THEORY OF COMPUTING, Volume 3 (2007), pp. 1–23
http://theoryofcomputing.org

Censorship Resistant Peer-to-Peer Networks

Amos Fiat [*] Jared Saia[†]

Received: March 3, 2006; published: January 21, 2007.

Abstract: We present a censorship resistant peer-to-peer network for accessing n data items in a network of n nodes. Each search for a data item in the network takes $O(\log n)$ time and requires at most $O(\log^2 n)$ messages. Our network is censorship resistant in the sense that even after adversarial removal of an arbitrarily large constant fraction of the nodes in the network, all but an arbitrarily small fraction of the remaining nodes can obtain all but an arbitrarily small fraction of the original data items. The network can be created in a fully distributed fashion. It requires only $O(\log n)$ memory in each node. We also give a variant of our scheme that has the property that it is highly spam resistant: an adversary can take over complete control of a constant fraction of the nodes in the network and yet will still be unable to generate spam.

ACM Classification: F.2.2

AMS Classification: 68W15

Key words and phrases: peer-to-peer; content addressable network (CAN); distributed hash table (DHT); censorship; spam; fault-tolerant; butterfly network; routing; Byzantine; probabilistic method; expander graphs.

[*]Work done while on sabbatical at University of Washington.
[†]Work done while a student at the University of Washington.

DOI: 10.4086/toc.2007.v003a001

Thomas Hobbes, René Descartes, Francis Bacon, Benedict Spinoza, John Milton, John Locke, Daniel Defoe, David Hume, Jean-Jacques Rousseau, Blaise Pascal, Immanuel Kant, Giovanni Casanova, John Stuart Mill, Émile Zola, Jean-Paul Sartre, Victor Hugo, Honore de Balzac, A. Dumas pere, A. Dumas fil, Gustave Flaubert, Rabelais, Montaigne, La Fontaine, Voltaire, Denis Diderot, Pierre Larousse, Anatole France

Partial list of authors in *Index Librorum Prohibitorum* (Index of Prohibited Books) from the Roman Office of the Inquisition, 1559–1966.

1 Introduction

Web content is under attack by state and corporate efforts to censor it, for political and commercial reasons [9, 17, 40]. Peer-to-peer networks are considered more robust against censorship than standard web servers [29]. However, while it is true that many suggested peer-to-peer architectures are fairly robust against random faults, the censors can attack carefully chosen weak points in the system. For example, the Napster [28] file sharing system has been effectively dismembered by legal attacks on the central server. Additionally, the Gnutella [14] file sharing system, while specifically designed to avoid the vulnerability of a central server, is highly vulnerable to attack by removing a very small number of carefully chosen nodes [34, 35].

A more principled approach than the centralized approach taken by Napster or the broadcast search mechanism of Gnutella is the use of a distributed hash table [38]. A distributed hash table (DHT) is a distributed, scalable, indexing scheme for peer-to-peer networks. Plaxton, Rajaram, and Richa [31] give a scheme to implement a distributed hash table (prior to its definition) in a web cache environment. Subsequent distributed hash tables have been suggested in [38, 32, 31, 41, 24, 25]. In the next two subsections, we give details of our two results: a DHT which is robust to adversarial node deletion and a DHT which is spam resistant.

1.1 Resistance to adversarial node deletion

We present a distributed hash table with n nodes used to store n distinct data items. The scheme is robust to adversarial deletion of up to half of the nodes in the network and has the following properties:

1. With high probability, all but an arbitrarily small fraction of the nodes can find all but an arbitrarily small fraction of the data items.

2. Search takes (parallel) time $O(\log n)$.

3. Search requires $O(\log^2 n)$ messages in total.

4. Every node requires $O(\log n)$ storage.

Some remarks. For simplicity, we have assumed that the number of items and the number of nodes is equal. However, for any n nodes and $m \geq n$ data items, our scheme will work, where the search time remains $O(\log n)$, the number of messages remains $O(\log^2 n)$, and the storage requirements are $O(\log n \times m/n)$ per node. Also for simplicity, we give our results and proofs for the case where the adversary deletes up to a $1/2$ fraction of the nodes. However we can easily modify our scheme to work for any constant less than 1. This would change the constants involved in storage, search time, and messages sent, by a constant factor.

As stated above, in the context of state or corporate attempts at censorship, it seems reasonable to consider *adversarial* attacks rather than random deletion. Our scheme is a distributed hash table that is robust against adversarial deletion of a $1/2$ fraction of the nodes. We remark that such a network is clearly resilient to having up to 1/2 of the nodes removed at random (in actuality, its random removal resiliency is much better). We further remark that if nodes come up and down over time, our network will continue to operate as required so long as, at any point in time, at least $n/2$ of the nodes are alive.

Finally, we note that in our model, it is unavoidable that after an attack some nodes may not be able to reach any data items and that some data items may not be able to be reachable by any nodes. In particular, if all nodes store only $O(\log n)$ pointers, then for any algorithm, the adversary can easily target a set, T, of $O(n/\log n)$ nodes and then delete all the nodes that are neighbors of any node in the set T. The nodes in T will then be completely isolated and unable to reach any data items. Similarly, the adversary can target some set of $O(n/\log n)$ of the data items, and then delete all nodes on which those data items are stored, ensuring that no node will be able to reach any data item in this targeted set. Thus, the robustness of our algorithm is optimal up to a $\log n$ factor among all algorithms that are *scalable*, in the sense that the each node requires $O(\log n)$ storage.

1.2 Spam resistance

Spamming has been a problem with peer-to-peer networks [7, 13]. Because the data items reside in the nodes of the network, and pass through nodes while in transit, it is possible for nodes to invent alternative data and pass it on as though it was the sought after data item.

We now describe a spam resistant variant of our distributed hash table. To the best of our knowledge this is the first such scheme of its kind. As before, assume n nodes used to store n distinct data items. The adversary may choose up to some constant $c < 1/2$ fraction of the nodes in the network. These nodes under adversary control may be deleted, or they may collude and transmit arbitrary false versions of the data item, nonetheless:

1. With high probability, all but an arbitrarily small fraction of the nodes will be able to obtain all but an arbitrarily small fraction of the *true* data items. To clarify this point, the search will *not* result in multiple items, one of which is the correct item. The search will result in one unequivocal true item.

2. Search takes (parallel) time $O(\log n)$.

3. Search requires $O(\log^3 n)$ messages in total.

4. Every node requires $O(\log^2 n)$ storage.

The rest of our paper is structured as follows. We review related work in Section 2. We give the algorithm for creation of our robust distributed hash table, the search mechanism, and properties of the distributed hash table in Section 3. The proof of our main theorem, Theorem 3.1, is given in Section 4. In Section 5 we sketch the modifications required in the algorithms and the proofs to obtain spam resistance, the main theorem with regard to spam resistant distributed hash tables is Theorem 5.1. We conclude and give directions for future work in Section 6. Acknowledgements are in Section 7.

2 Related work

2.1 Distributed hash tables – adversarial deletions and Byzantine faults

The work described in this paper was first published in [10]. Work subsequent to this publication has improved on the deletion-resistant network in various ways. Mayur Datar [8] gives a distributed hash table based on the multibutterfly network which improves in two ways on our deletion-resistant network. First, he improves on our resource costs by requiring only $O(\log n)$ messages per query and $O(1)$ pointers to be stored at each node in the network. Second, he shows how his network can be maintained dynamically when large numbers of nodes are added or deleted from the network. Unfortunately, his techniques do not carry over to the creation and maintenance of a spam-resistant network.

There is also subsequent work related to designing DHTs which are robust to Byzantine faults. Naor and Wieder describe a simple DHT which is robust to each node suffering a Byzantine fault independently with some fixed probability [27]. Hildrum and Kubiatowicz [16] describe how to modify two popular DHTs, Pastry [33] and Tapestry [41], in order to make them robust to this same type of attack. More recent work addresses the problem of designing DHTs which are robust to many Byzantine peers joining and leaving the network over a long period of time [4, 5, 36, 11].

2.2 Distributed hash tables – random deletions

Recent years have witnessed the advent of large scale real-world peer-to-peer applications such as eDonkey, BitTorrent, Morpheus, Kazaa, Gnutella, and many others. These networks can have on the order of hundreds of thousands or even millions of nodes in them. Several distributed hash tables (DHTs) have been introduced which are shown empirically and analytically to be robust to random peer deletions (i. e., fail-stop faults) [32, 38, 41, 33, 19, 24, 25].

