Robust Local Algorithms for Communication and Stability in Distributed Networks

Doctoral Thesis Proposal

Scott Roche *

Abstract In a world in which our technological infrastructure is increasingly reliant on platforms that are distributed in nature, there is a substantial need for distributed algorithms and network designs that are able to perform a wide range of tasks as or more efficiently than their non-distributed counterparts. Furthermore, these algorithms and networks must be resilient to changes and disruptions in the network. Arguably the best way to do this is to design algorithms that are decentralized. By exchanging messages and data with immediate neighbors and running other subprocesses locally (often involving some sort of random choice), a coordinated solution to problem still emerges from the system.

The work for my doctoral thesis in this proposal document focuses on two aspects of distributed algorithm design. Both involve a fully decentralized system running algorithms or processes locally, with limited system-wide knowledge. The first research area involves a protocol for information spreading with potential application to the study of epidemics on networks. We introduce a process which we call coalescing-branching random walks, that combine aspects of branching systems and coalescing particle systems and study the process on various types of finite graphs. Specifically, we study the length of time it takes for every node in a network to be visited by the walk at least once and provide results for various types of graphs, including expanders, trees, grids, and cliques.

The second branch of work for my thesis involves designing a distributed algorithm which maintains the expander topology of a network in the presence of a high degree of adversarial deletion and insertion of nodes. This algorithm makes heavy use of random walks as a source of randomness when choosing new connections to make between nodes of the network. We show that this algorithm can withstand up to $O(n/\text{polylog}(n))$ adversarial node deletions/additions *per round* while maintaining a network that is an almost-everywhere bounded-degree expander graph in each round. Furthermore, each node need only send a polylogarithmic (in n) number of bits in each round.

*College of Computer and Information Science, Northeastern University Boston, MA 02115, USA. Email: {str7}@cornell.edu.

1 Introduction

In a world in which our technological infrastructure is increasingly reliant on platforms that are distributed in nature (from the local, e.g. cluster or data center level, to the geographically diffuse, e.g. P2P systems, ad hoc or mobile networks, the Akamai network), there is substantial need for distributed algorithms and network designs that are able to perform a wide range of tasks as or more efficiently than their non-distributed counterparts. One of most attractive features of distributed systems is resilience to disruption, whether due to intentional causes (coordinated attacks or government interference) or other "natural" causes (changes in internet topology, damage to the physical infrastructure of the internet, congestion, etc.). In addition to being lightweight and efficient, these systems and the algorithms that run on them need to be able to adapt to a rapidly changing, possibly adversarial environment. Arguably the best way to do this is to design algorithms that are decentralized (one could also say local). What is meant by "local" is that the algorithm runs independently on each unit/computer in the system. This copy of the algorithm generally knows very little about the system as a whole (its topology, which computers hold which information, etc.). By exchanging messages and data with immediate neighbors and running other subprocesses locally (often involving some sort of random choice), a coordinated solution to problem still emerges from the system.

The work for my doctoral thesis in this proposal document focuses on two aspects of distributed algorithm design. Both involve a fully decentralized system running algorithms or processes locally, with limited system-wide knowledge. The first research area involves a protocol for information spreading with potential application to the study of epidemics on networks, while the second branch of work for my thesis involves designing a distributed algorithm which maintains the expander topology of a network in the presence of a high degree of adversarial deletion and insertion of nodes.

1.1 Coalescing-branching random walks: An information spreading protocol

An information spreading process in this context means an algorithm or mechanism that allows a piece of information, rumor, or infection at one node to be disseminated to all other nodes in the network through copying and forwarding of the information. There are a number of well-studied information spreading mechanisms, each with its own advantages and disadvantages. Here we propose a new mechanism based on a pairing of coalescing and branching processes. In this process. we start with one node holding a token. This token splits into two tokens (or in general, k tokens) and each token then picks a neighbor node independently and uniformly at random and moves to that neighbor. If more than one token lands at the same node, the two coalesce. This process continues for each token in all subsequent steps. Due to its resemblance to a random walk (and identity with the random walk when $k = 1$), we call this process a coalescing-branching random walk, or cobra walk.

The bulk of the work for my thesis explores the properties of cobra walks. In work already completed, we analyze the cover time of a cobra walk on various graph topologies. Cover times (the time until all nodes have been visited by the cobra walk at least once) play an important role in the understanding of random walks on graphs, and we extend this to cobra walks. Most of the remaining work I propose for my thesis also focuses on cobra walks. The primary goal is to find an upper bound on the cobra walk cover time for arbitrary graphs (which we conjecture to be $O(\log n)$). I propose several methods of attacking this problem, the most interesting (and difficult) of which has the potential to serve as a general model for a class of processes subsuming cobra walks, multiple random walks, or rumor spreading. In addition, I propose to explore the relationship between cobra walks and SIS epidemic processes further. There is much room in this area for progress. One major goal is to better understand the relationship between the topology of a graph and the long term behavior of an epidemic: does it die out quickly or persist for a long time. Finally, there is also opportunity to perform some simulation-based experiments probing various questions relating to epidemics through the lens of cobra walks.

1.2 Maintaining a self-healing expander topology in a highly dynamic network

The second branch of work for my thesis involves designing a distributed algorithm which maintains the expander topology of a network in the presence of a high degree of adversarial deletion and insertion of nodes. In this context, expansion refers to the number of edges that leave a subset of nodes of a network: a network with expander topology is one in which all sets have a content fraction of their edges leaving the set. In the domain of network construction, a graph with good expansion is considered a proxy for a well-connected network robust to the deletion of edges.

A common model for a distributed network is one in which nodes (i.e. individual computers) are constantly leaving and new ones are joining. Any system built on such a platform needs to be able to handle this *churn*: the underlying network topology must remain stable, essential communication functions must not be disrupted, data being stored in a distributed fashion on the network must not be deleted, and any computational tasks must continue to execute correctly. There is a large body of research investing the properties of dynamic networks (especially w.r.t. critical metrics of the network such as expansion or connectivity), and considerable work has been done in designing distributed algorithms that will operate in such a dynamic environment. More limited, but still strong, is the research done on algorithms that build, maintain, or heal a network being subjected to various kinds of disruption (e.g. adversarial vs. random, node churn vs. edge changes only, etc.).

In a series of papers [6, 5, 4] over the last several years, several authors and others have developed a toolkit of algorithms that perform important computational tasks on networks being subjected to a large churn of nodes, $O(n/\log^k n)$ nodes churned in every round, for some constant k. The churn is controlled by an oblivious adversary, and in some cases the allowance is made for the presence of Byzantine nodes in the network. An essential primitive of these algorithms is that large numbers of tokens are allowed to take a random walk through the graph – these tokens serve as a source of randomness for use in the algorithms in these papers.

However, there is one large missing link. Each of these algorithms assumes that the adversary will churn a large number of nodes in each round yet still maintain the network with an expander topology. Naturally, we have no reason to believe that an adversary in the real world would be so generous. What is needed, then, is an algorithm that runs in the background on the nodes of the network and which will, in every round, maintain the topology of a large portion of the network (ideally $n - o(n)$ of the nodes) as a bounded-degree expander sub-graph. We have designed such a distributed protocol for maintaining an expander topology.

This algorithm is (to date) one of the most promising algorithms for repairing and maintaining a network with a specified topology, in terms of the number of perturbations that are allowed to occur, the frequency with which they can occur, and the speed with which the network is repaired. The exciting consequence of this result is that there are a host of distributed algorithms for various tasks (search and storage, byzantine agreement, leader election) that operate on a graph with significant adversarial modification in every round.

1.3 Informal Description of a Distributed System

We can informally model a distributed network as follows: we are given a system of computers, V, which we will refer to interchangeably as **nodes** or **vertices**, and a set E of pairwise connections (u, v), where $u, v \in V$, which we refer to as the **edges**. Thus we model a system by graph $G = (V, E)$. A network can represent direct, physical connections (servers in a data center connected via an internal LAN), some sort of overlay network on top of the Internet, or even a model of some abstract or virtual relationships between arbitrary objects. Each node v is a separate computing unit, and is allowed some reasonable amount of memory and computing power, depending on the application. Let us also assume that v has limited information about the network it belongs to. It knows to which nodes it is directly connected (via an edge),

and that it knows the size of the network, n (or has a way to make a reasonably accurate guess). Node v does **not** know the overall topology of the network, no, in general, what data is stored at other nodes.

1.4 Completed Work

Here I provide a brief summary of the status of the work that has already been done. For cobra walks, we have already proven upper bounds on the cover time of a cobra walk on various graph topologies: trees, grids, the complete graph, and grids. Please see **??** for our results on the cover time of cobra walks and the techniques we developed for the analysis. For the distributed expander maintenance protocol, essentially all of the work has already been completed. We have devised the algorithm and provided a detailed proof of its accuracy. For inclusion in my final thesis, I intent to rewrite the paper to improve the narrative and flow of the analysis.

1.5 Proposed Work

I propose to devote the rest of my thesis work on further study of cobra walks. There are several promising techniques that may be of use in proving the conjectured $O(\log n)$ cover time of a cobra walk on an arbitrary graph. Furthermore, one of these techniques may provide a more general framework for studying a family of processes that include cobra walks and gossip protocols. Finally, on a separate track, I would like to devote some attention to the epidemiological side of cobra walks and make some contribution to the understanding the impact of a network's structure on the spread and duration of an infection.

1.6 Timeline

Finally, I present the following timeline for the completion of the proposed work.

Objective	**Target Completion Date**
Proposal	End of **February**
Research	End of **June**
Writing	Mid to End **August**
Defense	**September**

2 Coalescing-Branching Random Walks on Finite Graphs

In this section I provide a brief review of information spreading in graphs, give a detailed explanation of the cobra walk protocol, outline some of the potential applications of cobra walks, and describe the work already completed on cobra walks.

