Analyzing Network Coding Gossip Made Easy *

Bernhard Haeupler
Massachusetts Institute of Technology
32 Vassar Street, 32-G622
Cambridge, MA 02139, USA
haeupler@mit.edu

ABSTRACT

We introduce projection analysis – a new technique to analyze the stopping time of gossip protocols that are based on random linear network coding (RLNC). Projection analysis drastically simplifies, extends and strengthens previous results. We analyze RLNC gossip in a general framework for network and communication models that encompasses and unifies the models used previously in this context. We show, in most settings for the first time, that the RLNC gossip converges with high probability in optimal time. Most stopping times are of the form $O(k + T)$, where k is the number of messages to be distributed and T is the time it takes to disseminate one message. This means RLNC gossip achieves "perfect pipelining".

Our analysis directly extends to highly dynamic networks in which the topology can change completely at any time. This remains true, even if the network dynamics are controlled by a fully adaptive adversary that knows the complete network state. Virtually nothing besides simple $O(kT)$ sequential flooding protocols was previously known for such a setting.

While RLNC gossip works in this wide variety of networks our analysis remains the same and extremely simple. This contrasts with more complex proofs that were put forward to give less strong results for various special cases.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*network problems*

General Terms

Algorithms, Performance, Theory

Keywords

dynamic networks, gossip, multicast, network coding

*A preliminary version of this research was presented at an invited session of the 2010 Allerton Conference for Communication, Computing and Control; A full version of this paper is available at ArXiv:1010.0558.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
STOC'11, June 6–8, 2011, San Jose, California, USA.
Copyright 2011 ACM 978-1-4503-0691-1/11/06 ...$10.00.

1. INTRODUCTION

This paper presents a new way to analyze gossip protocols based on random linear network coding, which we call projection analysis. Projection analysis substantially simplifies, extends, and strengthens the results of previous work [4, 9, 10, 30, 31].

Gossip is a powerful tool to efficiently disseminate information. Its randomized nature is especially well-suited to work in unstructured networks with unknown, unstable or changing topologies. Because of this, gossip protocols have found a wide range of applications [1, 11, 15, 16, 26] and have been extensively studied over the past several decades [3, 5, 15, 20, 22, 24, 25, 29].

Recently, gossip protocols based on random linear network coding (RLNC) [2, 21, 28] have been suggested [8] to cope with the additional complexities that arise when multiple messages are to be distributed in parallel. RLNC gossip has been adopted in many practical implementations [6, 13–15, 23] and has performed extremely well in practice.

These successes stand in contrast to how little RLNC gossip is understood theoretically. Since its initial analysis on the complete graph [8–10], several papers [4,30,31] have tried to give good upper bounds on the stopping time of RLNC gossip in more general topologies. However, none of them address the case of unstable or changing topologies, and, even with the restriction to static networks, the guarantees are far from being general or tight on most graphs. In addition, all existing proofs are quite involved and do not seem to generalize easily.

1.1 Our Results:

This paper has two main contributions. The first is our new projection analysis technique which is both simpler and more powerful than previous approaches. Projection analysis relates the stopping time for k messages to the much easier to analyze time T needed to disseminate a single message. For the first time, and in practically all settings, this technique shows that RLNC gossip achieves perfect pipelining, i.e., it disseminates k messages in order optimal $O(T+k)$ time. Our results match, and in most cases improve, all previously known bounds and apply to more general models.

To formalize this, we give a general framework for network and communication models that encompasses and unifies the models suggested in the literature so far. We give concrete results for several instantiations of this framework and give more detailed comparisons with previous results in each section separately.

As a second major contribution, our framework extends all models to (highly) dynamic networks in which the topology is allowed to completely change at any time. All of our results hold in these networks, even if the network dynamics are controlled by a fully adaptive adversary that decides the topology at each time, based on the complete network state as well as all previously used randomness. Virtually nothing, besides simple $O(kT)$ sequential flooding protocols [27], is known for such truly pessimistic network dynamics so far. Having optimal "perfectly pipelined" stopping times in worst-case adaptive dynamic networks is among the strongest stability guarantees for RLNC gossip that one might hope for. To this end, our results are the first that formally explain RLNC gossip performance in the dynamic environments it is used in and was designed for. While the algorithm works in this wide variety of settings, our analysis remains mostly the same and extremely simple, in contrast with complex proofs that were previously put forward for the static setting.

1.2 Organization:

Section 3 reviews the RLNC algorithm and Section 4 describes the new projection analysis technique. In Section 5 we introduce the network model framework. Section 6 shows how to apply the projection analysis in various instantiations of this framework. Section 7 discusses several ways in which the intentionally simple proofs from Section 6 can be extended or sharpened.

2. BACKGROUND AND RELATED WORK

Gossip is the process of spreading information via a randomized flooding procedure to all nodes in an unstructured network. It stands in contrast to structured multi-cast in which information is distributed via an explicitly built and maintained structure (e.g., a selection of spanning trees). While structured multi-cast can often guarantee optimal use of the limited communication resources, it relies heavily on having a known and stable network topology and fails in distributed or uncoordinated settings. Gossip protocols were designed to overcome this problem. By flooding information in a randomized fashion they guarantee to deliver messages with high probability to all nodes with little communication overhead. This stability and distributed nature of gossip makes it an important tool for collaborative content distribution, peer-to-peer networks, sensor networks, ad-hoc networks and wireless networks; literature applying gossip in many areas and for many purposes is vast (e.g., [1, 11, 15, 16, 26]).

The gossip spreading of both a single message and multiple messages [3,5,15,20,22,24,25,29] has been intensely studied. The spreading of one message often follows a comparatively simple epidemic random process in which the message is flooded to a randomly chosen subset of neighbors. Spreading multiple messages in parallel is significantly more complicated because nodes need to select which information to forward. The main problem in this context is that widely spread messages get forwarded more often and quickly outnumber rarer messages. In many cases the slow spread of the rare messages dominates the time needed until all nodes know every message.

A powerful and elegant way to avoid this and similar problems is the use of network coding techniques. Network coding as introduced by the seminal work of Ahlswede, Cai, Li and Yeung [2] breaks with the traditional concept that information is transported by the network as an unchanged entity. Ahlswede et al. show that in many multi-cast scenarios the optimal communication bandwidth can be achieved if and only if intermediate nodes in the network code information together. Li et al. [28] showed that for multi-cast it is enough if intermediate nodes use linear coding, i.e., computing linear combinations of messages. Following this, Ho et. al [21] showed that the coefficients for these linear combinations need not be carefully chosen with regard to the network topology; indeed, using of random linear combinations works with high probability for any fixed network.

The strong performance guarantees and the independence of the coding procedure from any global information about the network makes random linear network coding (RLNC) the perfect tool for spreading multiple messages. This was first observed and made formal by Deb and Médard [8]. They show that using randomized gossip and RLNC in a complete network in which each of the nodes starts with one message, all information can be spread to all nodes in linear time, beating all non-coding approaches. After the introduction of this protocol in [8] and its follow-up [9, 10] it was used in many applications [6, 13, 14, 23], most notably the Microsoft Secure Content Distribution (MSCD) or Avalanche System [15]. There has also been more theoretical work [4, 30, 31] investigating the convergence time of the RLNC-algorithm on general static network topologies. We give a detailed description and comparison to these works in Section 6.

