A randomized algorithm for the joining protocol in dynamic distributed networks*

Colin Cooper† Ralf Klasing‡ Tomasz Radzik§

September 21, 2005

Abstract

We describe a randomized algorithm for assigning neighbours to vertices joining a dynamic distributed network. The aim of the algorithm is to maintain connectivity, low diameter and constant vertex degree. On joining each vertex donates a constant number of tokens to the network. These tokens contain the address of the donor vertex. The tokens make independent random walks in the network. A token can be used by any vertex it is visiting to establish a connection to the donor vertex. This allows joining vertices to be allocated a random set of neighbours although the overall vertex membership of the network is unknown. The network we obtain in this way is robust under adversarial deletion of vertices and edges and actively reconnects itself.

One model we consider is a network constructed in this fashion, in which vertices join but never leave. If t is the size of the network, then the diameter of the network is $O(\log t)$ for all t, with high probability. As an example of the robustness of this model, suppose an adversary deletes edges from the network leaving components of size at least $t^{1/2+\delta}$. With high probability the network reconnects itself by replacing lost edges using tokens from the token pool.

*This work was partially supported by Royal Society Grant ESEP 16244.

†Department of Computer Science, King's College London, London, UK, email ccooper@dcs.kcl.ac.uk.

‡MASCOTTE project, I3S-CNRS/INRIA/Université de Nice-Sophia Antipolis, 2004 Route des Lucioles, BP 93, F-06902 Sophia Antipolis Cedex (France), email Ralf.Klasing@sophia.inria.fr. Partially supported by the European projects RTN ARACNE (contract no. HPRN-CT-1999-00112) and IST FET CRESCCO (contract no. IST-2001-33135).

§Department of Computer Science, King's College London, London, UK, email radzik@dcs.kcl.ac.uk.

1 Introduction

The basic protocol

In the type of network we consider, each vertex knows only its immediate neighbours. The overall size, vertex list and structure of the network is unknown to any vertex. A central problem for such networks is how new vertices can join the network if there is no accurate current vertex list. Some possible ways of joining might include: join to an existing friend, join to a fixed (predetermined) vertex, use a look-up table from an earlier session. We call such joining behaviour *arbitrary*. We describe a randomized algorithm which overcomes possible consequences of arbitrary joining behaviour, by allocating a more suitable set of neighbours to the joining vertex.

We assume the network has properties which it wishes to maintain in a distributed fashion and that arbitrary joining behaviour is unfavourable to these properties. The simplest set of desirable network properties is: bounded degree (fairness in load sharing), connectivity (ability to communicate) and small diameter (fast broadcasting). Arbitrary joining behaviour can create long paths (join to previous joiner), high degree vertices (join to a fixed vertex), clustering and cliques and thus does not maintain these properties. Rather, it introduces communications bottlenecks which are vulnerable to adversarial attack. However, the network we obtain from our randomized joining algorithm is robust under adversarial attack and actively reconnects itself when edges are removed.

It is well known that some classes of random graphs (eg. random regular graphs [1]) usually have the desirable properties mentioned above (bounded degree, connectivity, small diameter). A good randomized algorithm for assigning neighbours to the joining vertex would be to maintain the network as a random regular graph. However, the vertex membership of the network is unknown, so we cannot directly assign a random subset of the existing vertices as neighbours of the joining vertex. In fact, one problem is to obtain such a random subset of the unknown vertex list.

A full description of the token protocol we use to randomize joining behaviour is given in Section 2. The basic idea is simple, and we give a brief outline here. After joining (as described below), each vertex w donates a fixed number d of tokens to the network containing the address (w) of the donor vertex. These tokens make independent random walks on the network. Because the neighbour structure is randomized by construction, the network is an expander and this random walk is rapidly mixing. Thus the tokens visiting a given vertex are (almost) random in terms of their donor addresses. Suppose a new vertex v joins the network via an initial (arbitrary) contact vertex u. The vertex u collects up a fixed number m of the tokens visiting it on their random walks. These m tokens constitute a (multi-)set of donor addresses A. The new vertex v connects to the donor vertices whose addresses are in A, and drops its initial connection with u. The tokens in A are

deleted, and instead v donates $d = cm,\ c > 1$ tokens with its own address to the network. As tokens can be used at most once, the degree of any vertex is at most $m + d = (c + 1)m$. A vertex distinguishes between the m edges from its original connections to A (out-edges) and any edges it acquires later as a result of tokens it donated to the network (in-edges).

The token approach has significant additional benefits apart from simplicity and randomization of the network structure. Tokens act as a kind of distributed memory of the vertex list which is difficult to destroy. As a consequence the network is robust under adversarial attack. If an adversary disconnects the network by breaking edges (communication links) then available tokens will be used to replace the missing out-edges. This helps to reconnect the network.

We briefly mention other possible applications, not studied here. The network could actively reconfigure itself in an on-going fashion, replacing out-edges by selecting new tokens at suitable time intervals. This makes it difficult for an adversary to learn its structure. The simulations of [7] modeled random networks which replaced edges randomly (as a surrogate for moving to a server whose file content is more interesting to the user). They found this edge replacement improved the efficiency of (repeated) search based on fixed depth flooding. Another application is for a joining vertex to attach text to the tokens it donates, giving its particular interests, thus extending the function of the distributed memory. As tokens visit vertices on the random walk, this text could be scanned for relevant items as an alternative to vertices actively seeking this information by broadcasting.

Related work

The idea of using a random walk to create random networks has been used by [2]. Other approaches to this randomization are given in [17], [12]. The token approach however, is as far as we know, new. We call networks which are constructed using tokens *self maintaining*, as they actively replace broken edges.

Our distributed construction of a random network should find applications in the design of dynamic networks in which evolve in an unstructured and unpredictable way. A typical example of this is the peer-to-peer (P2P) networks of eg. [17], [2], [12]. We briefly summarize what is known about algorithms for coping with arbitrary joining protocols, robustness and related problems in the context of P2P networks.

A P2P network is a decentralised, dynamic network for sharing data and computing resources among the vertices (participants) of the network. The network is dynamic in the sense that vertices join and leave the network. The vertices must follow the protocol of the network, and if they do, then the network should maintain some desirable properties such as connectivity, low degree and small network diameter.

We briefly mention some existing popular P2P networks [6]. An early P2P implementation is Gnutella [8]. This network has no centralised control, but also no explicit mechanism for maintaining low diameter. Although research has shown though that the Gnutella network often has small diameter (see [15]) this is by no means guaranteed.

At the other end of the spectrum w.r.t. centralisation is the (in)-famous Napster [16]. Napster is a centralised system in which all search queries are being processed by a fixed set of server machines, and results are then sent back to the user.

A currently very popular P2P system is Kazaa [11] which uses a network protocol called FastTrack. FastTrack is in some sense between centralised and decentralised: users with fast machines and fast connection to the internet are assigned the role of super nodes, and searches are performed by those nodes only. After a successful search the files are downloaded directly from the users who have the corresponding files. Other popular P2P networks include eDonkey and Overnet. Quoting [4]: "eDonkey2000 publishes to servers that can be set up by anyone. Once the network reaches a certain size these servers become a bottle neck to the performance. Users can no longer search the entire network for things they are interested in. And the servers become more and more bogged down." Overnet overcomes some of these problems by decentralization.

To provide a platform for more efficient searching, one approach is to make a deterministic construction based on known graphs of low diameter such as hypercubes or butterfly graphs and adapt this to the dynamic environment [19], [18]. A particularly good structure for maintaining connectivity is a cyclic list where new arrivals can push in anywhere (or be randomly assigned a location) and the list contracts as vertices leave. An instance of this is the Chord system [20, 13]. A backbone ring of nodes ordered according to the hash values of their identifiers is constructed. Data items are also hashed and are assigned to the node whose key immediately follows their hash key. A logarithmic number of Chordal edges per node (fingers) provides a fast look-up procedure. Such networks are called content-addressable (CAN), and [14] proposes a system, Viceroy, also based on the circular list concept. It embeds butterfly graphs around the ring so that the network has constant degree, whereas Chord has a logarithmic degree.

The difficulties of reconnecting if the network becomes disconnected, are mentioned in [20] where it is suggested it would be good to retain a random set of node addresses to do this. One approach to enable reconnection could be to generate a separate subgraph on the vertices of the cycle consisting of a self maintaining network of the type we propose here.

