HAL open science

A randomized algorithm for the joining protocol in dynamic distributed networks

Colin Cooper, Ralf Klasing, Tomasz Radzik

To cite this version:

Colin Cooper, Ralf Klasing, Tomasz Radzik. A randomized algorithm for the joining protocol in dynamic distributed networks. RR-5376, INRIA. 2004, pp.21. inria-00070627

HAL Id: inria-00070627

https://inria.hal.science/inria-00070627v1

Submitted on 19 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A randomized algorithm for the joining protocol in dynamic distributed networks

Colin Cooper — Ralf Klasing — Tomasz Radzik

N° 5376

November 2004

Thème COM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--5376--FR+ENG

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

Colin Cooper† , Ralf Klasing‡ , Tomasz Radzik§

Thème COM — Systèmes communicants
Projet Mascotte

Rapport de recherche n° 5376 — November 2004 — 21 pages

Abstract: We consider a randomized algorithm for assigning neighbours to vertices joining a dynamic distributed network. The algorithm acts to maintain connectivity, low diameter and constant vertex degree. This is effected as follows: On joining each vertex donates a fixed number of tokens to the network. The tokens contain the address of the donor vertex. 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 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. For example, an edge cut which leaves components of size at least $t^{(c+2)/2c}$ the network reconnects immediately (**whp**) on replacing lost edges from the token pool, where t is the size of the network and c is a constant of the protocol.

Key-words: Random dynamic networks, joining protocol, random graph processes.

* 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.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 —Télécopie : +33 4 92 38 77 65

Un algorithme aléatoire de protocole d'adhésion dans des réseaux distribués dynamiques

Résumé : Nous considérons un algorithme aléatoire pour l'attribution de voisins aux sommets joignant un réseau distribué dynamique. L'algorithme agit de manière à maintenir la connectivité, un diamètre faible et un degré de sommet constant. Ceci s'effectue comme suit : en joignant, chaque sommet donne un nombre fixe de jetons au réseau. Les jetons contiennent l'adresse du sommet donateur. Ces jetons suivent des chemins aléatoires indépendants dans le réseau. Un jeton peut être utilisé par n'importe quel sommet qu'il traverse pour établir un raccordement au sommet donateur. Ceci permet aux nouveaux sommets de recevoir un ensemble aléatoire de voisins sans connaître la structure globale du réseau. Le réseau obtenu de cette façon est persistent aux disparitions de sommets et d'arêtes et se reconnecte de lui-même dynamiquement. Par exemple, si la disparition d'une arête devait créer des composants d'une taille minimum à $t^{(c+2)/2c}$, alors le réseau se reconnecterait immédiatement (avec grande probabilité) en utilisant les jetons disponibles, t étant la taille du réseau et c une constante du protocole.

Mots-clés : Réseaux distribués dynamiques, protocole d'adhésion, processus de graphes aléatoires.

1 Introduction and the basic protocol

In the type of distributed network we consider, each vertex knows only its immediate neighbours, and the overall size and membership of the network is unknown. A central problem for such networks is how new members can join the network if there is no list of current members. Typical joining protocols include join to a friend or to a central node or set of nodes. We call such joining protocols *arbitrary*. We describe a randomized algorithm which overcomes the consequences of such behaviour by allocating a more suitable set of neighbours to the vertex once it has joined the network arbitrarily. The network we obtain in this way is robust under adversarial deletion of vertices and edges and actively reconnects itself when edges are deleted.

We assume the network has properties which it is committed to maintaining in a distributed fashion and that the arbitrary joining protocol may be unfavourable to this. The simplest set of desirable properties is bounded degree (fairness in load sharing and practical to implement), connectedness (ability to communicate) and small diameter (fast broadcasting). Arbitrary joining protocols can create long paths and high degree vertices, and can introduce inefficient bottlenecks which are vulnerable to adversarial attack. It is well known that some classes of random graphs (eg. random regular graphs [1]) have all the desirable properties mentioned above. A randomized algorithm for assigning neighbours should be able to do at least as much as a random graph. Because the complete vertex list of the network is unknown we cannot just assign a random subset of the vertices as neighbours of the joining vertex. In fact, one problem is to obtain such a random subset of the unknown vertex list (the used tokens are removed from the system).

Our basic approach is simple. On joining, each vertex donates a fixed number d of tokens to the network. Each token contains the address of the donor vertex. The tokens make an independent random walk on the network. This walk only stops when the token is used up (as described below). Because the neighbour structure is randomized by construction, the random walk is rapidly mixing and the tokens visiting any vertex are (almost) random. When a new vertex joins, its initial contact passes it a fixed number m of tokens. The new vertex then connects to the vertices which donated the tokens it was given, which are a random subset of the unknown vertex list.

Tokens can only be used once, so the degree of any vertex is at most $m + d$. The network we construct in this way is a random graph and has low diameter and other good connectivity properties. If an edge is deleted, the vertex simply acquires another token and connects to the vertex which donated it. Thus if a vertex leaves unannounced the links it supported will be replaced.