Experimental measurements of a connected component of the real Gnutella network have been studied [35], and it has been found to still contain a large connected component even with a $1/3$ fraction of random node deletions.

2.3 Faults on networks

2.3.1 Random faults

There is a large body of work on node and edge faults that occur independently at random in a general network. Håstad, Leighton, and Newman [15] address the problem of routing when there are node and edge faults on the hypercube which occur independently at random with some probability $p < 1$. They give a $O(\log n)$ step routing algorithm that ensures the delivery of messages with high probability even when a constant fraction of the nodes and edges have failed. They also show that a faulty hypercube can emulate a fault-free hypercube with only constant slowdown.

Karlin, Nelson, and Tamaki [18] explore the fault tolerance of the butterfly network against edge faults that occur independently at random with probability p. They show that there is a critical probability p^* such that if p is less than p^*, the faulted butterfly almost surely contains a linear-sized component and that if p is greater than p^*, the faulted butterfly does not contain a linear-sized component.

Leighton, Maggs, and Sitamaran [20] show that a butterfly network whose nodes fail with some constant probability p can emulate a fault-free network of the same size with a slowdown of $2^{O(\log^* n)}$.

2.3.2 Adversarial faults

It is well known that many common network topologies are not resistant to a linear number of adversarial faults. With a linear number of faults, the hypercube can be fragmented into components all of which have size no more than $O(n/\sqrt{\log n})$ [15]. The best known lower bound on the number of adversarial faults a hypercube can tolerate and still be able to emulate a fault free hypercube of the same size is $O(\log n)$ [15].

Leighton, Maggs, and Sitamaran [20] analyze the fault tolerance of several bounded degree networks. One of their results is that any n node butterfly network containing $n^{1-\varepsilon}$ (for any constant $\varepsilon > 0$) faults can emulate a fault free butterfly network of the same size with only constant slowdown. The same result is given for the shuffle-exchange network.

2.4 Other related work

One attempt at censorship resistant web publishing is the Publius system [39]; while this system has many desirable properties, it is not a peer-to-peer network. Publius makes use of many cryptographic elements and uses Shamir's threshold secret sharing scheme [37] to split the shares amongst many servers. When viewed as a peer-to-peer network, with n nodes and n data items, to be resistant to $n/2$ adversarial node removals, Publius requires $\Omega(n)$ storage per node and $\Omega(n)$ search time per query.

Alon et al. [1] give a method which safely stores a document in a decentralized storage setting where up to half the storage devices may be faulty. The application context of their work is a storage system consisting of a set of servers and a set of clients where each client can communicate with all the servers. Their scheme involves distributing specially encoded pieces of the document to all the servers in the network.

Aumann and Bender [3] consider tolerance of pointer-based data structures to worse case memory failures. They present fault tolerant variants of stacks, lists and trees. They give a fault tolerant tree with the property that if r adversarial faults occur, no more than $O(r)$ of the data in the tree is lost. This fault tolerant tree is based on the use of expander graphs.

Quorum systems [12, 22, 23] are an efficient, robust way to read and write to a variable which is shared among n servers. Many of these systems are resistant up to some number $b < n/4$ of Byzantine faults. The key idea in such systems is to create subsets of the servers called *quorums* in such a way that any two quorums contain at least $2b+1$ servers in common. A client that wants to write to the shared variable will broadcast the new value to all servers in some quorum. A client that wants to read the

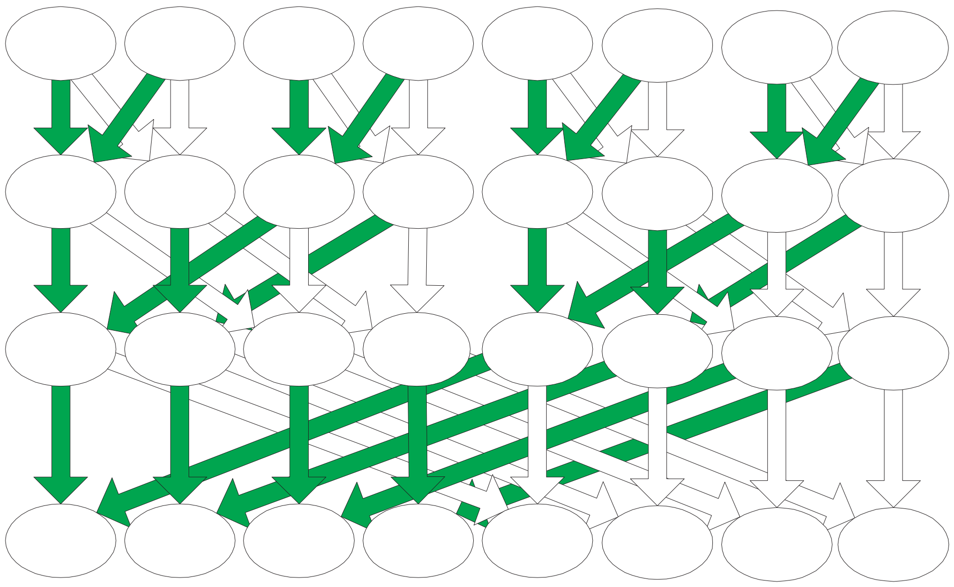

Figure 1: The butterfly network of supernodes.

variable will get values from all members in some quorum and will keep only that value which has the most recent time stamp and is returned by at least $b+1$ servers. For quorum systems that are resistant to $\Theta(n)$ faults the load on the servers can be high. In particular, $\Theta(n)$ servers will be involved in a constant fraction of the queries.

Recently Malkhi et al. [23] have introduced a probabilistic quorum system. This new system relaxes the constraint that there must be $2b+1$ servers shared between any two quoroms and remains resistant to Byzantine faults only with high probability. The load on servers in the probabilistic system is less than the load in the deterministic system. Nonetheless, for a probabilistic quorum system which is resistant to $\Theta(n)$ faults, there still will be at least one server involved in a constant fraction of the queries.

3 Our distributed hash table

We now state our mechanism for providing indexing of n data items by n nodes in a network that is robust to removal of any $n/2$ of the nodes. We make use of a butterfly network of depth $\log n - \log\log n$; we call the nodes of the butterfly network *supernodes* (see Figure 1). Every supernode is associated with a set of nodes. We call a supernode at the topmost level of the butterfly a top supernode, one at the bottommost level of the network a bottom supernode and one at neither the topmost or bottommost level a middle supernode.

3.1 The network

To construct the network we do the following:

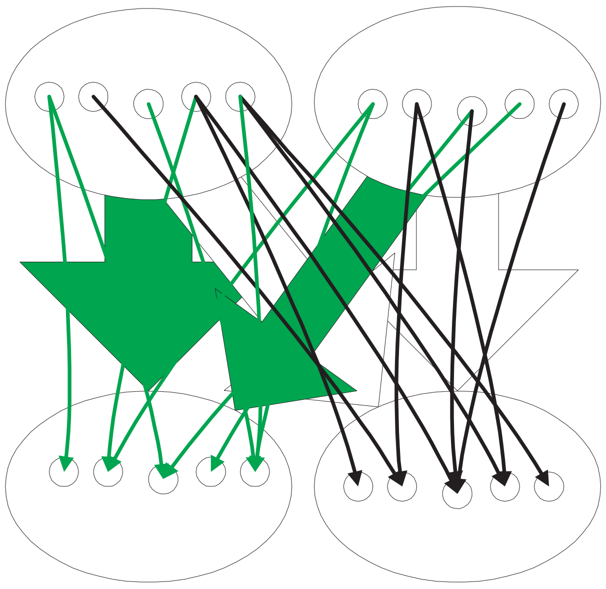

Figure 2: The expander graphs between supernodes.

- We choose an error parameter $\varepsilon > 0$, and as a function of ε we determine constants C, B, T, D, α and β. (See Theorem 3.1.)
- Every node chooses uniformly and independently at random C top supernodes, C bottom supernodes and $C \log n$ middle supernodes to which it will belong.
- Between two sets of nodes associated with two supernodes connected in the butterfly network, we choose a random constant degree expander graph of degree D (see Figure 2). (We do this only if both sets of nodes are of size at least $\alpha C \ln n$ and no more than $\beta C \ln n$.)
- We also map the n data items to the $n/\log n$ bottom supernodes in the butterfly. Every one of the n data items is hashed to B random bottom supernodes. (Typically, we would not hash the entire data item but only its title, e. g., "Singing in the Rain.") [1]
- The data item is stored in all the component nodes of all the (bottom) supernodes to which it has been hashed; if any bottom supernode has more than $\beta B \ln n$ data items hashed to it, it drops out of the network.
- In addition, every one of the nodes chooses uniformly and independently at random T top supernodes of the butterfly and points to all component nodes of these supernodes.

