Correctness of a Gossip Based Membership Protocol

André Allavena
andre@cs.cornell.edu

Alan Demers
ademers@cs.cornell.edu

John E. Hopcroft
jeh@cs.cornell.edu

Department of Computer Science
Cornell University
Ithaca NY 14853

ABSTRACT

The importance of scalability and fault-tolerance in modern distributed systems has led to considerable research in multi-cast protocols using *gossip*. In a gossip protocol, each node forwards messages to a small set of "gossip partners" chosen at random from the entire group membership. By discarding the strong reliability guarantees of traditional protocols in favour of probabilistic guarantees, gossip protocols can deliver greater scalability and fault tolerance. In early gossip algorithms, partners were chosen uniformly at random from the entire membership, limiting scalability because of the resources required to store and maintain complete membership views at each node. Later protocols avoided this issue by storing much smaller random subsets of the membership at each node, and choosing gossip partners only from these *local views*. Such protocols are subtle: at least some local views must change in response to group membership changes in order to preserve connectivity and performance guarantees. While these protocols have been the subject of much simulation and analysis, formal proofs of key properties – in particular the probability of partitioning – have remained elusive.

In this paper we give a new scalable gossip-based algorithm for local view maintenance, together with a proof that the expected time until a network partition is at least exponential in the square of the view size. We also develop probabilistic bounds on the in-degree (hence the load) of individual nodes, and argue that protocols lacking our *reinforcement* component eventually converge to star-like networks, whose connectivity depends on a small set of overloaded nodes. We also argue that the undirected connectivity graph is an expander, for which application-level gossip multi-cast protocols will converge rapidly.

Our theoretical results are supported by simulations.

Categories and Subject Descriptors

C.4 [**Performance of Systems**];
C.2.3 [**Computer Communication Network**]: Network Operations—*Network management*;
C.2.2 [**Computer Communication Network**]: Network Protocol

General Terms

Reliability Algorithms Design Management

Keywords

scalability, reliability, peer-to-peer, epidemic, gossip, membership, group communication, random graphs, probabilistic multi-cast

1. INTRODUCTION

The importance of scalability and fault-tolerance in modern distributed systems has led to considerable research in gossip-based multi-cast protocols, first introduced in [5]. In a gossip protocol, each node forwards messages to a small set of "gossip partners" chosen at random from the entire group membership. The resulting "epidemic" yields a probabilistic guarantee of delivery to all group members. By discarding the strong reliability guarantees of traditional protocols such as [2] in favour of probabilistic guarantees, a gossip protocol can deliver much greater scalability and fault tolerance.

The application that inspired [5] was the Xerox Clearinghouse, a distributed name service comprising hundreds, but not thousands, of nodes. At that scale, a strongly-consistent replication algorithm as described in [2] is infeasible, but it is still fairly cheap to store complete group membership information at every node. Thus, it was straightforward for the Clearinghouse system to select gossip partners uniformly at random from the entire group membership, using gossip to keep the entire membership data current at all nodes.

Modern distributed systems have grown to be orders of magnitude larger than the original Clearinghouse system. As a result, scalability of membership maintenance algorithms has become a serious limitation and much effort is being devoted to gossip-based algorithms that do not require knowledge of the full group membership at each node.

Broadly, there are two bodies of work relevant to this paper:

1. *Rigorous analysis of properties of gossip protocols.* Most existing work ignores the practical difficulty of implementing gossip partner selection distributions that require full knowledge of group membership. Such work includes [4, 10, 11, 12] and many others.

2. *Design of practical algorithms that scale beyond the limits imposed by full group membership.* A number of different approaches have been taken:

   - Nodes are clustered into (possibly overlapping) subgroups, with uniform gossip partner selection within subgroups, with a mechanism to propagate information between subgroups. Astrolabe [15] takes this approach, partitioning nodes according to an explicit, administrator-defined hierarchy.
   - The choice of gossip partner is limited using properties of the real underlying network topology. This is the approach taken by [13].
   - Instead of full group membership, the algorithm maintains a small *local view* of the membership at each node. The local views are chosen to maintain desired properties of the connection graph (e.g. it is k-connected, or it is an expander). The algorithm must respond to membership changes by updating some local views as needed to maintain these desired properties. This is the approach taken in this paper. Other examples include [6, 7, 14].

Up to now, analysis of practical algorithms based on local views has proven difficult, and most work has been validated primarily by simulation. For example, reference [6], the work closest in spirit to our own, contains an analysis of delivery latency and partition probability under the assumption that the view distribution remains uniformly random. In their simulations, however, this assumption is violated. Reference [14] gives a different local view maintenance algorithm. A claim is made for randomness of the local views, but the accompanying simulations do not fully demonstrate this. In fact, we are unaware of any algorithm yielding provably uniform views, and we doubt the existence of such an algorithm.

In this paper we give a new scalable gossip-based algorithm for local view maintenance. Using this membership information, any gossip algorithm using randomly selected gossip partners can be run at the application level, or even be piggybacked on our protocol.

The simplicity and elegance of the protocol were key assets to derive a framework in which we rigorously prove that the expected time until a network partitions is at least exponential in the square of the view size, without assuming the views to be uniform.

We also develop probabilistic bounds on the degree (hence the load) of individual nodes, and argue that protocols lacking our *reinforcement* component eventually converge to star-like networks, whose connectivity depends on a small set of high-degree (hence overloaded) nodes. We also argue that the undirected connectivity graph is an expander, for which application-level gossip multi-cast protocols will converge rapidly.

The remainder of the paper is organised as follows. In Section 2, we present the protocol. In Section 3, we present an intuitive explanation of its robustness. In Section 4, we prove our claims about the probability of partitioning, even in the presence of churn. We give bounds on the load of the nodes in Section 5. Section 6 presents some simulations matching our theoretical results. Section 7 presents alternative protocols, and a discussion of design choices.

2. PROTOCOL

In this section we describe our protocol and give an informal discussion of its behaviour.

2.1 Protocol

Our protocol is based on the notion of a *local view*, a fixed-size random subset of the group membership maintained by each node. Let n be the number of nodes and k the size of a local view. Two additional parameters, f, the *fanout*, and ω, the *weight of reinforcement*, are discussed below. Each node repeatedly updates its local view in *rounds*; in each round, a node s will:

- construct a list $\mathcal{L}_1$ comprising the local views of f nodes chosen at random from the local view of s,
- construct a list $\mathcal{L}_2$ of the other nodes that requested its view during the round,
- create a new local view by choosing k distinct elements at random from $\mathcal{L}_1$ and $\mathcal{L}_2$.[1] The reinforcement weight ω determines how much more likely nodes are to be selected from $\mathcal{L}_2$ than $\mathcal{L}_1$. If ω is 0, nodes in $\mathcal{L}_2$ are ignored; if ω is 1, we make no distinction between $\mathcal{L}_1$ and $\mathcal{L}_2$; and in the limit as ω goes to ∞, all nodes are taken from $\mathcal{L}_2$ if possible.

The protocol can be synchronous (all nodes are updated simultaneously), loosely synchronised (nodes are updated sequentially in some random order, each node being updated exactly once per round) or asynchronous (n nodes chosen uniformly at random with replacement are sequentially updated in the round, so some nodes may be updated more than once, and others not at all). Simulations show no significant differences in behaviour.