The token approach has significant additional benefits apart from simplicity and randomization of the network structure. Tokens act as a kind of distributed memory which is very 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 edges. This will tend to reconnect the network. A more precise description of this robustness is given later. We next briefly mention other possible applications, not studied here. The network can reconfigure itself in an on-going

fashion, replacing edges by selecting new tokens. This makes it difficult for an adversary to learn its structure. The simulations of [6] 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 idea is that a joining vertex could attach text to the tokens giving its particular interests. Other vertices could scan the text as the tokens visit them in the random walk and contact vertices of interest as an alternative to actively seeking information by broadcasting.

The idea of using a random walk to create random networks has been used by [2]. We discuss later in this section the protocol proposed in [2] and the other relevant randomized protocols given in [16], and [11]. 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 which evolve in an unstructured and unpredictable way. A typical example of this is the peer-to-peer (P2P) networks of eg. [16], [2], [11]. 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 the 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. Typically, these properties include connectivity, low degree and small network diameter. We mention now some existing popular P2P networks [5].

An early P2P implementation is Gnutella [7]. This network has no centralised control, but also no explicit mechanism for maintaining low diameter. Although research has shown that the Gnutella network often has small diameter (see [14]), this is by no means guaranteed. At the other end of the spectrum w.r.t. centralisation is the (in)-famous Napster [15]. 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 [10] which uses a network protocol called FastTrack. FastTrack is in some sense in 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 [3]: "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 [18], [17]. A particularly good structure for maintaining connectivity is a cyclic list where new arrivals can push in anywhere (or be randomly assigned a location within the cycle) and the list contracts as vertices leave. An instance of this is the Chord system [19, 12]. 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) enable a fast look-up procedure. Such networks are called content-addressable (CAN). The paper [13] 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 paper [19] mentions the difficulties of reconnecting the Chord network if it becomes disconnected, and suggests that retaining a random set of node addresses may help. A reconnection mechanism could be based on a self maintaining network of the type we propose here. The connections of the Chord network would be used for searching for data, while the connections of the self maintaining network would be used to reconnect a disconnected network.

Another approach to maintaining desirable connectivity properties of a dynamic network is to follow some sort of random graph model. The formative instance of this was the work of Pandurangan, Raghavan and Upfal [16] who proposed a protocol which ensures that the network is connected and has logarithmic diameter with high probability, and has always bounded degree. The crucial feature of their protocol is a central cache, which maintains addresses of a random subset of the set of vertices 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 which is 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, but cannot reconnect the network, if it becomes disconnected.

Law and Siu [11] 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 has the basic set of desired properties **whp** ("with high probability"). The model allows for the distributed construction of the network based on random walks as in [2].

Gkantsidis, Mihail and Saberi [6] simulate various random graph type models including one with two level clustering and compare the effectiveness of random walks as a search method for data retrieval on the network 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 [4] models content-addressable P2P networks resilient to adver-

sarial node deletion. It uses a butterfly graph which is robust under deletion of $n/2$ of the n data item nodes. The network is static and of logarithmic degree, with no explicit node addition mechanism.

We briefly mention the benefits of the token protocol (the detailed results for our model are given in Section 3). They are the randomizing of joining behaviour of new members, and that the model is robust under adversarial deletion, especially of edges. The tokens act as a distributed address memory. On losing edges, the vertices replace them from existing tokens many of whose donors lie in other components. On joining the network a new vertex is given m tokens which form a random (multi)-set of neighbours. The vertex must then donate $d = cm$ tokens to the network, where m and c are constants. Thus the maximum degree of a vertex is at most $(c+1)m$. Robustness under adversarial deletion is measured in terms of c, m and t – the current size of the network (the number of vertices). As a random graph the network already has good intrinsic connectivity properties and **whp** cannot be disconnected by the removal of a small (size $o(t^{1-1/m})$) set of vertices. For an edge cut which leaves components of size at least $t^{(c+2)/2c}$, the network reconnects immediately (**whp**) on replacing lost edges from the token pool. The worst case edge cut is to separate the first (oldest) s vertices. However, for $s < t/2^c$, all but at most $te(s(c+1)/t)^m$ of the $t-s$ other vertices reconnect to the first s vertices.

Our analysis is as follows. For the random graph properties, we show that a network of size t has constant conductivity and thus a random walk on this network mixes in time $O(\log t)$. After this time the tokens arriving at any vertex are effectively random and a set of these tokens constitutes a random neighbourhood. The $O(\log t)$ diameter follows immediately from the constant conductivity. We assume that the network "processes" the new vertices in batches. The network waits for $\Theta(\log t)$ steps to let the tokens reach the steady state, and then passes tokens to the next batch of new vertices – the vertices which approached the network during the period of mixing tokens. This building of a layered network is a convenience of the analysis and we do not suppose the actual protocol would wait for the steady state of all tokens. We analyse two cases: a growing network and a network which acts as a FIFO queue.

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. To do so, we have to consider availability of the tokens of a subset of vertices A. The availability of these tokens depends not only on the size of A but crucially on the age of A, and this makes the analysis considerably more complex and challenging than static random graph models.

