Building Low-Diameter Peer-to-Peer Networks

Gopal Pandurangan, *Member, IEEE*, Prabhakar Raghavan, *Fellow, IEEE*, and Eli Upfal, *Fellow, IEEE*

***Abstract*—Peer-to-Peer (P2P) computing has emerged as a significant paradigm for providing distributed services, in particular search and data sharing. Current P2P networks (e.g., Gnutella) are constructed by participants following their own uncoordinated (and often whimsical) protocols; they consequently suffer from frequent network overload and partitioning into disconnected pieces separated by choke points with inadequate bandwidth.**

In this paper, we propose a protocol for participants to build P2P networks in a distributed fashion, and prove that it results in connected networks of constant degree and logarithmic diameter. These properties are crucial for efficient search and data exchange. An important feature of our protocol is that it operates without global knowledge of all the nodes in the network.

***Index Terms*—Distributed protocol, peer-to-peer (P2P) networks, stochastic analysis, topology construction.**

I. Introduction

PEER-TO-PEER (P2P) networks are emerging as a significant vehicle for providing distributed services (e.g., search, content integration, and administration) both on the Internet [5]–[7], [9] and in enterprises. The idea is simple: Rather than have a centralized service (say, for search), each node in a distributed network maintains its own index and search service. Queries no longer go to a central server; instead they fan out over the network, and results are collected and propagated back to the originating node. This allows for search results that are fresh (in the extreme, admitting dynamic content assembled from a transaction database, reflecting—say in a marketplace—real-time pricing and inventory information). Such freshness is not possible with traditional static indices, where the indexed content is as old as the last crawl (in many enterprises, this can be several weeks). The downside, of course, is dramatically increased network traffic. In some implementations [6], this problem can be mitigated by adaptive distributed caching for replicating content; it seems inevitable that such caching will become more widespread.

How should the topology of P2P networks be constructed? Unlike static networks, P2P systems are very dynamic with a high peer turnover rate. For example, the study in [17] shows that in both Gnutella [8] and Napster [12], about half of the peers participating in the system are replaced within one hour. Thus maintaining even a basic property such as network connectivity becomes a nontrivial task.

Each node participating in a P2P network runs so-called *servent* software (for *ser*ver + cli*ent*, since every node is both a server and a client). This software embeds local heuristics by which the node decides, on joining the network, which neighbors to connect to. Note, that an incoming node (or for that matter, any node in the network) does not have global knowledge of the current topology, or even the identities [Internet protocol (IP) addresses] of other nodes in the current network. Thus, one cannot require an incoming node to connect (say) to "four random network nodes" (in the hope of creating an expander-like network [11]). What local heuristics will lead to the formation of networks that perform well? Indeed, what properties should the network have in order for performance to be good? In the Gnutella world [9], there is little consensus on this topic, as the variety of servent implementations (each with its own peculiar connection heuristics) grows—along with little understanding of the evolution of the network. Indeed, some services on the Internet [4] attempt to bring order to this chaotic evolution of P2P networks, but without necessarily using rigorous approaches (or tangible success).

A number of attempts are under way to create P2P networks within enterprises (e.g., Verity is creating a P2P enterprise infrastructure for search). The principal advantage here is that servents can be implemented to a standard so that their local behavior results in good global properties for the P2P network they create. In this paper, we begin with some desiderata for such good global properties, principally the diameter of the resulting network (the motivation for this becomes clear below). Our main contribution is a stochastic analysis of a simple local heuristic which, if followed by every servent, results in provably strong guarantees on network diameter and other properties. Our heuristic is intuitive and practical enough that it could be used in enterprise P2P products.

A. Case Study: Gnutella

To better understand the setting, modeling and objectives for the stochastic analysis to follow, we now give an overview of the Gnutella network. This is a public P2P network on the Internet, by which anyone can share, search for, and retrieve files and content. A participant first downloads one of the available (free) implementations of the search servent. The participant may choose to make some documents (say, all his IEEE papers) available for public sharing, indexes the contents of these documents, and runs a search server on the index. His servent joins the network by connecting to a small number (typically 3–5) of neighbors currently connected to the network. When

Manuscript received August 16, 2002; revised February 20, 2003. The work of G. Pandurangan was done while he was at Brown University. The work of G. Pandurangan and E. Upfal was supported in part by the Air Force and the Defense Advanced Research Projects Agency of the Department of Defense under Grant F30602-00-2-0599 and in part by the National Science Foundation (NSF) under Grant CCR-9731477 and Grant CCR-0121154. This paper was presented at the 42nd Annual IEEE Symposium on the Foundations of Computer Science (FOCS), Las Vegas, NV, 2001.