Another approach is to generate some sort of random graph model. The formative instance of this was the work of Pandurangan, Raghavan and Upfal [17] who proposed a protocol which ensures that the network has bounded degree, and is usually connected with logarithmic diameter. The crucial feature of their protocol

is a central cache, which holds addresses of a random subset of the vertex list and has to be accessed each time a node joins the network. The requirement of having a central cache leads to an infrastructure which is asymmetric and has a potential bottleneck. On the other hand, joining via a central website is the most obvious protocol.

Bourassa and Holt [2] proposed a fully decentralised protocol based on random walks. If a vertex in the network needs an address of a random vertex, then it initiates a random walk and gets the address of the vertex reached at some specified step of the walk. The protocol constructs a 4-regular random graph. Their protocol, however, cannot reconnect the network if it becomes disconnected.

Law and Siu [12] generate a network which is the union of d random edge disjoint Hamilton cycles (cyclic lists). As most random $2d$-regular graphs have logarithmic diameter and (for $d \geq 2$) can be expressed as the union of Hamilton cycles it seems appropriate that such a model can have the basic set of desired properties. The model allows for the distributed construction of the network based on random walks as in [2].

Gkantsidis, Mihail and Saberi [7] simulate various networks based on random graphs, including one with two level clustering. They compare the effectiveness of random walks as a search method for data retrieval against fixed depth flooding.

The robustness of P2P networks under adversarial deletion of vertices (censorship) or edges (communications links) can be addressed in both deterministic and random models. The work of Fiat and Saia [5] models content-addressable P2P networks resilient to adversarial node deletion. It uses a butterfly graph which is robust under deletion of $n/2$ of the n nodes. The network is static and of logarithmic degree, with no explicit node addition mechanism.

Summary of results

In Section 2 we define our protocol. In Section 3 we give the precise statements of our main results, and in Sections 4 – 6 we give the proofs. We briefly summarize the results here.

Our algorithm is randomized, and not all the network properties we describe are deterministic. We use the notation *with high probability* (**whp**) to mean a property holds with probability tending to one, as the network size t tends to infinity. The precise rate of convergence is given in the proofs in all cases. We often suppress the **whp** notation in general discussions of results, for legibility reasons.

We consider two examples of networks which use the protocol: a growing network and a network of fixed size which acts as a FIFO queue. Let t denote the network size. In both cases the algorithm builds the network as a connected random graph $G(t)$ with bounded degree, and diameter $O(\log t)$.

For the growing network, our analysis is as follows. We show that the network

has constant conductivity (is an expander) at all steps in its existence. A random walk on a constant conductivity network is rapidly mixing. After a time $\tau = O(\log t)$ the random walk is (almost) in the steady state and the tokens arriving at any vertex are effectively random. A set of these tokens constitute a random neighbourhood. The $O(\log t)$ diameter follows immediately from the constant conductivity.

For convenience of analysis we assume the network accepts new vertices in batches after waiting $\Theta(\log t)$ steps to let the tokens reach their steady state, followed by a $\sigma = \Theta(\log t)$ joining time. This building of a layered network in batches is a convenience of the analysis. We do not suppose the actual protocol would wait for the steady state.

The random graph we construct does not fit any standard model, as it is constructed in a layered fashion, and we need to establish its conductance properties. Availability of the tokens from a subset of vertices A depends not only on the size of A but crucially on the age of A (as older tokens tend to get used up). This makes the analysis considerably more complex and challenging than in static random graph models. It also has consequences for robustness.

For the growing network we study robustness under adversarial attack. As a random graph the network $G(t)$ has good intrinsic connectivity properties. We show that it cannot be disconnected by the removal of a set of vertices of size $o(t^{1-1/m})$. If edges are deleted leaving components of size larger than $t^{(c+1)/2c}$, the network reconnects by replacing lost edges using tokens from the token pool. The worst case edge cut is to separate the first (oldest) s vertices. For $t^{(c+1)/2c} \ll s \leq t/2$ however, a component of size at least $t[1-e(sm/(t(m-1)))^m]$ reforms by replacing lost edges from the token pool.

2 The token protocol

We assume that every vertex wishing to join the network has an absolute address (not a function of the network) at which it can be reached externally from the network we construct. Eg. the address is an IP address, and the vertices can be contacted over the internet.

The abstract process. When new vertices join the network, they connect to existing vertices by edges. Each edge adjacent to a vertex v has an intrinsic direction, and is classified as either an out-edge or an in-edge, but in either case the edge is a bi-directional communications link. Out-edges (v, w) are established when v joins the network, and their terminal vertices define the multi-set $N^+(v)$ of out-neighbours of v. In-edges (u, v) are established when a vertex u connects to v, and the initial vertices of these edges similarly define $N^-(v)$. To keep the vertex degrees within a fixed range $[m, (c+1)m]$, for some constant integers $m \geq 2$ and $c \geq 2$, we only allow

v to donate $d = cm$ tokens.

Random walk process. In the protocol we propose, *tokens* containing the addresses of available vertices randomly walk through the network. During one time step, for each token coming to a vertex w, a simple *random walk process* at w selects a random edge adjacent to w (not making any distinction between incoming and outgoing edges) and sends the token along that edge.

Joining the network. A new vertex v which wants to join the network approaches an arbitrary vertex w in the network. Vertex w passes to v the first m tokens it currently has. If w has fewer than m tokens, it waits until it accumulates m tokens, and then passes them to v. When v receives m tokens, it tries to establish m edges connecting to the vertices whose addresses are in the tokens (there may be parallel edges). It establishes connections to those vertices which are still in the network. The connection mechanism (unspecified) is external to the network. If the vertex which donated a particular token has left the network, vertex v asks vertex w for more tokens until it establishes m edges. All tokens received from w during this process are removed from the network (each token may be used for establishing only one edge). The initial connection to w is deleted, and vertex v completes the joining protocol by donating $d = cm$ tokens of its own to the network.

Leaving the network. If a vertex wants to leave the network, it simply stops reacting to any communication from its neighbouring vertices. We assume leaving vertices are polite, and thus push any tokens visiting them at the current step back into the network. Each vertex keeps checking whether its neighbouring vertices respond. If a vertex v notices that its neighbour w does not respond, then v behaves as follows: If the lost edge was outgoing from v, then v picks up an additional token to establish a new edge. If the edge was incoming to v it donates a new token of its own to the network.

This protocol also covers the case when vertex w has not left the network but only the connection between v and w has stopped working. In this case one vertex picks up a new token, while the other donates a token.

Simplifications of the basic model. We consider two simplifications of the basic model. The *growth model* and the *first-in first-out (FIFO) queue model*. In either case we assume that the initial network is a multi-graph of three vertices connected in a triangle, and each of them has m outgoing and m incoming edges. We prove that both these models are connected, with diameter logarithmic in the network size (**whp**).

Growth model. In this model, vertices join the network but never leave. Thus the size t of the increases over time. We study how this network restructures itself under adversarial deletion of vertices and edges.

FIFO queue model. Informally we can think of this network as an *obselescence network* in which inter-connected equipment fails or is replaced at a certain age, and this is effected in a distributed manner. Once the network has grown to a fixed size t, as a new vertex arrives the oldest vertex leaves. Vertices pass any spare tokens to neighbours before departing. Broken out-edges are replaced in the usual way, by sampling tokens from the network.

Batch protocol. For convenience of analysis we assume that the vertices are added to the network in batches. Let T denote the number of steps made by the random walk process since the start of the network, and let t denote network size. Let ΔT denote time between batches measured in steps of the random walk process.

Assume the network adds new vertices in batches during intervals which terminate at steps $T_i,\ i = 0, 1, ...$ We call an interval $(T_i, T_{i+1}]$ *epoch* i. Let $(T, T + \Delta T]$ be any of these intervals, where $\Delta T_i = T_i - T_{i-1}, i = 1, ...$ and $\Delta T_0 = T_0$.

Let $G(T) = (V(T), E(T))$ be the network at step T. At time $T+\Delta T$ the network updates to $G(T + \Delta T)$ to include those new vertices which have successfully joined the network between T and $T + \Delta T$.

Let t denote the number of vertices $|V(T)|$ in the network at step T of the random walk. The calculations in subsequent sections are all made in terms of network size t. If we wish to focus on the size of the network, we will write $G(t) = (V(t), E(t))$ where $G(T) \equiv G(t)$. For the purpose of analysis, the vertices are indexed with consecutive positive integers according to the order of their arrivals, with arbitrary ordering within batches. Thus $V(t) = \{1, 2, \ldots, t\} \equiv [t]$. Let $\mathcal{T} = (t_0, t_1, t_2, \ldots)$ where t_i is the index of the last vertex in batch i. Recall that at time $T = 0$ (the beginning) the network is a multi-triangle of size $t = 3$.