[1]We use the random oracle model [6] for this hash function, it would have sufficed to have a weaker assumption such as that the hash function is expansive.

3.2 Search

To perform a search for a data item, starting from node v, we do the following:

1. Take the hash of the data item and interpret it as a sequence of indices $i_1, i_2, \ldots, i_B$, $0 \le i_\ell \le n/\log n$.
2. Let $t_1, t_2, \ldots, t_T$ be the top supernodes to which v points.
3. Repeat in parallel for all values of k between 1 and T:
   (a) Let $\ell = 1$.
   (b) Repeat until successful or until $\ell > B$:
      i. Follow the path from t_k to the supernode at the bottom level whose index is i_ℓ:
         - Transmit the query to all of the nodes in t_k. Let W be the set of all such nodes.
         - Repeat until a bottom supernode is reached:
           – The nodes in W transmit the query to all of their neighbors along the (unique) butterfly path to i_ℓ. This transmission is done along the expander edges connecting the nodes in W to their neighbors in the supernode below. Let W be the new set of nodes in the supernode below the old W.
         - When the bottom supernode is reached, fetch the content from whatever node has been reached.
         - The content, if found, is transmitted back along the same path as the query was transmitted downwards.
      ii. Increment ℓ.

3.3 Properties of our distributed hash table

Following is the main theorem which we will prove in Section 4.

Theorem 3.1. *For all $\varepsilon > 0$, there exist constants $k_1(\varepsilon)$, $k_2(\varepsilon)$, $k_3(\varepsilon)$ which depend only on ε such that*

- *Every node requires $k_1(\varepsilon)\log n$ memory.*
- *Search for a data item takes no more than $k_2(\varepsilon)\log n$ time.*
- *Search for a data item requires no more than $k_3(\varepsilon)\log^2 n$ messages.*
- *All but εn nodes can reach all but εn data items.*

3.4 Some comments

1. **Distributed creation of our distributed hash table**

   We note that our distributed hash table can be created in a fully distributed fashion with n broadcasts or transmission of n^2 messages in total and assuming $O(\log n)$ memory per node. We briefly sketch the protocol that a particular node will follow to do this. The node first randomly chooses the supernodes to which it belongs. Let S be the set of supernodes which neighbors supernodes to which the node belongs. For each $s \in S$, the node chooses a set N_s of D random numbers between 1 and $\beta C \ln n$. The node then broadcasts a message to all other nodes which contains the identifiers of the supernodes to which the node belongs.

Next, the node will receive messages from all other nodes giving the supernodes to which they belong. For every $s \in S$, the node will link to the i-th node that belongs to s from which it receives a message if and only if $i \in N_s$.

If for some supernode to which the node belongs, the node receives less than $\alpha C \ln n$ or greater than $\beta C \ln n$ messages from other nodes in that supernode, the node removes all out-going connections associated with that supernode. Similarly, if for some supernode in S, the node receives less than $\alpha C \ln n$ or greater than $\beta C \ln n$ messages from other nodes in that supernode, the node removes all out-going connections to that neighboring supernode. Connections to the top supernodes and storage of data items can be handled in a similar manner.

2. **Insertion of a new data item**

   One can insert a new data item simply by performing a search, and sending the data item along with the search. The data item will be stored at the nodes of the bottommost supernodes in the search. We remark that such an insertion may fail with some small constant probability.

3. **Insertion of a new node**

   Our network does not have an explicit mechanism for node insertion. It does seem that one could insert the node by having the node choose at random appropriate supernodes and then forging the required random connections with the nodes that belong to neighboring supernodes. The technical difficulty with proving results about this insertion process is that not all live nodes in these neighboring supernodes may be reachable and thus the probability distributions become skewed.

   We note though that a new node can simply copy the links to top supernodes of some other node already in the network and will thus very likely be able to access almost all of the data items. This insertion takes $O(\log n)$ time. Of course the new node will not increase the resiliency of the network if it inserts itself in this way. We assume that a full reorganization of the network is scheduled whenever sufficiently many new nodes have been added in this way.

4. **Load balancing properties**

   Because the data items are searched for along a path from a random top supernode to the bottom supernodes containing the item, and because these bottom supernodes are chosen at random, the load will be well-balanced as long as the number of requests for different data items is itself balanced. This follows because a uniform distribution on the search for data items translates to a uniform distribution on top to bottom paths through the butterfly.

5. **Reducing storage costs**

   The scheme described stores each data item in $\Theta(\log n)$ nodes, resulting an $\Theta(\log n)$ blowup of space for storing the data items. We note that, in practice, it is possible to reduce the space required

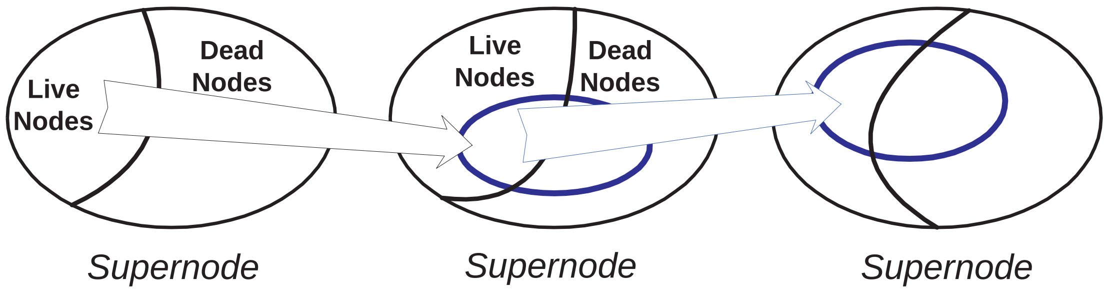

Figure 3: Traversal of a path through the butterfly.

for the data items by using erasure codes. In particular, instead of storing a copy of the data item at each of the $\Theta(\log n)$ nodes, we would just store an encoded piece of the data item with rate determined by our desired degree of fault tolerance. This would result in only a constant-factor blowup in storage without any loss in other performance measures. Any standard erasure code, such as tornado codes [21], can be used to achieve this reduction. We note that even with this change, each node will still require $\Theta(\log n)$ memory to store pointers to other nodes.

4 Proofs

4.1 Proof overview

Technically, the proof makes extensive use of random constructions and the probabilistic method [2].

We first consider the supernodes created in Section 3.1. In Sections 4.4 and 4.5, we show that with high probability, all but an arbitrarily small constant times $n/\log n$ of the supernodes are good, where good means that (a) they have $O(\log n)$ nodes associated with them, and, (b) they have $\Omega(\log n)$ live nodes after adversarial deletion. This implies that all but a small constant fraction of the paths through the butterfly contain only good supernodes.

We now consider the search protocol described in Section 3.2. Search is preformed by broadcasting the search to all the nodes in (a constant number of) top supernodes, followed by a sequence of broadcasts between every successive pair of supernodes along the paths between one of these top supernodes and a constant number of bottom supernodes. Fix one such path. The broadcast between two successive supernodes along the path makes use of the expander graph connecting these two supernodes. When we broadcast from the live nodes in a supernode to the following supernode, the nodes that we reach may be both live and dead (see Figure 4.1).

We now sketch the proof, given in Sections 4.6 and 4.7, that the search algorithm works correctly. Assume that we broadcast along a path, all of whose supernodes are good. One problem is that we are not guaranteed to reach all the live nodes in the next supernode along the path. Instead, we reduce our requirements to ensure that at every stage, we reach at least $\delta \log n$ live nodes, for some constant δ. The crucial observation is that if we broadcast from $\delta \log n$ live nodes in one supernode, we are guaranteed to reach at least $\delta \log n$ live nodes in the subsequent supernode, with high probability. This follows by using the expansion properties of the bipartite expander connection between two successive supernodes.

Recall that the nodes are connected to a constant number of random top supernodes, and that the data items are stored in a constant number of random bottom supernodes. The fact that we can broadcast along all but an arbitrarily small fraction of the paths in the butterfly implies that most of the nodes can reach most of the content.

In several statements of the lemmata and theorems in this section, we require that n, the number of nodes in the network, be sufficiently large to get our result. We note that, technically, this requirement is not necessary since if it fails then n is a constant and our claims trivially hold.