Joins and Leaves

Nodes join the network by copying the view of some node. If the rate of nodes joining the network is small, they can all bootstrap from the same node. Since the node's view changes at each iteration, this will not lead to a major imbalance in the graph. And even if it did, the graph automatically re-balances itself as we shall see below.

Since it is unreasonable to expect nodes always to leave gracefully, the protocol has been designed not to require any termination messages from a node leaving the network, and there are no "heartbeat" or "keepalive" messages. This solves the often overlooked scaling issue of the cost of nodes leaving the network. Whenever a node leaves, some dangling edges are left in the network. However, these edges gradually disappear. If the reinforcement weight is at least 1, which it should be, these edges will be purged from the network in an expected approximately k/f iterations. Dangling edges are not an issue for reliability, as we will prove below that the probability of the network partitioning is exponentially small even in the presence of churn.

2.2 Explanation of the functioning of the Protocol

Here we describe the main characteristics of the protocol behaviour. These characteristics are proven in the next few sections, backed by our simulations.

[1]The exact details of duplicate removal are unimportant.

An easy way to summarise the protocol is:

$$\text{New View(u)} = \text{SELECT k nodes from} \left(\underbrace{\text{Views requested by } u}_{\text{Mixing}} \cup \underbrace{\text{Nodes that requested } u\text{'s view}}_{\text{Reinforcement}} \right)$$

Due to its dynamic nature, the protocol automatically adapts and re-equilibrates the network (the connectivity graph) each time the connectivity graph (the directed graph built from the views) does not look like a uniform random graph with constant out degree, regardless of what caused that imbalance. There are two forces responsible for this, corresponding to the two different parts of the protocol:

The views requested by u. We call this *mixing*. Node u pulls views from several nodes, and almost all the nodes of the new view come from the pulled views. This process by itself, as we shall see later, ensures that the graph doesn't partition. This mixing is "pull" only. The push and push-pull alternatives are discussed in section 7.

The nodes that requested u's view. We call this *reinforcement*. The idea behind it is simple: by pulling, node u learns second hand of some nodes in the system. But u learns first hand that the nodes that pulled it exist. The node u positively reinforce the nodes that pulled it by adding them to the list out of which it creates its new view. Under some conditions (that the mixing part provides) this process ensures that the network stays relatively balanced. Also, the process removes older edges, thus ultimately eliminating edges pointing to dead nodes, and adds fresh edges, including some to newly joined members.

Without reinforcement the network would collapse into a star-like structure. This is why the "Weight of Reinforcement" parameter ω should be set to at least 1. Larger is better and will be either 1 or ∞ on a typical implementation.

Labelling this process *pushing* to oppose the pulling process described above would be quite misleading: it is correct that nodes are pushing some information. However, the nodes are pushing *their own names*, not the information in their views as done in the mixing (pulling) part. The crucial part is that nodes are adding their names to the pool of names, it is secondary that this is done by pushing.

These two processes will be analysed in detail in later sections. After connectivity and load balancing, the third desirable property of the protocol is to have views which are uniform samples of the membership set and changing over time so as to emulate each node having the complete membership set, choosing different gossip targets at each iteration. This pseudo uniformity of the views is outside of the scope of the present work, but will slightly be touched upon in Section 7.

3. DYNAMIC BEHAVIOUR OF THE PROTOCOL

In this section, we consider the synchronous version of the protocol and provide an intuitive analysis using expected values. More rigorous proofs are found in the following sections.

3.1 Definitions, Partitioning and Size Estimates

Consider a partition of the set of nodes into two sets A and B. Let x be the fraction of edges from nodes in A that go to nodes in A. Conversely, let y be the fraction of edges from nodes in B that go to nodes in A. Let $\gamma = {}^{|A|}/_{n}$ be the fraction of A nodes. A helpful interpretation is that x is the estimate made by the A nodes of the normalised[2] size of A, and y is the estimate made by the B nodes of the normalised size of A. If the edges were drawn uniformly at random, the expected number of edges to nodes in A (from nodes in either A or B) would be proportional to the size of A. Consider a set S of nodes. If these nodes think A represents 30 % of the nodes, and B the other 70 %, one expects 30 % of the edges from nodes in S to point to A, and 70 % to point to B, even if A is only 10 % of the nodes. The fraction of edges pointing to A is the estimate of the normalised size of A by the nodes of S. Applying this to set A instead of S, we see that x is the estimate by the A nodes of the size of A.

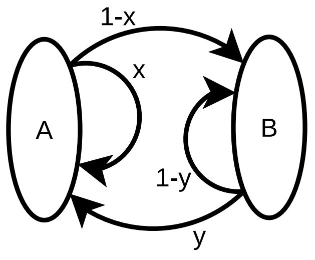

Figure 1: Edge fractions

For our purposes, the graph is partitioned if and only if there are no edges across the partition A-B, that is, both $x = 1$ and $y = 0$, or equivalently $x - y = 1$. However, if both parts agree on their size estimate, however far from the the correct size that estimate may be, we have $x - y = 0 \neq 1$, and (many) edges across the A-B cut. Luckily, the protocol ensures that $x \approx y$. Applying this fact to all possible A-B partitions, we see that the graph cannot be partitioned. In the next subsection we prove that in expected value the convergence to $x = y$ is extremely fast.

Furthermore, this also shows (in expected values) that an application level push-pull gossip will converge rapidly since the diameter of the undirected graph is small. Assume $|A| = \gamma n \leq |B|$. From $x = y$ there are at least $k\gamma n$ edges across the cut, thus making the (undirected) graph an expander.

3.2 Evolution of the estimates of the size of A

3.2.1 Mixing

Neglecting reinforcement, it is easy to see that both A and B converge to the same estimate of the size of A. The act of pulling and merging the views corresponds to asking nodes from both sides for their estimate, then averaging them.

Let x_n and y_n denote the fractions of edges to set A respectively from the A nodes and the B nodes after n iteration of the protocol. Considering the A nodes, with probability

[2]number of nodes in the considered set divided by the total number of nodes

x_n they pull nodes from set A (and get a fraction x_n of edges to A), and with probability $1 - x_n$, they pull nodes from B (and get a fraction y_n of edges to A). Similarly for y_{n+1}. Hence we get, in expected[3] values :

$$\left\{ \begin{array}{lcl} x_{n+1} & = & x_n * x_n + (1 - x_n) * y_n \\ y_{n+1} & = & y_n * x_n + (1 - y_n) * y_n \end{array} \right. \quad (1)$$

Both new estimates of the size of A are a weighted average between A's and B's current estimate, the weight being proportional to the estimated size of A.

By subtracting one equation from the other, we get $x_{n+1} - y_{n+1} = (x_n - y_n)^2$. In other words:

$$(x_n - y_n) = (x_0 - y_0)^{2^n} \quad (2)$$

The convergence to $x = y$ is extremely fast. The only cases not converging are $(x_0 = 1, y_0 = 0)$, we start with a partition and there exists no way to recover, and $(x_0 = 0, y_0 = 1)$, a degenerate case which leads to a partition only because we neglect reinforcement.