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.

2 The token protocol

The abstract process. We simulate the following process for a random network of bounded degree. When new vertices join the network, they connect to existing vertices by edges. Each edge adjacent to a vertex v is classified as either an outgoing edge or an incoming edge, but

in either case the edge is a bidirectional communication link. Out-edges are established when v joins the network, in-edges when a vertex which joins the network later connects to v. To keep the degrees of vertices within a fixed range $[m, (c+1)m]$, for some constant integers $m \geq 2$ and $c \geq 2$, each vertex has $(c+1)m$ "slots" for attaching edges adjacent to v: m slots for out-edges and $d = cm$ slots for in-edges. A vertex v joining the network selects from the available incoming slots of the existing vertices m random slots (sampling uniformly at random without replacement) and establishes m edges between itself and the vertices with the selected slots. The selected slots are not available for later connections. The above process may create parallel edges. Not to treat the initial network as a special case, we assume that initially we have three vertices connected in a triangle, and each of them has m outgoing and m incoming edges.

In the protocol we propose, the tokens with addresses of available slots (one token per one slot) randomly walk through the network. Each token contains the address of the vertex from where it originated. These tokens randomly spreading out through the network may be viewed as a distributed knowledge of the network. If the network is maliciously or accidentally split into two parts, then each part should have, through the tokens, some knowledge of the other part, and this knowledge can be used to reconnect the network. The details of the protocol are given next.

Random walks of the tokens. A number of tokens are always present in the network. A simple process running at each vertex makes these tokens walk randomly through the network: for each token coming to a vertex w, the process selects a random edge adjacent to w (not making any distinction between incoming and outgoing edges) and sends the token along that edge. In practical implementations the speed of forwarding the tokens would be adjusted to the expected rate of change of the network. Ideally, each token should walk for long enough to reach a random node before it is picked up to be used for connecting or reconnecting vertices.

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, then it tries to establish m edges connecting v to the vertices whose addresses are in the tokens (there may be parallel edges). Some of the tokens may have originated from the vertices which are no longer in the network. Vertex v keeps asking 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 cm tokens of its own to the network.

Leaving the network. A vertex leaves the network by simply no longer reacting to any communication from its neighbouring vertices. If the leaving is planned, then the vertex first pushes any tokens visiting it 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 picks up an additional token to establish a new edge if the lost edge was outgoing from v, or donates its own token to the network if that edge was incoming

to v. This protocol covers also 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.

Variations on the basic model. We analyse two variations of the basic model. In the first, vertices join the network but never leave. In the second, the network maintains a fixed size and as a new vertex arrives the oldest vertex leaves. It will be seen that both models are connected, with diameter logarithmic in the network size.

The first model is the simplest case. We show that it behaves as expected and study how well it manages to restructure itself under adversarial deletion of vertices and edges. We assume that the vertices are added to the network in batches. Although this is for convenience of analysis, it has the added benefit that it allows a high arrival rate. We show that this batch protocol can be implemented efficiently in a distributed manner. As the network evolves, it is natural that existing vertices should wish to leave the network. The generalization of this model to networks where vertices leave randomly is the subject of a later study.

The second model, which we refer to as a first-in first-out (FIFO) queue, maintains a fixed size network. Informally we can think of this network as an *obsolescence network* in which inter-connected equipment fails or is replaced at a certain age, and this is effected in a distributed manner.

Batch model. The network adds new vertices in batches at fixed intervals $[T_0, T_1]$, $[T_1, T_2], \ldots$ Let $[T, T + \Delta T]$ be any of these intervals and let $G(T) = (V(T), E(T))$ be the network at step T. This network updates to $G(T + \Delta T)$ at time $T + \Delta T$ to include the new vertices which have successfully joined the network between steps T and $T + \Delta T$. We will sometimes call an interval $[T_{i-1}, T_i]$ an *epoch*. We use capital and small letters to distinguish between the step number T and the number of vertices $t = |V(T)|$ in the network at step T. Similarly we write Δt for the number of vertices added between steps T and $T + \Delta T$. We also write $G(t)$ and $V(t)$ for $G(T)$ and $V(T)$, if the size of the network is the more important parameter than the step number. 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 $\mathcal{T} = (t_1, t_2, \ldots)$ where t_i is the index of the last note in batch i.

The size Δt of the next batch of vertices can be much larger than ΔT, which we only require to be at least logarithmic in the number of vertices (ie. $\Delta T \geq K \log t$ for some suitably large positive constant K). We assume in our analysis that $\Delta t = o(t)$ (an assumption that $\Delta t \leq \rho t$ for some constant ρ would do). We believe that this restriction on Δt is not necessary, but without it the proof would be considerably more complicated.