4.2 Definitions

Definition 4.1. A top or middle supernode is said to be (α,β)-good if it has at most $\beta \log n$ nodes mapped to it and at least $\alpha \log n$ nodes which are not under control of the adversary.

Definition 4.2. A bottom supernode is said to be (α,β)-good if it has at most $\beta \log n$ nodes mapped to it and at least $\alpha \log n$ nodes which are not under control of the adversary and if there are no more than $\beta B \ln n$ data items that map to the node.

Definition 4.3. An (α,β)-good path is a path through the butterfly network from a top supernode to a bottom supernode all of whose supernodes are (α,β)-good supernodes.

Definition 4.4. A top supernode is called (γ,α,β)-expansive if there exist $\gamma n/\log n$ (α,β)-good paths that start at this supernode.

4.3 Technical lemmata

Following are three technical lemmata about bipartite expanders that we will use in our proofs. The proof of the first lemma is well known [30] (see also [26]) and the proof of the next two lemmata are slight variants on the proof of the first.

Lemma 4.5. *Let ℓ, r, ℓ', r', d, and n be any positive values where $\ell' \le \ell$, $r' \le r$, and*

$$d \ge \frac{r}{r'\ell'}\left(\ell' \ln\left(\frac{\ell \mathrm{e}}{\ell'}\right) + r' \ln\left(\frac{r \mathrm{e}}{r'}\right) + 2 \ln n\right) \quad .$$

Let G be a random bipartite multigraph with left side L and right side R where $|L| = \ell$, $|R| = r$, and each node in L has edges to d random neighbors in R. Then with probability at least $1 - 1/n^2$, any subset of L of size ℓ' shares an edge with any subset of R of size r' .

Proof. We will use the probabilistic method to show this. We will first fix a set $L' \subset L$ of size ℓ' and a set $R' \subset R$ of size r' and compute the probability that there is no edge between L' and R' and will then bound the probability of this bad event for any such set L' and R'. The probability that a single edge does not fall in R' is $1 - r'/r$ so the probability that no edge from L' falls into R' is no more than $\mathrm{e}^{-r'\ell' d/r}$.

The number of ways to choose a set L' of the appropriate size is no more than $(\ell e/\ell')^{\ell'}$ and the number of ways to choose a set R' of the appropriate size is no more than $(r e/r')^{r'}$. So the probability that no two subsets L of size ℓ' and R of size r' have no edge between them is no more than:

$$\left(\frac{\ell e}{\ell'}\right)^{\ell'} \cdot \left(\frac{r e}{r'}\right)^{r'} \cdot e^{-\frac{r'\ell' d}{r}} \quad .$$

Below we solve for appropriate d such that this probability is less than $1/n^2$.

$$\left(\frac{\ell e}{\ell'}\right)^{\ell'} \cdot \left(\frac{r e}{r'}\right)^{r'} \cdot e^{-\frac{r'\ell' d}{r}} \leq 1/n^2 \tag{4.1}$$

$$\Longleftrightarrow \quad \ell' \ln\left(\frac{\ell e}{\ell'}\right) + r' \ln\left(\frac{r e}{r'}\right) - \frac{r'\ell' d}{r} \leq -2\ln n \tag{4.2}$$

$$\Longleftrightarrow \quad \frac{r}{r'\ell'}\left(\ell' \ln\left(\frac{\ell e}{\ell'}\right) + r' \ln\left(\frac{r e}{r'}\right) + 2\ln n\right) \leq d \quad . \tag{4.3}$$

We get step (4.2) from step (4.1) in the above by taking the logarithm of both sides. □

Lemma 4.6. *Let $\ell, r, \ell', r', d, \lambda$ and n be any positive values where $\ell' \leq \ell$, $r' \leq r$, $0 < \lambda < 1$ and*

$$d \geq \frac{2r}{r'\ell'(1-\lambda)^2}\left(\ell' \ln\left(\frac{\ell e}{\ell'}\right) + r' \ln\left(\frac{r e}{r'}\right) + 2\ln n\right) \quad .$$

Let G be a random bipartite multigraph with left side L and right side R where $|L| = \ell$ and $|R| = r$ and each node in L has edges to d random neighbors in R. Then with probability at least $1 - 1/n^2$, for any set $L' \subset L$ where $|L'| = \ell'$, there is no set $R' \subset R$, where $|R'| = r'$ such that all nodes in R' share less than $\lambda\ell' d/r$ edges with L'.

Proof. We will use the probabilistic method to show this. We will first fix a set $L' \subset L$ of size ℓ' and a set $R' \subset R$ of size r' and compute the probability that all nodes in R' share less than $\lambda\ell' d/r$ edges with L'. If this bad event occurs then the total number of edges shared between L' and R' must be less than $\lambda r'\ell' d/r$. Let X be a random variable giving the number of edges shared between L' and R'. The probability that a single edge from L' falls in R' is r'/r so by linearity of expectation, $E(X) = r'\ell' d/r$.

We can then say that:

$$\Pr\left(X \leq \frac{\lambda r'\ell' d}{r}\right) = \Pr(X \leq (1-\delta)\,E(X)) \leq e^{-E(X)\delta^2/2} \quad ,$$

where $\delta = 1 - \lambda$ and the last equation follows by Chernoff bounds if $0 < \lambda < 1$.

The number of ways to choose a set L' of the appropriate size is no more than $(\ell e/\ell')^{\ell'}$ and the number of ways to choose a set R' of the appropriate size is no more than $(r e/r')^{r'}$. So the probability that no two subsets L' of size ℓ' and R' of size r' have this bad event occur is

$$\left(\frac{\ell e}{\ell'}\right)^{\ell'} \cdot \left(\frac{r e}{r'}\right)^{r'} \cdot e^{-\frac{r'\ell' d\delta^2}{2r}} \quad .$$

Below we solve for appropriate d such that this probability is less than $1/n^2$.

$$\begin{aligned} \left(\frac{\ell \mathrm{e}}{\ell'}\right)^{\ell'} \cdot \left(\frac{r\mathrm{e}}{r'}\right)^{r'} \cdot \mathrm{e}^{-\frac{r'\ell' d\delta^2}{2r}} &\leq 1/n^2 & (4.4)\\ \iff \quad \ell' \ln\left(\frac{\ell \mathrm{e}}{\ell'}\right) + r' \ln\left(\frac{r\mathrm{e}}{r'}\right) - \frac{r'\ell' d\delta^2}{2r} &\leq -2\ln n & (4.5)\\ \iff \quad \frac{2r}{r'\ell'(1-\lambda)^2}\left(\ell' \ln\left(\frac{\ell \mathrm{e}}{\ell'}\right) + r' \ln\left(\frac{r\mathrm{e}}{r'}\right) + 2\ln n\right) &\leq d \ . & (4.6) \end{aligned}$$

We get step (4.5) from step (4.4) in the above by taking the logarithm of both sides. □

Lemma 4.7. *Let $\ell, \ell', r, r', d, \beta'$, and n be any positive values where $\ell' \leq \ell$, $\beta' > 1$ and*

$$d \geq \frac{4r}{r'\ell(\beta'-1)^2}\left(r' \ln\left(\frac{r\mathrm{e}}{r'}\right) + 2\ln n\right) \ .$$

Let G be a random bipartite multigraph with left side L and right side R where $|L| = \ell$ and $|R| = r$ and each node in L has edges to d random neighbors in R. Then with probability at least $1 - 1/n^2$, there is no set $R' \subset R$, where $|R'| = r'$, such that all nodes in R' have degree greater than $\beta'\ell d/r$.

Proof. We will again use the probabilistic method to show this. We will first fix a set $R' \subset R$ of size r' and compute the probability that all nodes in R' have degree greater than $\beta'\ell d/r$. If this bad event occurs then the total number of edges shared between L and R' must be at least $\beta' r'\ell d/r$. Let X be a random variable giving the number of edges shared between L and R'. The probability that a single edge from L falls in R' is r'/r so by linearity of expectation, $\mathrm{E}(X) = r'\ell d/r$.

We can then say that:

$$\Pr\left(X \geq \frac{\beta' r'\ell d}{r}\right) = \Pr(X \geq (1+\delta)\,\mathrm{E}(X)) \leq \mathrm{e}^{-\mathrm{E}(X)\delta^2/4} \ ,$$

where $\delta = \beta' - 1$ and the last equation follows by Chernoff bounds if $1 < \beta' < 2e - 1$.