2.1 Background

One of the most important primitives of a distributed system is information dissemination. That is, every node must have some mechanism to inform other nodes (usually **all** other nodes) of relevant information to the application at hand. However, as stated earlier, each node only knows how to directly message its immediate neighbors, and does not know the addresses of other nodes in the network. There are a number of different message- passing, information spreading protocols that can be used to do this. The most basic (found in the LOCAL model communication [49]) involves a node sending the same message to all of its neighbors. This is known as flooding, and while fast (assuming a stable network), flooding is not an efficient process because the message complexity (total number of messages sent) can be extremely high. In addition

to flooding, there are also structured, pre-set broadcast protocols – these, however, have the disadvantage of requiring a stable network with known topology [32]. Since flooding or predetermined broadcast protocols violate the spirit of lightweight, robust, decentralized communication protocols, alternatives needed to be developed.

The most robust of these alternatives, which has received the bulk of the research attention, are what are known as gossip or rumor-spreading protocols. The essential idea behind gossip is as follows: a node v of G starts with a rumor or piece of information (which we will sometimes refer to as a pebble or token). The goal is to distribute this rumor to all other nodes of G (whether they need it or not). There are many variants of gossip, the most basic of which is PUSH. In the first round, the origin node picks one of its neighbors and distributes a copy of that rumor to that neighbor. The choice of the neighbor, in the most basic version, is made uniformly at random from $N(v)$, the set of neighbors of v. In the next and all subsequent steps, every node that holds a copy of the rumor picks a neighbor $u.a.r$ and distributes a copy of the rumor. Note that it is possible that this neighbor already has the rumor. Another variant of the gossip protocol is PULL, which can be viewed as the complement of PUSH. In this version, every node v that does **not** have a copy of the rumor randomly selects a neighbor, and if that neighbor has a copy of the rumor, v "pulls" the rumor from the selected neighbor. Another common variant of gossip which is used for most of the analysis of rumor-spreading is PUSH-PULL, which is a combination of the two just described. Any node that holds the rumor operates in PUSH mode, while any node that does not hold the rumor operates in PULL mode.

One of the most important questions studied in gossip problems is the time it takes for every node to receive a copy of the rumor (also known as becoming informed). Clearly this is an essential property to know if one wants to use gossip as an effective information spreading communication protocol. A chain of papers [13, 14] improved the bounds on this process, with [28] providing a tight bound for the time until all nodes are informed via PUSH-PULL. The bound is $O(\phi^{-1} \log n)$ for all graphs, and there exists graphs for which $\Omega(\phi^{-1} \log n)$ rounds are required. Here ϕ is the **conductance** of the graph, which is defined as:

$$\phi = \min_{S \subseteq V(S), vol(S) \leq |E|} \frac{|cut(S, V \setminus S)|}{vol(S)} \tag{1}$$

where $cut(V, S)$ is the set of edges crossing the partition $(S, V \setminus S)$ and $vol(S)$ is the sum of the degrees of the vertices of S. Non-uniform random gossip protocols have also been considered (non-uniform in the selection probability of a neighbor of a node performing PUSH or PULL). In graphs where the conductance ϕ is low and there exist bottlenecks (for example a barbell graph), [12] provides an $O(\log^3 n)$ algorithm for the 1-local broadcast problem. (In the k-local broadcast problem, every $v \in V$ wants to exchange messages with all vertices within distance k. Hence the 1-local broadcast problem operates only on the immediate neighbors of vertex v.) Repeatedly applying this algorithm allows for global rumor spreading with no dependence on the conductance of the graph. Moving beyond algorithms that rely on randomness, [32] provided the first deterministic algorithm for solving the k-local broadcast problem in $2(k + \log n)$ rounds. Thus for the n-local problem (every node whats to inform all other nodes in G), this provides a coverage bound for **any** graph of $O(n \log n)$.

With rumor spreading "in the bag" so to speak, the search was on for other lightweight and fast information spreading communication protocols. But why would one need an alternative to rumor-spreading? The first reason involves the number of messages sent. In the case of gossip protocols, particularly PUSH-PULL, every node is participating in every round, regardless of whether it has no new rumors to share. It would be nice to have a protocol that does not involve such high overhead. Furthermore, in many real-world applications, the underlying network topology is dynamic. Such changes may even be under the influence of an adversary. In fact, [29] showed that even with graphs of regular vertex degree and high vertex expansion, an adaptive adversary can construct a sequence of regular graphs with high vertex expansion such that PUSH-PULL takes a polynomial (in n) number of rounds. It would be ideal to have a protocol that works even under adversarial dynamic settings.

Since many of the protocols we have already discussed involve the use of randomness in the selection of neighbors to move the rumor to, it makes sense to extend this idea and consider techniques based on random walks as a potential source of information spreading protocols. Indeed, random walks already have many applications in distributed computing. A subset of these applications includes search [56, 1, 30], load balancing [34], small world routing [35], information gathering and propagation [10], distributed construction of expander networks [37], and many others. The most basic of such possibilities is a simple random walk. However, a simple random walk has such a high latency for such metrics as cover time $O(n^3)$ in arbitrary graphs [25], a bound which is tight for some graphs, that using a random walk for a fast communication protocol is infeasible. Therefore, considerable effort has been made to "speed up" random walks. Alon et al. [2] use multiple, independent random walks to speed up cover time, with results improved upon in [23]. Modifications to random walks have also been made in which a walk tries to avoid traversing the same edge [9], a random walk is constructed in a distributed fashion [19], or through neighborhood exploration [8]

2.2 Coalescing-branching random walks

One natural extension of the random walk is inspired by mathematical models of epidemics on graphs. In this process, we start with a single token at a single node. In the first step of the process, this token becomes k copies of itself (i.e. where there was one token, there are now k). Each of these tokens then independently, uniformly at random chooses a neighbor of v to move to. If more than one token lands at the same vertex, those tokens coalesce into a single token. This continues independently, with each step involving a k-branching, single step walk for each newly generated pebble, and a coalescing. We have named this coalescing-branching, or cobra, walk. Cobra walks bear a resemblance to SIS-type epidemic models (the SIS stands for susceptible-infected-susceptible), in which an epidemic moves from infected to susceptible nodes, and the infected nodes recover and become susceptible again. (We will have more to say about this later). Cobra walks can also be viewed as a generalization of a random walk, where when $k = 1$ the cobra walk reduces exactly to a simple random walk.

There are several reasons why a cobra-walk may make a good candidate for a distributed information spreading protocol. Like gossip-based protocols, cobra walks have the property that each node acts independently and follows a simple set of instructions that do not rely on knowledge of the rest of the network (except for the immediate neighborhood). They also share the property that the number of rumor-containing tokens increases in expectation, often dramatically, thus seeming to provide a speed-up in the amount of time required to inform all nodes in the network. On the other hand, we can hypothesize that a cobra walk may be asymptotically more efficient than rumor spreading with respect to the total number of messages sent. A potential reason for this as follows: in gossip-based mechanisms, even after a node has informed all of its neighbors, it continues to broadcast the rumor, leading to unnecessary messages sent. With a cobra walk, on the other hand, when a node is "occupied" by the rumor, it informs somewhere between 1 and k neighbors and then is gone. If this happens often enough, we essential have the same effect as a gossip protocol (at least asymptotically in terms of cover time), but end up sending a far smaller number of messages in total. One can see this on the star graph, in which we have 1 central node connected to $n-1$ leaf nodes. No matter what node we start the process from, both the cobra walk and the PUSH process require $O(n \log n)$ rounds to complete (via a coupon-collector process). However, note that in the PUSH version, once a leaf node is informed, it continues to broadcast Thus the cobra walk will send $O(n \log n)$ messages, while PUSH will send in total $O(n^2)$ messages.

Of course what one potentially gains in message complexity one may lose in analytical tractability. While for gossip protocols the set of actives nodes (those nodes holding a token) is either monotonically non-decreasing or even increasing, the same is not true for cobra walks. Furthermore, in a gossip process the set of active nodes and the set of informed nodes is identical, in a cobra walk there is a difference. The

set of active nodes is not necessarily increasing or even non-decreasing. Despite this, we need to show that the set of informed nodes is growing (at least for some initial phase) and achieves a fast cover time (which we will more rigorously define in the next section). This is one of the most interesting facets of cobra walks. Whereas for gossip-based protocols it seems intuitively true that coverage will be fairly rapid due to the monotonically non-decreasing active set, it's not entirely clear that this would also be true for cobra-walks. Clearly, a cobra walk is no slower than a simple random walk, but it is a bit surprising that it would not lie somewhere in between gossip and the simple random walk.

The two major analytical challenges of cobra walks both involve keeping track of pebbles. As mentioned above, cobra walks have multiple active nodes at any given time. For most most graph topologies, it is difficult to compute and characterize the distribution of the pebbles at any arbitrary time t in the future, based merely on an initial distribution. That is to say, it differs from the simple random walk (or even multiple parallel random walks) in that that one can't use the tools of Markov chains (namely the transition matrix). While we will show later that a cobra walk is indeed a Markov chain, its state space is so complicated that it seems virtually impossible to make use of this fact. The second major challenge is that the branching and coalescing of the pebbles in the walk create a non-trivial level of dependence among the active nodes. Thus it would be extremely challenging to calculate the probability that a given node is active more than one step in the future.

2.3 Applications and Extensions of Cobra Walks

Tensor Product Chains
The challenges involved in analyzing cobra walks necessitate the invention of novel techniques that are interesting in their own right. Furthermore, some of the substeps required to proceed with the analysis also yield general results that may be of independent interest. Even more exciting, however, is the prospect that analyzing cobra walks will serve as the gateway to two interesting topics. In the proof of the cover time of cobra walks on expander graphs, we make use of what is known as the **tensor product** (c.f. [16, 39], which can roughly described as a graph which represents the process of multiple pebbles moving around copies of the original graph G. While the tensor product we study is relatively easy to analyze, the method may serve as a blueprint for creating the representation of a cobra walk on graph G as a Markov chain on a modification of the tensor product of G. In this Markov Chain, the state space is comprised of all subsets of vertex set V, where each subset represents a potential active set. Transitions between states (and their corresponding probabilities) exist if one active set S_t in the cobra walk at time t can lead to the active set S_{t+1} at time $t+1$. When generalized further, it is possible to view many of the protocols we discussed earlier as variants of the tensor product Markov chain, with modifications to the state space or the transitions of the chain. For example, we can represent gossip protocols with the same state space but remove all the transitions which map back to decreasing the size of the active/informed set. With further study (which I will outline in the Proposed Work section) it may be possible to develop a "unified theory" of information spreading processes, of which cobra walks, gossip, and multiple random walks fall out as examples.