Let Δt be the number of vertices added between T and $T+\Delta T$. Δt can be much larger than ΔT which we only require to be at least logarithmic in the number of vertices to ensure mixing (ie. $\Delta T \geq K \log t$ where K is a large positive constant). The size of Δt depends on the arbitrary (unknown) joining protocol. We assume in our analysis that $\Delta t = o(t)$. In fact an assumption that $\Delta t \leq \rho t$ for some constant ρ would do, but the proof would be more complicated.

The network $G(T)$ contains tokens (with addresses) which are making independent random walks on $G(T)$. After $\Delta T/2 \geq (K \log t)/2$ steps the distribution of tokens is very close to steady state. Tokens visiting a vertex v between $T + \Delta T/2$ and $T + \Delta T$, can be retained to pass to new vertices. A new vertex which has acquired m tokens by $T + \Delta T$ joins the network (otherwise it waits).

To join the network at time $T+\Delta T$, vertex u connects to the vertices $\{v_1, ..., v_m\}$ whose addresses are given on the m tokens. Each token generates one (possibly

parallel) directed edge. Edge (u, v_j) as an out-edge of u and in-edge of v_j. At $T + \Delta T$, a joining vertex u donates cm new tokens to the network labeled with the address of u. Each of these new tokens makes an independent random walk along with the existing tokens. Each joining vertex in $V(T)$ uses up m tokens. There are $(c-1)m|V(T)|$ active tokens walking in $G(T)$.

The deletion of a vertex u or edges incident with u means that the neighbours $v \in N^+(u)$ may have lost in-edges (u, v), and neighbours $w \in N^-(u)$ may have lost out-edges (w, u). The vertex w replaces a lost out-edge (w, u) by choosing a new token from the network. A token is donated to the network by v for each in-edge (u, v) lost. This maintains the total $(c-1)m|V(T)|$ of active tokens in $G(T)$.

Realization of the batch protocol. To allow the walk to mix, we need ΔT to be $\Theta(\log t)$ steps of the random walk, measured in terms of the network size t. As the vertices in the network do not know the size of the network, how can they figure out an appropriate value of ΔT?

One possible solution to this problem is to decide in the epoch lengths ΔT_i from the predicted (or desired) growth $\widehat{t} = f(T)$ of the network. The value of T_i and the function $f(T)$ are made known to all vertices on joining. The calculations $\widehat{t_i} = f(T_i)$ and $T_{i+1} = T_i + \lfloor f^{-1}(K \log \widehat{t_i}) \rfloor$ then give $\Delta T_i = T_{i+1} - T_i$. If the actual growth substantially exceeds $\widehat{t}$ there may be queueing; a separate problem which we do not analyse here. We simply assume that $\Delta T \geq K \log t$.

3 Results

Let $m \geq 400$ and $c \geq 20$. The large value of m is due to simplifying assumptions in the proofs. As mentioned above, we work in the batch model and assume that ΔT is always at least $K \log t$, for a suitably large constant K.

3.1 Network properties

Theorem 1 *The growth network model $G(T) \equiv G(t)$ has bounded maximum degree $(c+1)m$ and it is connected. For suitably large m and c and any constant $\varepsilon > 0$, there exists a constant $\gamma = \gamma(m, a, \varepsilon)$ such that with probability at least $1 - \varepsilon$, the diameter of $G(t)$ is at most $\gamma \log t$ for all t.*

In Section 4 we show that the conductance $\Phi(T)$ of $G(T)$ is constant (**whp**) and the diameter follows from this (see Lemma 3). The definition of conductance is given in (3) below. As $\Phi(T)$ is constant $G(T)$ is rapidly mixing and we can choose ΔT so that the distribution of any token on $G(T)$ at $T + \Delta T/2$ is (almost) stationary. Thus the tokens passed to the new vertices are a random sample (without replacement) of the tokens in $G(T)$. This inductively ensures that the conductance $\Phi(T + \Delta T)$

of graph $G(T + \Delta T)$ is also constant **whp**. To prove Theorem 1, which requires a logarithmic diameter for all t, we need to make careful probability estimates at each step. This is where most of the work in the proof occurs.

Theorem 2 *The FIFO queue network $G(T)$ has bounded maximum degree $(c+1)m$ and with probability $1 - O(t^{-2})$ is connected and has diameter $O(\log t)$.*

The proof of Theorem 2, given in Section 5, is in many ways similar to that of Theorem 1.

3.2 Growth model: Adversarial deletion

We study the robustness of the network under the adversarial deletion of edges and vertices. Clearly we cannot prevent disconnection of the network by edge or vertex deletion, but we show that in certain cases the network remains connected or is able to re-connect itself. The reconnection can be either *implicit* as vertices replace broken out-edges using existing tokens, or *eventual* as joining vertices form bridges between the components. We consider only implicit reconnection here. Any new arrivals can only help to reconnect existing components but we ignore this effect and insist that repairs are effected by the existing network. We emphasize that all decisions in the network are distributed and the global structure of the graph (eg. disconnection) is unknown to the individual vertices.

Various types of adversarial deletion of edges or vertices can be considered. For example an edge cut between sets of vertices S and $V(t) \setminus S$, or breaking the network into components of at least some given size, or isolating individual vertices. We examine the following cases:

(a) Deletion of the first s vertices $[s] = \{1, 2, ..., s\}$.
(b) An edge cut between $[s]$ and $G(t) \setminus [s]$.
(c) Disconnection of the network into components of size at least s.

Our reason for focusing on the set $[s]$ is as follows. Deletion or disconnection by edge cuts of the first s vertices $[s]$ is in expectation the most damaging option compared to any other set of size s. The set $[s]$ has the highest expected in-degree, so the most edges are broken. Under the protocol that vertices replace broken out-edges, and donate tokens for broken in-edges, the set $[s]$ has no means of actively re-connecting itself to $[s + 1, ..., t]$ as none of its out-edges are broken. We prove in Section 6 that the expected number of tokens from $[s]$ remaining unused in the network $G(t)$ is $\sim (c - 1)ms(s/t)^{1/(c-1)}$. Thus when $s = O(t^{1/c})$, most tokens from $[s]$ have been used.

Theorem 3 *Let $\delta = 1/2c$. The following results hold* **whp***:*

(a) For $s = o(t^{1-1/m})$, deletion of the set of $[s]$ leaves the network $G(t) \setminus [s]$ connected.

(b) Let $t^{1/2+\delta} \log t \leq s \leq t/2$. If the edges between between $[s]$ and $G(t) \setminus [s]$ are deleted, then the network reconnects implicitly except possibly for a set of vertices of size at most $te(sm/(t(m-1)))^m$.

(c) If edges are deleted, breaking the network into components of size at least $s = t^{1/2+\delta} \log t$, then the network reconnects itself implicitly.

The proof of Theorem 3 is given in Section 6. We do not suggest the results are optimal. Rather they serve to highlight the robustness of the network under adversarial attack.

The results of the theorem are for a one-off reconnection of the network. We do not give any analysis of the subsequent network properties in this paper, neither do we analyse the topic of eventual reconnection here.

4 Analysis of the growth model

In this section we prove Theorem 1. Before proceeding with the proof, we give some definitions and background material we will use.

In many places we have rounded values x down ($\lfloor x \rfloor$) or up ($\lceil x \rceil$). We have omitted the rounding notation if the error introduced is of the same order as in the rest of the analysis.

We frequently use the following approximations: $\sum_{j=s}^{t} \frac{1}{j} = \log t/s + O(1/s)$, $\binom{t}{a} \leq (te/a)^a$, for $0 < x < 1$, $1 - x = e^{-x-O(x^2)}$.

We next state the Chernoff and Hoeffding Inequalities (respectively) for the sum $X = \sum X_i$ of independent $0, 1$ random variables:

$$\mathbf{Pr}(X > \alpha \mathbf{E}X) \leq \left(\frac{e}{\alpha}\right)^{\alpha \mathbf{E}X} \qquad \alpha > e \tag{1}$$

$$\mathbf{Pr}(X < (1-\epsilon)\mathbf{E}X) \leq e^{-\epsilon^2 \mathbf{E}X/2} \qquad |\epsilon| < 1. \tag{2}$$

The Chernoff and Hoeffding Inequalities are also valid for uar sampling without replacement (see eg. [3], [9]), and we often use them in this context.