The estimate of the size of set A by the A nodes, and that estimate by the B nodes quickly converge to the same value. This value, however, has no tangible reason for being the true size of A. Not only does agreeing on the estimate ensure that the graph is not partitioned, it also provides the necessary conditions for reinforcement to drag the estimate to the correct value of the size. If we were to rewrite equations (1) to take into account the reinforcement, we would notice that the effect of reinforcement is negligible unless $x = y$, and when $x = y$, it pushes the estimates to the correct value γ, but at a much slower rate than (2).

3.2.2 Reinforcement

Consider set A, with both sides agreeing on their estimate of the size of A. Some nodes from A, and some nodes from B are going to pull A nodes. But what proportion of each? Actually, the same proportion as the sizes of A and B. To fix the ideas say $x = y = 30\,\%$. Then $30\,\%$ of the A nodes pull from A and $30\,\%$ of the B nodes pull from A. So A really sees $30\,\%$ of the A nodes, and $30\,\%$ of the B nodes, that is, it sees the same fraction of both sets and thus correctly estimate the fractions of A and B nodes in the network.

This is why reinforcement brings the estimate of the size to the correct value: the reinforcement process injects a little bit of true value in the size estimate at each iteration, thus pushing the estimate towards the correct value. Note, this only works when both sides agree on their estimate of the size of A. Otherwise, A would not be pulled by the correct proportion of nodes. Note also that by symmetry, the same applies to B.

[3]A comparable result for a given partition of the set of nodes can be obtained with high probability, even taking into account the fanout and the resulting dependency on the variables, thanks to some versions of the Chernoff Bounds and their insensitivity to probabilistic dependencies. This result is useful when considering a sudden in-balance in the graph across a given partition of the set of nodes. However, our result was not strong enough to capture the evolution of the whole graph: this required us to take a union bound on all 2^n possible partitions and a multiplicative factor of 2^n has an unhealthy tendency of rendering all bounds meaningless. A work around is to look for the limit probability distribution of the number of edges across each partition, an excessively complex computation.

4. NON PARTITIONING

In this section, we present our framework in which we prove that the expected time before a fraction γ of the nodes partitions away from the rest of the nodes is exponential in γkn, where k is the size of the views and n the number of nodes in the network. Let μ be the churn rate: at each round of the protocol, μ randomly selected nodes die, and μ new nodes join the network. Our proof holds as long as $\mu \ll \gamma kn$. See supporting simulations in Section 6.

The above result on the time to partition proves that if, and when, the network partitions, only a very small component disconnects, and the rest of the network stays connected. Furthermore, the nodes of the small disconnected component can easily detect they have partitioned away by looking at the (lack of) diversity in the content of their views over time. They then attempt to re-join the network. Setting $\gamma = {}^1\!/_n$ in the above formula means considering a single node and its probability of getting disconnected for having only dangling edges in its view. The view size needs to be larger than the churn rate for the expected time until partition to be exponential.

We present a detailed overview of the proof that the network does not partition. Here, for simplicity, we assume the churn rate μ to be zero. The complete proof can be found in appendix A. The modifications to the proof for $\mu \neq 0$ are detailed in appendix B.

4.1 Model and Definitions

4.1.1 Model Intuition

The model is obtained from a slightly modified version of the completely unsynchronised protocol further simplified by some sort of mean-field approximation.

At each iteration in our modified protocol, when pulling, one node chosen uniformly at random replaces a (randomly chosen) node v in its view by a (randomly chosen) node from the view of v. That is, it updates a single element of its view instead of all of them. Reinforcement works as follows: certain times, a node u, chosen uniformly at random, tags a randomly chosen node v from its view. Then v reinforces u by replacing one of the nodes in its view by u.

Our assumption is the following: consider a partition of the set of nodes into two sets A and B. Each node has some number of edges pointing to nodes in A, and some number of edges pointing to nodes in B. We assume a node's edge distribution to be independent of that of its neighbours. For a node u, this means that the number of edges pointing to A its neighbour $v \in$A has is the same as an arbitrary $x \in$A has. In other words, the (out)-edges of A nodes are identically distributed. The same holds for B nodes. The need for this assumption is clear when looking at Equations 3: consider a node u having an edge to a node $v \in$A, the assumption allows us to consider a random node $x \in$A instead of v.

While this is clearly incorrect for a system with a small number of nodes, we believe the approximation to hold when the number of nodes is large.

4.1.2 Model

Consider a partition of the set of nodes into two sets A and B, and two arrays $\mathcal{A}$ and $\mathcal{B}$. Consider the views of the A nodes. For each occurrence of an A node, place a 0 in $\mathcal{A}$, and for each occurrence of a B node, place a 1. Let a be the number of 0's (the number of edges from nodes in

A to nodes in A) and c (as in **c**ross edges) the number of 1's (the number of edges to nodes in B). The length of $\mathcal{A}$ is $a + c = \gamma kn$. Recall $\gamma = {}^{|A|}/_{n}$.

Conversely for $\mathcal{B}$: consider the views of the B nodes. Place a 0 in $\mathcal{B}$ for each occurrence of an A node, and place a 1 for each occurrence of a B node. Let b be the number of 0's, and d be the number of 1's. There are $b + d = (1 - \gamma)kn$ elements in $\mathcal{B}$. Each array represents the concatenation of the views of the nodes of one set where a 0 denotes an edge pointing to a node in A, and a 1 denotes an edge pointing to a node in B.

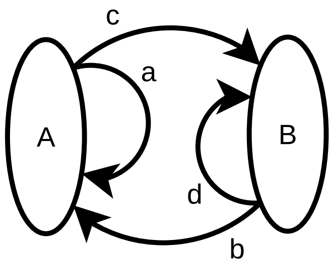

Figure 2: Number of edges
$a + c = \gamma kn$
$b + d = (1 - \gamma)kn$

Mixing (pull) updates can now be made precise. If the element being updated is a 0, the new value is the value of a position chosen uniformly at random from $\mathcal{A}$. If it is a 1, the value is the value of a position chosen uniformly at random from $\mathcal{B}$. As for reinforcement, if the node doing the tagging is in A, put a zero, otherwise, a 1.

The modifications we made to the protocol ensure that a single array entry is updated each time. To correctly mimic the protocol, with probability p the update will be reinforcement, and with probability $q = 1 - p \gg p$ the update will be by pulling, where the parameter p is an increasing function of the protocol parameter "Weight of Reinforcement" ω, taking value 0 when $\omega = 0$, ${}^{1}/_{k}$ when $\omega = 1$, and ${}^{f}/_{k}$ when $\omega = \infty$. In a later claim, we require $p \ll {}^{1}/_{\sqrt{\ln \gamma kn}}$. This inequality is verified even when $p = {}^{f}/_{k}$, for all values of γ by having $f \lessapprox \sqrt{k}$. This is the case since we take f to be a small constant.

4.1.3 State Diagram and Limit Probability

The system is fully characterised by the values a and b, or equivalently, by b and c as we use later. The state diagram is then a rectangular grid of size γkn by $(1 - \gamma)kn$, where the only possible state transitions are from a node to one of the four nearest neighbours on the grid. See Figure 3.