3. THE RLNC ALGORITHM

In this section, we give a brief description of the RLNC algorithm. The algorithm is simple and completely independent of the network structure or communication protocol. Alternative descriptions of the algorithm can be found in [8] or [6].

The RLNC algorithm sends out packets in the form of vectors over a finite field $\mathbb{F}_q$, where q is an arbitrary prime or prime power. We assume that there are k messages, $\vec{m_1}, \ldots, \vec{m_k}$, that are vectors from $\mathbb{F}_q^l$ of length l. Every packet that is sent around during the execution of the algorithm has the form $(\vec{\mu}, \vec{m})$, where $\vec{m} = \sum_{i=1}^{k} \mu_i \vec{m_i} \in \mathbb{F}_q^l$ is a linear combination of the messages, and $\vec{\mu} = (\mu_1, \ldots, \mu_k) \in \mathbb{F}_q^k$ is the vector of the coefficients. If enough packets of this form are known to a node, i.e., the span of the coefficient vectors is the full space $\mathbb{F}_q^k$, Gaussian elimination can be used to reconstruct all messages. For this, only k packets with linearly independent coefficient vectors are needed. Linearity furthermore guarantees that any new packet that is created by taking a linear combination of old packets has the same valid format. With this, it is easy to see that a node can produce any packet whose coefficient vector is spanned by the coefficient vectors of the packets it knows. The algorithm is now easily described:

Each node v maintains a subspace X_v that is the span

of all packets known to it at the beginning and received so far by simply storing all received packets. If v does not know any messages at the beginning, then X_v is initialized to contain only the zero vector. If v knows some message(s) $\vec{m}_i$ at the beginning, X_v is initialized to contain the packet $(\vec{\mu}, \vec{m}_i)$ in which $\vec{\mu}$ is the i^{th} standard basis vector. X_v also contains all linear combinations that complete the span of these packet(s). Whenever node v sends out a packet, it chooses a uniformly random packet from X_v by taking a random linear combination of the stored packets. At the end of each round, all received packets are added to X_v and again the span is taken. If the subspace spanned by the coefficient vectors is the full space, a node decodes all messages.

Throughout the rest of the paper we will solely concentrate on the "spreading" of the coefficient vectors; the linear combination of the messages implied by a coefficient vector $\vec{\mu}$ is always sent along with it. We therefore define Y_u to be only the coefficient part of X_u, i.e., the projection onto the first k components.

4. THE PROJECTION ANALYSIS

4.1 Previous Approaches

When analyzing the RLNC algorithm, Sub and Médard [8] were the first to use the dimensionality of the subspaces Y_v as a measure of progress. They defined a node u to be helpful for a node v if it knows something v does not, i.e., if the subspace Y_u is not contained in Y_v. Whenever node v receives a packet with a coefficient vector outside of Y_u the dimension of Y_v increases, which can happen at most k times. More importantly, Sub et al. made the observation that if a node u that is helpful to node v sends this node a packet, this dimension increase is very likely to happen. The reason for this is that the vectors $Y_u \cap Y_v$, that do not extend the dimensionality of Y_v, form a lower dimensional subspace in Y_u. Thus, whenever a node u sends a random vector from Y_u to a node v it is helpful for, the probability that the dimension of Y_v increases is at least $1 - 1/q$. This fact and the notion of helpfulness is used as a crucial tool in all further RLNC proofs [4, 9, 10, 30, 31].

4.2 The Projection Analysis Technique

We argue that the right way to look at the spreading of information is to look at the orthogonal (dual) complement $Y_u^\perp$ of the coefficient subspaces Y_u. While the coefficient subspaces grow monotonically to the full space their orthogonal complement decreases monotonically to the empty span. To see how quickly this happens, we first concentrate on one fixed (dual) vector $\vec{\mu} \in F_q^k$, determine the time that is needed until it disappears from all subspaces $Y_u^\perp$ with high probability and than take a union bound over all those dual vectors.

A different and maybe even simpler way of looking at this is that we are analyzing the spreading process by looking at its q^k one-dimensional projections. To keep track of these projections we introduce the following crucial notion of knowing:

DEFINITION 1. *A node u knows about $\vec{\mu} \in \mathbb{F}_q$ iff its coefficient subspace Y_u is not orthogonal to $\vec{\mu}$, i.e., if there is a vector $\vec{c} \in Y_u$ with $\langle \vec{c}, \vec{\mu} \rangle \neq 0$.*

Note that a node u knowing a vector $\vec{\mu}$ does not imply $\vec{\mu} \in Y_u$ or anything about u being able to decode a message associated with the coefficients $\vec{\mu}$. Counter intuitively, because we are not working over a positive-definite inner-product space, it can even be that $\vec{\mu} \in Y_u$ but u does not know $\vec{\mu}$. Knowing $\vec{\mu}$ solely expresses that the node is not completely ignorant about the set of packets that have a coefficient vector orthogonal to $\vec{\mu}$. The next lemma proves the two facts that make this notion of knowledge so useful:

LEMMA 2. *If a node u knows about a vector $\vec{\mu}$ and transmits a packet to node v then v knows about $\vec{\mu}$ afterwards with probability at least $1 - 1/q$. Furthermore, if a node knows about all vectors in $\mathbb{F}_q^k$ then it is able to decode all k messages.*

PROOF. Knowledge about a $\vec{\mu}$ spreads with a transmission with probability $1 - 1/q$ because the vectors in Y_u that are perpendicular to $\vec{\mu}$ form a lower dimensional hyperplane (with at most a $1/q$-fraction of the volume). For the second claim, we prove that any node u that is not able to decode does not know at least one vector $\vec{\mu}$: If u can not decode than Y_u is not the full space. Because Y_u is a subspace, it is lower-dimensional and we can find a vector $\vec{\mu}$ that is orthogonal to Y_u. This vector $\vec{\mu}$ is then by definition not known to u, a contradiction. □

Lemma 2 implies that the spreading of knowledge for a fixed vector $\vec{\mu} \in F_q^k$ is a monotone increasing, epidemic set growing process – or differently phrased – each projection looks essentially like a $1/q$-faulty one-message flooding process. It is usually relatively easy to understand this process and to determine the expected time T until all nodes know $\vec{\mu}$. Furthermore, because the set growing process is a monotone Markov chain its stopping probability has an exponentially decaying tail. In most cases this tail kicks in close to the expectation. This allows to pick a time t (usually $t = O(T+k)$) after which any vector in $\mathbb{F}_q^k$ has spread with probability at least $1 - 2^{-O(k)}$. Taking a union bound over all q^k vectors then completes the proof that with high probability everything has spread. The following theorem summarizes these ideas:

THEOREM 3. *Fix a prime (power) $q \geq 2$, a probability $\delta > 0$ and an arbitrary network and communication model whose dynamics do not depend on what nodes know.*
We define a projection or faulty one-message broadcast of the model as follows: One message starts at some node v, and in every round, every node, that knows the message and is supposed to communicate according to the communication model, forwards it with probability $1-1/q$ and remains silent otherwise.
If, for every node v, the probability that the message reaches all nodes after t rounds is at least $1 - \delta q^{-k}$ then k messages can be spread in the same model in time t with probability $1 - \delta$ using RLNC gossip protocol with field size q.