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

To join the network at time $T+\Delta T$, a joining vertex u connects to the vertices $\{v_1, ..., v_m\}$ whose addresses are given on the m tokens. Each token generates one (possibly parallel) edge. Edge (u, v_j) is an out-edge of u and in-edge of v_j. Vertex u donates new cm tokens to the network labelled with its address. Each of these new tokens makes an independent random walk along with the existing tokens. Since each vertex uses up m tokens and donates cm 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 some vertices lose their connections. A 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 a vertex 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. Our protocol will create a well-connected network, if $\Delta T \geq K \log t$. However, the vertices in the network do not know the size of the network, so how can they figure out what the value of ΔT should be? One possible approach to this problem is to decide in advance a function $i \mapsto \Delta T_i$ according to the predicted growth of the network. Another approach is to dynamically maintain an upper bound on the size of the network. This could be implemented in a distributed fashion in the following way. Assume that all vertices know the last epoch i when the upper bound $\bar{t}$ on the number of vertices was updated. The vertices assume that $\bar{t}$ is greater than, say, four times the number of vertices in the network at the end of epoch i, and use this bound to calculate the length of the subsequent epochs. During the second half of an epoch $j > i$, each vertex v checks the age of the tokens passing through it and calculates the ratio of the number of old tokens (those created up to epoch i) to the number of new tokens. If this ratio indicates that the network has increased by at least twice, then v broadcasts a message to all other vertices and all vertices update $\bar{t}$ by multiplying it by the same factor. One can show that under some reasonable probabilistic assumptions, $\log \bar{t}$ remains $\Theta(\log t)$. We do not analyse this process in this paper. We simply assume that always $\Delta T \geq K \log t$.

FIFO queue network. We consider the following model. The network builds up to size t in the usual fashion. When vertex $t+1$ is added, vertex 1 is deleted, and this repeats itself at all steps $t+k,\ k \geq 1$. For more generality, we assume that the arriving and departing vertices are organized into batches: the oldest Δt vertices leave and then Δt new vertices arrive. It is assumed that vertices leave in an orderly fashion (oldest first) passing any spare tokens to neighbours before departing. Broken out-edges are replaced in the usual way, by sampling tokens from the network.

3 Results

Let $m \geq 400$ and $c \geq 5$. The large value of m is due to simplifying assumptions in the proofs. As mentioned above, we assume that ΔT is always at least $K \log t$, for an appropriately large constant K. That is, we assume that if the conductance of the network is greater than a fixed constant, then the distribution of the tokens is close to stationary before they are used.

Theorem 1 *The batch network $G(T) \equiv G(t)$ has bounded maximum degree $(c+1)m$ and 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.*

We prove the bound on the diameter as follows. 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 (1) below. As $\Phi(T)$ is constant $G(T)$ is rapidly mixing and we can adjust 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 ensures that the conductance $\Phi(T+\Delta T)$ of graph $G(T+\Delta T)$ is also constant **whp**.

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

3.1 Batch 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/vertex deletion, but we show that in certain cases the network remains connected or is able to re-connect itself. The re-connection 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. connectivity) 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.

Deletion or disconnection of the first s vertices $[s]$ is in expectation at least the most damaging option over any other set of size s. As $[s]$ has the highest expected in-degree, 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. Moreover, it has few free tokens (at most $(c-1)ms(s/t)^{1/(c-1)}$) walking the network $[s+1, ..., t]$. The detailed results are established in Section 6.

Theorem 3

(a) Deletion of the set of $[s]$, $s \leq t^{\frac{m-2}{m-1}}$ leaves the network $G(t) \setminus [s]$ connected **whp**.

(b) *The edge cut between* $[s]$ *and* $G(t) \setminus [s]$ *is deleted. Then for* $s \geq t^{(c+1)/2c}$ **whp** *the network reconnects implicitly except for a set of size at most* $te(s(c+1)/t)^m$.

(c) *If edges are deleted, splitting the network into components of size at least* $s = t^{(c+2)/2c}$, *then* **whp** *the network reconnects itself implicitly.*

For smaller values of s in parts (b), (c) or larger s in part (a), we have to rely on eventual reconnection to repair the network. This is extremely effective for component sizes down to $t^{1/(c-1)}$, but we do not analyse this case here.

4 Analysis of the batch model

We consider graphs which are undirected for random walks, but each edge does have an underlying direction assigned to it. For a graph $G = (V, E)$, a vertex $v \in V$, subsets $A, B \subseteq V$, $d(v)$ is the (undirected) degree of v, $d(A) = \sum_{v \in A} d(v)$, and $d(A : B)$ is the number of directed edges 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 $A \subseteq V, \bar{A} = V \setminus A$.

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_u denote the stationary distribution of $W_u(\tau)$, that is, $\pi_u(v) = \lim_{\tau \to \infty} P_u^{(\tau)}(v)$. For an unbiased random walk on a connected non-bipartite graph G with $e(G)$ edges, the stationary distribution exists and $\pi_u(v) = \frac{d(v)}{d(V)} \equiv \pi(v)$.