G. Pandurangan is with the Department of Computer Science, Purdue University, West Lafayette, IN 47907-2066 USA (e-mail: gopal@cs.purdue.edu).

P. Raghavan is with Verity Inc., Sunnyvale, CA 94089 USA (e-mail: pragh@verity.com).

E. Upfal is with the Department of Computer Science, Brown University, Providence, RI 02912-1910 USA (e-mail: eli@cs.brown.edu).

Digital Object Identifier 10.1109/JSAC.2003.814666

any servent s wishes to search the network with some query q, it sends q to its neighbors. These neighbors return any of their own documents that match the query; they also propagate q to *their* neighbors, and so on. To control network traffic this fanning out typically continues to some fixed radius (in Gnutella, typically seven); matching results are fanned back into s along the paths on which q flowed outwards. Thus, every node can initiate, propagate and serve query results; clearly, it is important that the content being searched for be within the search radius of s. A servent typically stays connected for some time, then drops out of the network—many participating machines are personal computers on dialup connections. The importance of maintaining connectivity and small network diameter has been demonstrated in a recent performance study of the public Gnutella network [4].

Note, that the above discussion lacks any mention of which three to five neighbors a servent joining the network should connect to, and indeed, this is the current free-for-all situation in which each servent implementation uses its own heuristic. Most begin by connecting to a generic set of neighbors that come with the download, then switch (in subsequent sessions) to a subset of the nodes whose names the servent encountered on a previous session (in the course of remaining connected and propagating queries, a servent gets to "watch" the names of other hosts that may be connected and initiating or servicing queries). Note also that there is no standard on what a node should do if its neighbors drop out of the network (many nodes join through dialup connections, and typically dial out after a few minutes—so the set of participants keeps changing). This free-for-all situation leads to partitioning of the network into disconnected pieces as documented in [4].

B. Main Contributions and Organization of the Paper

Our main contribution is a new protocol by which newly arriving servents decide which network nodes to connect to, and existing servents decide when and how to replace lost connections. We show that our protocol results in a constant degree network that is likely to stay connected and have small diameter. A nice feature of our protocol is that it operates without any global knowledge (such as the topology of the network or even the identities of all other nodes) and can be implemented by a simple distributed local message passing scheme. Also, our protocol is easily scalable both in terms of degree (which remains bounded irrespective of size) and diameter (grows slowly as a function of network size).

Our protocol for building a P2P network is described in Section II. Section III presents a stochastic analysis of our protocol. Our protocol involves one somewhat nonintuitive notion, by which nodes maintain "preferred connections" to other nodes; in Section IV, we show that this feature is essential. Our analysis assumes a stochastic setting in which nodes arrive and leave the network according to a probabilistic model. Our goal is to show that even as the network changes with these arrivals/departures, it remains connected with small diameter. Our main result is that at *any* time (after a short initial period), with large probability, the network is *connected* and its diameter is *logarithmic* in the size of the network at that time. Furthermore, our analysis proves that the protocol has strong fault tolerance properties: if the network gets partitioned into disconnected pieces it rapidly recovers its connectivity. The technical core of our analysis is an analysis of an evolving graph as nodes arrive and leave, with edges being dictated by the protocol; the analysis of evolving graphs is relatively new, with virtually no prior analysis in which both nodes and edges (connections) arrive and leave the network.

We mention related work in Section V and discuss open issues in Section VI.

II. P2P Protocol

The central element of our protocol is a *host server*[1] which, at all times, maintains a *cache*[2] of K nodes, where K is a constant. The host server is reachable by all nodes at all times; however, it need not know of the topology of the network at any time, or even the identities of all nodes currently on the network. We only require that 1) when the host server is contacted on its IP address it responds and 2) any node on the P2P network can send messages to its neighbors. In this sense, our protocol demands far less from the network than do (for instance) current P2P proposals (e.g., the *reflectors* of dss.clip2.com, which maintain knowledge of the global topology).