4.1.4 Limit Probability Distribution

The limit probability distribution is 1 for the partitioned state ($a = \gamma kn$, $b = 0$), and 0 for the rest of the grid, because there is no transition out of the partitioned state. We add a transition out of the partitioned state to the two neighbour states ($a = \gamma kn - 1$, $b = 0$) and ($a = \gamma kn$, $b = 1$). Each time the graph partitions, we restart with a single directed edge across the partition. The fraction of time spent in the partitioned state in this never ending process is clearly a lower bound on the expected time until the (real) system partitions.

4.1.5 Claim

In the model described above, the fraction of time spent in the partitioned state of this finite (2-dimensional) Markov Chain is exponentially small in γkn when $\mu \ll \gamma kn$ and $p \ll {}^{1}/_{\sqrt{\ln \gamma kn}}$.

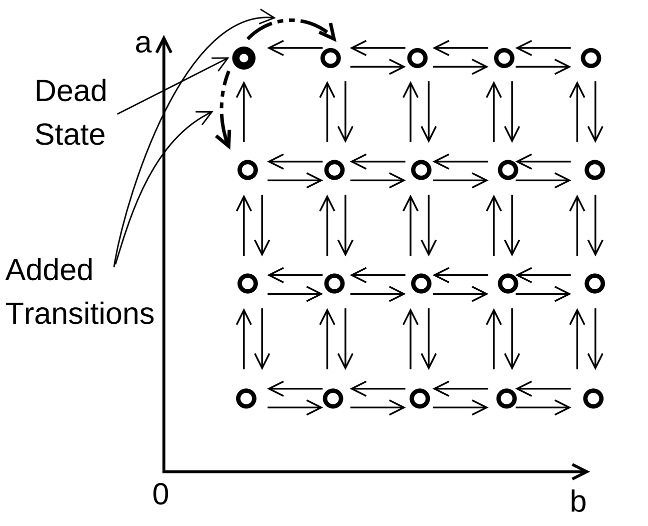

Figure 3: State Diagram

4.2 Equations

The probabilities of changing the number of zeros in $\mathcal{A}$ are:

$$\begin{aligned} \Pr[(a,b) \to (a-1,b)] &= (1-p)\frac{a}{kn}\frac{c}{\gamma kn} + p\frac{b}{kn}\frac{a}{\gamma kn} \\ &= \frac{a}{k^2n^2}\left(\frac{(1-p)c}{\gamma} + \frac{pb}{\gamma}\right) \\ \Pr[(a,b) \to (a+1,b)] &= (1-p)\frac{c}{kn}\frac{b}{(1-\gamma)kn} + p\frac{c}{kn}\frac{a}{\gamma kn} \\ &= \frac{c}{k^2n^2}\left(\frac{(1-p)b}{1-\gamma} + \frac{pa}{\gamma}\right) \qquad (3) \end{aligned}$$

The first term on the right-hand side of the first equation, corresponds to updating by pulling. To decrease the number of 0's in the array $\mathcal{A}$, we need to pick a 0 in $\mathcal{A}$, which happens with probability ${}^{a}/_{kn}$. This 0 denotes an edge pointing to a node in A. We follow that edge and use the target's value for our update. In our model, "following the edge" means "choose a position in $\mathcal{A}$ uniformly at random".[4] We need to pick a one in $\mathcal{A}$, which happens with probability ${}^{c}/_{\gamma kn}$.

With probability p, we do some reinforcement. To decrease the number of zeros in $\mathcal{A}$, we need to have a node from B tag a node from A, which happens with probability ${}^{b}/_{kn}$, and the element in $\mathcal{A}$ receiving the reinforcement needs to be a zero, which happens with probability ${}^{a}/_{\gamma kn}$. The reasoning is similar to increase the number of zeros by 1 in $\mathcal{A}$; hence Equations (3). We shall not present the analysis for the evolution of $\mathcal{B}$, since it can be obtained by symmetry, exchanging a by d, b by c, and γ by $1 - \gamma$.

The following analysis confirms the behaviour suggested by our earlier analysis in expected value. If one neglects reinforcement, the probability to step towards the diagonal line ${}^{a}/_{\gamma} = {}^{b}/_{(1-\gamma)}$ (corresponding to both sides agreeing on their estimate of the size of A), is larger than stepping away. In other words, the system has a natural tendency of re-equilibrating itself. The same is true of the reinforcement, as will be made apparent in Section 5.

[4]This is where our assumption about neighbour independence is being used.

4.3 Proof Intuition and Summary

Denote by $P_{(a,b)}$ the limit probability of state (a,b). Assume for one instant that

$$P_{(a,b)} \Pr[(a,b) \to (a+1,b)] = P_{(a+1,b)} \Pr[(a+1,b) \to (a,b)]$$
$$P_{(a,b)} \Pr[(a,b) \to (a,b-1)] = P_{(a,b-1)} \Pr[(a,b-1) \to (a,b)] \quad (4)$$

holds for all a's and b's. Then

$$\frac{\Pr[(a,b) \to (a+1,b)]}{\Pr[(a+1,b) \to (a,b)]} = \frac{P_{a+1,b}}{P_{a,b}} \approx \frac{\gamma}{1-\gamma}\frac{b}{a}$$

$$\frac{\Pr[(a,b) \to (a,b-1)]}{\Pr[(a,b-1) \to (a,b)]} = \frac{P_{a,b-1}}{P_{a,b}} \approx \frac{(1-\gamma)}{\gamma}\frac{(\gamma kn - a)}{b-(1-\gamma)kn} = \frac{(1-\gamma)}{\gamma}\frac{c}{d} \quad (5)$$

These two terms, $\gamma/(1-\gamma)\frac{b}{a}$ and $(1-\gamma)/\gamma\frac{c}{d}$ are smaller than one for any position above the diagonal, $a/\gamma n \geq b/(1-\gamma)n$, and decrease as we step away from the diagonal and get closer to the partitioned (dead) state. These two approximations come from Equations 3 where we neglected the terms in p.

Consider any path between a state on the diagonal and the partitioned state such that for each transition, either a increases, or b decreases, that is, we only use the two transitions written in (5). Express the ratio between the limit probability of these two states as the cascading product of the limit probability distribution of the intermediary points on the path:

$$\mathcal{R} = \frac{P_{\gamma kn,0}}{P_{a_{\text{start}},b_{\text{start}}}} = \frac{P_{a_1,b_1}}{P_{a_{\text{start}},b_{\text{start}}}}\frac{P_{a_2,b_2}}{P_{a_1,b_1}}\frac{P_{a_3,b_3}}{P_{a_2,b_2}}\cdots\cdot\frac{P_{\gamma kn,0}}{P_{a_{\text{finish}-1},b_{\text{finish}-1}}} \quad (6)$$

This ratio $\mathcal{R}$ is also a product of many right-hand terms from (5), hence making the ratio exponentially small. All the terms in the ratio are non-zero since all states of the Markov Chain are reachable.