Epidemics and the SIS model A perhaps less theoretical use of cobra walks is as a method of modeling epidemics. While mathematical models of epidemics have been around for well over a century(see description in [3], it is only within the last fifteen to twenty years have researchers begun to study epidemics that occur on networks [50, 44, 22]. In this approach, which much more closely models how epidemics spread *in situ*, individual organisms are modeled as nodes in what is referred to as the contact network. Connections between individuals are represented by edges, and the existence of an edge depends on the pathogen under consideration. For example, an edge would imply casual contact for a rhinovirus, close family or caregiver contact for ebola, sexual contact for HIV (cf. [40, 55, 47, 42]). In a typical model setup, we begin with an network of individuals connected with some specified topology. We then begin the infection at some

node or some subset of nodes. Time is usually treated as a continuous variable, and an infection at a node is governed by two (independent) Poisson processes: the infection rate (per each edge) with parameter β, and the cure rate (per each node) with parameter δ. Note that here we are discussing what is known as an SIS model, for "susceptible infected susceptible", which which a node can either be in the infected or susceptible but uninfected state, and can transition between the two arbitrarily often. This is in contrast to the SIR model, for "susceptible, infected, recovered", in which there are three states. The transitions here are unidirectional: susceptible to infected, infected to recovered. Once a node becomes recovered, in most cases it is considered as removed from the graph's topology for the purposes of the spread of the infection. There is a substantial body of research on SIR models, and I will merely note here there is a large overlap with the field of bond percolation – in fact, many useful results for SIR models have been found by analyzing a SIR model as a percolation process [44]. For SIS models, there are many variations on the process [18]. I will only provide a few here as examples: the link infection rate may depend on the number of neighbors, there may be an incubation period, or the curing process for an infected node may have a minimum fixed duration. Essentially, if there is some real world feature of an epidemic that can be inserted into a model, it most likely has been.

While the SIS network model is fairly simple to describe and conceptualize, the analysis of it is another story. Almost all work to date involves some sort of mean-field approximation, whether at the level of the entire network (i.e. we assume the graph is a clique) all the way to the state transition of a single node (between infected and susceptible). Perhaps one of the best results thus far comes from [27]. In this work, the authors study the impact that the network's topology has on the duration of the epidemic. Note that for any finite graph, in an SIS model as described above, there is a single absorbing state which corresponds to the infection going extinct. However, prior to this occurring, the epidemic may stick around for an extremely long time (exponential in the size of the network). Indeed, in [27], the authors establish two bounds on the duration of an epidemic. They show that, on one hand, if the spectral radius (i.e. the dominant eigenvalue of the adjacency matrix A of the graph) is $\rho(A) < 1/\beta$, where β is the infection rate parameter, then the epidemic dies out quickly, in $O(\log n)$ time. For a bound on exponentially long persistence, they make use of a generalized isoperimetric constant:

$$\eta(G, m) = \inf_{S \subset \{1,\ldots,n\}, |S| \leq m} \frac{E(S, \bar{S})}{|S|}, 0 < m \leq \lfloor n/2 \rfloor. \tag{2}$$

and show that if $\frac{1}{\beta\eta(m)} < 1$, then the epidemic will last for $\Omega(e^{n^\alpha})$ length of time, in expectation, for some $\alpha > 0$. There is quite a late gap between these two bounds, especially coming from the perspective of percolation, wherein one might expect there is some fairly sharp threshold between the two behaviors. There are some graphs (such as the hypercube) for which the two conditions are very close, but for arbitrary graphs there is still a lot of opportunity to narrow the gap.

It should be fairly obvious that cobra walks present a different sort of approximation for an SIS model. The main differences are that cobra walks operate in discrete time rather than continuous time, the epidemic can never die out (though it can be reduced to only one pebble), and the infection rate, probability distribution, and method of selecting neighbors to spread to are slightly different. However, cobra walks provide the opportunity to model an SIS process explicitly (i.e. tracking the progress of the infection at every node across all time steps) rather than through some mean-field approximation. As part of my present work, I will discuss the possibility of modifying the cobra walk to even more closely resemble an SIS epidemic process and provide some questions that might be answerable through the use of (possibly) modified cobra walks.

2.4 Preliminaries: Definitions and Assumptions

Before proceeding with this summary, I would first like to define a few of the terms and concepts that we will use throughout. We have previously described the coalescing-branching process. By **active set** we mean the set S_t of all vertices at time t of graph G that have a pebble. We define $N(S)$ to be the inclusive neighborhood of a set S: that is, the set of all neighbors of the vertices of S, including members of s itself. Define $\Gamma(S)$ to be the non-inclusive neighborhood, consisting of the set of all neighbors of the nodes of S that are **not** themselves members of S.

The expected **maximum hitting time** h_{max} of a cobra walk on G be defined as $\max_{u,v \in V} E[h_{u,v}]$ where $h_{u,v}$ is the random variable representing the time it takes for the first pebble in a cobra walk starting at u to reach v for the first time. Then the quantity τ_v is the random variable representing the minimum time t at which every vertex has belonged to the active set at least once. The **cover time** of a cobra walk on G is $\max_{v \in V} \tau_v$. We define the **expected cover time** to be $\max_{v \in V} E[\tau_v]$. While in the literature on random walks the cover time usually refers to the expected cover time, in this work we show high-probability bounds on the cover time.

In the results dealing with expanders, we need to clarify which definition of expansion we use. While to a certain extent the various definitions are more-or-less interchangeable, to be precise here we will use a spectral definition for expanders and use Tanner's theorem to translate that definition into neighborhood and cut-based notions of expanders.

Definition 1. *A d-regular ϵ-expander is a d-regular graph whose adjacency matrix A has eigenvalues α_i such that $|\alpha_i| \leq \epsilon d$ for $i \geq 2$, where $\epsilon \in (0, 1)$.*

And here is Tanner's theorem:

Theorem 2. *[53] Let G be a d-regular graph ϵ-expander. For all $S \subseteq V$ with $|S| = \delta n$, we have $|N(S)| \geq \dfrac{|S|}{\epsilon^2(1-\delta)+\delta}$.*

2.5 Results

Here I present a quick overview of the major results that have come out of our study of cobra walks to this point.

2.5.1 Matthews bound for cobra walks

One of the more famous results describing the cover time of a simple random walk is known as Matthew's bound [41] , and it relates the cover time to the maximum expected hitting time over all pairs of vertices in the graph. It turns out that there is an analogue for Matthew's bound for cobra walks, which we prove:

Theorem 3. *Let G be a connected graph on n vertices. Let W be a cobra walk on G starting at an arbitrary node v. Then the cover time of W on G, $C(G)$, is bounded from above by $O(h_{max} \log n)$ in expectation and with high probability.*

This version of Matthew's theorem is the key to the first type of technique we use to prove cover time bounds on graphs. If we can calculate h_{max}, we can then bound the cover time. Calculating h_{max} is slightly different for cobra walks than it is for simple random walks. In a walk between u and v, we only need to follow the pebble closest to v at any time step. Pebbles used in consecutive time steps may not even be directly related. Though this technique is used to different effect in grids and trees, the basic idea is the same.

2.5.2 Cover time of trees

We are able to show that any tree of n nodes is covered by a cobra walk starting at any vertex in the tree in $O(n \log n)$ steps. As mentioned above, the proof will follow if we can can establish that a cobra walk starting at node any node u will reach any node v in a linear number of steps (in expectation).

We first show that for any tree T of size n, a cobra walk starting at (an arbitrarily picked) root r of T will return to r (meaning a pebble of the cobra walk will reach r) in expectation in $4n/d$ steps, where d is the degree of the root. This result is proved by showing that if we follow the pebble closest to r in each step, we are mimicking a biased random walk (biased towards the root) and can establish a recurrence relation that yields the linear return time.

Next we show that if we fix a path in the tree, the expected time it takes the cobra walk to take one step along the path can be calculated:

Lemma 4. *Fix a path in a tree T made up of vertices $1, \ldots, l, (l+1), \ldots, t$. Then the expected time it takes for a cobra walk starting at vertex l to get to $l+1$ with at least one token can be bounded as:*

$$h_{l,(l+1)} \leq \frac{5}{4} + \frac{12}{5} \sum_{i=l}^{2} \left(\frac{1}{5}\right)^{l-i} |T_i| \tag{3}$$

The proof again makes use the fact that closest pebble to vertex $l+1$ is taking a biased random walk. If we calculate the probabilities associated with moving to $l-1$, $l+1$, or the (possibly empty) subtree rooted at l we can again establish a recurrence relation to achieve the above expression.

We finally prove the $O(n)$ hitting time by showing that $h_{1,l} \leq h_{1,2} + h_{2,3} + \ldots + h_{t-1,t} \leq 4n$. The cover time is proven by applying Matthew's bound.