When a node is in the cache, we refer to it as a *cache node*. A node is *new* when it joins the network, otherwise, it is *old*. Our protocol will ensure that the degree (number of neighbors) of all nodes will be in the interval $[D, C+1]$ for two constants D and C.

A new node first contacts the host server, which gives it D random nodes from the current cache to connect to. The new node connects to these and becomes a *d-node*; it remains a d-node until it, subsequently, either enters the cache or leaves the network. The degree of a d-node is always D. At some point the protocol may put a d-node into the cache. It stays in the cache until it acquires a total of C connections, at which point, it leaves the cache as a *c-node*. (Thus, the set of cache nodes keeps changing with time.) A c-node might lose connections after it leaves the cache, but its degree is always at least D. A c-node has always one *preferred* connection, made precise below. Our protocol is summarized below as a set of rules applicable to various situations that a node may find itself in.

Peer-to-Peer Protocol for Node v

1. **On joining the network:** Connect to D cache nodes, chosen uniformly at random from the current cache.
2. **Reconnect rule:** If a neighbor of v leaves the network, and that connection was not a preferred connection, connect

[1]The host server is similar to (or models) websites that maintain list of host IP addresses which clients visit to get entry points into the P2P network; for example, *http://www.gnufrog.com/* is a website which maintains a list of active Gnutella servents. New clients can join the network by connecting to one or more of these servents. Another point to note is that we have assumed a single host server for clarity of presentation. The protocol can be easily extended to work with multiple host servers.

[2]This is just a terminology used to denote the set of nodes which can accept connections—analogous to the list of active Gnutella clients mentioned in the previous footnote.

to a random node in cache with probability $D/d(v)$, where $d(v)$ is the degree of v before losing the neighbor.

3. **Cache Replacement rule:** When a cache node v reaches degree C while in the cache (or if v drops out of the network), it is replaced in the cache by a d-node from the network. Let $r_0(v) = v$, and let $r_k(v)$ be the node replaced by $r_{k-1}(v)$ in the cache. The replacement d-node is found by the following rule:

$k = 0$;
while (a d-node is not found) **do**
 search neighbors of $r_k(v)$ for a d-node;
 $k = k + 1$;
endwhile

4. **Preferred Node rule:** When v leaves the cache as a c-node it maintains a *preferred connection* to the d-node that replaced it in the cache. (If v is not already connected to that node this adds another connection to v).

5. **Preferred Reconnect rule:** If v is a c-node and its preferred connection is lost, then v reconnects to a random node in the cache and this becomes its new preferred connection.

We end this section with brief remarks on the protocol and its implementation.

1) It is clear from our protocol that it is essential for a node to know whether it is in the cache or not; thus, each node maintains a flag for this purpose.
2) The cache replacement rule can be implemented in a distributed fashion by a local message passing scheme with constant storage per node. Each c-node v stores the address of the node that it replaced in the cache, i.e., $r(v)$. Node v sends a message to $r(v)$ when v itself does not have any d-node neighbors.
3) Note that the overhead in implementing each rule of the protocol is constant (or expected constant). This is very important in practice, because even if a protocol is local, it is desirable that neither too much (local) computation nor too many local messages be sent per node. Rules 1, 2, 4, and 5 can be easily implemented with constant overhead. It follows from our analysis that the overhead incurred in replacing a full cache node (rule 3) is constant on the average, and w.h.p. (with high probability) is at most logarithmic in the size of the network (see Section III-B).
4) We note that the host server is contacted whenever a node needs to reconnect (rules 2 and 5), and when a new node joins the network. We show that the expected number of contacts the host server receives per unit time interval is constant in our model and w.h.p. only *logarithmic* in the size of the network; this implies that the network also scales well in terms of the number of "hits" the host server receives.
5) We assume that a node knows when any of its neighbors leave the network. One way of realizing this in practice is (as in the Gnutella protocol [8]) that each node can periodically ping its neighbors to check whether any of them have gone offline.
6) In the stochastic analysis that follows, the protocol does have a minuscule probability of catastrophic failure: for instance, in the cache replacement step, there is a very small probability that no replacement d-node is found. A practical implementation of this step would either cause some nodes to exceed the maximum capacity of $C + 1$ connections, or to reject new connections. In either case, the system would rapidly "self-correct" itself out of this situation.