For a graph $G = (V, E)$ and a vertex $v \in V$, $d(v)$ is the (undirected) degree of v. For subsets $A, B \subseteq V$, $d(A) = \sum_{v \in A} d(v)$ and $d(A : B)$ is the number of edges directed from A to B. If A and B are disjoint, then $E(A : B)$ denotes the number of undirected edges between A and B, that is $E(A : B) = d(A : B) + d(B : A)$. For $A \subseteq V, \overline{A} = V \setminus A$.

Let $G = (V, E)$ denote a fixed connected graph, and let u be an arbitrary vertex from which a random walk W_u is started. Let $W_u(\tau)$ be the vertex reached at

step τ, and let $P_u^{(\tau)}(v) = \mathbf{Pr}(W_u(\tau) = v)$. Let π denote the stationary distribution (if any) of $W_u(\tau)$, that is, $\pi(v) = \lim_{\tau\to\infty} P_u^{(\tau)}(v)$. For an unbiased random walk on an ergodic (connected, non-multi-partite) undirected graph G, the stationary distribution π exists and is given by $\pi(v) = d(v)/(2|E|)$.

It is important for the structure of the network that the distribution of tokens in the network should be close to stationary when sampled by joining vertices, as this makes all tokens visiting a given vertex equiprobable. We use the method of conductance to prove convergence to stationarity.

The *conductance* Φ of the graph G is defined by

$$\Phi = \min_{S:\pi(S)\leq 1/2} \Phi(S), \qquad \Phi(S) = \frac{E(S:\overline{S})}{d(S)}, \tag{3}$$

where $\pi(S) = \sum_{v\in S} \pi(v)$. It follows from Jerrum and Sinclair [10] that for a walk starting at vertex u

$$|P_u^{(\tau)}(v) - \pi(v)| = (\pi(u)/\pi(v))^{1/2} \left(1 - \frac{\Phi^2}{2}\right)^{\tau}. \tag{4}$$

The ratio $\pi(u)/\pi(v)$ is $d(u)/d(v) \leq c + 1$. Thus from (4) above, if $V(T) = t$ and $\tau = (2K/\Phi^2)\log t$, after τ steps the random walk will be within $O(t^{-K})$ of the steady state. There is a technical point here: The result of [10] assumes that the walk is *lazy*, and only makes a move to a neighbour with probability $1/2$ at any step. The purpose of this is to ensure that the stationary distribution of a random walk exists so that (4) holds equally for bipartite graphs. We can assume in our analysis that our random walks are lazy, ie. the random walk process only forwards the tokens with probability $1/2$ at any step. The analysis can be easily adapted to non-lazy walks since our protocol creates networks which with high probability are not bipartite.

The graphs $G(t)$ we consider are undirected for random walks, but each edge has an underlying direction. All vertices have out-degree m so the total number of edges is mt, and thus $\pi(v) = d(v)/2mt$. For any set A, we have and $d(A) = m|A| + d(A:A) + d(B:A)$, by counting in-degree and out-degree of the vertices of A. By counting arcs originating in A we have $m|A| = d(A:A) + d(A:B)$.

We need the following lemma, which gives an upper bound on the size of a subset of vertices A, if $\pi(A) \leq 1/2$.

Lemma 1 *If $\lambda > 0$, $A \subseteq V(t)$ and $\pi(A) \leq 1/2$, then either $\Phi(A) > \lambda$ or $|A| \leq (1+\lambda)t/2$.*

Proof Let $B = V(t) - A$. Assuming that $\Phi(A) \leq \lambda$ we have

$$\begin{aligned} \lambda &\geq \Phi(A) = \frac{d(A:B) + d(B:A)}{2d(A:A) + d(A:B) + d(B:A)} \geq \frac{d(A:B)}{2d(A:A) + d(A:B)} \\ &= \frac{m|A| - d(A:A)}{m|A| + d(A:A)}. \end{aligned} \tag{5}$$

Since $\pi(A) = d(A)/(2mt)$ then as $\pi(A) \leq 1/2$,

$$\begin{aligned} d(A:A) &= d(A) - m|A| - d(B:A) \leq d(A) - m|A| \\ &\leq mt - m|A|. \end{aligned} \tag{6}$$

Inequalities (5) and (6) imply that $|A| \leq (1+\lambda)t/2$. □

Before proceeding there is a point we need to address: Recall that $\mathcal{T} = (t_0, t_1, t_2, \ldots t_i, \ldots)$ where t_i is the size of the network at step T_i of the random walk process corresponding to the start of epoch i. The vertices are labeled in the order in which they are added to the network. In terms of the mixing properties of the network the conductivity $\Phi(t)$ is only relevant at vertex steps $t_i \in \mathcal{T}$ and we assume $t \in \mathcal{T}$. Given $t_i, t_{i+1} \in \mathcal{T}$, for $t_i < t \leq t_{i+1}$, the vertices available as neighbours of t are drawn from $[t_i]$. Also, using t instead of t_i introduces a small error in eg. $\pi(v) = d(v)/2mt_i = d(v)/2m(t - o(t))$ and we correct for this in the proofs.

For a subset $A \subseteq V(t)$ we write $A(s)$ for $A \cap V(s)$ and $a(s)$ for $|A(s)|$ (in particular, $A(t) = A$ and $a(t) = a = |A|$). The tokens which originate from vertices in A are called A-tokens. The following lemma will allow us to omit in the further analysis some cases when the number of A-tokens available is too limited.

Lemma 2 *If $A \subset V(t)$, $s \leq t$, $a(s) \geq a(t)/2$, and there are fewer than $(c-3)ma(s)$ $A(s)$-tokens available in $G(t)$ then $\Phi(A) \geq 1/5$.*

Proof Let $B = V(t) - A$. If there are fewer than $(c-3)ma(s)$ $A(s)$-tokens available in $G(t)$, then B must have used at least $ma(s)$ $A(s)$-tokens (A can use at most $ma(t) \leq 2ma(s)$ $A(s)$-tokens), so there are at least $ma(s)$ edges between A and B. Thus $d(B:A) \geq ma(s)$ and as $d(A:A) + d(A:B) = ma(t)$,

$$\begin{aligned} \Phi(A) &= \frac{d(A:B) + d(B:A)}{2d(A:A) + d(A:B) + d(B:A)} \\ &\geq \frac{d(B:A)}{2d(A;A) + d(B:A)} \geq \frac{ma(s)}{2ma(t) + ma(s)} \geq \frac{1}{5}. \end{aligned}$$

□

Theorem 1 follows from Theorem 4 and Lemma 3 below. $\Phi(t)$ is the conductance of $G(t)$.

Theorem 4 *Let Let $m \geq 400$ and $c \geq 20$. There exist constants $\lambda = \lambda(m, c, t_0)$and $K = K(\lambda)$ such that, provided $\Delta t = o(t)$ and $\Delta T \geq K \log t$, $\Phi(t) \geq \lambda$ for all $t \geq t_0$, with probability at least $1 - 1/t_0$.*

Proof As the network is connected by construction, then for any constant network size t_0, $\Phi(t_0) \geq \lambda_0$ for some constant λ_0. For $t \geq t_0$, we prove that $\Phi(t) \leq \lambda$ with probability at most $1/t^2$. Thus for all $t \geq t_0$, $\Phi(t) \geq \lambda$, with probability

$$1 - \sum_{s=t_0+1}^{\infty} 1/s^2 \geq 1 - 1/t_0.$$

Let $K = 22/\lambda^2$. Since $\Delta S = K \log s$ and $\Phi(s) \geq \lambda$, for $s > t_0$ it follows from (4) and the assumptions of the theorem that at step $S + \Delta S/2$, the tokens are within $o(s^{-10})$ of the stationary distribution. For $v \in V(s)$, consider the sequence of tokens visiting v in the interval $(S + \Delta S/2, S + \Delta S]$. The probability a given token x visits v at a given step is $\pi(v)(1 + o(s^{-10}))$. Thus if $X = \{x_1, \ldots, x_J\}$ is the set of tokens in the network, then the next token to visit v is x_j with probability $(1 + o(s^{-10}))/J$. To simplify the presentation, we omit this deviation factor $1 + o(s^{-10})$ from the arguments below. To account for it, one would need to introduce factors $1 + o(1)$ to (7), (10) and (13), but the subsequent derivations would subsume such factors anyway.