PROOF. The theorem follows directly from the discussion above and Lemma 2. Initially every non-zero vector $\vec{\mu} \in \mathbb{F}_q^k$ is known to at least one node, namely the one that knows about the i^{th} message, where i is a non-zero component of $\vec{\mu}$. Whenever the network and communication model dictates that a node u that knows $\vec{\mu}$ sends a message to a node v, Lemma 2 shows that with probability $1 - 1/q$ the node v afterwards knows $\vec{\mu}$. Therefore, the spreading of each vector

$\vec{\mu}$ behaves like a faulty flooding process that floods $\vec{\mu}$ in every transmission with probability $1-1/q$. By assumption we have that after t time steps every vector from $\mathbb{F}_q^k$ fails to spread to all nodes with probability at most δq^{-k}. Taking a union bound over all q^k vectors gives the guarantee that the probability that after t rounds all nodes know all vectors is at least $1-\delta$. According to Lemma 2 all nodes can decode in this case and have learned the k messages. □

4.3 A Typical Template

Next, we give a typical and easy way to apply Theorem 3. We show that, even for $q=2$ the time for one vector $\vec{\mu}$ to spread is often dominated by a negative binomial distribution $NB(T,1-p)$, where T is the expected time to spread one message, and p is a constant probability. Such a distribution has a strong enough tail to prove optimal $O(T+k)$ stopping times. In what follows we give a simple template for this argument:

What is needed for this template is a definition of a "successful round", such that at most T such rounds are needed to spread a single vector $\vec{\mu}$ and such that a round is not a success with probability at most p. The appropriate definition of success depends on the network model and is usually centered around how many additional nodes come to know the vector in a "good round".

Since nodes do not forget any information, this spreading process is monotone, i.e., no progress gets lost in a bad round. Thus, if the knowledge about $\vec{\mu}$ has not spread after $t=c(k+T+\log\delta^{-1})$ steps, then there were at least $c(k+T+\log\delta^{-1})-T>(c-1)(k+T+\log\delta^{-1})$ failures, whereas one would only expect $pc(k+T+\log\delta^{-1})$. If we choose the constant c large enough, a standard Chernoff bound implies that the probability for this to happen is at most $2^{-O(k+T+\log\delta^{-1})}$. This is small enough that, after a union bound over all q^k vectors (e.g., for $q=2$), the probability that all k messages have not spread is at most δ.

This is usually all that is needed to prove order optimal $O(k+T)$ stopping times. It also shows that we actually obtain high probability results. In particular as shown here, an additive $\Theta(\log\delta^{-1})$ additional rounds typically suffices to obtain a $1-\delta$ success probability for any $\delta>0$. This strong and optimal guarantee is (up to some scaling) true for all our results, even so we do not state it explicitly for sake of simplicity.

5. NETWORK MODEL FRAMEWORK

In this section, we elaborate on our network model framework that encompasses and extends the models suggested in the literature so far. The models and the results can be easily extended further; we chose the following framework as a trade-off between simplicity and generality.

The Network:
We consider networks that consist of n nodes. A network is specified by a (directed) graph $G(t)$ on these nodes for every time t. Edges in $G(t)$ are links and present potential communication connections between two nodes in round t.

(Adversarial) Dynamics:
In all previous papers that analyzed the RLNC algorithm, the network topology was assumed to be *static*, i.e., $\forall t: G(t)=G$. As discussed in the introduction, we allow the network topology to change completely from round to round and allow a fully adaptive adversary to choose the network. Because we are dealing with randomized protocols, we have to specify precisely what the adversary is allowed to adapt to. In our models (similar to [27]) an *adaptive adversary* gets to know the complete network state and all previously used randomness when choosing the topology. After that, independent randomness is used to determine the communication behavior and the messages of the nodes on this topology. The last assumption is not essential; in further work of the author [17] it was shown that, for a large enough q, the analysis presented here works even against omniscient adversaries, that get to know all randomness used for the coding coefficients in advance.

The Goal: Gossip:
Distributed over the network are k messages, each with a unique index from $1,\dots,k$. Each message is initially known to at least one node. Throughout this paper, we assume a worst-case starting configuration for all messages, including the case in which all messages are exclusively known to only one node (see also Section 7.1). The goal of gossip protocols is to make all messages known to all nodes in the network using as little time as possible (in expectation and with high probability).

Communication:
Nodes communicate along links during transactions that are atomic in time. In each round, one packet is transmitted over a link, if this link is activated in this round. From the view of a node there are four commonly considered types of connections. Either a node sends to all its neighbors, which is usually referred to as BROADCAST, or it establishes a connection to one (e.g., uniformly random) neighbor and sends (PUSH) or receives (PULL) a message or both (EXCHANGE). In all cases, the packet is chosen without the sender knowing which node(s) will receive it.

Message and Packet Size:
As described in Section 3, we assume that all messages and packets have the same size, and that a packet exactly contains one encoded message and its RLNC-coefficients. Note that the restriction on the message size is without loss of generality, since one can always cut a big message into multiple messages that fit into a packet. We also assume that the message size is large enough that the size of the RLNC-coefficients that are sent along is negligible, i.e., $l \gg k$. This assumption was made by all previous work and is justified by simulations and implementations in which the overhead is only a small fraction (e.g., $< 1\%$ [8]) of the packet size.

Synchronous versus Asynchronous Communication:
We consider two types of timing models. In the synchronous case, all nodes get activated at the same time, choose their messages independently, and messages get delivered according to the current network $G(t)$ and who sends and receives from whom. Note that this model is inherently discrete, and we assume that $t=1,2,\dots$ are the times when nodes communicate. For the asynchronous case, we assume that every node communication is triggered independently by a Poisson clock, which implies (with probability one) that at most one node sends its message at any time. This model can be directly translated into a discrete time model that defines round i as the i^{th} time such a communication takes place. The model considered in the literature so far assumes that

every node is activated uniformly at random to communicate and then chooses a uniformly random neighbor for a PUSH, PULL or EXCHANGE. They also scale the time in the asynchronous model by a factor of $1/n$ so that each node gets activated once per time unit in expectation. We do not assume uniformity in either of the two distributions, and we present results for this more general model in Section 6.2.

6. APPLICATIONS AND RESULTS

In this section we take the models from Section 5 and describe the results that can be obtained for them using the projection analysis technique. There is a section for each communication model. We concentrate on showing simple proofs that solely use the template from Section 4.3. In Section 7, we revisit the models covered here and discuss some proof extensions.