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

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

It follows from Jerrum and Sinclair [9] that

$$|P_u^{(\tau)}(v) - \pi_u(v)| = O\left(n^{1/2}\left(1 - \frac{\Phi^2}{2}\right)^{\tau}\right), \tag{2}$$

where $n = |V(G)|$. Thus if $\tau = K \log n$, for sufficiently large K, (1) above will be $O(n^{-10})$ at τ. We remark that there is a technical point here. The result of [9] assumes that the walk is *lazy*, and only makes a move to a neighbour with probability $1/2$ at any step. This way the stationary distribution of a random walk exists, and (2) holds equally for bipartite graphs. We assume in our analysis that random walks are lazy, but the analysis can be easily adapted to the non-lazy walks since our protocol creates networks which with high probability are not bipartite.

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

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

Proof Let $V = V(t)$ and $B = V - A$. Assuming that $\Phi(A) \leq \lambda$ and substituting $m|A| - d(A:A) = d(A:V) - d(A:A)$ for $d(A:B)$, we have

$$\begin{aligned} \lambda &\geq \Phi(A) = \frac{d(A:B) + d(B:A)}{2d(A:A) + d(A:B) + d(B:A)} \geq \frac{d(A:B)}{2d(A:A) + d(A:B)} \\ &= \frac{m|A| - d(A:A)}{m|A| + d(A:A)}. \end{aligned} \tag{3}$$

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

$$\begin{aligned} d(A:A) &= d(A) - d(A:V) - d(B:A) = d(A) - m|A| - d(B:A) \leq d(A) - m|A| \\ &\leq mt - m|A|. \end{aligned} \tag{4}$$

Inequalities (3) and (4) imply that $|A| \leq (1+\lambda)t/2$. □

For a subset $A \subseteq V(t)$ and $s \in \mathcal{T} \cap [t]$, we write $A(s)$ for $A \cap V(s)$ and $a(s)$ for $|A(s)$ (in particular, $A(t) = A$ and $a(t) = |A|$). The tokens which originate 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 $A \subset V(t)$, $s \in \mathcal{T} \cap [t]$, $a(s) \geq a(t)/2$, and there are fewer than $(c-3)ma(s)$ $A(s)$-tokens available at any time after the time T_s when the network $G(s)$ is established, 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 at some time after T_s, 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 $d(A:A) = ma(t)$, so

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

□

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

Theorem 4 *Let m and c be suitably large constants and let $\Phi(t_0) \geq \lambda_0 > 0$. There exist constants $\lambda = \lambda(m, c, t_0, \lambda_0)$ and $K = K(\lambda)$ such that with probability at least $1 - 1/t_0$, $\Phi(t) \geq \lambda$ for all $t \geq t_0$,*

Proof We prove inductively that with probability at least $1-\sum_{s=t_0+1}^{t} 1/s^2 (\geq 1-1/t_0)$, $\Phi(t) \geq \lambda$ for all $t \geq t_0$. To do so, we assume that for each $t_0 \leq s < t$, $\Phi(s) \geq \lambda$, and show that $\Phi(t) \leq \lambda$ with probability at most $1/t^2$.

Let $K = 22/\lambda^2$. Since $\Delta S = K \log s$ and $\Phi(s) \geq \lambda$ ($s > t_0$, $G(S) \equiv G(s)$), then it follows from (1) and the assumptions of the theorem that at step $S+(\Delta S)/2$, the tokens are within $o(s^{-10})$ of the stationary distribution. This means that for $v \in V(s)$, considering the sequence of tokens visiting v in the interval $[S+(\Delta S)/2, S+\Delta S]$, if $X = \{x_1, \ldots, x_J\}$ is the set of tokens in the network when the i-th token of this sequence visits v, then this token is x_j with probability $(1+o(s^{-10}))/J$. We omit below this deviation factor $1+o(s^{-10})$ to simplify the presentation. To account for it, one would need to introduce factors $1+o(1)$ to (5), (8) and (10), but the subsequent derivations would subsume such factors anyway.

We consider now 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 $V(s+\Delta s)$ ($s \notin V(s)$) generating edges $e_1, .., e_m$. The probability that the terminal vertex of edge e_i is in $A(s)$ is

$$p(v,i;A(s)) = \frac{X_A(s) - U_A(v,i)}{m(c-1)s - U(v,i)}, \tag{5}$$

where $U(v,i)$ (resp. $U_A(v,i)$) is the number of tokens already used up from the $V(s)$ (resp. $A(s)$) between $s+\Delta s/2$ 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 $\lambda \leq 1/M$. We consider four cases depending on the size of A and the placement of the vertices of A in the whole sequence if 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 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 $b/4$ B-vertices are added after t_A ($b(t_B) \leq b(t)/2 + o(t_B)$) so at least $b/4$ B-vertices are added when there are already at least $a/2$ A-vertices in the network. Similarly, if $t_B \leq t_A$, then at least $a/4$ A-vertices are added when there are already at least $b/2$ B-vertices in the network.