We now consider the graph $G(t)$. Let $A \subset V(t)$ and $B = V(t) - A$. Let $X_A(s)$ denote the number of available $A(s)$-tokens in network $G(s)$, $s \in \mathcal{T}, s < t$. Let v be a new vertex in $V(s + \Delta s)$ ($v \notin V(s)$) generating edges $e_1, .., e_m$. The probability that the terminal vertex of edge e_i is in $A(s)$ is

$$p(v, i; A(s)) = \frac{X_A(s) - U_A(v, i)}{m(c-1)s - U(v, i)}, \tag{7}$$

where $U(v, i)$ (resp. $U_A(v, i)$) is the number of tokens already used up from $V(s)$ (resp. $A(s)$) at the moment between steps $S + \Delta S/2$ and $S + \Delta S$ when the token for e_i is passed to v.

Let $a = a(t)$ and $b = b(t)$. Let M be a large constant (to be deduced). From Lemma 1 we know that either $\Phi(A) \geq \lambda$ or $|A| \leq (1+\lambda)t/2$, so we assume $a = |A| \leq (1 + 1/M)t/2$ and $\lambda \leq 1/M$. The value of M depends on m, eg. when $m = 400$ we can choose $M = 6$ in the proofs below. We prove there are always at least a edges between A and B (for Corollary 5)). There are several cases depending on the size of A and the placement of the vertices of A in the whole sequence of vertices.

Case 1: $\frac{t}{M} \leq a \leq \frac{t}{2}\left(1 + \frac{1}{M}\right)$.

Let t_A (resp. t_B) be the end vertex of the first batch such that $a(t_A) \geq a(t)/2$ (resp. $b(t_B) \geq b(t)/2$). If $t_A \leq t_B$, then at least $b/3$ B-vertices are added after t_A ($b(t_A) \leq b(t)/2 + o(t_A)$) so at least $b/3$ B-vertices are added when there are already

at least $a/2$ A-vertices in the network. Similarly, if $t_B \le t_A$, then at least $a/3$ A-vertices are added when there are already at least $b/2$ B-vertices in the network.

Case 1(i): $t_A \le t_B$.

Let $N = |d(B:A)|$ be the number of edges from $B(t)$ to $A(t)$. Let $s \ge t_A$ be the end vertex of a batch (ie. $s \in \mathcal{T}$) and let $v \in B(t),\ v > s$. Then (7) implies that $p(v,i;A(s)) \ge a/3s$, for $c \ge 20$, $a(s) \ge a/2$ and Lemma 2 allows us to assume that $X_A - U_A(v,i) \ge (c-3)ma(s)$. Thus, as $s \le t$

$$\mathbf{E}N > m\frac{a}{3t}\frac{b}{3}.$$

Since $b = t-a \ge t/2(1-1/M)$, we have $\mathbf{E}N \ge (ma/18)(1-1/M)$. By the Hoeffding Inequality [9], we have (2)

$$\mathbf{Pr}(N \le \delta \mathbf{E}N) \le \exp -\frac{1}{2}(1-\delta)^2\mathbf{E}N.$$

Hence, for $M = 6$,

$$\mathbf{Pr}(|d(B:A)| \le ma/400) \le e^{-ma/50} < e^{-t}.$$

Thus there exists a constant $\lambda = \lambda(m,c,t_0)$ such that

$$\mathbf{Pr}(\exists A : \text{case 1(i) applies and } \Phi(A) \le \lambda) \le t2^t e^{-t} < \frac{1}{5t^2}. \tag{8}$$

Case 1(ii): $t_B \le t_A$. Similarly to case 1(ii), we can show that $\mathbf{Pr}(|d(A:B)| \le ma/400) \le e^{-t}$, so there exists a constant $\lambda = \lambda(m,c,t_0)$ such that

$$\mathbf{Pr}(\exists A : \text{case 1(ii) applies and } \Phi(A) \le \lambda) \le t2^t e^{-t} < \frac{1}{5t^2}. \tag{9}$$

Case 2: $1 \le a \le t/M$.

From the way the network is constructed, the graph $G(t)$ is connected, so it is impossible for any subset of vertices $A \subset V(t)$ to be disconnected. This implies that $\Phi_A \ge 1/(2m|A|+1)$. Thus we only need to consider $|A| > C$ for some constant C.

Let κ be the first entry in $\mathcal{T} \cap [t]$ of value at least $\frac{1}{2}\sqrt{at}$. Let $A^- = A(t) \cap [\kappa]$.

Case 2(i): $|A^-| \ge a/2$. Let $a^- = |A^-|$, and

$$\begin{aligned}\mathcal{F} &= \{v > \kappa :\ \text{no } A^- \text{ token chosen }\}\\ \mathcal{H} &= \{v > \kappa :\ \text{at least one } A^- \text{ token chosen }\}\end{aligned}$$

Let $\mathcal{H}(v) = \mathcal{H} \cap [\kappa+1, ..., v-1]$ be the set of vertices from $\kappa+1$ up to v which have used A^- tokens. We assume when vertex v gets tokens to join the network, the number of available A^--tokens is at least $m(c-3)a^-$, (see Lemma 2). Thus

$$\mathbf{Pr}(v \in \mathcal{F}) \le \left(1 - \frac{m(c-3)a^-}{(c-1)m(v-o(v))}\right)^m. \tag{10}$$

If $|\mathcal{H}(t)| \geq 3a/2$, then as $|A(t) \setminus A^-| < a/2$ we must have $|\mathcal{H}(t) \cap B| \geq a$, implying that $d(B : A) \geq a$ and thus $\Phi(A)$ is greater than some constant. Hence we can assume that $|\mathcal{H}(t)| < a$. For an arbitrary subset of vertices $H \subseteq [\kappa+1, \ldots, t]$ of size $|H| < a$, denote $F = [\kappa+1, \ldots, t] - H$ and derive

$$\begin{aligned}
\mathbf{Pr}(\mathcal{H}(t) = H) &= \mathbf{Pr}(\mathcal{F}(t) = F) \\
&\leq \prod_{v \in F} \left(1 - \frac{(c-3)a^-}{(c-1)v}\right)^m \leq \exp\left\{-\frac{c-3}{c-1}\, ma^- \sum_{v \in F} \frac{1}{v}\right\} \\
&\leq \exp\left\{-\frac{c-3}{2(c-1)}\, ma \log \frac{t}{\kappa + 3a/2}\right\} \leq \exp\left\{-\frac{c-3}{2(c-1)}\, ma \log\left(\frac{2}{3}\sqrt{\frac{t}{a}}\right)\right\}.
\end{aligned}$$

The last inequality follows from the fact that $\kappa = \sqrt{at}/2 + o(t)$, and we choose $M = 6$. Thus for $\beta = m\frac{c-3}{4(c-1)}$, $\mathbf{Pr}(\mathcal{H}(t) = H) \leq (9a/4t)^{\beta a}$, and for some constant $\lambda = \lambda(m, c, t_0)$,

$$\begin{aligned}
&\mathbf{Pr}(\exists A : \text{case 2(i) applies and } \Phi(A) \leq \lambda) \\
&\quad \leq \mathbf{Pr}(\exists a, A^-, H \,:\, C \leq a \leq t/M,\ A^- \subseteq [\kappa],\ a/2 \leq |A^-| \leq a, \\
&\qquad\qquad H \subseteq [\kappa+1 \ldots t],\ |H| < a,\ \mathcal{H}(t) = H) \\
&\quad \leq \sum_{a=3}^{t/M} a^2 \binom{t}{a}^2 (9a/4t)^{\beta a} \leq \sum (te/a)^{2a} (9a/4t)^{ma/5} < \frac{1}{5t^2}, \qquad (11)
\end{aligned}$$

as $\beta \geq m/5$ for $c \geq 20$. The last but one inequality above holds for sufficiently large m, M (eg. choose $m = 400$ and $M = 6$ will do).

Case 2(ii): $|A^-| < a/2$, $\epsilon t \leq |A| \leq t/M$, $\epsilon = e^{-m/16+2}$.
We need this subcase to get sensible values for m.