2.5.3 Cover time of grids

We are able to establish that for a grid G of dimension d on n nodes, a cobra walk covers G in (almost) root-d of n steps:

Theorem 5. *Let G be a finite d-dimensional grid for some constant d, without wrap-around edges. Then the cover time of a cobra walk on G is $\widetilde{O}(n^{1/d})$ w.h.p. for branching factor $k = 2$.*

The proof of this is a variation on a theme of tracing the closest pebble on a cobra walk starting at $s = (0, \ldots, 0)$ and aiming to reach target $t = (n^{1/d}, \ldots, n^{1/d})$. However, we do not track the pebble that is **globally** closest to t. Rather, we focus on the progress the cobra walk is making one dimension at a time. Let's say we are focusing on coordinate i of a cobra walk's progress. With respect to this coordinate, we do keep track of the pebble closest to t's i^{th} coordinate, and we show that this pebble is making a biased random walk towards that target. After $n^{1/d}$ steps, the pebble is within $n^{1/2d}$ distance from the target. With respect to all other dimensions, while we are following dimension i, the pebble is taking an **unbiased** random walk, and hence its expected drift is zero. We can therefore state with high probability that it does not move more than a distance of $n^{1/2d}$ away from its initial position. We cycle through each of the coordinates and make the same progress. We then repeat this entire process $\log \log n$ times, continually shrinking the distance by another square root factor, until we are within constant distance in each coordinate away from t. Then this final gap will be closed in expectation in a constant number of steps. Thus proving the hitting time, we again apply Matthew's bound to get a cover time.

2.5.4 Cover time of the complete graph

A cobra walk covers a complete graph rapidly, in $O(\log n)$ rounds. This is perhaps not surprising, but we include this result because the proof technique is interesting in its own right, and serves as a sort of warm

up for the proof of the cover time of the expander (as well as a contrast to show where technical challenges arise in the expander).

The main result stated formally:

Theorem 6. *Let $G = K_n$, the complete graph on n nodes. A cobra walk starting from any node in G will cover the entire graph w.h.p. in $O(\log n)$ time.*

The key to understanding the coverage of K_n is to understand what happens at an arbitrary step t, given that the active set is already of size $s = |S_t|$. For simplicity, assume that every node in K_n also has a self loop. Let us also assume that $|S_t| << n$. There are two tokens at each vertex of S_t, so we have $2|S_t|$ tokens choosing from n vertices independently, uniformly at random. It is easy to show via a combinatorial argument paired with a concentration bound that $|S_{t+1}|$ will be of size $(1 + c)|S_t|$ for some constant c with some (overwhelmingly) high probability. Thus, in each step, the active set will grow, meaning after $\log n$ steps we will have δn active nodes for some constant δ. Once we reach this state, simply by δn pebbles randomly sampling from n vertices, we will hit all of the vertices of K_n w.h.p. in another $O(\log n)$ steps.

2.5.5 Cover time of expanders

As hinted at earlier, showing that an expander is covered by a cobra walk in $O(\log n)$ as yet has not been possible. To quote the Bush Administration, I prefer to think of this not as a failure, but as a success that hasn't happened yet. However, we have been able to show that coverage occurs w.h.p. in $O(\log^2 n)$ time. The analysis is broken up into two phases. In the first phase, we show that the cobra walk's active set grows by a multiplicative factor in every round w.h.p. up until the active set reaches a size of δn for some constant δ. This takes $O(\log n)$ time. In the second phase, we need to get a bit more creative. We invent a process called W_{alt} in which the number of pebbles is frozen (at δn). However, the pebbles move around in such a way as that their transition probabilities resemble the transitions of a cobra walk. We show that this process stochastically dominates a cobra walk, and that this process (which we call W_{alt}) covers G in $O(\log^2 n)$ time with high probability.

Thus we have the following two results formally stated, which when put together imply that a cobra walk covers an expander with suitably high expansion (specified in the first lemma below) in $O(\log^2 n)$:

Theorem 7. *Let G be any d-regular ϵ-expander with ϵ, δ not depending on n (number of vertices in G), with $\delta < \frac{1}{2}$, and ϵ, a sufficiently small constant such that*

$$\frac{1}{\epsilon^2(1-\delta)+\delta} > \frac{d(de^{-k} + (k-1)) - \frac{k^2}{2}}{d(e^{-k} + (k-1)) - \frac{k^2}{2}}, \tag{4}$$

then in time $O(\log n)$, w.h.p. a cobra walk on G with branching factor k, will attain an active set of size δn.

We note that the condition in the above theorem is satisfied if either ϵ is sufficiently small, or k is sufficiently large. For instance when $k = 2$, the above condition holds for strong expanders, such as the Ramanujan graphs, which have $\epsilon \leq 2\sqrt{d-1}/d$, and random d-regular graphs, for d sufficiently large.

Theorem 8. *Let G be as above, and let W be a cobra walk on G that at time T has reached an active set of size δn. Then w.h.p in an additional $O(\log^2 n)$ steps every vertex of G will have visited by W at least once.*

The proof of 7 is technically complicated but not intuitively complicated. We argue directly using the expansion of G that the active set initially grows by a constant multiplicative factor in each round in, expectation. Then we use a standard martingale argument to show that for a single step, the growth occurs with high probability. The only challenging part of the proof is that within a window of $O(\log n)$ rounds

there will be $\Omega(\log n)$ growth steps. This is done by showing that the growth is stochastically dominated by a negative binomial random variable representing the size of the active set and applying a concentration bound.

The proof of 8 is far more interesting. As mentioned, rather than work with the cobra walk W directly, we work with a process W_{alt}. In this process, we start with δn arbitrarily distributed pebbles (at most one per vertex). The pebbles then take walks around G, acting almost like independent random walks except in the case that three or more pebbles are co-located at the same vertex at the same time. Then, two of the pebbles choose neighbors to walk to uniformly at random, and the remaining pebbles pick one of these two destinations with probability $1/2$ each. (Do you see the resemblance to the cobra walk?). We show that if we pick any vertex, this vertex will be covered by at least one pebble in $O(\log n)$ time with constant probability. Unlike as would be the case for independent random walks, we can't assume that the stationary distribution of a pebble is $1/n$ for each vertex, and that the walks mix rapidly. Rather, we need to find a way to calculate the stationary distribution directly. To do this we need to calculate the stationary distribution of the joint walk of two pebbles, where we assume one pebble is the "leader" and the other the follower. Thus, we end up analyzing the (weighted, directed) random walk on a graph that is derived from the tensor product of G with itself. Using a result from [15], we are able to show that this walk approaches its limiting distribution rapidly (in logarithmic time), and that this distribution is close to uniform $(1/n^2)$. Thus, we can show that in $O(\log^2 n)$ time every vertex in G is covered by W_{alt}, which carries through to the cobra walk.

3 Enabling Efficient and Robust Distributed Computation in Highly Dynamic Networks

In this section I provide a brief overview of the state of research on algorithms for computing on and repairing distributed networks. I then describe an algorithm which is able to maintain a bounded degree, almost-everywhere expander in the face of $o(n)$ adversarial churn, in every round, with high probability. I then provide a brief sketch of how we prove that this algorithm meets its guarantee.

3.1 Background

In a dynamic network we assume that the structure is capable of changing or being changed in every time step (using a discrete time interpretation). The changes to the network can either be to the nodes of V, the edges E, or to both. There are many different varieties of dynamism. The most basic is to assume some sort of randomness in the rearrangement of edges or the addition or deletion of nodes. To be truly robust, however, any distributed algorithm must be be able to withstand network changes introduced by an adversary. There are two major categories of adversarial dynamic networks: the oblivious adversary, in which the adversary lays out a sequence of graphs prior to the start of the process, and the adaptive adversary, in which the adversary can respond and set out a new list of network changes after each round, incorporating some degree of knowledge of the state of the network and the internal state of each node.

A primary example of a dynamic distributed network is P2P networks, in which every node is an independent agent, and computation and other tasks (storage, for example) are performed through the cooperation and collaboration of the member nodes of the network. A key property of P2P networks is their high degree of dynamism, especially with respect to node churn (the process by which nodes in the network leave and are replaced by new nodes freshly joining). It is estimated that almost 50% of nodes churn out of the network in each hour in many recent real-world P2P networks [24, 31, 52]. Yet despite the churn, P2P networks need to be able to perform tasks such as persistent storage of data, which at first glance seems to be incompatible with a high level of churn. Distributed algorithms have been proposed to perform a variety of tasks, from data storage and retrieval [48, 21] to more diverse tasks such as data mining [20], collabora-

tive filtering [11], and worm detection [54], and many others. However, it has been difficult designing P2P systems that are both fully distributed and also provably robust to changes in the network. Many of the most used P2P applications in the last 15 years (Kazaa, Napster, Gnutella) have had some degree of centralization: for example, a central server to manage newly connecting users, or a subset of users known as "super nodes". Furthermore, even if a network is fully decentralized, many of the algorithms for the applications described above do not have provable guarantees of performance or robustness in the presence of a high degree of adversarial churn of nodes or modification of edges.

Thus, more recent work has focused on developing such algorithms: [26, 33, 43, 51]. Many algorithms geared towards dynamic networks assume that at some point the network will stop changing and stabilize or that there will be a rest period between changes to allow the algorithm to repair the network. The authors of [36] study a network that can change in every round (as long as the network remains connected) in an arbitrary fashion and provide reasonable upper bounds on the running time of various distributed tasks. In the last several years, Gopal Pandurangan and a dynamic set of collaborators have developed several distributed algorithms which provide very useful functionality and are provably robust under a high degree of adversarial churn. [4] gave an $O(\log n)$-round search algorithm with up to $n/\log^{1+\delta} n$ churn per round. It also provides an efficient storage mechanism while only requiring $\Theta(\log n)$ copies of each data item to be stored. The work in [6] provides an algorithm that achieves distributed agreement even with a linear churn rate in a polylogarithmic number of rounds. Finally, [5] provides a distributed algorithm for Byzantine agreement in the presence of up to $O(\sqrt{n}/\text{polylog}(n))$ Byzantine nodes and $O(\sqrt{n}/\text{polylog}(n))$ churn per round.