III. Analysis

In evaluating the performance of our protocol, we focus on the long term behavior of the system in a fully decentralized environment in which nodes arrive and depart in an uncoordinated and unpredictable fashion. This setting is best modeled by a stochastic, memoryless, continuous-time setting. The arrival of new nodes is modeled by Poisson distribution with rate λ, and the duration of time a node stays connected to the network is independently and exponentially distributed with parameter μ. We are inspired by models in queueing theory which have been used to model similar scenarios, e.g., the classical telephone trunking model [10]. Also, a recent measurement study of real P2P systems [17] (Gnutella and Napster) provides evidence that the above model approximates real-life data reasonably well.

Let G_t be the network at time t (G_0 has no vertices). We analyze the evolution in time of the stochastic process $\mathcal{G} = (G_t)_{t \geq 0}$.

Since the evolution of $\mathcal{G}$ depends only on the ratio λ/μ we can assume w.l.o.g. that $\lambda = 1$. To demonstrate the relation between these parameters and the network size, we use $N = \lambda/\mu$ throughout the analysis. We justify this notation in Section III-A by showing that the number of nodes in the network rapidly converges to N. Furthermore, if the ratio between arrival and departure rates is changed later to $N' = \lambda'/\mu'$, the network size will then rapidly converge to the new value N'. Next, we show that the protocol can with high probability[3] maintain a bounded number of neighbors for all nodes in the network, i.e., w.h.p. there is a d-node in the network to replace a cache node that reaches full capacity. In Section III-C, we analyze the connectivity of the network, and in Section III-D, we bound the network diameter.

A. Network Size

Let $G_t = (V_t, E_t)$ be the network at time t.

Theorem III.1:

1) For any $t = \Omega(N)$, w.h.p. $|V_t| = \Theta(N)$
2) If $t/N \to \infty$ then w.h.p. $|V_t| = N \pm o(N)$.

Proof: Consider a node that arrived at time $\tau \leq t$. The probability that the node is still in the network at time t is

[3]Throughout this paper w.h.p. (with high probability) denotes probability $1 - N^{-\Omega(1)}$.

$e^{-(t-\tau)/N}$. Let $p(t)$ be the probability that a random node that arrives during the interval $[0,t]$ is still in the network at time t, then (since in a Poisson process the arrival time of a random element is uniform in $[0,t]$)

$$p(t)=\frac{1}{t}\int_0^t e^{-(t-\tau)/N}d\tau=\frac{1}{t}N\left(1-e^{-t/N}\right).$$

Our process is similar to an infinite server Poisson queue. Thus, the number of nodes in the graph at time t has a Poisson distribution with expectation $tp(t)$ (see [15, pp. 18 and 19]).

For $t=\Omega(N)$, $E[|V_t|]=\Theta(N)$. When $t/N\to\infty$, $E[|V_t|]=N-o(N)$.

We can now use a tail bound for the Poisson distribution [1, p. 239] to show that for $t=\Omega(N)$

$$\Pr\left(||V_t|-E[|V_t|]|\le\sqrt{bN\log N}\right)\ge 1-\frac{1}{N^c}$$

for some constants $b>0$ and $c>1$. ■

The above theorem assumed that the ratio $N=\lambda/\mu$ was fixed during the interval $[0,t]$. We can derive similar result for the case in which the ratio changes to $N'=\lambda'/\mu'$ at time τ.

Theorem III.2: Suppose that the ratio between arrival and departure rates in the network changed at time τ from N to N'. Suppose that there were M nodes in the network at time τ, then if $\frac{(t-\tau)}{N'}\to\infty$ w.h.p. G_t has $N'\pm o(N')$ nodes.

Proof: The expected number of nodes in the network at time t is

$$Me^{-\frac{(t-\tau)}{N'}}+N'(1-e^{-\frac{t-\tau}{N'}})=N'+(M-N')e^{-\frac{(t-\tau)}{N'}}.$$

Applying the tail bound for the Poisson distribution, we prove that w.h.p. the number of nodes in G_t is $N'\pm o(N')$. ■

B. Available Node Capacity

To show that the network can maintain a bounded number of connections at each node, we will show that w.h.p., there is always a d-node in the network to replace a cache node that reaches capacity C and that the replacement node can be found efficiently. We first show that at any given time the network has w.h.p. a large number of d-nodes.