The number of ways to choose a set R' of the appropriate size is no more than $(r\mathrm{e}/r')^{r'}$. So the probability that no subset R' of size r' has this bad event occur is

$$\left(\frac{r\mathrm{e}}{r'}\right)^{r'} \cdot \mathrm{e}^{-\frac{r'\ell' d\delta^2}{4r}} \ .$$

Below we solve for appropriate d such that this probability is less than $1/n^2$.

$$\begin{aligned} \left(\frac{r\mathrm{e}}{r'}\right)^{r'} \cdot \mathrm{e}^{-\frac{r'\ell d\delta^2}{4r}} &\leq 1/n^2 & (4.7)\\ \iff \quad r' \ln\left(\frac{r\mathrm{e}}{r'}\right) - \frac{r'\ell d\delta^2}{4r} &\leq -2\ln n & (4.8)\\ \iff \quad \frac{4r}{r'\ell(\beta'-1)^2}\left(r' \ln\left(\frac{r\mathrm{e}}{r'}\right) + 2\ln n\right) &\leq d \ . & (4.9) \end{aligned}$$

We get step (4.8) from step (4.7) in the above by taking the logarithm of both sides. □

4.4 (α,β)-good supernodes

Lemma 4.8. *Let α,δ',n be values where $\alpha < 1/2$ and $\delta' > 0$ and let $k(\delta',\alpha)$ be a value that depends only on α,δ' and assume n is sufficiently large. Let each node participate in $k(\delta',\alpha)\ln n$ random middle supernodes. Then removing any set of $n/2$ nodes still leaves all but $\delta' n/\ln n$ middle supernodes with at least $\alpha k(\delta',\alpha)\ln n$ live nodes.*

Proof. For simplicity, we will assume there are n middle supernodes (we can throw out any excess supernodes).

Let $\ell = n$, $\ell' = n/2$, $r = n$, $r' = \delta' n/\ln n$, $\lambda = 2\alpha$ and $d = k(\delta',\alpha)\ln n$ in Lemma 4.6. We want probability less than $1/n^2$ of being able to remove $n/2$ nodes and having a set of $\delta' n/\ln n$ supernodes all with less than $\alpha k(\delta',\alpha)\ln n$ live nodes. This happens provided that the number of connections from each supernode is bounded as in Lemma 4.6:

$$k(\delta',\alpha)\ln n \geq \frac{4\ln n}{\delta' n(1-2\alpha)^2}\left(\frac{n\ln(2e)}{2} + \frac{\delta' n}{\ln n}\cdot\ln\left(\frac{\ln n}{\delta'}\right) + 2\ln n\right) \tag{4.10}$$

$$= \frac{2\ln(2e)\cdot\ln n}{\delta'(1-2\alpha)^2} + o(1) \tag{4.11}$$

$$\iff \quad k(\delta',\alpha) \geq \frac{2\ln(2e)}{\delta'(1-2\alpha)^2} + o(1)\ . \tag{4.12}$$

□

Lemma 4.9. *Let β,δ',n,k be values such that $\beta > 1$, $\delta' > 0$ and assume n is sufficiently large. Let each node participate in $k\ln n$ of the middle supernodes, chosen uniformly at random. Then all but $\delta' n/\ln n$ middle supernodes have less than $\beta k\ln n$ participating nodes with probability at least $1-1/n^2$.*

Proof. For simplicity, we will assume there are n middle supernodes (we can throw out any excess supernodes and the lemma will still hold). Let $\ell = n$, $r = n$, $r' = \delta' n/\ln n$, $d = k\ln n$ and $\beta' = \beta$ in Lemma 4.7. Then the statement in this lemma holds provided that:

$$k\ln n \geq \frac{4\ln n}{\delta' n(\beta-1)^2}\left(\frac{\delta' n}{\ln n}\cdot\ln\left(\frac{\ln n}{\delta'}\right) + 2\ln n\right) \tag{4.13}$$

$$\iff \quad k \geq \frac{4}{(\beta-1)^2\ln n}\cdot\ln\left(\frac{\ln n}{\delta'} + \frac{2}{\delta' n}\right)\ . \tag{4.14}$$

The right hand side of this equation goes to 0 as n goes to infinity. □

Lemma 4.10. *Let α,δ',n be values such that $\alpha < 1/2$, $\delta' > 0$ and let $k(\delta',\alpha)$ be a value that depends only on δ' and α and assume n is sufficiently large. Let each node participate in $k(\delta',\alpha)$ top (bottom) supernodes. Then removing any set of $n/2$ nodes still leaves all but $\delta' n/\ln n$ top (bottom) supernodes with at least $\alpha k(\delta',\alpha)\ln n$ live nodes.*

Proof. Let $\ell = n$, $\ell' = n/2$, $r = n/\ln n$, $r' = \delta' n/\ln n$, $\lambda = 2\alpha$ and $d = k(\delta',\alpha)$ in Lemma 4.6. We want probability less than $1/n^2$ of being able to remove $n/2$ nodes and having a set of $\delta' n/\ln n$ supernodes all with less than $\alpha k(\delta',\alpha)\ln n$ live nodes. We get this provided that the number of connections from each supernode is bounded as in Lemma 4.6:

$$\begin{aligned} k(\delta',\alpha) &\geq \frac{4}{\delta' n(1-2\alpha)^2}\left(\frac{n\ln(2\,\mathrm{e})}{2}+\frac{\delta' n}{\ln n}\cdot\ln(1/\delta')+2\ln n\right) && (4.15)\\ &= \frac{2\ln(2\,\mathrm{e})}{\delta'(1-2\alpha)^2}+o(1)\ . && (4.16)\end{aligned}$$

□

Lemma 4.11. *Let β,δ',n,k be values such that $\beta>1$, $\delta'>0$ and n is sufficiently large. Let each node participate in k of the top (bottom) supernodes (chosen uniformly at random). Then all but $\delta' n/\ln n$ top (bottom) supernodes consist of less than $\beta k\ln n$ nodes with probability at least $1-1/n^2$.*

Proof. Let $\ell=n$, $r=n/\ln n$, $r'=\delta' n/\ln n$, $d=k$ and $\beta'=\beta$ in Lemma 4.7. Then the statement in this lemma holds provided that:

$$\begin{aligned} k &\geq \frac{4}{\delta' n(\beta-1)^2}\left(\frac{\delta' n}{\ln n}\cdot\ln\left(\frac{e}{\delta'}\right)+2\ln n\right) && (4.17)\\ &= \frac{4}{\ln n(\beta-1)^2}\cdot\left(\ln\left(\frac{e}{\delta'}\right)+\frac{2\ln n}{\delta' n}\right)\ . && (4.18)\end{aligned}$$

The right hand side of this equation goes to 0 as n goes to infinity. □

Corollary 4.12. *Let β,δ',n,k be values such that $\beta>1$, $\delta'>0$ and n is sufficiently large. Let each data item be stored in k of the bottom supernodes (chosen uniformly at random). Then all but $\delta' n/\ln n$ bottom supernodes have less than $\beta k\ln n$ data items stored on them with probability at least $1-1/n^2$.*

Proof. Let the data items be the left side of a bipartite graph and the bottom supernodes be the right side. The proof is then the same as Lemma 4.11. □

Corollary 4.13. *Let $\delta'>0$, $\alpha<1/2$, $\beta>1$. Let $k(\delta',\alpha)$, be a value depending only on δ' and assume n is sufficiently large. Let each node appear in $k(\delta',\alpha)$ top supernodes, $k(\delta',\alpha)$ bottom supernodes and $k(\delta',\alpha)\ln n$ middle supernodes. Then all but $\delta' n$ of the supernodes are $(\alpha k(\delta',\alpha),\beta k(\delta',\alpha))$-good with probability $1-O(1/n^2)$.*

Proof. Use

$$k(\delta',\alpha)=\frac{10}{3}\cdot\frac{2\ln(2\,\mathrm{e})}{\delta'(1-2\alpha)^2}$$

in Lemma 4.10. Then we know that no more than $3\delta' n/(10\ln n)$ top supernodes and no more than $3\delta' n/(10\ln n)$ bottom supernodes have less than $\alpha k(\delta',\alpha)\ln n$ live nodes. Next plugging $k(\delta',\alpha)$ into Lemma 4.8 gives that no more than $3\delta' n/(10\ln n)$ middle supernodes have less than $\alpha k(\delta',\alpha)\ln n$ live nodes.