The critical assumption in these algorithms is that the adversary can do whatever he likes to change the graph in each round as long as he makes each graph an expander. This is a fairly strong restriction, because an expander graph has very nice properties that allow for many of the necessary sub operations. For example, the three works cited above make heavy use of the ability to randomly sample nodes of the graph. This sampling is done via many tokens performing independent random walks through the network. Since the network is an expander, a walk has reached its limiting distribution after only $O(\log n)$ steps, even in a dynamic graph. Clearly, if the graphs the adversary dictates are not expanders, the random walks can no longer provide an efficient way to sample the nodes of the graph. Therefore, it would be extremely useful if one could develop a distributed algorithm that runs in the background on the nodes of the network which is able to rapidly repair the network so that it is an expander graph almost everywhere. Note that almost everywhere, by which we mean for all but some $o(n)$ nodes, is the best we can hope for when dealing with churn, since new nodes are connecting into the network in each round and it would be impossible to incorporate them fully into the expander topology immediately.

To date there have been a number of attempts to design algorithms for which we term "self-healing expanders". However, most either impose limitations on the ability of the adversary to change the graph (such as by limiting the number of nodes that can be deleted or added) or by allowing for generous intervals where no change are made and the graph has a chance to repair itself. Here I present a brief summary of the most salient results. In [38], a distributed algorithm is given which constructs and then maintains an expander graph (w.h.p.) under a limited number of insertions and deletions. [45] provides a self-healing algorithm that repairs a single node insertion or deletion in $O(\log n)$ steps and using $O(\log n)$ messages (assuming no other changes during the healing). In [46], the authors present an algorithm Xheal, which can repair an expander when an adversary deletes or adds a node in each time step, while preserving expansion and making only local changes to the network topology. In [17], the authors propose an algorithm that can maintain a well- connected graph of small diameter in the presence of adversarial modifications to the graph. However, for more than a small fraction of adversarial churn, the algorithm can only guarantee connectivity of the graph and not good expansion. However, one notable feature of this algorithm is the use of random tokens generated by each node bearing that node's address. These tokens each make independent random walks around the graph and are used to sample the remaining nodes of the graph to create random edges.

We will employ a variation of this technique in our self-healing algorithm.

3.2 Healing an expander graph with $O(n/\log^c n)$ churn in constant time, in every round, with polylogarithmic overhead

In this proposal I outline the algorithm that we developed to deal with some of the challenges above. It is the first algorithm that we know of that has the following properties: It allows for an oblivious adversary to delete and insert (i.e. churn) up to $O(n/\text{polylog}(n))$ nodes in every round of the algorithm. Furthermore, it will make its repairs in every round, so that w.h.p. the network will contain a subgraph of size $n - o(n)$ that forms an expander graph (i.e. its isoperimetric number is bounded below by a constant. Furthermore, this algorithm requires each node to send and receive up to only a polylogarithic (in n) number of bits in each round.

The key primitive of this algorithm is the use random walks. Each node, in each round, generates a polylogarithmic number of tokens bearing its address. These tokens then take random walks around the (dynamic) network, in the process either being deleted by being at a node that gets churned out or sticking around long enough in the network to approach a uniform distribution among the nodes. If they survive long enough, the tokens are used by any node in the network (be it newly joined or an existing node) to request a new edge to be formed between that node and the node that originated the token. In this fashion, as the graph it evolves it remains essentially a bounded-degree random regular graph. Naturally, there is quite a bit under the hood that needs to be shown. For example, it is not at all clear that the random tokens will indeed mix rapidly, since mixing is a function of the network topology, which is a function of the edges formed by sampling from mixed tokens. Furthermore, it needs to be shown that in a short number of rounds, if a disconnected or recently connected node sends a constant number of requests in each round, it will be (randomly) reconnected back to the network quickly and will not suffer from too many message collisions. We address all off these challenges and more.

As this work is in many ways the last piece of a puzzle of a body of work already published, I do not intend to try to develop more content for my thesis work. However, when going back over the paper to summarize it for this section, I decided that the current layout and flow of both the algorithm and analysis sections of the papers is somewhat confusing and deserves a major reorganization. I will include this in my thesis.

3.2.1 Main result and technical contribution

The major result of this work is an efficient randomized distributed protocol that guarantees the maintenance of a *sparse* (bounded degree) topology with *high expansion* despite the presence of a large adversarial churn. We assume that the *number* of nodes in the graph is stable. We assume that the churn is limited to $O(n/\text{polylog}(n))$ nodes per round and is controlled by an oblivious adversary, which has complete knowledge of what nodes join and leave at what time, but is unaware of any random choices made by nodes of the graph (including a decision to create or delete an edge between nodes). The adversary can remove any set of nodes up to the churn limit in every round. It also adds (an equal amount) of nodes to the network. Each new node added must be connected to at least one existing node (i.e. a node that was not just churned out), and the number of new nodes connected to an existing node may not exceed some fixed constant.

Our protocol requires only polylogarithmic in n bits to be processed and sent by each node in each round; each node's computation cost per round is small (also polylogarithmic in n). Our protocol guarantees that in each round, even in the presence of adversarial churn that obeys the $O(n/\text{polylog}(n))$ churn limit, there is a subgraph of size $n - o(n)$ with expansion lower bounded by some fixed constant. Furthermore, the degree of each node is at all times upper bounded by a fixed constant. The one significant requirement of our protocol is that a "bootstrap" phase of $O(\log n)$ rounds is required at the beginning of the process – during this phase,

the graph is connected as an expander graph and there is no churn. This phase is necessary to ensure that the process by which edges can be sampled is initially random. We show that any strategy that relies on random walks will also require this bootstrap phase.

This algorithm is lightweight, distributed, and local (no global topological knowledge is required). It also should be easy to implement. The algorithm serves as a building block of enabling distributed algorithms for non-trivial distributed competing problems such as agreement, search and storage, and leader election in dynamic P2P networks.

The main technical challenge to overcome in designing such a protocol is maintaining good topological properties in highly dynamic networks where both edges and nodes can change by a large amount on an ongoing basis. In this work, this challenge is overcome by using random walks to maintain an expander-like topology. In this random-walk based procedure, each node, in every round, generates some number of tokens (containing the generating node's ID and a counter) and sends them on a random walk around the network. After a logarithmic number of rounds, the random walks "mix" – are roughly uniformly distributed around the network – and can then be used to form random edges between nodes. Thus the backbone of the network (and of the algorithm) is a random, bounded degree graph, which is known to have good expansion properties.

Naturally, it needs to be shown that this process (mixing of random walks and sampling from mixed tokens) still works even when the network is being modified by the adversary and by the algorithm itself, which one would imagine can create rather complicated complexities. Many of the tokens taking the random walks can be deleted (since they are nodes that are churned out), and thus a bias may arise in the distribution of the remaining tokens. Since these tokens are then used to create edges to maintain the graph, and these edges might now be drawn from a biased distribution, the random walks may then even be further biased by modifications made by the algorithm itself. Thus we must ensure that the random walks remain mixed and asymptotically close to a uniform distribution over the nodes of the graph despite the changes occurring. We overcome this challenge by implementing at each node a "local" criterion for deciding if tokens being received are sufficiently random – if an insufficient number of tokens, mixed or otherwise, are received, the node knows it is not well connected to the network and "refreshes" its edges, thus ensuring that, in aggregate, almost all edges are indeed random.

The two other main challenges are one, that nodes can become isolated from the rest of the graph and used as a beachhead by the adversary to add newly churned in nodes and disconnect the graph, and two, that nodes can only send a constant number of messages (and hence a constant number of requests to connect) in each round. This could mean that a node fails to connect sufficiently well before it becomes isolated from the graph. We present mechanisms to overcome both of these challenges.

3.2.2 Preliminaries

The Adversarial Network Model. We consider a synchronous dynamic network controlled by an oblivious adversary. The adversary fixes a sequence of graphs $\mathcal{H} = (H_1, H_2, \ldots, H_i, \ldots)$ with each H_i is defined on a vertex set V_i. The size of the vertex set is assumed to be stable, i.e., $|V_i| = n$ for all i, and $|V_i \setminus V_{i+1}| = |V_{i+1} \setminus V_i| \in O(n/\log^k n)$, where k is a constant that will be fixed in the analysis[1]. The vertex set V_i refers to the set of nodes present in the network in round i. Each node has a unique ID chosen from an ID space of size polynomial in n. The edges must be viewed as a rudimentary set of connections provided by the adversary. The degree of each graph H_i is bounded by a constant denoted by $\deg(\mathcal{H})$.

For the first $\Theta(\log n)$ rounds, called the *bootstrap phase*, the H_i's are constant degree expanders with *expansion at least* $\alpha > 0$, a fixed constant. This means that, for any set S of $\leq n/2$ nodes, the number of edges across the cut given by S and the remaining nodes, is at least $\alpha|S|$.

[1]For the sake of clarity, we do not seek to optimize constants and $\log n$ factors. Our analysis shows that k need not be > 3.

Moreover, there is no churn during the bootstrap phase, so the vertex sets are identical. We can think of the bootstrap phase as a period of stability provided for the protocol to get started.

After the bootstrap phase, the requirements on the adversary are significantly reduced. It is only required to ensure that any new node u that enters at, say, round i, must be connected in H_i to at least one other pre-existing node v, i.e., if $u \in V_i \setminus V_{i-1}$, then, u must be connected to some $v \in V_{i-1} \cap V_i$ in round i. The adversary must also ensure that the degree bound stipulated for the H_is is not violated. The following subtleties are noteworthy. Firstly, we only require that a new node is connected to a pre-existing node in its very first round upon entering the network. Secondly, after the bootstrap phase we don't even require each H_i to be connected — in fact, the only required edges in H_i are the edges connecting new nodes to pre-existing nodes.