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

Let $N = |d(B:A)|$ be the number of edges from $B(t)$ to $A(t)$. Let $s \geq t_A$ be the end of a batch and let $v \in B(t)$, $v > s$. Then (5) implies that $p(v,i,A(s)) \geq a/(4s)$, since Lemma 2 allows us to assume that $X_A - U_A(v,i) \geq (c-3)ma(s)$, and $a(s) \geq a/2$. Thus

$$\mathbf{E}N \geq m\frac{a}{4}\sum_{s=t-b/4}^{t}\frac{1}{s} > \frac{ma}{4}\frac{b}{4t}.$$

Since $b = t - a \geq t/3$, we have $\mathbf{E}N \geq ma/48$. By the Hoeffding Inequality [8], for $\delta < 1$ we have

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

Hence, for $\delta = 1/2$ and $M \leq m/384$,

$$\mathbf{Pr}(|d(B:A)| \leq ma/96) \leq e^{-Ma} < e^{-t}.$$

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

$$\mathbf{Pr}(\exists A : \text{case 1(i) applies and } \Phi(A) \leq \lambda) \leq 2^t e^{-t} < \frac{1}{4t^2}. \tag{6}$$

Case 1(ii): $t_B \leq t_A$. Similarly to case 1(ii), we can show that $\mathbf{Pr}(|d(A:B)| \leq ma/96) \leq e^{-t}$, so there exists a constant $\lambda = \lambda(m, c, t_0)$ such that

$$\mathbf{Pr}(\exists A : \text{case 1(ii) applies and } \Phi(A) \leq \lambda) \leq 2^t e^{-t} < \frac{1}{4t^2}. \tag{7}$$

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

From the way the network is constructed, it is impossible for a subset of vertices $A \subset V(t)$ to be disconnected which implies that $\Phi_A \geq 1/(2m|A|+1)$. Thus we only need to consider $|A| > S$ for some large constant S.

Let κ be the index in $\mathcal{T} \cap [t]$ which is nearest to $\frac{1}{2}\sqrt{at}$. Let $A^- = A(t) \cap [\kappa]$.

Case 2(i): $|A^-| \geq a/2$. Let

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

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

$$\mathbf{Pr}(v \in \mathcal{N}) \leq \left(1 - \frac{m(c-3)a^-}{(c-1)mv}\right)^m. \tag{8}$$

If $|\mathcal{H}(t)| \geq a$, then $|\mathcal{H}(t) \cap B| \geq a/2$, implying that $d(B:A) \geq a/2$ and $\Phi(A)$ is greater than some constant. Hence we can assume that $|\mathcal{H}(t)| \leq a$. 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

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

The last inequality follows from the fact that $\kappa \leq \sqrt{at}$. Thus for $\beta = m\frac{c-3}{5(c-1)}$, $\mathbf{Pr}(\mathcal{H}(t) = H) \leq (a/t)^{\beta a}$, and for some constant $\lambda = \lambda(m, c, t_0)$,

$$\begin{aligned}
&\mathbf{Pr}(\exists A : \text{case 2(i) applies and } \Phi(A) \leq \lambda) \\
&\leq \mathbf{Pr}(\exists a, A^-, H : S \leq a \leq t/M,\ A^- \subseteq [\kappa],\ a \geq |A^-| \geq a/2, \\
&\qquad\qquad H \subseteq [\kappa+1 \ldots t],\ |H| < a,\ \mathcal{H}(t) = H) \\
&\leq ta^2 \binom{t}{a}^2 (a/t)^{\beta a} \leq (a/t)^{-a} < \frac{1}{4t^2}. \qquad (9)
\end{aligned}$$

The last but one inequality above holds for sufficiently large m.

Case 2(ii): $|A^-| < a/2$.

Let $N = |d(A^+ : A)|$. We will prove that the probability that $N > ma/4$ is small. This will imply that since there are at least $ma/2$ edges from A^+, there are with high probability at least $ma/4$ edges from A^+ to B, so $\Phi(A)$ is greater than a constant.

Let $\kappa \leq s \leq t-1$. When vertex $s+1$ is getting 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

$$\frac{a(s)}{s} \leq \frac{a}{\kappa} \leq 4\sqrt{\frac{a}{t}}.$$

Thus

$$\mathbf{E}N \leq a \cdot \frac{2mc}{c-1} \cdot 4\sqrt{\frac{a}{t}} = \frac{ma}{4}\left(\frac{32c}{c-1}\sqrt{\frac{a}{t}}\right). \qquad (10)$$

The Chernoff inequality implies that

$$\mathbf{Pr}(N \geq \delta(\mathbf{E}N)) \leq \left(\frac{e}{\delta}\right)^{\delta(\mathbf{E}N)}. \qquad (11)$$

Using (10) and (11) for $\delta = (ma/4)/(\mathbf{E}N)$, we get

$$\mathbf{Pr}(N \geq ma/4) \leq \left(\frac{e}{\delta}\right)^{ma/4} \leq \left(\frac{a}{t}\right)^{\beta a},$$

for $\beta = m/16$ and t sufficiently large. Thus for sufficiently large m and for some constant $\lambda = \lambda(m, c, t_0)$,

$$\mathbf{Pr}(\exists A : \text{case 2(ii) applies and } \Phi(A) \leq \lambda) \leq \binom{t}{a}\left(\frac{a}{t}\right)^{\beta a} \leq \left(\frac{a}{t}\right)^{a} \leq \frac{1}{4t^2}. \qquad (12)$$

Inequalities (6), (7), (9) and (12) 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.