Next using $k(\delta',\alpha)$ in Lemma 4.11 and Lemma 4.9 gives that no more than $\delta' n/(20\ln n)$ of the supernodes can have more than $\beta k(\delta',\alpha)\ln n$ nodes in them. Finally, using $k(\delta',\alpha)$ in Lemma 4.12 gives that no more than $\delta' n/(20\ln n)$ of the bottom supernodes can have more than $\beta k(\delta',\alpha)\ln n$ data items stored at them. If we put these results together, we get that no more than $\delta n/\ln n$ supernodes are not $(\alpha k(\delta',\alpha),\beta k(\delta',\alpha))$-good with probability $1-O(1/n^2)$. □

4.5 (γ,α,β)-expansive supernodes

Theorem 4.14. *Let $\delta > 0$, $\alpha < 1/2$, $0 < \gamma < 1$, $\beta > 1$. Let $k(\delta,\alpha,\gamma)$ be a value depending only on δ,α,γ and assume n is sufficiently large. Let each node participate in $k(\delta,\alpha,\gamma)$ top supernodes, $k(\delta,\alpha,\gamma)$ bottom supernodes and $k(\delta,\alpha,\gamma)\ln n$ middle supernodes. Then all but $\delta n/\ln n$ top supernodes are $(\gamma,\alpha k(\delta,\alpha),\beta k(\delta,\alpha))$-expansive with probability $1-O(1/n^2)$.*

Proof. Assume that for some particular $k(\delta,\alpha,\gamma)$ that more than $\delta n/\ln n$ top supernodes are not $(\gamma,\alpha k(\delta,\alpha,\gamma),\beta k(\delta,\alpha,\gamma)$-expansive. Then each of these bad top supernodes has $(1-\gamma n)/\ln n$ paths that are not $(\alpha k(\delta,\alpha,\gamma),\beta k(\delta,\alpha,\gamma))$-good. So the total number of paths that are not $(\alpha k(\delta,\alpha,\gamma),\beta k(\delta,\alpha,\gamma))$-good is more than

$$\frac{\delta(1-\gamma)n^2}{\ln^2 n} \ .$$

We will show there is a $k(\delta,\alpha,\gamma)$ such that this event will not occur with high probability. Let $\delta' = \delta(1-\gamma)$ and let

$$k(\delta,\alpha,\gamma) = \frac{10}{3}\cdot\frac{2\ln(2\mathrm{e})}{\delta(1-\gamma)(1-2\alpha)^2} \ .$$

Then we know by Lemma 4.13 that with high probability, there are no more than $\delta(1-\gamma)n/\ln n$ supernodes that are not $(\alpha k(\delta,\alpha,\gamma),\beta k(\delta,\alpha,\gamma))$-good. We also know that each of these supernodes which are not good cause at most $n/\ln n$ paths in the butterfly to be not $(\alpha k(\delta,\alpha,\gamma),\beta k(\delta,\alpha,\gamma))$-good. Hence the number of paths that are not $(\alpha k(\delta,\alpha,\gamma),\beta k(\delta,\alpha,\gamma))$-good is no more than $\delta(1-\gamma)n^2/(\ln^2 n)$ which is what we wanted to show. □

4.6 (α,β)-good paths to data items

We will use the following lemma to show that almost all the nodes are connected to some appropriately expansive top supernode.

Lemma 4.15. *Let $\delta > 0$, $\varepsilon > 0$ and n be sufficiently large. Then exists a constant $k(\delta,\varepsilon)$ depending only on ε and δ such that if each node connects to $k(\delta,\varepsilon)$ random top supernodes then with high probability, any subset of the top supernodes of size $(1-\delta)n/\ln n$ can be reached by at least $(1-\varepsilon)n$ nodes.*

Proof. We imagine the n nodes as the left side of a bipartite graph and the $n/\ln n$ top supernodes as the right side and an edge between a node and a top supernode in this graph if and only if the node and supernode are connected.

For the statement in the lemma to be false, there must be some set of εn nodes on the left side of the graph and some set of $(1-\delta)n/\ln n$ top supernodes on the right side of the graph that share no edge. We can find $k(\delta,\varepsilon)$ large enough that this event occurs with probability no more than $1/n^2$ by plugging in $\ell = n$, $\ell' = \varepsilon n$, $r = n/\ln n$ and $r' = (1-\delta)(n/\ln n)$ into Lemma 4.5. The bound found is:

$$k(\delta,\varepsilon) \geq \frac{1}{(1-\delta)\varepsilon n}\left(\varepsilon n\cdot\ln\left(\frac{\mathrm{e}}{\varepsilon}\right)+\frac{(1-\delta)n}{\ln n}\cdot\ln\left(\frac{\mathrm{e}}{(1-\delta)}\right)+2\ln n\right) \tag{4.19}$$

$$= \frac{\ln\left(\frac{\mathrm{e}}{\varepsilon}\right)}{1-\delta}+o(1) \ . \tag{4.20}$$

$\square$

We will use the following lemma to show that if we can reach γ bottom supernodes that have some live nodes in them that we can reach most of the data items.

Lemma 4.16. *Let γ, n, ε be any positive values such that $\varepsilon > 0$, $\gamma > 0$. There exists a $k(\varepsilon,\gamma)$ which depends only on ε, γ such that if each bottom supernode holds $k(\varepsilon,\gamma)\ln n$ random data items, then any subset of bottom supernodes of size $\gamma n/\ln n$ holds $(1-\varepsilon)n$ unique data items.*

Proof. We visualize the n data items as the left side of a bipartite graph and the $n/\ln n$ bottom supernodes as the right side of this graph. There is an edge between a data item and a bottom supernode if and only if the bottom supernode contains that data item. The bad event is that there is some set of $\gamma n/\ln n$ supernodes on the right that share no edge with some set of εn data items on the right. We will find $k(\varepsilon,\gamma)$ large enough that this event occurs with probability no more than $1/n^2$. We do this by plugging in $\ell = n$, $\ell' = \varepsilon n$, $r = n/\ln n$, and $r' = \gamma n/\ln n$ into Lemma 4.5.

We get:

$$k(\varepsilon,\gamma)\ln n \geq \frac{\ln n}{\varepsilon\gamma n}\left(\frac{\gamma n}{\ln n}\cdot\ln\frac{e}{\gamma} + \varepsilon n\cdot\ln\frac{e}{\varepsilon} + 2\ln n\right) \tag{4.21}$$

$$\iff \quad k(\varepsilon,\gamma) \geq \frac{1}{\gamma}\cdot\ln\frac{e}{\varepsilon} + o(1) \ . \tag{4.22}$$

$\square$

4.7 Connections between (α,β)-good supernodes

Lemma 4.17. *Let $\alpha, \beta, \alpha', n$ be any positive values where $\alpha' < \alpha$, $\alpha > 0$ and let C be the number of supernodes to which each node connects. Let X and Y be two supernodes that are both $(\alpha C, \beta C)$-good. Let each node in X have edges to $k(\alpha,\beta,\alpha')$ random nodes in Y where $k(\alpha,\beta,\alpha')$ is a value depending only on α, β and α'. Then with probability at least $1-1/n^2$, any set of $\alpha' C\ln n$ nodes in X has at least $\alpha' C\ln n$ live neighbors in Y.*

Proof. Consider the event where there is some set of $\alpha' C\ln n$ nodes in X which do not have $\alpha' C\ln n$ live neighbors in Y. There are $\alpha C\ln n$ live nodes in Y so for this event to happen, there must be some set of $(\alpha-\alpha')C\ln n$ live nodes in Y that share no edge with some set of $\alpha' d\ln n$ nodes in X. We note that the probability that there are two such sets which share no edge is largest when X and Y have the most possible nodes. Hence we will find a $k(\alpha,\beta,\alpha')$ large enough to make this bad event occur with probability less than $1/n^2$ if in Lemma 4.5 we set $\ell = \beta C\ln n$, $r = \beta C\ln n$, $\ell' = \alpha' C\ln n$, and $r' = (\alpha-\alpha')C\ln n$. When we do this, we get that $k(\alpha,\beta,\alpha')$ must be greater than or equal to:

$$\left(\frac{\beta}{\alpha'(\alpha-\alpha')}\right)\cdot\left(\alpha'\ln\left(\frac{\beta e}{\alpha'}\right) + (\alpha-\alpha')\ln\left(\frac{\beta e}{\alpha-\alpha'}\right) + \frac{2}{C}\right) \ .$$

$\square$

4.8 Putting it all together

We are now ready to give the proof of Theorem 3.1.