Requirements. Our goal is to design a distributed protocol to maintain a graph process $\mathcal{G} = (G_1, G_2, \ldots, G_i, \ldots)$ over time such that each G_i constructed at the end of round i has good expansion while maintaining a constant degree bound Δ. The protocol must achieve this goal by dynamically adding/deleting edges over time. Each $G_i = (V_i, E_i)$ has the same vertex set as that of H_i. In each round i, the protocol can add a set $E_i^{\downarrow}$ of new edges. Also, a set $E_i^{\uparrow}$ of edges that were added by the protocol in some previous round is lost — some explicitly removed by the protocol and others removed implicitly because vertices to which they were incident were churned out. Thus, each $E_i = (E_{i-1} \setminus E_i^{\uparrow}) \cup E_i^{\downarrow}$. Our goal, more formally, is to show that for some constant $\alpha > 0$, if G_{i-1} has maximum degree Δ and expansion at least α, then, the graph $G_i = (V_i, E_i)$ thus formed must, with high probability[2], also have expansion $\geq \alpha$ and a maximum degree of $\leq \Delta$. We emphasize that the graph H_i is only presented at round i and the protocol is unaware of future graphs.

Communication Model. Communication is via message passing. If a node u knows the ID of another node v either directly because u and v are neighbors in the current graph in $\mathcal{G}$, or because they are neighbors in H_i (assuming we are in round i), or indirectly through received messages, then u can communicate with v[3]. Each node can send and receive messages of size polylogarithmic (in n) bits. Furthermore, each node is restricted to some constant $M = \Delta + \deg(\mathcal{H})$ in the number of messages (and therefore, the number of nodes to which) it can send and receive in a single round.

Notice that a sender u can ensure that it sends out at most M messages intended for at most M recipients. Let v be one of the nodes to which u has sent a message. Notice that u has no information about how many other nodes are also trying to send messages to v. If more than M nodes send messages intended for v, it can only receive M messages, while others must be dropped. We assume the following priority by which messages are either accepted or dropped.

1. All messages (say, of cardinality $j \leq M$) sent along edges in $\mathcal{H}$ and $\mathcal{G}$ are sent successfully, and
2. Among the remaining messages intended for v, only an arbitrarily chosen subset of $M - j$ messages reach v; the rest are all dropped.

Sequence of Events in a Round. We now describe the precise sequence of events that comprise a single round i in our model:

1. The adversary presents the new graph H_i in which some of the old nodes have been churned out and some new nodes have been churned in (assuming round i occurs after the bootstrap phase). The graph G_{i-1} loses all edges that were incident to nodes that were churned out.
2. Each node is then allowed to perform some local computation including private coin flips and send messages of size polylogarithmic in n to up to C other nodes. This communication step, among other purposes, is particularly useful for nodes to send requests for adding edges.
3. Nodes can again perform some more local computation, and again send messages of size polyloga-

[2]We say that an event occurs *with high probability* (whp) if its probability is at least $1 - n^{-c}$ for a sufficiently large constant c.

[3]This is a typical assumption in the context of P2P and overlay networks, where a node can establish communication with another node if it knows the other node's IP address.

rithmic in n to up to C other nodes. The messages sent in this step are typically responses to messages sent out in the previous step. For instance, a node v that had received a request from a node u to form an edge could either send an accept message or ignore it (implicitly rejecting the request).

4. Once v accepts node u's edge request, edge (u, v) is added to the communication graph. The communication graph now includes all the newly added edges, and is designated G_i.

We note that a round is comprised of a two-step protocol to add an edge: the requesting node sends a message to the other node, which takes one more step to accept/reject. The above assumption is slightly different from the classic notion of a round, where messages are sent only one way per round (hence implementing these two steps will actually take two rounds.) This assumption is not just a mere technicality; on the other hand it is needed. We prove an impossibility result that that shows there is no protocol in the classic synchronous round model (cf. [49]) that can maintain connectivity (let alone expansion) for a large majority of the nodes, even if the churn is limited to only $\Theta(\sqrt{n}/\text{polylog}(n))$ nodes per round.

Adding/Deleting Edges in $\mathcal{G}$. We assume the existence of a simple two-step "handshake" protocol (described in the next paragraph) that allows any node u to propose an edge between itself and another node v (provided u knows v's id); the node v, in turn, can either explicitly accept or ignore (thereby implicitly rejecting) the proposal. Furthermore, a node u can drop any edge (u, v) incident to it without consulting v; we assume that v will be immediately notified that (u, v) has been dropped.

3.2.3 Description of the Protocol

Our protocol works by ensuring two interdependent invariants (whp). The first invariant is that the nodes can sample from the set of IDs in the network almost uniformly at random. We implement this sampling via random walks. For the sampling technique to work efficiently, we need our second invariant, namely, that the network maintains good expansion, and for this, we employ a technique of connecting each node to a constant number of other nodes chosen (almost) uniformly at random, thus bringing us back to our first invariant. With the two invariants being so intimately dependent on each other, the bootstrap phase in which expansion is guaranteed is crucial in helping the protocol to get started. During this phase, the random walks based sampling procedure gets started, which, in turn, allows the network to continue maintaining good expansion beyond the bootstrap phase. While this high level idea is quite straightforward, there are some significant challenges to be overcome in order to get the protocol to work.

Here we give an high-level overview of the main ideas.

Degree Bound : Red and Blue Edges Every node v in the network has upper bound Δ on its degree. We can thus say it has Δ ports. It marks $\delta < \Delta/6$ of those ports as *blue* ports. These will be used to form *blue* edges (from v's perspective), which are edges formed at v's request. The remainder of the ports are marked as *red* ports, which will be used to form edges between v and nodes which request the edge to be formed. A blue port without an incident edge is called a *dangling port*, while each red port without an incident edge forms a self-loop with itself.

Random Walks Based Sampling In each round (including the bootstrap phase), each node v generates $z \in \Theta(\text{polylog}(n))$ tokens. Each token has the originating node's address and a counter (set to 0 initially). Each time the token moves to a new node, its counter is incremented by 1.

Each token then takes a random walk for up to $\tau = \Theta(\log n)$ rounds. If it survives long enough to have its counter equal to τ, it is deemed a *mature* token.

If a node v at any point receives fewer than $z/2$ mature tokens (each taking an independent random walk) in a round, the mature tokens are discarded. This is because the mature tokens are used to sample from the vertex set when forming an edge, and receiving fewer than $z/2$ mature tokens indicates that the node is in a poorly connected portion of the network and would do best to just start over forming edges. If

u receives more than $z/2$ tokens, it places them in what we call a *prioritized token buffer*.

Prioritized Token Buffer Each node v has a prioritized token buffer which holds up to $C \in O(polylog(n))$ mature tokens. These tokens are assigned a priority based on their age (younger means higher priority and whether or not they have been copied (see below). A node will reserve the top $c \in \Theta(\text{polylog}n)$ tokens for its own use. Node v can be contacted by another node requesting fresh (not copied) tokens. If there are at least $C/2$ tokens in the buffer, c tokens are given to the requesting node. If the number of fresh tokens in the buffer falls below $C/2$, then the tokens are marked *stale* and are copied. These copied tokens (with the stale marking) are given to the requested node.

Distributed Expander Maintenance Protocol The protocol runs on all nodes in the network, and is initiated as soon as the node joins the network. The random walk token sampling processes runs in the background at all times (implemented at the node level but operating as a global property). Once the bootstrap phase is over, the adversary is allowed to make changes to the network.

Each node can be in one of two states: `normal` mode or `reconnect` mode. We describe what is happening from the perspective of a single node v.

The `Normal` mode If v is in `Normal` mode, it checks to see how many tokens it has received. If it has received below $z/2$ mature tokens, it deletes all of its existing blue edges and switches into `reconnect` mode.

If it has received a sufficient number of mature tokens, with probability $1/\text{polylog}(n)$, v drops all of its blue edges and switches to `reconnect` mode.

The `Reconnect` mode Node v will be in this mode if it has either just joined the network, or if some or all of its blue edges have been dropped.

If v is new to the graph, the adversary must connect it with some existing node w. Then v will receive $c \in \Theta(\text{polylog}(n))$ tokens from w. These tokens can be either fresh or stale (copied). If they are fresh tokens, v makes edge requests until it has formed δ blue edges. If v has received copied tokens, then v uses these tokens to send requests for fresh tokens. Once it receives them, it begins making connection requests to form δ blue nodes. We prove that this two-step reconnection process ensures that no component of the network becomes disconnected from the network as a whole.

If v already has c fresh tokens in its priority token buffer, it immediately begins using these tokens to replenish its δ blue edges.

3.3 Proof of Correctness

Here I will provide a brief overview of how we prove that our algorithm operates as intended, as well as a brief summary of results. The main result is:

Theorem 9. *In each round, the expander maintenance protocol maintains the network communication graph as a graph that contains a large $n-o(n)$-size expander graph with expansion at least $\alpha \in \Theta(1)$ for a number of rounds that is at least a polynomial in n with high probability.*

3.3.1 Using the random walk tokens to form edges produces a graph with high expansion

Here we describe Lemma 10, one of the main lemmas used in the analysis of the protocol. The gist of the lemma is that if we can show that, when a *blue* edge is formed by a node, the endpoint of the new blue edge is drawn almost uniformly at random from a set of nodes representing a large constant fraction of the nodes

of the network, then a network formed from such an edge set selection process is an expander with high probability.

Lemma 10. *Let $G = (V, E)$ be a random graph on n vertices formed by the following random process. For each $v \in V$, we create δ incident blue edges such that the red end of each such edge e is chosen from a probability distribution D over V defined in the following manner. For every $v \in V, D(v) \leq 1/n + 1/n^c$ for a suitably large constant c. Furthermore, for an arbitrarily chosen subset $S_e \subseteq V$ of cardinality at least $0.8|V|$, $D(v)$ is guaranteed to be in $\Omega(1/n)$ for every $v \in S_e$. Then, with high probability, G has an expansion of at least α' for some suitable constant $\alpha' > 0$.*

3.3.2 Proving that the tokens are taking rapidly-mixing random walks

Most of the rest of the work in the analysis involves showing that the random walk token technique of the maintenance protocol will in fact successfully mix enough of the tokens to guarantee that there are enough of them to perform the random sampling needed to create the blue edges, despite the changes going on in the network.