6.1 Random Phone Call Model

We first consider the paper [8] by Deb and Médard and the follow-up [9, 10] and show how to simplify and improve the analysis. The papers use a fairly simple model from our framework, namely the synchronous PUSH or PULL model on the complete graph, i.e., $G(t) = K_n$. This means in each round each node picks a random other node to exchange information with. This model is also known as the random phone call model and was introduced by [11]. It is shown in [8] that it is possible in this model to spread $k = \Theta(n)$ messages in $O(n)$ time if $q = n$. This beats the $O(n \log n)$ time of n sequential $O(\log n)$-phases of flooding just one message (see also [12] for a different non-coding approach to this problem). The follow-up papers [9, 10] generalize this result to smaller number of messages k and allow q to be as small as k. They show that the running time of the algorithm is $t = O(k+\sqrt{k}\log k \log n)$, i.e., order optimal as long as $k \geq \log^{2+\epsilon} n$ for any $\epsilon > 0$. In order to prove this result, they have to assume that each node knows initially only one message and that initially the messages are equally spread. Even with these assumptions the analysis is long and complicated and the authors state themselves in their abstract that "While the asymptotic results might sound believable, owing to the distributed nature of the system, a rigorous derivation poses quite a few technical challenges and requires careful modeling and analysis of an appropriate time-varying Bernoulli process."

Our next lemma shows that RLNC gossip actually finishes with high probability in order optimal stopping time $O(k + \log n)$. Our analysis is much simpler and has many further advantages: It holds for all choices of k and allows q to be as small as 2. The proof is (almost) the same for all communication models while two completely separate proofs for the PUSH and the PULL protocol were given in previous works. Furthermore, our proof does not rely on any assumptions on the initial message distribution. This is important since we show in Section 7.2, that the well-mixed initial state assumed in [8–10] actually provably speeds up the stopping time compared to the worst-cast distribution for which our result holds. Our proof gives a success probability of $1 - 2^t$ if the algorithm runs for $O(t)$ time. In the setting of [8] with $k = n$, this is $1 - 2^{-n}$ instead of the $1 - 1/n$ stated there. Lastly, it is interesting to note that later, more general approaches like [4] and [30] were unable to prove any running time that beats the simple non-coding $O(n \log n)$ sequential flooding approach when applied to the complete graph ([4] even gives only a $O(n^2)$ convergence time).

LEMMA 4. *RLNC gossip in the random phone call model with $q = 2$ spreads k messages in $O(k + \log n)$ time with high probability. This holds independently from the initial distribution of the messages and of the communication model (e.g., PUSH, PULL, EXCHANGE).*

PROOF. We use the template from Section 4.3: For this we fix a coefficient vector $\vec{\mu}$ and define a round as successful if the number of nodes that know it increases by at least a constant factor $\lambda > 1$ or if the number of nodes that do not know $\vec{\mu}$ decreases by a factor of λ. There are at most $O(\log n)$ successful rounds needed until at least $n/2$ nodes know $\vec{\mu}$ and at most another $O(\log n)$ successful rounds until all nodes know $\vec{\mu}$. It remains to be shown that each round succeeds with constant probability.

We first consider the PULL model. At first we have $i < n/2$ nodes that know $\vec{\mu}$ and at least $n/2$ nodes pulling for it. Each of those nodes has a probability of i/n to hit a knowing node. We expect a i/n fraction of the ignorant nodes, i.e., at least $i/2$ nodes, to receive a packet from a node that knows about $\vec{\mu}$. The independence of these successes and Lemma 2 prove that with constant probability at least $\Omega(i)$ nodes learn about $\vec{\mu}$. Once there are at least $n/2$ nodes that know $\vec{\mu}$, each of the ignorant nodes pulls a packet from a knowing node with probability at least $1/2$ and applying Lemma 2 again finishes this case, too.

The proof for the PUSH model is similar. If there are $i < n/2$ nodes that know $\vec{\mu}$ and push out a packet, then there are at least $n/2$ ignorant nodes that each receive at least one packet from one of the i nodes with probability $1 - (1 - 1/n)^i$. It is not hard to see that, in total, $\Omega(i)$ ignorant nodes receive a packet from a node that knows $\vec{\mu}$ with constant probability. Lemma 2 now guarantees that, with constant probability, the number of ignorant nodes that learn $\vec{\mu}$ is only a small factor smaller. Once there are $n/2$ nodes knowing $\vec{\mu}$ and each of these pushes out, each node that does not know $\vec{\mu}$ has a chance of $(1 - 1/n)^{n/2} \approx e^{-2}$ per round to receive a packet from a node that knows $\vec{\mu}$. Applying Lemma 2 again finishes the proof. □

6.2 Asynchronous Single Transfer Protocols

After the helpfulness of RLNC gossip was established for the complete graph by [8], the papers [30], [31] and [4] generalized it to general static topologies and consider asynchronous and synchronous PUSH, PULL and EXCHANGE gossip. In this section we first review the previous results and than show how to improve over them giving an exact characterization of the stopping time or RLNC gossip for $k = n$ messages using the template of Section 4.3.

The paper "Information Dissemination via Network Coding" [30] by Mosk-Aoyama and Shah was the first to consider general topologies. They consider a similarly general version of the synchronous and asynchronous gossip as presented here and analyze the stopping times for $k = n$ in dependence on the conductance. Their analysis implies that with high probability $O(n \log n)$ phases of n asynchronous rounds suffice for the complete graph and constant degree expanders and $O(n^2)$ such phases for the ring-graph. While the analysis is very interesting, these results do not beat the simple (non-coding) sequential flooding protocol and the stopping

time of the ring-graph and many other graphs is even off by a factor of n. Their running times for the synchronous model are similar but loose another $\log n$-factor. Their dependence is on the success probability $1-\delta$ is furthermore multiplicative in $\log \delta^{-1}$ because it stems from a standard probability amplification argument.

More recently, [31] and the paper "Tight bounds on Algebraic Gossip on Arbitrary Graphs" [4] also analyzed RLNC gossip using two completely different approaches. The authors of [4] point out that the analysis of [31] is flawed and prove that the asynchronous RLNC gossip on a network with maximum degree Δ takes with high probability $O(\Delta n^2)$ time (using the scaling of time used in this paper). Their proof uses an interesting reduction to networks of queues and applies Jackson's theorem. They also give a tight analysis and lower bounds for a few special graphs with interesting behavior (see below). While their analysis is exact for few selected graphs the analysis is far from tight; in most graphs the maximum degree has nothing to do with the stopping time of RLNC gossip. The major question asked in [4] is to find a characterizing property of the graph that determines the stopping time.