Proof. Let $\delta, \alpha, \gamma, \alpha', \beta$ be any values such that $0 < \delta < 1$, $0 < \alpha < 1/2$, $0 < \alpha' < \alpha$, $\beta > 1$ and $0 < \gamma < 1$. Let

$$C = \frac{10}{3} \cdot \frac{2\ln(2\,\mathrm{e})}{\delta(1-\gamma)(1-2\alpha)^2}, \qquad T = \frac{\ln\left(\frac{\mathrm{e}}{\varepsilon}\right)}{1-\delta}, \qquad B = \frac{1}{\gamma}\ln\left(\frac{\mathrm{e}}{\varepsilon}\right)$$

and

$$D = \left(\frac{\beta}{\alpha'(\alpha-\alpha')}\right)\left(\alpha'\ln\left(\frac{\beta\,\mathrm{e}}{\alpha'}\right) + (\alpha-\alpha')\ln\left(\frac{\beta\,\mathrm{e}}{\alpha-\alpha'}\right) + \frac{2}{C}\right) \quad .$$

Let each node connect to C top, C bottom and C middle supernodes. Then by Theorem 4.14, at least $(1-\delta)n/\ln n$ top supernodes are $(\gamma, \alpha C, \beta C)$-expansive. Let each node connect to T top supernodes. Then by Lemma 4.15, at least $(1-\varepsilon)n$ nodes can connect to some $(\gamma, \alpha C, \beta C)$-expansive top supernode. Let each data item map to B bottom supernodes. Then by Lemma 4.16, at least $(1-\varepsilon)n$ nodes have $(\alpha C, \beta C)$-good paths to at least $(1-\varepsilon)n$ data items.

Finally, let each node in a middle supernode have D random connections to nodes in neighboring supernodes in the butterfly network. Then by Lemma 4.17, at least $(1-\varepsilon)n$ nodes can broadcast to enough bottom supernodes so that they can reach at least $(1-\varepsilon)n$ data items.

Each node requires T links to connect to the top supernodes; $2D$ links for each of the C top supernodes it plays a role in; $2D$ links for each of the $C\ln n$ middle supernodes it plays a role in and $B\beta\ln n$ storage for each of the C bottom supernodes it plays a role in. The total amount of memory required is thus

$$T + 2DC + C\ln n(2D + B\beta) \quad ,$$

which is less than $k_1(\varepsilon)\log n$ for some $k_1(\varepsilon)$ dependent only on ε.

Our search algorithm will find paths to at most B bottom supernodes for a given data item and each of these paths has less than $\log n$ hops in it so the search time is no more than

$$k_2(\varepsilon)\log n = B\log n \quad .$$

Each supernode contains no more that $\beta\ln n$ nodes and in each search, exactly T top supernodes send no more than B messages so the total number of messages transmitted during a search is no more than

$$k_3(\varepsilon)\log^2 n = (TB\beta C)\log^2 n \quad .$$

□

5 Modifications for spam resistant distributed hash table

We now describe the changes in the network necessary for the spam resistant distributed hash table. We only sketch the required proofs since the arguments are based on slight modifications to the proofs of Section 4.

The first modification is that rather than have a constant degree expander between two supernodes connected in the butterfly, we will have a full bipartite graph between the nodes of these two supernodes. Since we have insisted that the total number of adversary controlled nodes be strictly less than $n/2$, we can guarantee that a $1-\varepsilon$ fraction of the paths in the butterfly have all supernodes with a majority of good (non-adversary controlled) nodes. In particular, by substituting appropriate values in Lemma 4.6 and Lemma 4.7 we can guarantee that all but $\varepsilon n/\log n$ of the supernodes have a majority of good nodes. This then implies that no more than an ε fraction of the paths pass through such "adversary-majority" supernodes. As before, this implies that most of the nodes can access most of the content through paths that don't contain any "adversary-majority" supernodes.

For a search in the new network, the paths in the butterfly network along which the search request and data item will be sent are chosen exactly as in the original construction. However, we modify the protocol so that in the downward flow, every node passes down a request only if the majority of requests it received from nodes above it are the same. This means that if there are no "adversary-majority" supernodes on the path, then all good nodes will take a majority value from a set in which good nodes are a majority. Thus, along such a path, only the correct request will be passed downwards by good nodes. After the bottommost supernodes are reached, the data content flows back along the same links as the search went down. Along this return flow, every node passes up a data value only if a majority of the values it received from the nodes below it are the same. This again ensures that along any path where there are no "adversary-majority" supernodes, only the correct data value will be passed upwards by good nodes. At the top, the node that issued the search takes the majority value amongst the $(O(\log n))$ values it receives as the final search result.

To summarize, the main theorem for spam resistant distributed hash tables is as follows:

Theorem 5.1. *For any constant $c < 1/2$ such that the adversary controls no more than cn nodes, and for all $\varepsilon > 0$, there exist constants $k_1(\varepsilon)$, $k_2(\varepsilon)$, $k_3(\varepsilon)$ which depend only on ε such that:*

- *Every node requires $k_1(\varepsilon)\log^2 n$ memory.*
- *Search for a data item takes no more than $k_2(\varepsilon)\log n$ time. (This is under the assumption that network latency overwhelms processing time for one message, otherwise the time is $O(\log^2 n)$.)*
- *Search for a data item requires no more than $k_3(\varepsilon)\log^3 n$ messages.*
- *All but εn nodes can search successfully for all but εn of the true data items.*

6 Discussion and open problems

We conclude with some open issues:

1. Can one improve on the construction for the spam resistant distributed hash table described in this paper?
2. Can one deal efficiently with more general Byzantine faults that occur all at once? For example, the adversary could use nodes under his control to flood the network with irrelevant searches, this is not dealt with by either of our solutions.

3. We conjecture that our network has the property that it is poly-log competitive with any fixed degree network. I. e., we conjecture that given any fixed degree network topology, where *n* items are distributed amongst *n* nodes, and any set of access requests that can be dealt with fixed sized buffers, then our network will also deal with the same set of requests by introducing no more than a polylog slowdown.

7 Acknowledgments

We gratefully thank Anna Karlin, Prabhakar Raghavan, Stefan Saroiu, and Steven Gribble for their great help with this paper.

References

[1] * NOGA ALON, HAIM KAPLAN, MICHAEL KRIVELEVICH, DAHLIA MALKHI, AND JULIEN STERN: Scalable secure storage when half the system is faulty. In *Proc. 27th Internat. Colloquium on Automata, Languages and Programming (ICALP'00)*, pp. 576–587. Springer, 2000. [ICALP:3l87mp6xvnxr0cfg]. 2.4

[2] * NOGA ALON AND JOEL SPENCER: *The Probabilistic Method, 2nd Edition*. John Wiley & Sons, 2000. 4.1

[3] * YONATAN AUMANN AND MICHAEL BENDER: Fault tolerant data structures. In *Proc. 37th FOCS*, pp. 580–589. IEEE Computer Society, 1996. [FOCS:10.1109/SFCS.1996.548517]. 2.4

[4] * BARUCH AWERBUCH AND CHRISTIAN SCHEIDELER: Group spreading: A protocol for provably secure distributed name service. In *Proc. 31st Internat. Colloquium on Automata, Languages, and Programming (ICALP'04)*, pp. 183–195. Springer, 2004. [ICALP:782vxmb2mlxxrmru]. 2.1

[5] * BARUCH AWERBUCH AND CHRISTIAN SCHEIDELER: Robust distributed name service. In *Proc. 3rd Internat. Workshop on Peer-to-Peer Systems (IPTPS'04)*, pp. 237–249. Springer, 2004. [Springer:crpp90cx7r3p61t0]. 2.1

[6] * M. BELLARE AND P. ROGAWAY: Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1st ACM Conf. on Computer and Communications Security*, pp. 62–73. ACM Press, 1993. [ACM:168588.168596]. 1

[7] * JOHN BORLAND: Gnutella girds against spam attacks. *CNET News.com*, August 2000. http://news.cnet.com/news/0-1005-200-2489605.html. 1.2

[8] * MAYUR DATAR: Butterflies and peer-to-peer networks. In *Proc. 10th European Symp. on Algorithms (ESA'02)*, pp. 310–322. Springer, 2002. [ESA:w83mmlkyt13lx90f]. 2.1

[9] * Electronic Freedom Foundation — Censorship — Internet censorship legislation & regulation (CDA, etc.) — Archive. http://www.eff.org/pub/Censorship/Internet_censorship_bills. 1