In the paper, the proof is handled somewhat indirectly. Instead of working directly with the network generated by the algorithm (and adversary), we describe a family of processes which we refer to as Υ processes. An Υ process is a process on a sequence of graphs that includes adversarial change, the generation of random edges, and the performance of many random walks. We prove that every Υ process has the desired properties, and then prove that our expander maintenance protocol is in fact an Υ process.

Without getting into the technical details, I briefly describe how we show that every node will receive a sufficient number of mature (i.e. mixed) random tokens. The first step is to note that random walks will mix rapidly in a graph that is dynamic in edges but not nodes if the graph is an expander in each round. Then, we imagine a version of our process in which when a new node is churned in, it is assigned the edges that belonged to a node that had just been churned out. These edges that are reassigned are called ghost edges. The tokens continue their random walks around the network, and if they at any point cross a ghost edge, they are labeled ghost tokens. Tokens that do not cross a ghost edge are called real tokens.

We then prove that:

1. If a node receives a sufficient number of real tokens, then with high probability each real token's origin can be drawn from an almost uniform distribution over a large set of nodes of the network $(1 - \epsilon)n$, for some constant $\epsilon < 1/2$.
2. The number of nodes that receive this sufficient number of tokens is of cardinality $n - o(n)$.

Thus, at each step, an $n - o(n)$ size fraction of the (real) graph, in which ghost edges are deleted, gets mature tokens, and these tokens are well-mixed. We can then apply **??** to show that each graph (including its ghost edges) is an expander. Thus, a subset of the graph, by citing a result in [7], forms an induced subgraph that also has constant (though slightly lower) expansion.

4 Proposed Additional Work

As noted earlier, there are a number of open problems for cobra walks: the most immediate being tightening up the cover time of expanders to $O(\log n)$ and proving a general cover time bound for arbitrary graphs. Beyond cover time, I believe that there are a number of other interesting questions. At the top of that list would be a study of the long term dynamics of cobra walks. Of interest to me would be whether the number of active/infected nodes remains fairly stable or is subject to wide fluctuations, and whether the conductance of the underlying graph has any impact on this. For example, one might expect a cobra walk on a tree to perhaps travel in "waves" over the graph, while any graph that had a subgraph of high conductance might

have a more stable pattern of infection. These questions are broadly applicable to the study of cobra walks as an SIS epidemic model for a disease endemic to a population.

Before describing my proposed work in more detail, I would like to spend a few minutes justifying going down this road on a practical level. Given the original paper's good reviews and appearance at SPAA in 2013, the invitation to submit it to a special edition journal (TOPC), and the interest the topic generated in discussions at ICERM's Networks semester in Spring of 2014, I think there is sufficient interest in general to devote more time to this study. Further, as I will outline below, there is a chance that a new approach to studying cobra walks will provide an even more general framework to view other similar processes (gossip protocols, multiple random walks, and voter models).

The cobra walk was originally envisioned as a generalization of a random walk. However, despite he observation that setting the branching factor $k = 1$ produces a simple random walk, the complexity of analyzing a cobra walk (specifically, keeping track of the many interactions and dependencies among the pebbles), obscures this original view. The view of cobra walks we have thus far is as a large (possibly $\Omega(n)$) number of pebbles at each time step making independent, single-step, random walks. bookended by the branching and coalescing steps. The "states" of this system, if they could be interpreted as such, are the active sets. That is, S_t is the set of vertices that have a pebble at time t. Though complicated, if we know the topology of G and the membership of S_t, for any set $S \subseteq V$, we can compute $\Pr[S_{t+1} = S|S_t]$. Informally, this means that the cobra walk, which seems superficially far more complicated than a simple random walk, is in fact itself a Markov Chain.

Of course, there are a few substantial differences: for example, the state space of the chain is $2^{|V|} - 1$, so it is exponentially sized (in n). Furthermore, while irreducible, it is not reversible. For example, take the situation in which, for some node v, all of the neighbors of v (and only the neighbors of v) have a pebble. There is a non-zero (and indeed in a bounded-degree graph a constant) probability that all of the pebbles in the next step will move v. Viewing this as a walk on the markov chain, we can denote this probability $p_{i,j}$, where this is the probability of transition from state i to state j, defined appropriately. Since the chain is irreducible, the stationary distribution π_i for vertex i is non-zero. However, the reverse probability $p_{j,i}$ is 0, and hence the definition of reversibility of the chain, $p_{i,j}\pi_i = p_{j,i}\pi_j$ is violated, since the term on the right of the equality is 0. Thus, what we have described here is essentially a random walk on a directed graph whose vertices consist of all of the subsets of V (except for the empty set) and whose edges are defined by probabilities of a cobra walk with active set S having an active set T in the next step. For simplicity, from here on out I will refer to this directed random walk as an **Cobra Markov Chain (CMC)**.

There are a number of potentially fertile questions regarding CMC that I propose to work on:

1. Formalization and Basic Properties of CMC

Up until this point, I have been necessarily somewhat vague or imprecise about how I describe CMCs. The first step in this work will be coming up with a formal description and terminology for CMCs. I will also prove some basic properties that are intuitive but still need to be demonstrated: That the chain is irreducible, that a stationary distribution exists, and that it is (in general) irreversible. This step will be quick and will set up a good infrastructure for the rest of my work.

2. Application to Phase 2 for the Cover Time of Expanders

Recall that in our original work on cobra walks, we were only able to show the full coverage of a d-regular ϵ-expander in $O(\log^2 n)$ time. This was done by showing an $O(\log^2 n)$ bound on the cover time of a process W_{alt}, a modified version of a cobra walk that stochastically dominated it. In turn, the analysis of W_{alt} hinged on calculating the mixing time of the walk of 2 non-independent pebbles viewed as the walk of a pebble on a directed graphed formed from the 2-way tensor product of G.

This analysis leads to the natural question: what if we increase the number of pebbles taking a "joint walk" from 2 to δn. This would then be a walk on a directed graph formed from the δn-way tensor product

of G. We would be interested in calculating the mixing time of such a walk. To have coverage of W_{alt} in $O(\log n)$ steps, we would need to show that the walk on the tensor product mixes rapidly, and that the probability of the walk being at any state that mapped back to at least one pebble being at arbitrary vertex v in W_{alt} was sufficiently high (and this would be true independently for each vertex).

While the array of tools and results for random walks on directed graphs is not as large or as pleasant as that for undirected random walks, there is still a considerable amount of work. [TODO: insert references here]. Potentially the most useful of these papers is [15], in which the authors develop a Cheeger bound for non-reversible (i.e. directed) Markov chains and are able to show convergence to stationarity as a function of the Cheeger number (so, for example, when the Cheeger number is constant, the chain converges rapidly).

3. Calculating Bounds Directly on the CMC of a well-structured graph

Before tackling the general case (i.e. analyzing a CMC on a tensorized arbitrary graph), it would be useful to analyze the process on a simpler graph either with a very regular structure (e.g. lattice, hypercube), or on a graph for which we already have a decent bound (e.g. tree, lattice). In this case, the best candidate may be a lower-dimensional lattice (most likely 2). If a tight correspondence between a CMC and a cobra walk can be established (i.e. some form of isomorphic relationship), then we can actually prove results about the CMC and its underlying tensor product graph from known results and proceed from there.

4. Bounds on General Graphs The main goal in studying the cover time of the cobra walk is to find an upper bound on the cover time of an arbitrary graph. We have long conjectured (and experimental results support) that the cover time for any graph is in $O(n \log n)$, making it more-or-less equivalent to the rumor spreading process. This is perhaps the most ambitious result to strive for as well. If the cover time was indeed $O(n \log n)$, this would imply that the corresponding CMC graph will under all circumstances have high conductance and small diameter. Ultimately, a general result about cover time of cobra walks would nicely round out our initial work on the process, mirroring results that have been shown for other processes, from random walks to gossip-type processes.

On the other hand, due to the enormous state space and difficult nature of calculating transition probabilities for CMCs, there is a reasonable chance it will be difficult to make progress in this area. Therefore, I need to consider other options towards proving general bounds on the cover time of cobra walks on graphs:

5. Alternatives to CMC. There is likely an alternative avenue to proving at least the general bound for the cover time of a cobra walk. In [12], the authors present an algorithm for clustering any graph into subsets with lower-bounded (and relatively high) conductance:

Corollary 11. *Every unweighted graph on m edges has a clustering that cuts at most $m/2$ edges and each cluster has conductance at least $\frac{1}{3 \log_{3/4} 2m}$.*

Separate experimental results that I have conducted have shown that a cobra walk will cover a subgraph with high conductance in $\log n$ time, even if every node has a fraction of its edges leaving the subgraph. In addition, the subgraph will quickly to go to some equilibrium state in which a constant fraction of its vertices have a pebble at each time step. If we can make this a provably true result, we then may be able to provide a cover time bound for the cobra walk on the entire graph as follows: Once a subgraph receives at least one pebble from the cobra walk, that subgraph is rapidly covered. Furthermore, the subgraph remains infected and for every time step thereafter "seeds" subgraphs it is adjacent to. This actually resembles a gossip protocol, and we may then be able to use the large body of gossip-centered results to establish the bound on cover time.

As a departure from the above questions that deal directly with cobra walks, I am also interested in the epidemiological applications of cobra walks. Furthermore, work in the SIS epidemic space opens the door for some of my work to take the form of simulation experiments, as a form of hedge against not being able to make progress in theoretical results:

6. Applications of cobra walks to SIS-type epidemics While this last proposed work area is broader than those above, it also presents an opportunity to include simulation-based experiments as part of my research. There are a number of directions to go in here, but I will outline two here: In [27] the authors present an upper bound involving the spectral radius of a graph, below which an epidemic will go extinct rapidly, as well as a lower bound on the isoperimetric number of the graph (thus involving the second eigenvalue) above which the epidemic will persist an exponentially long time. For general graphs there is a wide gap between these two conditions. I would like to devote some time and thought into searching for a "threshold" condition that marks the bifurcation in behavior between dying out and long-term persistence. I am also interested in the effect of a graph's topology (mostly its spectral properties) on the long-term dynamics of an epidemic. One can ask questions such as: Does the epidemic persist in a large fraction of the nodes? Does it cycle? Is there a stable state?