We give such a characterization for the asynchronous case with $k = n$ assuming a worst-cast message initialization. The model we use is a generalization of the classical PUSH, PULL and EXCHANGE model: We allow the topology in every round to be specified by a graph with directed and/or undirected edges and a probability weight p_e on every edge e, such that the sum over all edges is at most 1. In every round each edge gets exclusively selected with probability p_e, i.e., in each round at most one edge gets selected. If the edge is undirected an EXCHANGE is performed and if a directed edge gets activated a packet is delivered in the direction of the edge. Note that this model is a generalization of the "classical" communication models. To obtain the probability graph from an undirected network with PUSH or PULL, one just has to replace every undirected edge $\{u, v\}$ by two directed edges with probability weight $\frac{1}{n\Delta_u}$ and $\frac{1}{n\Delta_v}$, where Δ_u and Δ_v are the degrees of u and v respectively. To obtain the EXCHANGE protocol each undirected edge $\{u, v\}$ simply has the probability weight $\frac{1}{n\Delta_u} + \frac{1}{n\Delta_v}$.

Given such a network graph G with probability weights p_e we define the min-cut $\gamma(G)$ as:

$$\gamma(G) = \min_{\emptyset \neq S \subset V} \sum_{e \in \Gamma_G^+(S)} p_e,$$

where $\Gamma_G^+(S)$ are all edges leaving a non-empty vertex-subset $S \subset V$ in G. The next two lemmas show that this quantity exactly captures how long the RLNC gossip takes to spread n messages.

LEMMA 5. *If for every time t the min-cut of $G(t)$ is at least γ then the asynchronous single transfer algorithm with $q = 2$ spreads n messages with probability at least $1 - 2^{-n}$ in $O(\frac{n}{\gamma})$ time.*

PROOF. Our proof proceeds along the lines of the simple template from Section 4.3 and concentrates on the spreading of one coefficient vector $\vec{\mu}$. We define a round as a success if and only if one more node learns about $\vec{\mu}$. It is clear that exactly n successes are needed. From the definition of γ and Lemma 2 follows that each round is successful with probability at least $\gamma(1-1/q)$. Thus, if we run the protocol for $t = \frac{cn}{(1-1/q)\gamma}$ rounds we expect at least cn successes, where c is a constant greater than 1. A standard Chernoff bound implies that the probability that we get less than n is at most $2^{-O(n)}$. If we choose c appropriately this is small enough to end up with 2^{-n} after taking the union bound over the $q^k = 2^n$ vectors. □

The next lemma proves that the $O(\frac{n}{\gamma})$ bound is optimal.

LEMMA 6. *With high probability, the asynchronous single transfer algorithm takes exactly $\Theta(\frac{n}{\gamma})$ rounds to spread n messages if it is used on any fixed graph G with min-cut $\gamma(G)$ on which at least $\Theta(n)$ messages are initialized inside this cut.*

PROOF. In each round, at most one packet can cross the min-cut. For this to happen, an edge going out of the cut has to be selected and the probability for this is by definition exactly γ. In order to be able to decode the $\Theta(n)$ messages at least $\Theta(n)$ packets have to cross the cut, each taking in expectation $O(\frac{1}{\gamma})$ rounds. A standard Chernoff bound shows that it takes with high probability at least $\Omega(\frac{n}{\gamma})$ rounds until $\Theta(n)$ packets have crossed the cut. This proves the lower bound and Lemma 5 gives the matching upper bound. □

Applying Lemma 5 to the standard PUSH/PULL model gives a $O(\Delta n^2)$ stopping time for any connected static or dynamic graph whose maximum degree is bounded by Δ. Restricted to the static case, this is the main result of [4]. It also gives $O(n^2)$ for the complete graph (instead of the $O(n^3)$ of [4]) and nicely explains the behavior of the barbel graph and the extended barbel-graph that were considered by [4]. The proof of Lemma 5 can furthermore easily be extended to show that the dependency on the success probability is only logarithmic and additive in contrast to both [30] and [4].

6.3 BROADCAST

In this section we give stopping times for synchronous and asynchronous BROADCAST protocols in arbitrary dynamic networks. These are to our knowledge the first results for the RLNC algorithm in such a setting. We think the results in this section are of particular interest for highly dynamic networks because many of the unstable or dynamic networks that occur in practice, like ad-hoc-, vehicular- or sensor-networks, are wireless and thus have an inherent broadcasting behavior.

To fix a model, we first consider the simple synchronous broadcast model. We assume without loss of generality that the network graph G is directed because any undirected edge can be replaced by two directed, anti-parallel edges. Having wireless networks in mind, we also assume that in each round each nodes computes only one packet which is then send out to all neighbors. Our results also hold for the easier but less realistic model where nodes send out a different packet to each neighbor.

The parameter that governs the time to spread one message in a static setting is (not surprisingly) the diameter D and it is easy to prove $\Theta(D + k)$ stopping times for k messages using the projection analysis. In a dynamic setting this is not true. Even for just one message, an adaptive adversary can, for example, always connect both the set of nodes that know it and the set of nodes that do not know it to a clique and connect the two cliques by one edge. Even

though the graph $G(t)$ has diameter 3 at all times, it clearly takes at least n rounds to spread one message. In order to prove stopping times in the adaptive adversaries model we switch to a parameter that indirectly gives a good upper-bound on the diameter for many graphs. The parameter we use is the isoperimetric number $h(G)$, which measures the "node expansion" of a graph as follows:

$$h(G) := \min_{S \subseteq V} \frac{|N_G^+(S)|}{\min(|\overline{S}|, |S|)},$$

where $N_G^+(S)$ are the nodes in G outside of the subset S that are in the directed neighborhood of S.

To give a few example values: for disconnected graphs $h(G)$ is zero and for connected graphs it ranges between 1 and $\frac{2}{n}$; for a k-vertex-connected graph G we have $h(G) = \Omega(\frac{k}{n})$ and $h(G) = \Theta(1)$ holds if and only if G is a vertex-expander (or a complete graph).

We are going to show that the expected time for one message to be broadcast is at most $T = \frac{\log(nh(G))}{h(G)}$. This is $O(n)$ for a line or cycle, $O(\frac{n \log k}{k})$ for a k-vertex-connected graph, and $O(\log n)$ for any vertex-expander. Our bound is tight in the sense that for any value h with $1 \geq h \geq \frac{2}{n}$ there is a static graph G that has diameter at least $O(T)$ and isoperimetric number $h(G) = \Theta(h)$. Having an upper bound on the time T it takes to spread one message we again prove a perfectly pipelined time of $O(T + k)$ for k messages:

LEMMA 7. *With high probability the synchronous broadcast gossip protocol takes at most $O(\frac{\log(nh)}{h} + k)$ rounds to spread k messages as long as the isoperimetric number of the graph $G(t)$ is at least h at every time t.*

PROOF. We use the simple template from Section 4.3 and concentrate on the spreading of one coefficient vector $\vec{\mu}$. We define a round to be a success if and only if the number of nodes that know $\vec{\mu}$ grows at least by a $\frac{h}{7}$ fraction or if the number of nodes that do not know $\vec{\mu}$ shrinks at least by the same factor.