Unfortunately, Equations (4) do not hold. However, the cascading product (6) is still useful and meaningful if we replace the equalities of (4) by inequalities in the appropriate direction. We can build a path such that the inequalities are always in the direction of the next point in the path. Sadly, now we can have transitions where b increases or a decreases, that is, some of the numbers in the cascading ratio are now larger than 1. Some algebra work overcomes that issue except when the start state – that we cannot choose anymore – is close to the edge of the grid. Starting on the diagonal, we cannot be close to both bad sides, which solves this last issue.

This however requires p to be moderately small. While this is the case for our protocol and all the other protocols we are aware of, one might want to design a protocol where p is a constant. We believe the result to hold true even with larger p's. The constraint on p came in when we had to consider the worst possible starting point for the path. This was seemingly due to a technical difficulty, not a fundamental one. Any reasonable starting point of the path lifts the requirement on p.

The complete proof can be found in appendix A.

5. NETWORK BALANCE

Each view consists of k distinct nodes. Consider the array $\mathcal{V}$ formed by concatenation of all the views. This array contains kn nodes. Ideally, each node would appear exactly k times. However, some nodes appear more than k times and some less. The number of times node u appears in $\mathcal{V}$ is the same as the number of nodes having node u in their view. This the best indicator of how many times node u will be pulled in one round of the protocol.

Consider $\mathcal{V}$: it is updated at every round by the protocol. Instead of analysing this very complex process, we introduce a simpler model which can be analytically analysed and which captures the essence of the protocol, while being general enough to be valid for other protocols as well. In essence, this is the model from Section 4 with the added assumption that the nodes from sets A and B are indistinguishable. This corresponds to assuming there is agreement on the size estimate of the partition as defined in Section 3.

5.1 Simplified Model

We have the array $\mathcal{V}$ of length kn representing the concatenation of all views. Assume no node leaves or joins the network. At each iteration, a randomly chosen array element is replaced by the following value:

- with probability p pick a node uniformly at random, that is, randomly pick a number between 1 and n where n is the number of nodes and use this value; this corresponds to the reinforcement,
- with probability $1-p$ copy the value of an element chosen uniformly at random from the array; this corresponds to the pulling of the views.

The parameter p is the same as in the previous section. Consider a partition of the nodes into sets A and B. Set the array elements to 0 or 1, denoting whether the element is in A or B.

In this simplified model, when reinforcing, a node replaces an element in its view by a node randomly chosen from the list of nodes currently in the system. When pulling, a node replaces a node in its view by picking a node from an *arbitrary* view. Remember, in Section 4, we had assumed that the distribution of the number of edges to A nodes was the same for all views of A nodes, but different from the distribution of the views of the B nodes. Here, we furthermore assume there is no such difference between A and B: picking from a view from A or B does not matter, the distribution governing the edge distribution is the same for all nodes.

This model is justified when, and only when, both sets agree on their partition size estimates. Then, nodes in A are indistinguishable from nodes in B since all nodes have the same (out-)edge distribution. Because both sides of the partition behave in the same manner, there is no longer the need to distinguish sides when selecting a node, allowing us to obtain stronger results than in Section 4. Again, we make the assumption that the conditioning on neighbours to be negligible: if v is a neighbour of u, we assume the edges of v are independent of those of u. That is, given a node u pointing to a neighbour v, we can consider the edges out-going a random node x instead of the ones out-going v.

Note: in the model, when pulling, an element is updated by selecting some element from the array and copying its value. We assumed the probability distribution of the element selected to be copied is uniform. Uniformity is not crucial. However, being *identical* for all nodes when pulling is. This happens (only) when there is agreement on the estimate.

5.2 Without Reinforcement (case p=0 in the simplified model)

Reinforcement is crucial to the functioning of the protocol. Indeed, without reinforcement, many nodes quickly disappear from the union of the views as we show below. They do not partition out, but they do not get pulled anymore, since all the edges point to the small core of nodes left in the union of the views, which bear the whole load by themselves.

Choose a set X of i nodes. After an expected number of iterations $E[i]$ in the simplified model there will be either no node from X, or only nodes from X left in the views, with

$$E[i] = ikn \sum_{j=1}^{kn-1} \frac{1}{j} - k^2n^2 \sum_{j=1}^{i-1} \frac{i-j}{j(m-j)} \leq ikn \ln kn$$

Proof omitted.

The $ikn \ln kn$ iterations correspond to $i \ln kn$ rounds. This result suggests that the number of rounds it takes for *some i* nodes to disappear from the views is much smaller. Indeed, from our simulations, it looks like the number of iterations before more than 90 % of the nodes have disappeared from the views[5] is logarithmic in n.

We believe the protocol of [14] to be accurately captured by our model with no reinforcement. As such, after the initial phase where the views are filled, the network will rapidly converge to a star, with a core whose size is the view size.

In fact, any membership protocol which re-samples randomly from the views without adding the names of the nodes currently in the system in some way or another is doomed to collapse. Consider $\mathcal{V}$, the concatenation of all the views. Iterating the protocol once corresponds to creating a new $\mathcal{V}$ by some kind of sampling with replacement from the old $\mathcal{V}$. Some nodes might disappear from $\mathcal{V}$ at each iteration. Once a node has disappeared, it cannot reappear without an external mechanism like reinforcement. The diversity of the content of $\mathcal{V}$ decreases over time, and in fact rather rapidly, indicating a star like network. Note that it is theoretically possible to evade the issue by creating a protocol which would correspond to a permutation on $\mathcal{V}$, but this is rather tricky to implement, and doesn't necessarily behave nicely in the presence of nodes leaving or joining the network. Otherwise, one needs to actively add the names of the nodes currently in the network to $\mathcal{V}$, a process we call reinforcement. LwPBCast [6] has some reinforcement, even though not specifically mentioned in the article: each process adds itself to the "subs" field when sending a message. The same holds for Newscast [9] and Cyclon [16] as well: nodes add their own address to their view that they then disseminate.

Note the following interesting behaviour in say the context of news propagation [9]: assume all the "News Events" are created by some subset $\mathcal{S}$ of the nodes. Let only the nodes creating "News Events" reinforce: nodes add their names to their views when creating a "News Events" instead of every T seconds. Then the network will converge to having a core ($\mathcal{S}$, the nodes creating messages) and a fringe (the other nodes). Were these nodes to change, the star would adapt and recenter itself on the nodes currently emitting messages.

[5]We only simulated with views of logarithmic size, fanout didn't have any effect.

5.3 With Reinforcement

When there is reinforcement in the system, one can solve for the limit distribution, with P_i the number of 0's in $\mathcal{V}$:

$$\frac{P_i}{P_0} = \binom{kn}{i} \frac{\gamma p}{1 - \gamma p - \frac{i}{kn}(1-p)} \prod_{j=1}^{i-1} \frac{\gamma p + \frac{j}{kn}(1-p)}{1 - \gamma p - \frac{j}{kn}(1-p)}$$

For all but p extremely small (less than $^2/_{kn}$) the limit probability distribution looks like a peak function centred around $i_{\max} = \gamma kn$. Looking at how fast the function drops around the peak in comparison with the number of partitions of size γ yields estimates of the extrema numbers of occurrence of A nodes in the views. For example: with $\gamma = {}^1/_n$, solving $nP_i \approx 1$ yields the likely expected maximum occurrence of a node in the views. The peak is narrower for higher values of p, but keep in mind that $p \ll 1$ for the protocol to make sense. In Section 3 terms, $i_{\max}$ is the correct value for the estimate of the size of the partition.