[10] * AMOS FIAT AND JARED SAIA: Censorship resistant peer-to-peer content addressable networks. In *Proc. 13th ACM-SIAM Symp. on Discrete Algorithms (SODA'02)*, pp. 94–103. ACM Press, 2002. [SODA:545381.545392]. 2.1

[11] * AMOS FIAT, JARED SAIA, AND MAXWELL YOUNG: Making chord robust to byzantine attack. In *Proc. 13th European Symposium on Algorithms (ESA'05)*, pp. 803–814. Springer, 2005. [ESA:422llxn7khwej72n]. 2.1

[12] * D.K. GIFFORD: Weighted voting for replicated data. In *Proc. 7th ACM Symp. on Operating Systems Principles*, pp. 150–159. ACM Press, 1979. [ACM:800215.806583]. 2.4

[13] * Gnutella: To the bandwidth barrier and beyond. http://dss.clip2.com/gnutella.html. 1.2

[14] * Gnutella Website. http://www.gnutella.com. 1

[15] * J. HÅSTAD AND F. THOMSON LEIGHTON: Fast computation using faulty hypercubes. In *Proc. 21st STOC*, pp. 251–263. ACM Press, 1989. [STOC:73007.73031]. 2.3.1, 2.3.2

[16] * KRISTEN HILDRUM AND JOHN KUBIATOWICZ: Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks. In *Proc. 17th Internat. Symposium on Distributed Computing (DISC'03)*, pp. 321–336. Springer, 2003. [Springer:7emt7u01cvbb6bu6]. 2.1

[17] * Index On Censorship Homepage. http://www.indexoncensorship.org. 1

[18] * ANNA R. KARLIN, GREG NELSON, AND HISAO TAMAKI: On the fault tolerance of the butterfly. In *Proc. 26th STOC*, pp. 125–133. ACM Press, 1994. [STOC:195058.195117]. 2.3.1

[19] * M. KASHOEK AND D. KARGER: Koorde: A simple degree-optimal distributed hash table. In *Proc. 2nd Internat. Workshop on Peer-to-Peer Systems (IPTPS'03)*, pp. 98–107. Springer, 2003. [Springer:unmqcqy0yxpu32xp]. 2.2

[20] * F. THOMSON LEIGHTON, BRUCE MAGGS, AND RAMESH SITAMARAN: On the fault tolerance of some popular bounded-degree networks. *SIAM Journal on Computing*, 27(5):1303–1333, 1998. [SICOMP:10.1137/S0097539793255163]. 2.3.1, 2.3.2

[21] * MICHAEL G. LUBY, MICHAEL MITZENMACHER, M. AMIN SHOKROLLAHI, DANIEL A. SPIELMAN, AND VOLKER STEMANN: Practical loss-resilient codes. In *Proc. 29th STOC*, pp. 150–159. ACM Press, 1997. [STOC:258533.258573]. 5

[22] * DAHLIA MALKHI, MICHAEL REITER, AND AVISHAI WOOL: The load and availability of byzantine quorum systems. *SIAM Journal on Computing*, 29(6):1889–1906, 2000. [SICOMP:10.1137/S0097539797325235]. 2.4

[23] * DAHLIA MALKHI, MICHAEL REITER, AVISHAI WOOL, AND REBECCA N. WRIGHT: Probabilistic byzantine quorum systems. In *Proc. 17th Ann. ACM Symp. on Principles of Distributed Computing (PODC'98)*, p. 321. ACM Press, 1998. [ACM:277697.277781]. 2.4

[24] * G. MANKU, M. BAWA, AND P. RAGHAVAN: Symphony: Distributed hashing in a small world. In *Proc. 4th USENIX Symp. on Internet Technologies and Systems (USITS'03)*, pp. 127–140, 2003. 1, 2.2

[25] * P. MAYMOUNKOV AND D. MAZIERES: Kademlia: A peer-to-peer information system based on the XOR metric. In *Proc. 1st Internat. Workshop on Peer-to-Peer Systems (IPTPS'02)*, pp. 53–65. Springer, 2002. [Springer:2ekx2a76ptwd24qt]. 1, 2.2

[26] * RAJEEV MOTWANI AND PRABHAKAR RAGHAVAN: *Randomized Algorithms*. Cambridge University Press, 1995. 4.3

[27] * MONI NAOR AND UDI WIEDER: A simple fault tolerant distributed hash table. In *Proc. 2nd Internat. Workshop on Peer-to-Peer Systems (IPTPS'03)*, pp. 88–97. Springer, 2003. [Springer:4e756fgyq4ff4kay]. 2.1

[28] * Napster Website. http://www.napster.com. 1

[29] * ANDY ORAM, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, July 2001. 1

[30] * M. PINSKER: On the complexity of a concentrator. In *Proc. 7th Internat. Teletraffic Conference*, pp. 318/1–318/4, 1973. 4.3

[31] * C.G. PLAXTON, R. RAJARAMAN, AND A.W. RICHA: Accessing nearby copies of replicated objects in a distributed environment. In *Proc. 9th Ann. ACM Symp. on Parallel Algorithms and Architectures (SPAA'97)*, pp. 311–320. ACM Press, 1997. [SPAA:258492.258523]. 1

[32] * SYLVIA RATNASAMY, PAUL FRANCIS, MARK HANDLEY, RICHARD KARP, AND SCOTT SHENKER: A scalable content-addressable network. In *Proc. ACM SIGCOMM 2001 Technical Conference*, pp. 161–172. ACM Press, 2001. [ACM:964723.383072]. 1, 2.2

[33] * ANTONY I. T. ROWSTRON AND PETER DRUSCHEL: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proc. of the IFIP/ACM Internat. Conf. on Distributed Systems Platforms*, pp. 329–350. Springer, 2001. [Springer:7y5mjjep0hqlctv6]. 2.1, 2.2

[34] * STEFAN SAROIU, P. KRISHNA GUMMADI, AND STEVEN D. GRIBBLE: A measurement study of peer-to-peer file sharing systems. In *Proc. 9th Ann. Symp. on Multimedia Computing and Networking (MMNC'02)*. SPIE Press, 2002. 1

[35] * STEFAN SAROIU, P. KRISHNA GUMMADI, AND STEVEN D. GRIBBLE: Measuring and analyzing the characteristics of napster and gnutella hosts. *Multimedia Systems*, 9(2):170–184, 2003. 1, 2.2

[36] * CHRISTIAN SCHEIDELER: How to spread adversarial nodes? Rotate! In *Proc. 37th STOC*, pp. 704–713. ACM Press, 2005. [STOC:1060590.1060694]. 2.1

[37] * ADI SHAMIR: How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979. [ACM:359168.359176]. 2.4

[38] * ION STOICA, ROBERT MORRIS, DAVID LIBEN-NOWELL, DAVID R. KARGER, M. FRANS KAASHOEK, FRANK DABEK, AND HARI BALAKRISHNAN: Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, 2003. [doi:10.1109/TNET.2002.808407]. 1, 2.2

[39] * MARC WALDMAN, AVIEL D. RUBIN, AND LORRIE FAITH CRANOR: Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proc. 9th USENIX Security Symposium*, pp. 59–72, August 2000. 2.4

[40] * ERPING ZHANG: Googling the great firewall: Google kowtowed to communist censorship. *The New York Sun*, 31 January 2006. http://www.nysun.com/article/26791. 1

[41] * B.Y. ZHAO, K.D. KUBIATOWICZ, AND A.D. JOSEPH: Tapestry: An infrastructure for fault-resilient wide-area location and routing. Technical Report UCB//CSD-01-1141, University of California at Berkeley Technical Report, April 2001. 1, 2.1, 2.2

AUTHORS

Amos Fiat [About the author]
Professor
University of Tel Aviv, Tel Aviv, Israel
fiat@tau.ac.il
http://www.math.tau.ac.il/~fiat

Jared Saia [About the author]
Assistant Professor
University of New Mexico, Albuquerque, New Mexico
saia@cs.unm.edu
http://www.cs.unm.edu/~saia

ABOUT THE AUTHORS

AMOS FIAT graduated from the Weizmann Institute in 1986. His advisor was Adi Shamir. His reseach interests include online algorithms, cryptography, data mining, web search, and peer-to-peer networks. He also enjoys photography and sailing.

JARED SAIA graduated from the University of Washington in 2002. His advisor was Anna Karlin. His research interests include randomized algorithms, distributed algorithms, and graph theory. He also enjoys hiking, skiing, and mountain biking.