We want to argue that at most $T = O(\frac{\log(nh)}{h})$ successes are needed to spread $\vec{\mu}$ completely. Note that this is slightly better than the straight forward $(1 + \frac{h}{7})^T \geq n$ bound that would lead to $T = O(\frac{\log(n)}{h})$. The improvement comes from exploiting the fact that the number of nodes that learn is an integral quantity: In the first $\frac{7}{h}$ successful rounds at least one node learns about $\vec{\mu}$. The next $\frac{7}{2h}$ successful rounds at least 2 nodes learn about $\vec{\mu}$ and the following $\frac{7}{3h}$ successful rounds it is 3 new nodes and so on. There are $\frac{n}{2} \cdot \left(\frac{7}{h}\right)^{-1}$ such phases until at least $n/2$ nodes know $\vec{\mu}$. The downward progression than follows by symmetry. The total number of successes sums up to:

$$T \leq 2\frac{7}{h} \sum_{i=1}^{O(nh)} \frac{1}{i} = O(\frac{\log nh}{h}).$$

To finish the proof, we show that every round has a constant success probability. This follows from Lemma 2 if for a success only one node is supposed to learn about $\vec{\mu}$. If at least $\lceil i \rceil \geq 2$ nodes are supposed to learn then by the definition of a success and of $h(G(t))$ there are $k \geq \lceil 7i \rceil \geq 4\lceil i \rceil$ nodes on the knowledge cut, i.e., at least k nodes that do not know $\vec{\mu}$ are connected to a node that knows about $\vec{\mu}$. We invoke Lemma 2 again to see that each of these nodes fails to learn about $\vec{\mu}$ with probability at most $1/q \leq 1/2$. Finally, Markov's inequality gives that the probability that more than $k - \lceil i \rceil \geq \frac{3}{4}k$ nodes fail to learn $\vec{\mu}$ is at most 2/3. A round is therefore successful with probability at least 1/3. □

A similar result to Lemma 7 can be proven for the asynchronous BROADCAST model in which at every round each node gets selected uniformly independently at random (i.e., with probability $\frac{1}{n}$) to broadcast its packet to its neighbors:

LEMMA 8. *With high probability the asynchronous broadcast gossip protocol takes at most $O(n \cdot (\frac{\log(nh)}{h} + k))$ rounds to spread k messages as long as the isoperimetric number of the graph $G(t)$ is at least h at every time t.*

7. EXTENSIONS

In this section we discuss how the simple proofs from Section 6 that use only the template from Section 4.3 can be extended to give more detailed or sharper bounds.

7.1 Exploiting a Well-Mixed Message Initialization

As stated in Section 5 we assume throughout the paper that k messages are to be spread that are initially distributed in a worst-case fashion. All earlier papers restricted themselves to the easier special case that $k = n$ and that each node initially holds exactly one message [4, 30], or that k is arbitrary but the network starts in a similarly well-mixed state in which each message is known by a different node and all messages are equally spread over the network [10]. In many cases the worst-case and any well-mixed initialization take equally long to converge because the running time is lower bounded and bottlenecked by the flooding time T for a single message or the time it takes for a node to receive at least k packets. Nevertheless, there are cases where a well-mixed initialization can drastically improve performance.

Our proof technique explains this and we give a simple way to exploit assumptions about well-mixed initializations to prove stronger performance guarantees: If, e.g., each node initially holds exactly one of $k = n$ messages then most vectors $\vec{\mu}$ are already known to most nodes initially. More precisely exactly the $\binom{n}{i}(q-1)^i$ vectors with i non-zero components are initially known to exactly i nodes. With many vectors already widely spread initially the union bound over the failure probabilities for all vectors to spread after t rounds can decrease significantly. Taking the different quantities and probabilities for nodes that are initially known to a certain number of nodes in account one can prove in theses cases that a smaller t suffices.

One example for a mixed initialization being advantageous is discussed in Section 7.2. Another one is the convergence time of the asynchronous PUSH and PULL protocol on the star-graph: For both PUSH and PULL the network induced by the star-graph has a min-cut of $1/n^2$ which leads according to Lemma 5 and 6 to a stopping time of $\Theta(n^3)$ under a worst-case initialization. The corresponding lower bound from Lemma 6, which relates the convergence time to the min-cut of the network graph, has to assume that at least a constant fraction of the messages are initialized inside a bad cut. For the "classical" initialization in which each node

starts with exactly one message this is true for the PUSH model but not in the PULL model in which every bad cut only contains few messages. Indeed assuming a well-mixed initialization the PUSH protocol still takes $\Theta(n^3)$ time to while a much lower $\Theta(n^2 \log n)$ stopping time for the PULL model can be easily derived using the projection analysis.

7.2 Exact Dependence on k

In many (highly connected) networks the spreading time T for one message is short and $O(k)$ becomes the dominant term in the order optimal $O(k+T)$-type upper bounds presented in this paper. So is, for example, $T = O(\log n)$ for most expanding networks. While it is clear that at least k packets need to be received at each node it becomes an interesting question how large the constant factor hidden by the O-notation is. Differently stated, we ask how large the fraction of helpful or innovative packets received by a node is over the execution of the protocol.

Determining and even more optimizing proofs to obtain such constants is usually a big hassle or even infeasible due to involved proofs. In those cases simulation is used in practice to get a good estimation of the constants (e.g., [10]). Our method is simple enough that it is often possible to prove (optimal) constants (and lower order terms). All that is needed is to replace the Chernoff bound in the template from Section 4.3 by an argument that gives the correct base in the exponential tail-bound. In many cases one can show that this constant is arbitrarily close to the optimal constant one, i.e., we get $t = k + O(T)$ stopping times. We demonstrate this on the synchronous BROADCAST from Section 6.3:

LEMMA 9. *With high probability the synchronous broadcast gossip protocol that uses logarithmic size coding coefficients, i.e.,* $\log q = \Omega(\log n)$*, takes at most* $k + O(T)$ *rounds to spread* k *messages, where* $T = \frac{\log(nh)}{h}$ *if the isoperimetric number of the graph* $G(t)$ *is at least* h *at any time* t*.*

For the proof of Lemma 9 we need the following proposition:

PROPOSITION 10. *Let* $X_1, X_2, \ldots, X_l$ *be i.i.d. Bernoulli variables with probability* $P(X_1 = 0) = p \leq \frac{1}{2}$*. The probability that a positively weighted sum of the variables is at most* $\frac{1}{4}$ *its expectation is at most* p*, i.e.:*

$$\forall w_1, \ldots, w_l > 0: \quad P(\sum_j w_j X_j \leq \frac{1}{4}(1-p)\sum_j w_j) \leq p.$$

PROOF. We first scale the weights such that $\sum_j w_j = 1$ and than use the second moment method:

$$\begin{aligned} &P\left(\sum_j w_j X_j \leq \frac{1}{4}(1-p) \right) \\ &= P\left(\sum_j w_j(1-X_j) - p\sum_j w_j \geq 1 - \frac{1}{4}(1-p) - p \right) \\ &= P\left(\left(\sum_j w_j(1-X_j) - p\sum_j w_j \right)^2 \geq \frac{9}{16}(1-p)^2 \right) \end{aligned}$$