Corollary 5 *For sufficiently large constants m and c,* **whp** $|E(A:\overline{A})| > |A|$ *for all* $A \subseteq V(t)$, $1 \le |A| \le 2t/3$.

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 $\pi(S) \le 1/2$, then $E(S:\overline{S}) \ge \lambda d(S)$. The vertex set $N(S)$ of the next tree consists of the vertices in S and all their neighbours. Therefore $d(N(S)) \ge d(S) + E(S:\overline{S}) \ge (1+\lambda)d(S)$. 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

We use the following labeling for the vertices in or leaving the queue. The current network is $[1, ..., t]$. The $t+1$ vertices before addition of vertex 1 are labelled $[-t, ..., 0]$. 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 this edge the *final replacement edge of v*.

Theorem 6 *The current network of size t is connected and has diameter $O(\log t)$* **whp**.

Proof We repeat the conductivity arguments of Lemma 1. For brevity we only consider the most difficult cases (small sets). Assume $V(t) = A \cup B$, $a \le t/M$. As before let $\kappa = \lceil \sqrt{at}/2 \rceil$ and $A^- = A(\kappa)$.

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

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

$$\mathbf{E}N(A^- : \kappa) \le \frac{cm\kappa}{(c-1)mt} ma = \frac{cm}{2(c-1)} a\sqrt{\frac{a}{t}}.$$

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

$$\mathbf{Pr}(N(A^- : \kappa) \ge \lfloor ma/8 \rfloor) \le \left(\frac{e}{\beta}\right)^{ma/8}.$$

Thus

$$\begin{aligned}\mathbf{Pr}(\exists A, |A| = a,\ N(A^- : \kappa) \geq ma/8) &\leq \binom{t}{a}\left(\frac{e}{\beta}\right)^{ma/8} \\ &\leq \exp -a\left(\frac{m}{16}\log t/a - \log te/a - -\frac{m}{8}\log 4ec/(c-1)\right) \\ &\leq \left(\frac{a}{t}\right)^{a\left(\frac{m}{16}\log M - \log M - 1 - \frac{m}{8}\log 4ec/(c-1)\right)},\end{aligned}$$

which is $o(1)$ for $M \geq e^8$ and $m \geq 50$. By a similar argument at most $ma/8$ edges of A^- go to $[-t+1, ..., -t+\kappa]$ so that at least $\lceil 3ma/4 \rceil$ edges of A^- are replaced after κ by deletion of vertices in $[-t+\kappa+1, ..., 0]$.

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

Let $N'(A)$ count the subset of the (first) $3ma/4$ edges of A^- finally replaced after κ which point to A. Thus $\mathbf{E}N'(A) \leq a\sqrt{a/t}(3mc/2(c-1))$, and using $\beta = \sqrt{t/a}(c-1)/6c$ we have

$$\mathbf{Pr}(N'(A) \geq ma/4) \leq (e/\beta)^{ma/4}.$$

It follows that

$$\mathbf{Pr}(\exists A, |A| = a,\ N'(A) \geq ma/4) \leq \exp -a\left(\frac{m}{8}\log t/a - \log t/a - O(1)\right)$$

very much as before.

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

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

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

where $j = |A(x(j)|$ at step $x(j) \geq \kappa$ at which the final replacement is made. Thus $\mathbf{E}N'(A) \leq a\sqrt{a/t}(2mc/(c-1))$ and by calculations similar to above

$$\mathbf{Pr}(\exists A, |A| = a, 1 \leq a \leq t/M \text{ such that } N'(A) \geq ma/4) = o(1).$$

□

6 Batch model: robustness under adversarial deletion

Given a set S, the numberX_t of available S-tokens in the network at step t is

$$X_t = cms - d(S \times S) - d((V \setminus S) \times S),$$

where $s = |S|$. Let Y_t be the number of S-tokens removed at step t. Thus $X_{t+1} = X_t - Y_t$. Y_t is hypergeometric $H(m, X_t, (c-1)mt)$ and $\mathbf{E}Y_t = mX_t/((c-1)mt)$. In the case where $S = [s]$ we have

$$\mathbf{E}X_t = X_s \prod_{\tau=s+1}^{t} \left(1 - \frac{1}{(c-1)\tau}\right) = (1+o(1))(c-1)ms\left(\frac{s}{t}\right)^{1/(c-1)}.$$

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

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

The value of $X_t([s])$ is sharply concentrated by martingale arguments provided $s/\sqrt{t} \to \infty$. For any other set S of size s, $\mathbf{E}X_t(S)$ is at least the value of $\mathbf{E}X_t([s])$.

6.1 Adversarial deletion of vertices

We consider the case where the first s vertices $S = [s]$ are deleted at step t. The set $[s]$ has the largest expected in-degree $(c-1)ms(1-(s/t)^{1/(c-1)})$ among all subsets of vertices of size s. Thus deletion of $[s]$ can, in expectation at least, cause more damage than deletion of any other set of size s. In the theorem and the proof below we consider only the simplified case when vertices are added one by one, but the proof can be extended to cover the case when vertices are added in batches. The extension requires an upper bound t^α on the size of the batch at step t, for some constant $0 < \alpha < 1$.