This analysis shows that once both sides have agreed on estimates of the size of the partition, the estimate will converge to the correct value and stay there, ensuring that the network stays well balanced.

6. SIMULATIONS

Simulations for large numbers of nodes have shown that the performance of the protocol is quite good. For $2^{17} \approx 100,000$ nodes, view sizes of 17, a fanout of 3, and a loosely synchronised system, the maximum in-degree was always below 4.5 times that of a random graph and the standard deviation was not more than 3.2 times larger than that of a random graph. These values would improve with increased fanout, but even a fanout of 2 gives satisfactory performance.

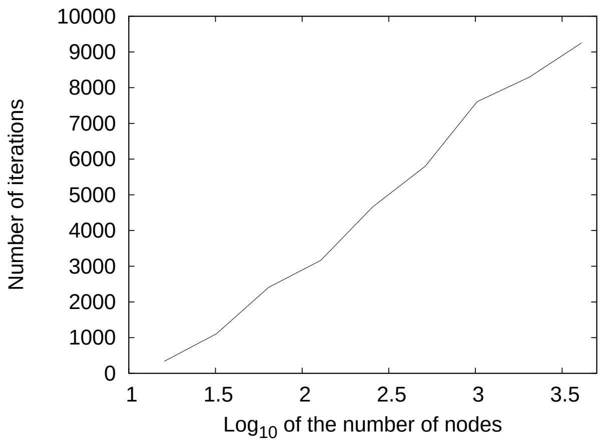

Figure 4: Number of iterations until partitioning

We were interested in matching our theoretical results about partitioning and churn. We ran simulations evaluating the number of iterations until partitioning. By partitioning, we include single nodes getting disconnected for having only dangling edges. Unsurprisingly, increasing view sizes or decreasing the churn increased the number of iterations before partitioning. More interestingly, so did increasing the number of nodes *without* increasing the view size. For example, with no churn, the average number of iterations until partitioning with a view of 4 elements and a fanout of 2 was respectively 2×10^5, 3×10^6 and 7×10^6 for 16, 64 and 256 nodes. Due to the exponential nature of the phenomena, it is only possible to simulate for small view sizes and / or high churn rates. Also, increasing the fanout from 2 to 3 very significantly increased the time to partition. More details will appear in [1].

In Figure 6, we verified the scaling properties of the protocol. With n the number of nodes in the system, we set the view size to $k = \log_2 n$ and the churn to $\mu = {}^{n}\!/_{32}$. Such a high churn is required to actually see partitioning in a reasonable time. Even though there were over a 120 runs for each point (except 40 for the last one), the standard deviation was of the same order of magnitude as the average. The fact that the curve is increasing suggests that the system scales. It also suggests that the requirement $\mu \ll k$ of our theorem may not be necessary.

The size of the small partition decreases when the churn increases. It goes from an average of 5.8 when there is no churn down to 1 or 2 for high churn.

7. ALTERNATIVE PROTOCOLS

There were several choices made in the design of the algorithm: push, pull, randomised, etc. Here we explore some of the alternatives and cite some relevant work. Reference [8] simulates many of these alternatives. Note however that they do not distinguish between reinforcement and pulling (view mixing), using the same process for both. We point out their results where applicable. SCAMP [7] and follow up papers explore mechanisms to dynamically adjust the view size. This is outside the scope of our work.

7.1 Reinforcement

Reinforcement was implemented using a push mechanism: every node pushes its name onto the view of some (random) node. The idea is to actively add one's name to a (multi) set defined by the concatenation of the views ($\mathcal{V}$ in section 5 terms) in order to compensate for the drop due to selection randomness in the pulling part, node failures, etc.

The alternative implementation, a pull mechanism, makes little sense. Adding node u to one's view is only useful if u is not already present, but the nodes for which this is the case cannot contact u! As mentioned above, lack of reinforcement yields a star network. This has also been noted in [8].

7.2 Mixing

This part of the protocol was implemented by a pull mechanism for simplicity. Really, it is about mixing the content of the views. Pushing, Pulling or Push-Pull are valid options. For example, the analysis of Section 3 (neglecting reinforcement) yields the exact same result considering a push only mechanism: while the equations 1 are different and rather intractable, one can numerically verify that the expected speed of convergence given by Equation 2 is the same in both push and pull cases. We believe this push analysis to reasonably capture the essence of LwPBCast.

However, we chose Pulling and have the following non-rigorous argument to support our choice. Set reinforcement aside and consider the (directed) connectivity graph $\mathcal{G}$ at $t = 0$. Under a pull mechanism, an element of the view of node i at time t is obtained by a random walk of 2^t steps in $\mathcal{G}$ starting from i. This might suggests the following memoryless property: after $\ln n$ rounds, the views are independent of their initial value.[6]

Under a push mechanism, the random walk in not necessarily directed and can be significantly shorter since the walk can actually backtrack. Reference [8] also notes that Push increases the risk of partitioning when the network grows.

When implementing view mixing and reinforcement as a whole, push-pull ensures the presence of the good features of both.

7.3 Randomisation

Instead of making all choices at random, each view entry could have a time-stamp. Then the random choices (communication partner, replacement) can be based on these time-stamps. Again, see [8] for some simulations. Randomisation ensured that old edges were eventually removed from the views, replaced by fresh ones. The variance in the life span of the edges is the primary source of node in-degree variance. Time-stamps can advantageously replace randomisation, sharply decreasing the in-degree variance. See Cyclon [16] for a protocol quite similar to ours, but using time-stamps.

Conclusion

We have analysed an algorithm for local view maintenance without requiring the assumption of uniformly random views. The strong guaranties offered by our proofs should apply to other protocols as well. Extending them to cover protocols with time-stamps in the views is of particular interest since these protocols have sharp node in-degree concentration.

8. REFERENCES

[1] A. Allavena. *On the Correctness of Gossip-based Membership Protocols.* PhD thesis, Cornell University, 2005.

[2] K. Birman and T. Joseph. Exploiting virtual synchrony in distributed systems. In *Proc. 11th ACM Symp. on Operating Systems Principles*, 1987.

[3] Fan Chung. Laplacian and the cheeger inequality for directed graphs. *Annals of Combinatorics*, to appear.

[4] C. Cooper, M. Dyer, and C. Greenhill. Sampling regular graphs and a peer-to-peer network. In *Proc. of 16th ACM-SIAM Symp. on Discrete Algorithms*, 2005.

[5] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proc. 6th ACM Symp. on Principles of Distributed Computing*, 1987.

[6] P. Eugster, R. Guerraoui, S. Handurukande, A.-M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems*, 21(4), Nov 2003.

[7] A.J. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-peer membership management for

[6]Developing this argument suggests modifying the protocol to only update a randomly chosen half of the view at each update so as to make the random walk lazy, ignoring the fact that the walks for different elements are correlated, and a liberal application of the results of [3]. Fan Chung proves that lazy random walk on directed regular graphs mix in small polynomial time. Here our graph is close to regular.

gossip-based protocols. *IEEE Transactions on Computers*, 52(2), Feb 2003.