Now the left-hand side is the variance of a weighted sum of i.i.d. Bernoulli variables with probability $1-p$, and as such its expectation is exactly $\sum_j w_j^2(1-p)p$. Using Markov's inequality on this expectation, we get that the probability we want to bound is at most:

$$\begin{aligned} \left(\sum_j w_j^2(1-p)p \right) \left(\frac{9}{16}(1-p)^2 \right)^{-1} &= \frac{16}{9}\frac{p}{1-p}\sum_j w_j^2 \\ &\leq \frac{16}{9} \cdot 2p \cdot 1/4 \leq p. \end{aligned}$$

The last transformation holds because $1-p \geq 1/2$ and because we can assume that all weights are at most $1/4$. This is true because if there is a $w_i \geq 1/4$ then already $X_i = 1$ leads to an outcome of at least one fourth of the expectation and the probability for this to happen is p. □

PROOF OF LEMMA 9 (SKETCH). We modify the proof of Lemma 7 only in the way that we use a slightly more precise tail bound, namely that the probability that after $t = k + O(T)$ independent trials there are less than T successes is at most p^k, where p is the failure probability (as long as $-\log p \geq \Omega(\log t)$). To see this, we pick $t = k - (T+1)\log t/\log p + T$ and get that

$$p^k = p^{t-T}t^{T+1} > \sum_{i=t-T}^{t} \binom{t}{t-i} p^i (1-p)^{t-i},$$

which is exactly the probability for having at least $t-T$ failures in t rounds.

We keep the same definition of success as in the proof of Lemma 7 but prove that the success probability of a round is at least $1 - 1/q$, instead of $1/4$ as in Lemma 7:

If only one node is supposed to learn for a success, this is again clear by Lemma 2. If at least $\lceil i \rceil$ nodes are needed to a success, we know also, by the definition of a success, that at least $4\lceil i \rceil$ nodes that do not know $\vec{\mu}$ are connected to a node that knows about it. We assign each ignorant node to exactly one node that knows about $\vec{\mu}$, breaking ties arbitrarily. Now, according to Lemma 2, with probability $1 - 1/q$ each such node independently sends out a message that is not perpendicular to $\vec{\mu}$. In this case all ignorant nodes connected to it learn $\vec{\mu}$. We can now directly apply Lemma 10 and obtain that we indeed have a success probability of at least $1 - 1/q$ per round. This finishes the proof. □

Another interesting case, that was considered in [10], is the Rumor Mongering process from Section 6.1. The authors of [10] give a theoretical analysis in the regime where the $O(k)$ term clearly dominates and prove an upper bound of $3.46k$ for the PUSH protocol and $5.96k$ for the PULL model. They also simulated the protocol and estimated the stopping time to be $1.5k + \log_2 n$. Both their analytic bounds and the simulation assume that messages start out in separate nodes and are equally spread over the network (see also Section 7.1).

The projection analysis technique can be used to prove an upper bound of $1.82462135k$ for the PULL model (even for $q = 2$) starting from a worst-case initialization. On the other hand, we have a lower bound showing that a leading constant of $1.58197671k$ is best possible in this case. Interestingly, if one assumes that messages start out in separate nodes, we can use the techniques from Section 7.1 to prove that the PULL model achieves the optimal constant 1. More extensive simulation results than the ones in [10] confirm that the constant for the dependency on k should indeed be smaller than the projected $1.5k$.

7.3 Asynchronous Single Transfer Protocols with Few Messages

Section 6.2 proves convergence times for spreading $k = n$ messages using the asynchronous single transfer protocols. These bounds are tight and directly extend to a $\Theta(\frac{k}{\gamma})$ bound for $k = \Omega(n)$ messages. In what follows, we want to generalize this to smaller number of messages and discuss the bounds that can be obtained using the projection analysis technique.

For small number of messages, e.g., $k = 1$, the convergence time of RLNC single transfer gossip can be much faster than $O(\frac{n}{\gamma})$ but still be $\omega(\frac{k}{\gamma})$. This shows that the min-cut γ is not the right quantity to look at in this scenario. Again, as in Section 6.3, conductance quantities capture much better how fast a small number of messages spreads. The quantity we consider is:

$$\lambda(G) = \min_{S \subset V} \frac{\sum_{e \in \mathbf{out}(S)} p_e}{\min(|S|, |\overline{S}|)}$$

It is easy to see that it takes at most $T = O(\frac{\log n}{\lambda})$ time for one message to spread if the conductance is bounded by λ at every time t. This is a tight bound for many regular graphs and gives, e.g., a convergence time of $\Theta(n \log n)$ for the complete graph or any other regular expanders. The other lower bound that kicks in for large enough k is the $\Omega(\frac{k}{\gamma})$ lower bound from Lemma 6.

Similar to the results for the other models we can show that essentially the total running time is either dominated by the $T = \frac{\log n}{\lambda}$ rounds to spread one message or for larger number of messages k the $O(\frac{k}{\gamma})$ rounds coming from the communication lower bound that the k messages have to cross the worst case cut. The exact statement that is proven in the full version of this paper has an additional $\log n$ factor and is also obtained by a simple analysis of the tail probability for the spreading time of vector $\vec{\mu}$. This probability is dominated by a sum of negative binomial random variables with increasing success probability and we suspect that using a tighter tail bound for this sum should get rid of the extra $\log n$-factor:

LEMMA 11. *With high probability disseminating k messages in the asynchronous single transfer model with $q = 2$ takes at most $t = O(\frac{k}{\gamma} + \frac{\log^2 n}{\lambda})$ rounds, if the graph G has a min-cut of at most γ and a conductance of at least λ at all times t.*

7.4 Weaker Connectivity Requirements

The idea behind proving performances in the extremely strong and pessimistic adaptive adversarial model used in this paper is that the guarantees directly extend to the widest possible range of dynamic networks, including random models. Most of our proofs, like the ones of Lemma 5, 7 or 8, demand that the network graph $G(t)$ has a certain connectivity requirement at any time t. These requirements might be slightly too strong for random network models. In the following we briefly discuss how these requirements can be easily weakened in many ways:

The simple fact that no progress in the spreading of knowledge gets lost, makes it easy to deal with the case that the connectivity fluctuates (e.g., randomly). Increasing the stopping time by a constant factor easily accounts for models in which the desired connectivity occurs only occasionally or with constant probability. Looking at the average connectivity is another possibility. It is furthermore not necessary to require the entire graph to be expanding on average but it suffices to demand that each subset expands on average with constant probability according to its size. This way, convergence can be proven even for always disconnected graphs (see also [19]). In many models it is furthermore possible and helpful to consider the union of the network graphs of consecutive rounds, i.e., $G'(t) = G(3t') \cup G(3t'+1) \cup G(3t'+2)$.