Theorem 7 *For $s = o(t^{1-1/m})$ deletion of $S = [s]$ does not disconnect $G(t) \setminus S$* **whp**. *In general, deletion of S may disconnect a set of size $te(s(c+1)/t)^m$.*

Proof Even this simple case requires a rather careful analysis. Let $A \subseteq [s+1, ..., t]$, $B = [s+1, ..., t] \setminus A$, $A \neq \emptyset$, and $B \neq \emptyset$. We bound the probability that $E(A:B) = 0$, that is, the probability that there are no edges between A and B so that $G(t) \setminus S$ is disconnected.

Let V_A be the steps in $[s+1, ..., t]$ at which vertices of A are added, and let $A(v)$ be A after step v and $a(v) = |A(v)|$. define analogously V_B, $B(v)$, and $b(v)$. Assume that there are no edges between A and B by step v. If $v \in V_A$, then during step v there are total $(c-1)m(v-1)$ 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 of $cmb(v)$ B-tokens have been used). Thus 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 $v \in V_B$, then the probability that at this step v 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 $s+a\le t$ as $a+b\le t-s$ and assume that $a\le b$ to derive $P\le\left(\frac{s+a}{t}\right)^a$. Thus

$$\mathbf{Pr}(\exists A, |A|=a \text{ and } E(A:B)=0) \le \left(\frac{te}{a}\left(\frac{s+a}{t}\right)^m\right)^a. \tag{13}$$

The maximum in-degree of S is cms and the out-degree of A is ma, so whenever $a>cs$, we must have $E(A:B)>0$. If $a\le cs$, then the right-hand side of (13) is $o(1)$, if $s=o(t^{1-1/m})$.

6.2 Adversarial deletion of edges

We give a brief analysis of this to indicate the types of arguments which can be used.

Theorem 8 *Let $t^{(c+1)/2c}\ll s<t2^{-c}$ and let $m\ge 10$. Let the edges between $[s]$ and $G(t)\backslash[s]$ be cut, and let the tokens in the network be in the stationary distribution when the cut was made. All but at most $te(s(c+1)/t)^m$ vertices reconnect to $[s]$ immediately* **whp**.

Proof For $s\le t2^{-c}$ **whp** $G(t)\backslash[s]$ contains a large component L, and some small components A of total size at most $te(s(c+1)/t)^m$. Thus the size of L is at least

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

Note also that $s\gg|A|(c+1)m$. As there were at least $(s+a)$ edges between L and $S\cup A$ (Corollary 5), at least $s+a-(c+1)ma>s/2$ of these edges are between L and $[s]$ and at least $s/2m$ vertices B of L have broken edges.

The number X_t of available $[s]$-tokens has expected value $\mathbf{E}X_t(1+o(1))(c-1)ms(s/t)^{1/(c-1)}$ which is concentrated provided $\mathbf{E}X_t/\sqrt{t}\to\infty$, ie. $s\gg t^{(c+1)/2c}$.

By the uniformity assumption at least $Y=\mathbf{E}X_t/2$ of these tokens are in stationary distribution on L, and the expected number of these on B is at least $Yd(B)/2mt$. The expected number of tokens on B at the break is $(d(B)/2mt)(c-1)mt$. Thus

$$\mathbf{Pr}(\text{no replacement edges points to } [s]) \le \left(1-\frac{Y}{(c-1)mt}\right)^{s/2m} = o(1).$$

Theorem 9 *Let $G(t)$ be disconnected by edge cuts into components A_j, $j \in J$, where $V(t) = \cup_{j \in J} V(A_j)$ and $\min |V(A_j)| \geq t^{(c+2)/2c}$, then $V(t)$ will reconnect* **whp**.

Proof Let H be the graph $H = \cup_{j \in J} A_j$. Let A be a component of size at most $t/2$, let $B = G(t)/A$, and let $|A| = a$. By Corollary 5 in $G(t)$ we have that there are at least a edges between A and B. Either there are at least $a/2$ edges from A to B or vice versa. We consider the first case here. the proof for the second case is similar.

In the case of at least $a/2$ edges from A to B, there are at least $\beta = (c-1)mt - (c+1)ma$ B-tokens available, and some number Y of these are on A in H, where $\mathbf{E}Y \geq \beta d(A)/2mt \geq \beta a/2t$. Thus

$$\mathbf{Pr}(Y \leq \tfrac{1}{2}\mathbf{E}Y) \leq \exp -\frac{1}{16}\frac{a\beta}{t} = o(e^{-\sqrt{t}}).$$

Let Z be the number of reconnecting edges from A to B, then $\mathbf{E}Z \geq (a/2)a\beta/4t$. Thus $\mathbf{Pr}(Z \leq \mathbf{E}Z/2) = o(e^{-\sqrt{t}})$ very much as before.

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

[4] 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.

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

[6] 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.

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

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

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399