Let $B^- = [\kappa] \setminus A^-$, $b^- = |B^-|$. For $s \in \mathcal{T}$ and $s \geq \kappa$, the number of $B(s)$ tokens available at s is at least $m(c-3)b(s)$ (or $\Phi(A) > \lambda$, by considering $d(A(s) : B(s))$). Let $v > s$, $v \in A$ be added in the next batch. Then

$$\begin{aligned}
p(v, i; B(s)) &\geq \frac{m(c-3)b(s) - o(s)}{m(c-1)(a(s) + b(s)) - o(s)} \\
&\geq \frac{(c-4)b^-}{(c-1)(a+b^-)} \\
&\geq \frac{c-4}{c-1}\frac{\kappa - a/2}{\kappa + a}.
\end{aligned}$$

For $c \geq 20$, we have $\mathbf{E}d(A : B) \geq ma/5$ and thus

$$\mathbf{Pr}(d(A : B) < \mathbf{E}d(A : B)/80) \leq e^{-ma/16}.$$

Finally we have

$$\begin{aligned}\mathbf{Pr}(\exists A:\ \text{case 2(ii) applies and } \Phi<\lambda) &\leq \sum_{a=\epsilon t}^{t/M}\binom{t}{a}e^{-ma/16} \\ &\leq \sum\left(\frac{te}{a}e^{-m/16}\right)^a \\ &\leq te^{-\epsilon t}\leq 1/5t^2.\end{aligned} \tag{12}$$

Case 2(iii): $|A^-|<a/2$, $1\leq|A|\leq\epsilon t$.

Let $A^+=A\setminus A^-$, and let $N=|d(A^+:A)|$. We will prove the probability that $N>ma/4$ is small. It follows that since there are at least $ma/2$ out-edges of A^+, at least $ma/4$ edges go from A^+ to B, so $\Phi(A)$ is greater than a constant.

Let $\kappa\leq s\leq t-1$. When vertex $s+1$ gets m tokens to join the network, there are at least $m(c-1)(s-o(s))$ tokens in the network, and at most $mca(s)$ of them are $A(s)$-tokens. Thus the expected number of edges from vertex $s+1$ to $A(s)$ is at most $(2mc/(c-1))a(s)/s$ and

$$\frac{a(s)}{s}\leq\frac{a}{\kappa}\leq 2\sqrt{\frac{a}{t}}.$$

Thus

$$\mathbf{E}N\leq a\cdot\frac{2mc}{c-1}\cdot 2\sqrt{\frac{a}{t}}=\frac{ma}{4}\left(\frac{16c}{c-1}\sqrt{\frac{a}{t}}\right). \tag{13}$$

The Chernoff inequality implies that

$$\mathbf{Pr}(N\geq\delta(\mathbf{E}N))\leq\left(\frac{e}{\delta}\right)^{\delta(\mathbf{E}N)}. \tag{14}$$

Using (13) and (14), as $\delta\geq(\sqrt{t/a})/17$ for sufficiently large c, we get

$$\mathbf{Pr}(N\geq ma/4)\leq\left(\frac{e}{\delta}\right)^{ma/4}\leq\left((17e)^2\frac{a}{t}\right)^{ma/8}.$$

Thus for sufficiently large m and for some constant $\lambda=\lambda(m,c,t_0)$,

$$\mathbf{Pr}(\exists A:\text{case 2(ii) applies and }\ \Phi(A)\leq\lambda)\leq\sum\left(\frac{te}{a}\left(e^8\frac{a}{t}\right)^{\frac{m}{8}}\right)^a\leq\frac{1}{5t^2}. \tag{15}$$

Inequalities (8), (9), (11), (12) and (15) imply that there exists a constant λ such that $\Phi(t)\leq\lambda$ with probability at most $1/t^2$. □

We also note the following for the adversarial deletion proofs. The proof of this statement can be traced in the proof of Theorem 4 (for $m=400, c=20, M=6$).

Corollary 5 *For sufficiently large constants m and c, with probability $1 - 1/t^2$, $|E(A : \overline{A})| > |A|$ for all $A \subseteq V(t)$, $1 \leq |A| \leq (t/2)(1 + 1/M)$.*

The lower bound on the conductance of the graph implies an upper bound on the diameter.

Lemma 3 *If the conductance of a graph $G = (V, E)$ is at least λ, then its diameter is at most $\frac{2}{\log(1+\lambda)} \log |E|$.*

Proof Starting at any vertex $v \in V$ we build a search tree in a breadth first manner, adding one layer of the tree in each iteration. If S is the vertex set of the current tree and $\pi(S) \leq 1/2$, then $E(S : \overline{S}) \geq \lambda d(S)$. The vertex set $N(S)$ of the next tree consists of the vertices in S and all their neighbours. Therefore $d(N(S)) \geq d(S) + E(S : \overline{S}) \geq (1 + \lambda)d(S)$. After at most $\log |E| / \log(1 + \lambda)$ iterations, the tree will cover a set of vertices of π-measure greater than $1/2$. □

5 Analysis of FIFO queue model

The network grows from its initial size of 3 to size t and thereafter as a new vertex arrives, the oldest vertex leaves. After $s \geq t$ vertices have joined the network, we use the following labeling for vertices in $[s]$. The current network is always $[1, ..., t]$. The $t+1$ vertices before addition of vertex 1 are labeled $[-t, ..., 0]$. An edge directed from $v \in [1, ..., t]$ to $w \in [-t + 1, ..., 0]$ will be broken when w leaves at step $t + w$. Of course the replacement edges may be directed to $[w+1, ..., 0]$ but eventually they will be inserted into $[1, ..., t]$ as this is the network after step t. We call such an edge the *final replacement edge of v*.

Theorem 6 *Assume $\Delta t = o(t)$. With probability $1 - O(1/t^2)$ the network $G(t)$ has diameter $O(\log t)$.*

Proof We repeat the conductivity arguments of the previous section. For brevity we only consider the most difficult case (small sets). Let $A \subset V(t)$, where $a = |A| \leq t/M$. As before let $\kappa = (1 + o(1))\sqrt{at}/2$ and $A^- = A(\kappa)$.

Case $|A^-| \geq a/2$.

We first prove that most out-edges from A^- go to $[-t + 1, ..., 0]$. At any step $\tau \in [1, ..., \kappa]$ there are $(c - 1)mt$ tokens in the network, of which at most $cm\tau$ are from $[1, ...\tau]$. Let $N(A^- : \kappa)$ be edges from A^- to $[1, ..., \kappa]$. Thus

$$\mathbf{E}N(A^- : \kappa) \leq \frac{cm\kappa}{(c-1)mt} ma.$$

Using the Chernoff inequality with $\beta = ((c - 1)/(4c))\sqrt{t/a}$ we have

$$\mathbf{Pr}(N(A^- : \kappa) \geq ma/8) \leq \left(\frac{e}{\beta}\right)^{ma/8}.$$

Assuming $c \geq 20$ and $M \geq e^8$ (which implies $e \leq (t/a)^{1/8}$),

$$\begin{aligned}\mathbf{Pr}(\exists A, |A| = a,\ N(A^- : \kappa) \geq ma/8) &\leq \binom{t}{a}\left(5e\sqrt{\frac{a}{t}}\right)^{ma/8} \\ &\leq \left(\frac{a}{t}\right)^{a\left(\frac{m}{64}-\frac{9}{8}\right)},\end{aligned}$$

which is $o(t^{-3})$ for $m \geq 400$. By a similar argument at most $ma/8$ edges of A^- go to $[-t+1, ..., -t+\kappa]$ so that at least $\lceil ma/4 \rceil$ edges of A^- are replaced after κ by deletion of vertices in $[-t+\kappa+1, ..., 0]$. For $v \in A^-$

$$\mathbf{Pr}(\text{ final replacement edge of } v \text{ goes to } A) \leq \frac{cma}{(c-1)m\kappa}.$$

Let $N'(A)$ count the subset of the (first) $ma/4$ edges of A^- finally replaced after κ which point to A. Thus $\mathbf{E}N'(A) \leq (ma/4)(ca/(c-1)\kappa)$, and using $\beta = (19/50)\sqrt{t/a}$ we have

$$\mathbf{Pr}(N'(A) \geq ma/5) \leq (e/\beta)^{ma/5}.$$

As $50/19 < e$ it follows that

$$\begin{aligned}\mathbf{Pr}(\exists A, |A| = a\ : N'(A) \geq ma/5) &\leq \binom{t}{a}\left(e^2\sqrt{\frac{a}{t}}\right)^{ma/5} \\ &\leq \left(\frac{a}{t}\right)^{a\left(\frac{m}{20}-\frac{9}{8}\right)},\end{aligned}$$

very much as before.