[8] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In *Proc. 5th ACM/IFIP/USENIX International Middleware Conference*, 2004.

[9] M. Jelasity, W. Kowalczyk, and M. van Steen. Newscast computing. Technical report, Vrije Universiteit Amsterdam, November 2003.

[10] R. M. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *Proc. 41st IEEE Symp. on Foundations of Computer Science*, 2000.

[11] D. Kempe and J. Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In *Proc. 43rd IEEE Symp. on Foundations of Computer Science*, 2002.

[12] D. Kempe, J. Kleinberg, and A. Demers. Spatial gossip and resource location protocols. In *Proc. 33rd ACM Symp. on Theory of Computing*, 2001.

[13] M.-J. Lin and K. Marzullo. Directional gossip: Gossip in a wide area network. In *Proc. 3rd European Dependable Computing Conference on Dependable Computing*, 1999.

[14] K. Shen. Structure management for scalable overlay service construction. In *Proc. 1st USENIX/ACM Symp. on Networked Systems Design and Implementation*, 2004.

[15] R. van Renesse, K. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2), May 2003.

[16] S. Voulgaris, D. Gavidia, and M. van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, to appear, 2005.

APPENDIX

A. EXPONENTIALLY SMALL PROBABILITY OF PARTITIONING (PROOF)

Non-negative-Flow Path

There exists a path $\mathcal{P} = \{(a_0, b_0), (a_1, b_1), \ldots, (a_i, b_i), \ldots, (a_l, b_l)\}$ between any initial state (a_0, b_0) and any final state (a_l, b_l) such that, for each transition $(a_i, b_i) \to (a_{i+1}, b_{i+1})$ on the path, we have $P_{a_i,b_i} \Pr[(a_i, b_i) \to (a_{i+1}, b_{i+1})] \geq P_{a_{i+1},b_{i+1}} \Pr[(a_{i+1}, b_{i+1}) \to (a_i, b_i)]$.
This can be interpreted as having a non-negative (probability) flow between (a_i, b_i) and (a_{i+1}, b_{i+1}).

Proof: Consider the set S of states reachable from (a_0, b_0). By reachable we mean there exists a path from (a_0, b_0) satisfying our non-negative flow constraint. If S is not the whole set, there is a positive (probability) flow from the outside to S, since all transitions into S are positive. This situation violates the steady state definition (no sinks, flows are at equilibrium).

Simplified Path

Let us define an upper parallel $\mathcal{D}$ to the diagonal $^a/_\gamma = {}^b/_{(1-\gamma)}$:

$$\mathcal{D}: \quad a = \frac{\gamma b}{1-\gamma} + 0.5\gamma kn \tag{7}$$

We consider a non-negative flow path between a random start point of $\mathcal{D}$ and $(\gamma kn, 0)$ and simplify it by getting rid of all loops, if any. Furthermore, we only keep the tail of the path, the part between the last time it intersects $\mathcal{D}$ and the partitioned state. This ensures that our path $\mathcal{P}$ stays in the region above $\mathcal{D}$. Our choice of the diagonal makes the bound in (5) less than $^1/_2$. Keeping only the tail also ensures that $\mathcal{P}$ does not spiral around the starting point, which we need later for the algebra to work out.

Ratio and Analysis

We break $\mathcal{R} = \mathcal{R}_\mathcal{A} \cdot \mathcal{R}_\mathcal{B}$ from (6) into two parts, $\mathcal{R}_\mathcal{A}$ for all the transitions on the path where the change in the number of 0's and 1's is in $\mathcal{A}$ (that is, a and c change, b and d are constant) and $\mathcal{R}_\mathcal{B}$ when the changes are is $\mathcal{B}$.

$$\mathcal{R}_\mathcal{A} = \prod_{\substack{(a_i,b_i)\in\mathcal{P}\\ b_i=b_{i+1}}} \frac{P_{a_{i+1},b_i}}{P_{a_i,b_i}} \leq \prod_{\substack{(a_i,b_i)\in\mathcal{P}\\ b_i=b_{i+1}}} \frac{\Pr[(a_i,b_i) \to (a_{i+1},b_i)]}{\Pr[(a_{i+1},b_i) \to (a_i,b_i)]}$$

Group the terms for a given value of a

$$\begin{aligned}\mathcal{R}_\mathcal{A}(a) &= \prod_{\substack{b,\text{ such that}\\ \{(a,b)\to(a+1,b)\}\in\mathcal{P}'}} \frac{P_{a+1},b}{P_{a,b}} \prod_{\substack{b,\text{ such that}\\ \{(a+1,b)\to(a,b)\}\in\mathcal{P}'}} \frac{P_a,b}{P_{a+1,b}} \\ &\leq \prod_{\substack{b,\text{ such that}\\ \{(a,b)\to(a+1,b)\}\in\mathcal{P}'}} \mathcal{UP}(a,b) \prod_{\substack{b,\text{ such that}\\ \{(a+1,b)\to(a,b)\}\in\mathcal{P}'}} \frac{1}{\mathcal{UP}(a,b)}\end{aligned} \tag{8}$$

so that $\mathcal{R}_\mathcal{A} = \prod_a \mathcal{R}_\mathcal{A}(a)$, and with

$$\mathcal{UP}(a,b) = \frac{\Pr[(a,b) \to (a+1,b)]}{\Pr[(a+1,b) \to (a,b)]} = \frac{c}{a+1} \cdot \frac{\frac{(1-p)b}{1-\gamma} + \frac{pa}{\gamma}}{\frac{(1-p)(c-1)}{\gamma} + \frac{pb}{\gamma}} \tag{9}$$

$\mathcal{UP}(a,b)$ is a monotonically increasing function of b, for all but large values of a where it is monotonically decreasing. From now on, it is easier to work with c instead of a, so we shall mostly use c, keeping in mind that $a + c = \gamma kn$, and that the partitioned state occurs when $b = c = 0$. Let $c_0 = \gamma kn - a_0$ be such that $\mathcal{UP}(a,.)$ is increasing for $a \leq a_0$, and decreasing for $a \geq a_0$. We have

$$c_0 = \frac{p^2(1-\gamma)kn + (1-p)^2}{(1-p)^2 + p\,{}^{(1-\gamma)}/_\gamma} \approx \frac{p^2(1-\gamma)kn}{(1-p)^2} \leq \frac{f^2 n}{k} \tag{10}$$

The terms in $\mathcal{R}_\mathcal{A}(a)$ correspond to the points of intersection of $\mathcal{P}$ with the line for constant a, with the following orientation: if the direction is towards the partitioned state, then it is "Up", and the corresponding term is $\mathcal{UP}(a,b)$. Otherwise, it is "Down" with $^1/_{\mathcal{UP}(a,b)}$. Consider the terms of $\mathcal{R}_\mathcal{A}(a)$ sorted in increasing order of b. We have a succession of "Up", "Down", "Up", "Down" etc. There is an odd number of terms in $\mathcal{R}_\mathcal{A}(a)$ when $c \leq c_{\text{start}}$ (we finish with a "Down"), and an even number otherwise. The fact that "Up" and "Down" alternate is a property of our path, coming from a simple topological argument.

