraction of helpful node becomes at least $1/2$. At this point, node u is interested in communication again. However, any helpful node can increase the dimension of u with probability at least $(1/n)(1 - 1/q)$, since any helpful node communicates with u with probability $1/n$ and increases the dimension with probability $1 - 1/q$ at least. Since there are at least $n/2$ helpful nodes now, the probability that u does not increase its dimension is at most $(1 - \frac{1}{n}(1 - \frac{1}{q}))^{\frac{n}{2}} \approx \frac{1}{\sqrt{e}}$ (for large n). Thus, once there are at least $n/2$ helpful nodes, the mean

time for u to increase its dimension is $1/(1 - 1/\sqrt{e})$. Thus, on an average, the total time T_i that u spends while it has dimension i is no more than the sum of, the time it takes till there $n/2$ helpful nodes, and $1/(1 - 1/\sqrt{e})$. Thus,

$$T_i \leq \frac{\ln(k/(2(k-i)))}{\ln(2-1/q)} + \frac{1}{(1-1/\sqrt{e})}$$

which implies

$$\begin{aligned} \sum_{i=1}^{k-1} T_i &\leq \sum_{i=1}^{k-1} \frac{\ln(k/(2(k-i)))}{\ln(2-1/q)} + (k-1) \frac{\sqrt{e}}{\sqrt{e}-1} \\ &= \frac{\ln(k^{k-1}/(2^{k-1}(k-1)!))}{\ln(2-1/q)} + (k-1) \frac{\sqrt{e}}{\sqrt{e}-1} \\ &= O(k) \end{aligned}$$

Thus, it is believable that RLC with *push* attains the optimal order.

An almost similar heuristic can be provided for *pull*. The only difference is that, here we keep track of the unhelpful nodes. More precisely, starting with the fraction of unhelpful nodes $(1 - F)$, after one more round of message exchange, the fraction of unhelpful nodes becomes at most $(1 - F)^2 + (1 - F)F(1/q)$. The first term accounts for the event that an unhelpful node stays so if it pulls from another unhelpful node, and the second term accounts for the event that even if an unhelpful node pulls from a helpful node, with probability at most $1/q$ (Lemma 2.1) it may not increase its dimension. Using this, we can find the time after which there are at most $n/2$ unhelpful nodes.

We end this section with a few words on the proofs. Intuitively, for the first $k/2$ dimensions (or $k/2$ messages with RMS) any communication is likely to be helpful in any case, with or without coding. However, we show that, the benefits of a coding based approach remains till the dimension is almost k , more precisely, till the dimension is $k - \sqrt{k} \ln(k)$ using *push*, and $k - \Theta(\sqrt{k} \ln(k))$ using *pull*. We show that it takes $O(n)$ time for the dimension of a node to reach, $k - \sqrt{k} \ln(k)$ using *push*, and $k - \Theta(\sqrt{k} \ln(k))$ using *pull*. In fact, this is the regime where we make the heuristic mean-field approach precise by studying suitable random variables that can account for the mean-field approach formally. Finally, in the regime $k \geq k - \sqrt{k} \ln(k)$, the time to increase the dimension by one cannot be worse than the time to receive a message with a single message based dissemination which takes $\ln(n)$ rounds. Thus, increasing the dimension from $k - \Theta(\sqrt{k} \ln(k))$ to k will take $O(\sqrt{n}(\ln(n))^2)$ time in the worst case. Thus its takes $O(n)$ time to decode all the messages.

The details of the analysis with the all the protocols are provided in [3].

4 Concluding Remarks

We considered the problem of disseminating multiple messages simultaneously in a large network using *gossip-based* dissemination mechanisms. We have presented a protocol based on *random linear coding* that disseminates the messages in optimal time in an order sense. The RLC protocol is quite general and does not depend on the underlying communication model. However, we have demonstrated the benefits of the protocol over a *gossip-based* communication model and in a worst case demand scenario when all the nodes want everything. Demonstrating the advantages of such a coding based approach in a more generic setting, and with a fixed communication graph is a topic of further work.

Acknowledgments

It is a pleasure to thank Dr. Tracey Ho and Prof. David Karger for helpful discussions.

References

- [1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46:1024–1016, 2000.
- [2] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip and mixing times of random walks on random graphs. Preprint available at http://www.stanford.edu/boyd/gossip_gnr.html.
- [3] S. Deb and M. Médard. Algebraic gossip: A network coding approach to optimal multiple rumor mongering. *M.I.T. LIDS Technical Report, Also submitted to IEEE Transactions on Information Theory*, April 2004.
- [4] A. Demers et. al. Epidemic algorithms for replicated database maintenance. In *Proc. ACM Symposium on Principles of Distributed Computing*, 1987.
- [5] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. J. Wiley & Sons, New York, 1964.
- [6] T. Ho, M. Médard, M. Effros, and D. Karger. The benefits of coding over routing in a randomized setting. In *Proc. IEEE Symposium on Information Theory*, 2003.
- [7] T. Ho, M. Médard, M. Effros, and D. Karger. On randomized network coding. In *Proc. 41st Allerton Annual Conference on Communication, Control and Computing*, October 2003.
- [8] Network Coding Homepage. <http://www.comm.csl.uiuc.edu/koetter/nwc/>.
- [9] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *Proc. Foundations of Computer Science*, 2000.
- [10] D. Kempe and J. Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In *Proc. 43rd IEEE Symposium on Foundations of Computer Science*, 2002.
- [11] David Kempe, Jon M. Kleinberg, and Alan J. Demers. Spatial gossip and resource location protocols. In *Proc. ACM Symposium on Theory of Computing*, 2001.
- [12] R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, October 2003.
- [13] S.-Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, February 2003.
- [14] Y. Minski. Spreading rumors cheaply, quickly, and reliably, 2002. Ph.D. Thesis, Cornell University.