Case $|A^-| < a/2$**.**

Considering the first $\lceil a/2 \rceil$ vertices v of A^+

$$\mathbf{Pr}(\text{ final replacement out-edge of } v \text{ hits } A) \leq \frac{cj}{(c-1)x(j)} \leq \frac{ca}{(c-1)\kappa},$$

where $j = |A(x(j))|$ at step $x(j) \geq \kappa$ at which the final replacement is made. Thus, defining $N'(A)$ as above, but for edges of A^+ pointing to A, $\mathbf{E}N'(A) \leq (ma/4)(ca/(c-1)\kappa)$, and as above

$$\mathbf{Pr}(\exists A, |A| = a, 1 \leq a \leq t/M \text{ such that } N'(A) \geq ma/5) = o(t^{-2}).$$

□

6 Growth model: robustness under adversarial deletion

As before, we assume the random walk on tokens is in the steady state when vertex t is added, and suppress the $(1 + o(t^{-10}))$ error incurred by this approximation to simplify presentation. The steps t (below) refer to the addition of vertices. We first make some preliminary calculations on availability of tokens from a given vertex set S.

Given a set $S \subseteq V(t)$, let $X_t(S)$ be the number of S-tokens remaining in the network after step t. Then

$$X_t = cm|S| - d(S : S) - d((V(t) \setminus S) : S).$$

Let $s = |S|$, and let Y_{t+1} be the number of S-tokens removed at step $t + 1$. Thus $X_{t+1} = X_t - Y_{t+1}$. The vertex $t+1$ chooses m tokens uar without replacement from a set of size $(c-1)mt$, of which $X_t(S)$ are from S, and thus $\mathbf{E}Y_{t+1} = mX_t/((c-1)mt)$. When $S = [s]$ we have $X_s = (c-1)ms$ and taking expectations recursively

$$\begin{aligned}
\mathbf{E}X_t([s]) &= X_s \prod_{\tau=s}^{t-1} \left(1 - \frac{1}{(c-1)\tau}\right) \\
&= (1 + o(1))X_s \exp\left(-\frac{1}{(c-1)} \sum_{\tau=s}^{t-1} \frac{1}{\tau}\right) \\
&= (1 + o(1))(c-1)ms \left(\frac{s}{t}\right)^{1/(c-1)}.
\end{aligned}$$

It follows that $Z_t([s])$, the in-degree of $[s]$ has expected value

$$\mathbf{E}Z_t = (c-1)ms \left(1 - (1 + o(1)) \left(\frac{s}{t}\right)^{1/(c-1)}\right). \tag{16}$$

Thus when $t = \Omega(s^c)$, most tokens from $[s]$ have been used. For any set S of size s, we can argue as follows that $\mathbf{E}X_t(S)$ is at least as large as $\mathbf{E}X_t([s])$:

$$\mathbf{E}X_t(S) = cm \sum_{v \in S} \prod_{\tau=v}^{t-1} \left(1 - \frac{1}{(c-1)\tau}\right) \geq \mathbf{E}X_t([s]).$$

Provided $\mathbf{E}X_t(S)/\sqrt{t} \to \infty$, the value of $X_t(S)$ (resp. $Z_t(S)$) is sharply concentrated. For, using the Azuma martingale inequality on the ordered sample of tokens of length mt made by the network, we have

$$\mathbf{Pr}(|X_t(S) - \mathbf{E}X_t(S)| > \theta) \leq \exp - \left(\frac{\theta^2}{8mt}\right).$$

6.1 Adversarial deletion of vertices

We consider the case where the set $S = [s]$ is deleted at step t. Among all subsets of vertices of size s, the set $[s]$ has the largest expected in-degree $\sim (c-1)ms(1-(s/t)^{1/(c-1)})$ (see (16)). Thus deletion of $[s]$ can, in expectation at least, cause more damage than deletion of any other set of size s.

Theorem 7 *The following hold* **whp** *for the deletion of* $S = [s]$
(a) For $s = o(t^{1-1/m})$ *deletion of* $[s]$ *does not disconnect* $G(t) \setminus [s]$.
(b) For $s \le t/2$, *deletion of* $[s]$ *disconnects a set of size at most* $te(sm/t(m-1))^m$.

Proof Let A, B be disjoint non-empty sets whose union is $[s+1, ..., t]$. If there are no edges between A and B then $G(t) \setminus S$ is disconnected. We next bound the probability that $E(A:B) = 0$.

Let V_A be the steps in $[s+1, ..., t]$ at which vertices of A are added, let $A(v)$ be A after step v and let $a(v) = |A(v)|$. Define analogously V_B, $B(v)$, and $b(v)$. Assume that there are no edges between A and B before step v. If $v \in V_A$, then during step v there are $(c-1)m(v-1)$ tokens available and at least $(c-1)mb(v)$ of them are B-tokens. The B-tokens have been used only by vertices from B, so at most $mb(v)$ of the total $cmb(v)$ B-tokens have been used. The probability that at this step v no edge between A and B is created, is at most $((v-1-b(v))/(v-1))^m$. Analogously, if $v \in V_B$, the probability at this step v that no edge between A and B is created is at most $((v-1-a(v))/(v-1))^m$. Hence the probability that $E(A:B) = 0$, is at most P^m, where

$$\begin{aligned} P &= \prod_{v\in V_A} \frac{v-1-b(v)}{v-1} \prod_{v\in V_B} \frac{v-1-a(v)}{v-1} \\ &= \frac{\prod_{v\in V_A}(v-1-b(v)) \prod_{v\in V_B}(v-1-a(v))}{s(s+1)\cdots(t-1)} \\ &= \frac{[s(s+1)\cdots(s+a-1)][s(s+1)\cdots(t-a-1)]}{s(s+1)\cdots(t-1)}. \end{aligned}$$

To see that the last equality above holds, consider any two consecutive steps v' and v'' in V_A. The steps $v'+1, v'+2, \ldots, v''$ consist of $b(v'') - b(v')$ steps from V_B and one step from V_A. Thus $v'' = v' + (b(v'') - b(v')) + 1$, and $v'' - b(v'') = v' - b(v') + 1$.

Note that $s+a < t$ as $a+b \le t-s$ and assume $a \le b$ to derive $P = \prod_{j=0}^{a-1} \frac{s+j}{t-a+j} \le \left(\frac{s+a}{t}\right)^a$. Thus

$$\mathbf{Pr}(\exists A, |A| = a \text{ and } E(A:B) = 0) \le \left(\frac{te}{a}\left(\frac{s+a}{t}\right)^m\right)^a. \tag{17}$$

The maximum in-degree of S is cms and the out-degree of A is ma, so, by connectivity we must have $E(A:B) > 0$ whenever $a > cs$. For $s = o(t^{1-1/m})$, the sum of

the right-hand side of (17) for $1 \le a \le cs$ is $o(1)$, which proves the first part of the theorem.

Assume now that the value of s is fixed. Let $f(a) = (et/a)((s+a)/t)^m$, then $f(a)$ has a unique minimum at $a^* = s/(m-1)$. As $a \le b$, the maximum value a can take is $(t-s)/2$. For $s \le t/2$, $f((t-s)/2) < 1$. Thus $f(a) > 1$ only for $1 \le a \le \alpha$ where $\alpha < a^* = s/(m-1)$. Using a^* as an upper bound on α we find $\alpha \le et(sm/t(m-1))^m$. □

6.2 Adversarial deletion of edges

We note the following: If $f(B)$ tokens from set B are available at step t, and X is the number of these tokens in the set A, then the expected number $\mathbf{E}X = f(B)d(A)/2mt$. Provided $\mathbf{E}X = \Omega(\log t)$, by the Hoeffding Inequality, the actual number X is sharply concentrated about this value.

Theorem 8 *Let $t^{(c+1)/2c} \log t \le s \le t/2$. Suppose the edges between $[s]$ and $G(t)\setminus[s]$ are cut. Then* **whp** *all but at most $te(sm/t(m-1))^m$ vertices reconnect to $[s]$ implicitly.*

Proof By Theorem 7, **whp** $G(t)\setminus[s]$ contains a large component L, and some small components A of total size at most $te(sm/t(m-1))^m$. The size of L is at least

$$|L| \ge t - s - te(sm/t(m-1))^m \ge t\left(\frac{1}{2} - e\left(\frac{m}{(m-1)2}\right)^m\right).$$

For sufficiently large m, $s \gg |A|(c+1)m$. By Corollary 5 there were at least $(s+a)$ edges between L and $S \cup A$ (**whp**). Thus at least $s + a - (c+1)ma > s/2$ of these edges were between L and $[s]$. Denote by B the vertices of L which have broken edges. Thus $|B| \ge s/2m$.