Since $\mathcal{UP}(a,.)$ is monotonic, when there is an odd number of terms, we can always combine an "Up" with the next (or previous) "Down" to form ratios less than one. We are left with the last (or first) "Up" unpaired. Because we are above the upper diagonal defined in (7), this last term is less than a half, except for two exceptions where it can be bounded by 2: on the edge of the grid and for the hand-created transition out of the dead state.

When there is an even number of terms in $\mathcal{R}_{\mathcal{A}}(a)$, when $\mathcal{UP}(a,.)$ is increasing, we pair each "Up" with the following "Down", getting ratios less than one. When $\mathcal{UP}(a,.)$ is decreasing, we pair the first "Up" with the last "Down", then each "Down" with the next "Up". All these ratios will be less than one, except the first one for which some algebra shows that it is bounded by $1 + \frac{p(1-\gamma)kn}{1-p}$.

Putting everything together, we get

$$\mathcal{R}_{\mathcal{A}} \leq \left(\frac{1}{2}\right)^{c_{\text{start}}} \left(1 + \frac{p(1-\gamma)kn}{1-p}\right)^{\max(c_0 - c_{\text{start}},0)}$$

We get $\mathcal{R}_{\mathcal{B}}$ by symmetry. Combining both, we have:

$$\mathcal{R} \leq \left(\frac{1}{2}\right)^{b_{\text{start}}+c_{\text{start}}} \left(1 + \frac{p}{1-p}\gamma kn\right)^{\max(b_0 - b_{\text{start}},0)} \times \left(1 + \frac{p}{1-p}(1-\gamma)kn\right)^{\max(c_0 - c_{\text{start}},0)}$$

At most one of the two right most terms is raised to a non-zero power. When $\gamma \leq 1/2$, our bound is worst, that is, the above ratio is largest for $b_{\text{start}} = 0$, yielding $c_{\text{start}} = \gamma kn/2$. By symmetry for equation (10), we have $b_0 = (p/1-p)^2\gamma kn$. The bound simplifies to:

$$\mathcal{R} \leq \left(\frac{(1+\frac{p}{1-p}\gamma kn)^{(\frac{p}{1-p})^2}}{\sqrt{2}}\right)^{\gamma kn}$$

As long as $(1-p/p)^2 \leq \ln kn$ the upper term is negligible. This is the case since we set $p \geq 1/k$.

We now apply a union bound on all possible partitions of size γ. This multiplies our bound by something negligible as long as $k \geq \mathcal{O}(\ln n)$. Finally:

$$\mathcal{R} \leq \left(\frac{1}{2}\right)^{\gamma kn/2}$$

B. CHANGES TO APPENDIX A AND SECTION 4 TO INCLUDE CHURN

We detail here the changes necessary to make the proof outlined in Section 4 apply when there is a churn rate of $\mu \neq 0$. In evaluating whether a set A of nodes partitions away from the network, it makes sense to assume that these nodes don't die. In our proof, all the nodes joining and dying are B nodes, the A nodes do not change.

Handling dead edges

The definitions of a, b, c and d are unchanged. Let α be the number of dangling edges in A, and β the number of dangling edges in B. We have $a+c+\alpha = \gamma kn$, and $b+d+\beta = (1-\gamma)kn$.

We cannot analyse this 4-dimensional model: we do not know how to prevent the path $\mathcal{P}$ from spiralling around the start point. Instead, we assume the fraction of dead edges to be constant: $\alpha = \lambda c$ and $\beta = \lambda b$. Edges pointing to A nodes cannot be dangling since the A nodes don't fail.

In steady state, the number of dangling edges removed and created at each iteration of the protocol are equal. We have $\lambda = \frac{\mu}{pn}$.

Equations

There were $(1-p)kn$ pulls and pkn reinforcements per round. Now there are also $(b+c)\frac{\mu}{(1-\gamma)n}$ creations of dangling edges per round. Dangling edges are removed by the reinforcement process. In $\mathcal{B}$, the view of a failing node needs to be removed, replaced by the view of a joining node. It is easier to assume that the view of a failing node is taken over by a joining node. Not doing so would mean adding an extra term (insignificant in the end) to the following probabilities.

$$\Pr[(b,c) \rightarrow (b,c-1)] = \frac{\frac{c\mu}{(1-\gamma)n}}{kn + \frac{(b+c)\mu}{(1-\gamma)n}} + \frac{kn}{kn + \frac{(b+c)\mu}{(1-\gamma)n}} \left(p\frac{a}{kn}\frac{c}{\gamma kn} + (1-p)\frac{c}{kn}\frac{b+\beta}{(1-\gamma)kn}\right)$$

$$\Pr[(b,c) \rightarrow (b,c+1)] = \frac{kn}{kn + \frac{(b+c)\mu}{(1-\gamma)n}} \left(p\frac{b}{kn}\frac{a+\alpha}{\gamma kn} + (1-p)\frac{a}{kn}\frac{c}{\gamma kn}\right) \quad (11)$$

Up down ratio

For $\mathcal{UP}(a,b)$, we now have: Note that $a+c$ do not add up to γkn anymore.

$$\mathcal{UP}(b,c) = \frac{p\frac{\lambda ac}{\gamma} + (1-p)\frac{bc}{(1-\gamma)(1+\lambda)} + (1-p)\frac{\lambda ckn}{1+\lambda} + \frac{ck\mu}{(1-\gamma)}}{(1-p)\frac{(a+1)(c-1)}{\gamma} + pb(kn - \frac{c-1}{\gamma})}$$

For our results to hold, $\mathcal{UP}(b,c)$ needs to be less than one, meaning that we are less likely to step toward the partitioned state than away. The main difference with equation (9) in Section 4 is the third and fourth terms of the numerator, corresponding to the creation of dangling edges. They need to be small – otherwise the result won't hold – and they are.

The monotonicity of $\mathcal{UP}(a,.)$ changes direction at c_0 where $c_0 \approx \frac{(1-\gamma)p^2kn+\gamma k\mu}{(1-p)^2}$ By symmetry we get b_0:

$$b_0 \approx \frac{\gamma p^2 kn + (1-\gamma)k\mu}{(1-p)^2} \quad (12)$$

Again, our bound on $\mathcal{R}$ is worse (for $\gamma \leq 1/2$) when $c_{\text{start}} = \gamma kn/2$ and $b_{\text{start}} = 0$. The bound is:

$$\mathcal{R} \leq \frac{poly(n)^{b_0}}{2^{\gamma kn/2}} = \left(\frac{poly(n)^{b_0/\gamma kn}}{\sqrt{2}}\right)^{\gamma kn}$$

The ratio $b_0/\gamma kn$ needs to go to 0 for the bound to be small.[7] From (12), dropping irrelevant terms we have the claimed result when the following ration is small:

$$\frac{b_0}{\gamma kn} \approx \frac{\mu}{\gamma kn} \ll 1$$

Comment on model:

The case of nodes dying and joining in *both* sets of the partition is a straight forward modification. The third term of equation (11) changes, and α and β are now constant. The algebra giving b_0 and c_0 is also simpler.

[7]Being less than $Cst \ln n$ is sufficient