The current best idea for approaching the epidemic topic is to make minor modifications to the cobra walk process so that it even more closely resembles an SIS epidemic, and then proceed by both direct analysis of the modified cobra walk and also through simulation studies.

References

[1] Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, and Bernardo A. Huberman. Search in power-law networks. *Phys. Rev. E*, 64:046135, Sep 2001.

[2] Noga Alon, Chen Avin, Michal Koucky, Gady Kozma, Zvi Lotker, and Mark R. Tuttle. Many random walks are faster than one. In *Proceedings of the Twentieth Annual Symposium on Parallelism in Algorithms and Architectures*, SPAA '08, pages 119–128, New York, NY, USA, 2008. ACM.

[3] R. M. Anderson and R. M. May. *Infectious Diseases of Humans: Dynamics and Control*. Oxford University Press, 1992.

[4] John Augustine, Anisur Rahaman Molla, Ehab Morsy, Gopal Pandurangan, Peter Robinson, and Eli Upfal. Storage and search in dynamic peer-to-peer networks. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '13, pages 53–62, New York, NY, USA, 2013. ACM.

[5] John Augustine, Gopal Pandurangan, and Peter Robinson. Fast byzantine agreement in dynamic networks. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, pages 74–83. ACM, 2013.

[6] John Augustine, Gopal Pandurangan, Peter Robinson, and Eli Upfal. Distributed agreement in dynamic peer-to-peer networks. *Journal of Computer and System Sciences*, 2015.

[7] Amitabha Bagchi, Ankur Bhargava, Amitabh Chaudhary, David Eppstein, and Christian Scheideler. The effect of faults on network expansion. *Theory of Computing Systems*, 39(6):903–928, 2006.

[8] Petra Berenbrink, Colin Cooper, Robert Elsässer, Tomasz Radzik, and Thomas Sauerwald. Speeding up random walks with neighborhood exploration. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1422–1435, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[9] Petra Berenbrink, Colin Cooper, and Tom Friedetzky. Random walks which prefer unvisited edges.: Exploring high girth even degree expanders in linear time. In *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing*, PODC '12, pages 29–36, New York, NY, USA, 2012. ACM.

[10] Ashwin R. Bharambe, Mukesh Agrawal, and Srinivasan Seshan. Mercury: Supporting scalable multi-attribute range queries. *SIGCOMM Comput. Commun. Rev.*, 34(4):353–366, August 2004.

[11] John Canny. Collaborative filtering with privacy. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 45–57. IEEE, 2002.

[12] Keren Censor-Hillel, Bernhard Haeupler, Jonathan Kelner, and Petar Maymounkov. Global computation in a poorly connected world: Fast rumor spreading with no dependence on conductance. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 961–970, New York, NY, USA, 2012. ACM.

[13] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Almost tight bounds for rumour spreading with conductance. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 399–408, New York, NY, USA, 2010. ACM.

[14] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Rumour spreading and graph conductance. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1657–1663, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[15] Fan Chung. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 9:1–19, 2005.

[16] Colin Cooper, Robert Elsässer, Hirotaka Ono, and Tomasz Radzik. Coalescing random walks and voting on graphs. In *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing*, PODC '12, pages 47–56, New York, NY, USA, 2012. ACM.

[17] Colin Cooper, Ralf Klasing, and Tomasz Radzik. A randomized algorithm for the joining protocol in dynamic distributed networks. *Theoretical Computer Science*, 406(3):248–262, 2008.

[18] Daryl J. Daley and Joseph M. Gani. *Epidemic Modeling: an introduction*. Cambridge University Press, 1999.

[19] Atish Das Sarma, Danupon Nanongkai, and Gopal Pandurangan. Fast distributed random walks. In *Proceedings of the 28th ACM Symposium on Principles of Distributed Computing*, PODC '09, pages 161–170, New York, NY, USA, 2009. ACM.

[20] Souptik Datta, Kanishka Bhaduri, Chris Giannella, Ran Wolff, and Hillol Kargupta. Distributed data mining in peer-to-peer networks. *IEEE Internet Computing*, 10(4):18–26, 2006.

[21] P. Druschel and A. Rowstron. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. *In Proc. of ACM SOSP*, 2001.

[22] Rick Durrett. Some features of the spread of epidemics and information on a random graph. *Proceedings of the National Academy of Sciences*, 107(10):4491–4498, 2010.

[23] Robert Elssser and Thomas Sauerwald. Tight bounds for the cover time of multiple random walks. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming*, volume 5555 of *Lecture Notes in Computer Science*, pages 415–426. Springer Berlin Heidelberg, 2009.

[24] Jarret Falkner, Michael Piatek, John P. John, Arvind Krishnamurthy, and Thomas E. Anderson. Profiling a million user dht. In *Internet Measurement Comference*, pages 129–134, 2007.

[25] Uriel Feige. A tight upper bound on the cover time for random walks on graphs. *Random Struct. Algorithms*, 6(1):51–54, January 1995.

[26] Amos Fiat and Jared Saia. Censorship resistant peer-to-peer content addressable networks. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 94–103. Society for Industrial and Applied Mathematics, 2002.

[27] A. Ganesh, L. Massoulie, and D. Towsley. The effect of network topology on the spread of epidemics. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 1455–1466 vol. 2, March 2005.

[28] George Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In *Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 57–68, March 10–12 2011.

[29] George Giakkoupis, Thomas Sauerwald, and Alexandre Stauffer. Randomized rumor spreading in dynamic graphs. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 495–507, July 8–11 2014.

[30] C. Gkantsidis, M. Mihail, and A. Saberi. Hybrid search schemes for unstructured peer-to-peer networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1526–1537 vol. 3, March 2005.

[31] P. Krishna Gummadi, Stefan Saroiu, and Steven D. Gribble. A measurement study of napster and gnutella as examples of peer-to-peer file sharing systems. *Computer Communication Review*, 32(1):82, 2002.

[32] Bernhard Haeupler. Simple, fast and deterministic gossip and rumor spreading. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 705–716. SIAM, 2013.

[33] Kirsten Hildrum and John Kubiatowicz. Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks. In *Distributed Computing*, pages 321–336. Springer, 2003.

[34] David R. Karger and Matthias Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '04, pages 36–43, New York, NY, USA, 2004. ACM.

[35] Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 163–170, New York, NY, USA, 2000. ACM.

[36] Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 513–522. ACM, 2010.

[37] C. Law and K.-Y. Siu. Distributed construction of random expander networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 2133–2143 vol.3, March 2003.

[38] Ching Law and K-Y Siu. Distributed construction of random expander networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 2133–2143. IEEE, 2003.

[39] David Asher Levin, Yuval Peres, and Elizabeth Lee Wilmer. *Markov chains and mixing times*. Providence, R.I. American Mathematical Society, 2009. With a chapter on coupling from the past by James G. Propp and David B. Wilson.

[40] Fredrik Liljeros, Christofer R. Edling, Luis A. Amaral, H. Eugene Stanley, and Yvonne Aberg. The web of human sexual contacts. *Nature*, 411(6840):907–908, June 2001.

[41] Peter Matthews. Covering problems for brownian motion on spheres. *The Annals of Probability*, pages 189–199, 1988.

[42] Lauren Ancel Meyers, Babak Pourbohloul, M. E. J. Newman, Danuta M. Skowronski, and Robert C. Brunham. Network theory and sars: Predicting outbreak diversity. *Journal of Theoretical Biology*, 232:71–81, 2005.

[43] Moni Naor and Udi Wieder. A simple fault tolerant distributed hash table. *Peer-to-Peer Systems II*, pages 88–97, 2003.

[44] M. E. J. Newman. Spread of epidemic disease on networks. *Physical Review E*, 66(1):016128, July 2002.

[45] Gopal Pandurangan, Peter Robinson, and Amitabh Trehan. Dex: Self-healing expanders. In *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium*, IPDPS '14, pages 702–711, Washington, DC, USA, 2014. IEEE Computer Society.

[46] Gopal Pandurangan and Amitabh Trehan. Xheal: a localized self-healing algorithm using expanders. *Distributed Computing*, 27(1):39–54, 2014.

[47] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic dynamics and endemic states in complex networks. *Phys. Rev. E*, 63:066117, May 2001.

[48] Arjan Peddemors. Cloud storage and peer-to-peer storage - end-user considerations and product overview. *http://www.novay.nl/okb/publications/152*, 2010.

[49] David Peleg. *Distributed Computing: A Locality-sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[50] CJ Rhodes and RM Anderson. Persistence and dynamics in lattice models of epidemic spread. *Journal of theoretical biology*, 180(2):125–133, 1996.

[51] Christian Scheideler. How to spread adversarial nodes?: Rotate! In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 704–713, New York, NY, USA, 2005. ACM.

[52] Subhabrata Sen and Jia Wang. Analyzing peer-to-peer traffic across large networks. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, IMW '02, pages 137–150, New York, NY, USA, 2002. ACM.

[53] R Michael Tanner. Explicit concentrators from generalized n-gons. *SIAM Journal on Algebraic Discrete Methods*, 5(3):287–293, 1984.

[54] Vasileios Vlachos, Stephanos Androutsellis-Theotokis, and Diomidis Spinellis. Security applications of peer-to-peer networks. *Computer Networks*, 45(2):195–205, 2004.

[55] D. J. Watts and S. H. Strogatz. Collective dynamics of'small-world'networks. *Nature*, 393(6684):409–10, 1998.

[56] Ming Zhong and Kai Shen. Random walk based node sampling in self-organizing networks. *SIGOPS Oper. Syst. Rev.*, 40(3):49–55, July 2006.