From (16) the number of available $[s]$-tokens $X_t([s])$, has expected value $\mathbf{E}X_t(1+o(1))(c-1)ms(s/t)^{1/(c-1)}$. By Azuma's inequality X_t is concentrated provided $\mathbf{E}X_t/\sqrt{t} \to \infty$, ie. $s \gg t^{(c+1)/2c}$.

As B is large ($|B| \ge s/2m$), the number of $[s]$-tokens and of all tokens on L, B is sharply concentrated. Thus **whp** at least $Y = \mathbf{E}X_t/5$ of $[s]$-tokens are on L, and the expected number of these on B is at least $Yd(B)/2mt$. The total number of tokens on B is at most $2(d(B)/2mt)(c-1)mt$. Thus

$$\mathbf{Pr}(\text{no replacement edge from } B \text{ points to } [s]) \le \left(1 - \frac{1}{10}\left(\frac{s}{t}\right)^{c/(c-1)}\right)^{s/2m} = o(1).$$

□

Theorem 9 *Let $V(t) = \cup_{j\in J} V(A_j)$, where the sets A_j, $j \in J$ are disjoint and $\min |V(A_j)| \geq t^{(c+2)/2c} \log t$. Suppose $G(t)$ is disconnected by edge cuts into components with vertex sets A_j, $j \in J$. Then $V(t)$ will reconnect implicitly by token sampling* **whp**.

Proof Let (A, B) be a partition of the components, where the vertices of A are of total size $a \leq t/2$. By Corollary 5, in $G(t)$ there were at least a edges between A and B. Thus, there were at least $a/2$ edges from A to B or vice versa.

Let $\omega = \log t$ and let $s = t^{(c+1)/2c}\omega$. The number, $X(A)$, of A-tokens remaining in the network satisfies $X(A) \geq ((c-1)ma/2)(a/t)^{1/(c-1)}$ **whp**. Thus

$$\mathbf{Pr}(\text{no replacement edge from } A \text{ points to } B) \leq \left(1 - \frac{1}{2}\left(\frac{a}{t}\right)^{\frac{c}{c-1}}\right)^{a/2}.$$

The total number of components is $N \leq t/s$, and the number M of components in A satisfies $M \leq a/s$. Let P be the probability that there exists some partition (A, B) of the components with no replacement edges formed between A and B. Then

$$\begin{aligned}
P &\leq \sum_{M=1}^{N/2} \binom{N}{M} \exp\left(-\frac{a}{4}\left(\frac{a}{t}\right)^{\frac{c}{c-1}}\right) \\
&\leq \sum \left(\frac{2et}{a}\right)^{M} \exp\left(-\frac{Ms}{4}\left(\frac{a}{t}\right)^{\frac{c}{c-1}}\right) \\
&\leq \sum \exp\left(-M\omega\left(-\frac{t^{(c+1)/2c}}{4}\left(\frac{s}{t}\right)^{\frac{c}{c-1}} - \frac{\log 2et/a}{\omega}\right)\right) \\
&\leq \sum_{M} \exp\left(-M\omega t^{1/2c}\right) \\
&= o(t^{-K}),
\end{aligned}$$

for any $K > 0$. □

7 Conclusions

Our initial investigation suggests that the token protocol is a successful method of randomizing network structure to maintain low diameter and considerable robustness against adversarial attack. The models we have studied (growth model, FIFO queue) are simple examples based on this approach. It would be interesting to see if the general model, with arrivals and departures suitably regulated, also has such good properties, and this work is on-going.

With the benefit of hindsight, it seems very possible that the best way to maintain robustness is to allow the vertices of the network to re-randomize their out-neighbourhood at suitable time intervals. This alteration can be implemented without fundamental change to the token protocol; for, when an edge (u, v) is dropped, the terminal vertex v donates a new token to maintain the token pool in the usual manner.

This would overcome the problem in the growth model that all edges (u, v) point from newer vertices u to older vertices v. For example, the number X of vertices in $V(t) \setminus [s]$ whose only network edges point to $[s]$ has expected value $\mathbf{E}X \sim (t-s)(s/t)^{cm/(c-1)}$. At present, an edge cut between $[s]$ and $V(t)\setminus[s]$ leaves a constant fraction of such vertices isolated, whereas this effect could be largely avoided by re-randomizing.

There are several important points about the token protocol which need discussing.

The first concerns the arbitrary joining behaviour. The actual number of new vertices which can be added at any step depends on the joining behaviour, which is assumed to be unpredictable. The expected flow of tokens through a given vertex u in σ steps of the random walk is $(c-1)mt\ \sigma\ (d(u)/2mt) \geq \sigma(c-1)m/2$. As long as the number of joiners approaching vertex u is not m times greater than the number of tokens flowing through u during σ steps, all joiners can be accommodated. If too many vertices arrive at a vertex, they will have to queue to join. There are various ways to adapt to this, eg. increase the speed of the random walk and the value of c.

The overhead at a vertex due to maintaining the token flow can be analysed as follows: The number of tokens arriving at any vertex v is a uar sample from $(c-1)mt$ items with probability $d(v)/2mt \leq (c+1)/2t$. Thus the expected number of tokens arriving at any vertex during one time step is at most $(c-1)(c+1)m/2$. The probability that more than $K \log t$ tokens arrive at a given step is $O(t^{-K})$ for any K, by the Chernoff Inequality. We assume that this overhead is insignificant compared to the network traffic.

Increasing the values of m, c increases the network connectivity and robustness respectively. It also increases the load on each vertex both in terms of vertex in-degree and token forwarding. To simplify the proofs we have assumed $c \geq 20,\ m \geq 400$, but we suppose the graph will be an expander (and hence rapidly mixing with logarithmic diameter) for any $m \geq 2$. The role of c (other than robustness) is to ensure there are an adequate number of tokens available to joining vertices. Probably any value of $c > 1$ will do.

Finally we remark that the token protocol is completely distributed, requires no complicated constructions, only the simplest actions on the part of the participants, and makes no assumptions about joining behaviour. Yet (with high probability) it yields a connected, robust, low diameter network.

References

[1] B. Bollobás. *Random Graphs*. Academic Press, 1985.

[2] V. Bourassa and F. B. Holt. SWAN: Small-world Wide Area Networks. In *Proc. SSGRR-2003s, International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet*, 2003.

[3] V. Chvátal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25:285–287.

[4] Overnet eDonkey2000. `http://www.edonkey2000.com/documentation/onvsed2k.html`.

[5] 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.

[6] T. Friedetzky. Personal communication. July 2004.

[7] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *Proc. 22-nd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong*, March 2004.

[8] Gnutella. `http://gnutella.wego.com`.

[9] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal American Statistical Association*, 58:13–30, 1963.

[10] M. Jerrum and A. Sinclair. The markov chain monte carlo method: an approach to approximate counting and integration. In D. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, pages 482–520. PWS, 1996.

[11] Kazaa. `http://www.kazaa.com`.

[12] C. Law and K.-Y. Siu. Distributed construction of random expander graphs. In *Proc. 22-nd Annual Joint Conference of the IEEE Computer and Communications Societies*, April 2003.

[13] David Liben-Nowell, Hari Balakrishnan, and David Karger. Analysis of the evolution of peer-to-peer systems. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 233–242. ACM Press, 2002.

[14] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: a scalable and dynamic emulation of the butterfly. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 183–192. ACM Press, 2002.

[15] M.Ripeanu. Peer-to-peer architecture case study: Gnutella network. Technical report, University of Chicago Technical Report TR-2001-26.

[16] Napster. `http://www.napster.com`.

[17] G. Pandurangan, P. Raghavan, and E. Upfal. Building low-diameter peer-to-peer networks. *IEEE Journal on Selected Areas in Communications*, 21(6):995–1002, August 2003.

[18] Mario T. Schlosser, Michael Sintek, Stefan Decker, and Wolfgang Nejdl. Hypercup - hypercubes, ontologies, and efficient search on peer-to-peer networks. In *Agents and Peer-to-Peer Computing, First International Workshop, AP2PC 2002*, pages 112–124, July 2002.

[19] Mario T. Schlosser, Michael Sintek, Stefan Decker, and Wolfgang Nejdl. A scalable and ontology-based p2p infrastructure for semantic web services. In *2nd International Conference on Peer-to-Peer Computing (P2P 2002)*, pages 104–111, September 2002.

[20] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.