We demonstrate the usefulness of these ideas on a small example, namely, an alternative way to prove Lemma 4: Instead of analyzing the random phone call model as a gossip process on the complete graph one can alternatively see it as a synchronous BROADCAST on a random network. The network graph $G(t)$ is in this case simply formed by a random directed in-edge, directed out-edge or undirected edge at each node, depending on whether the PUSH, PULL or EXCHANGE model is used. For these random topologies $G(t)$ Lemma 7 will not directly give any stopping time bound, simply because the network graph $G(t)$ is with high probability disconnected. Using either of the two ideas introduced above solves this problem: with constant probability every set has a constant expansion; alternatively one can use that the union of a constant number of rounds forms most likely with an expander.

8. CONCLUSIONS

We introduced the projection analysis technique as a new way to analyze (gossip) protocols based on random linear network coding. Our analysis drastically simplifies, extends and strengthens previous results. In all settings considered in this paper we prove that the RLNC gossip spreads messages in a perfectly pipelined manner and stops with high probability in optimal time. As most notable extension all our results hold in highly dynamic networks that are controlled by a fully adaptive adversary.

Theorem 3 gives a direct way to transfer results for a single-message gossip process to the corresponding multi-message RLNC gossip process, given that strong enough tail bounds are provided. One candidate for which this could be [5], which can be interpreted as giving bounds on a synchronous single transfer gossip for one message. Other interesting candidates are the stationary Markovian evolving graphs studied in [7].

The projection analysis technique is quite simple and flexible and we expect that it has more applications and extensions. Since the initial submission of this paper the author, in collaboration with Karger, Médard, and Kim, demonstrated some of these: among other things, [17] shows that RLNC, with a large enough q, continues to work optimally against omniscient adversaries. This also leads to a (non-uniform) derandomization. [19] uses the projection technique to show that optimal stopping times are preserved if nodes keep only one instead of all packets for further coding purposes. Lastly, [18] shows, using techniques based on [21], that in any model with an oblivious adversary RLNC stops in exactly optimal time, i.e., when there is enough connectivity in the time-expanded graph. This implies that the projection technique can be seen as proving tight bounds on the connectivity of many network models, an important but usually very challenging task.

Acknowledgments

The author wants to thank Jon Kelner for his support and helpful discussions. He also wants to thank David Karger, Chen Avin, Muriel Médard and the anonymous reviewers.

9. REFERENCES

[1] D. Agrawal, A. El Abbadi, and R. C. Steinke. Epidemic algorithms in replicated databases (extended abstract). In *Proc. of the 16th Symposium on Principles of Database Systems (PODS)*, pages 161–172, 1997.

[2] R. Ahlswede, N. Cai, S. Li, and R. Yeung. Network information flow. *Transactions on Information Theory (TransInf)*, 46(4):1204–1216, 2000.

[3] J. Aspnes and E. Ruppert. An introduction to population protocols. *Middleware for Network Eccentric and Mobile Applications*, pages 97–120, 2009.

[4] M. Borokhovich, C. Avin, and Z. Lotker. Tight Bounds for Algebraic Gossip on Graphs. In *Proc. of the Int'l Symposium on Information Theory (ISIT)*, pages 1758–1762, 2010.

[5] F. Chierichetti, S. Lattanzi, and A. Panconesi. Almost tight bounds for rumour spreading with conductance. In *Proc. of the 42nd Symposium on Theory of Computing (STOC)*, pages 399–408, 2010.

[6] P. Chou, Y. Wu, and K. Jain. Practical network coding. In *Proc. of the 41st Allerton Conference on Communication Control and Computing*, pages 40–49, 2003.

[7] A. Clementi, A. Monti, F. Pasquale, and R. Silvestri. Information spreading in stationary Markovian evolving graphs. In *Proc. of the Int'l Symposium on Parallel & Distributed Processing (IPDPS)*, pages 1–12, 2009.

[8] S. Deb and M. Médard. Algebraic gossip: a network coding approach to optimal multiple rumor mongering. In *Proc. of the 42rd Allerton Conference on Communication, Control, and Computing*, 2004.

[9] S. Deb, M. Médard, and C. Choute. On random network coding based information dissemination. In *Proc. of the Int'l Symposium on Information Theory (ISIT)*, pages 278 –282, 2005.

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

[11] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proc. of the 6th Symposium on Principles of Distributed Computing (PODC)*, pages 1–12, 1987.

[12] Y. Fernandess and D. Malkhi. On collaborative content distribution using multi-message gossip. *Journal of Parallel and Distributed Computing*, 67(12):1232–1239, 2007.

[13] C. Fragouli, J. Le Boudec, and J. Widmer. Network Coding: An Instant Primer. *Computer Communication Review*, 36(1):63, 2006.

[14] C. Fragouli, J. Widmer, and J.-Y. Le Boudec. Efficient broadcasting using network coding. *Transactions on Networking (TON)*, 16(2):450–463, 2008.

[15] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *Proc. of the 24th Int'l Conference on Computer Communications (INFOCOM)*, volume 4, pages 2235–2245, 2005.

[16] Z. J. Haas, J. Y. Halpern, and L. Li. Gossip-based ad hoc routing. *Transactions on Networking (TON)*, 14(3):479–491, 2006.

[17] B. Haeupler and D. Karger. Faster Information Dissemination in Dynamic Networks. In *Proc. of the 30th Symposium on Principles of Distributed Computing (PODC)*, 2011.

[18] B. Haeupler, M. Kim, and M. Médard. Optimality of Network Coding in Packet Networks. *ArXiv:1102.3569*, 2011.

[19] B. Haeupler and M. Médard. One Packet Suffices - Highly Efficient Packetized Network Coding With Finite Memory. *ArXiv:1102.3204*, 2011.

[20] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18:319–349, 1988.

[21] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *Transactions on Information Theory (TransInf)*, 52(10):4413–4430, 2006.

[22] R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking. Randomized rumor spreading. In *Proc. of the 41st Symposium on Foundations of Computer Science (FOCS)*, pages 565–574, 2000.

[23] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. XORs in the air: practical wireless network coding. *Transactions on Networking (TON)*, 16(3):497–510, 2008.

[24] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. of the 44th Symposium on Foundations of Computer Science (FOCS)*, pages 482–491, 2003.

[25] D. Kempe and J. Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In *Proc. of the 43rd Symposium on Foundations of Computer Science (FOCS)*, pages 471–480, 2002.

[26] D. Kempe, J. Kleinberg, and A. Demers. Spatial gossip and resource location protocols. *Journal of the ACM (JACM)*, 51(6):943–967, 2004.

[27] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proc. of the 42nd Symposium on Theory of Computing (STOC)*, pages 557–570, 2010.

[28] S. Li, R. Yeung, and N. Cai. Linear network coding. *Transactions on Information Theory (TransInf)*, 49(2):371–381, 2003.

[29] Y. Minski. *Spreading rumors cheaply, quickly, and reliably.* PhD thesis, Cornell University, 2002.

[30] D. Mosk-Aoyama and D. Shah. Information dissemination via network coding. In *Proc. of the Int'l Symposium on Information Theory (ISIT)*, pages 1748–1752, 2006.

[31] D. Vasudevan and S. Kudekar. Algebraic gossip on Arbitrary Networks. *ArXiv:0901.1444*, 2009.