Lemma III.1: Let $C>3D+1$; then at any time $t\ge a\log N$ (for some fixed constant $a>0$), w.h.p. there are

$$\left(1-\frac{2D+1}{C-D}\right)\min[t,N](1-o(1))$$

d-nodes in the network.

Proof: Assume that $t\ge N$ (the proof for $t<N$ is similar). Consider the interval $[t-N,t]$; we bound the number of new d-nodes arriving during this interval and the number of nodes that become c-nodes.

The arrival of new nodes to the network is Poisson-distributed with rate 1; using the tail bound for the Poisson distribution, we show that w.h.p. the number of new d-nodes arriving during this interval is $N(1+o(1))$ and that the number of connections to cache nodes from the new arrivals is $DN(1+o(1))$.

By Theorem III.1, the expected size of the network at any time in the interval is bounded by $N(1+o(1))$. The expected number of connections to the cache nodes in unit time in this interval is bounded by

$$\sum_{v\in V}\left((1+o(1))\frac{d(v)}{N}\frac{D}{d(v)}+(1+o(1))\frac{1}{N}\right)$$

$$=(D+1)(1+o(1))$$

(The two terms within the sum bounds the number of reconnections due to non-preferred and preferred neighbors leaving a node.) Thus, the expected number of connections to the cache from old nodes in this interval is bounded by $N(D+1)(1+o(1))$. Let $u_1,\ldots,u_\ell$ be the set of nodes that left the network, in that interval, and let $X_{v,u_i}=1$ if v makes connection to the cache when u_i left the network, else $X_{v,u_i}=0$. Then

$$E\left[\sum_{i=1}^{\ell}\sum_{v}X_{v,u_i}\right]\le N(D+1)(1+o(1))$$

and each variable in the sum is independent of all but C other variables. By partitioning the sum into C sums such that in each sum all variables are independent and applying the Chernoff bound ([11, pp. 67-71]) to each sum individually, we show that w.h.p. the total number of connections to the cache from old nodes during this interval is bounded w.h.p. by $N(D+1)(1+o(1))$.

Thus, w.h.p. the total number of connections to cache is bounded by $(2D+1)N(1+o(1))$. Since a node receives $C-D$ connections while in the cache, w.h.p. no more than $\frac{2D+1}{C-D}N(1+o(1))$ d-nodes convert to new c-nodes in the interval; thus, w.h.p., we are left with $(1-\frac{2D+1}{C-D})N(1-o(1))$ d-nodes that joined the network in this interval.

Lemma III.2: Suppose that the cache is occupied at time t by node v. Let $Z(v)$ be the set of nodes that occupied the cache in v's slot during the interval $[t-c\log N,t]$. For any $\delta>0$ and sufficiently large constant c, w.h.p. $|Z(v)|$ is in the range $\frac{(2D+1)c}{(C-D)K}\log N(1\pm\delta)$. ■

Proof: As in the proof of Lemma III.1, the expected number of connections to a given cache node in an interval $[t-c\log N,t]$ is $\frac{(2D+1)c\log N}{K}(1+o(1))$. Applying the Chernoff bound, we show that w.h.p. the number of connections is in the range $\frac{(2D+1)c}{K}\log N(1\pm\delta)$. Since a cache node receives $C-D$ connections while in the cache the result follows. ■

The following lemma shows that most often the algorithm finds a replacement node for the cache by searching only a few, i.e., $O(\log N)$ nodes.

Lemma III.3: Assume that $C>3D+1$. At any time $t\ge c\log N$, with probability $1-O(\log^2 N/N)$ the algorithm finds a replacement d-node by examining only $O(\log N)$ nodes.

Proof: Let $v_1,\ldots,v_K$ be the K nodes in the cache at time t. By Lemma III.2, w.h.p. $|Z(v_i)|=d\log N$, for some constant d. With probability at least

$$e^{-\frac{(Kcd\log^2 N)}{N}}\ge 1-O\left(\frac{\log^2 N}{N}\right)$$

no node in $Z(v_i)$, $i=1,\ldots,K$ leaves the network in the interval $[t-c\log N,t]$.

Suppose that node v leaves the cache at time t, then the protocol tries to replace v by a d-node neighbor of a node in $Z(v)$. As in the proof of Lemma III.1, w.h.p., $Z(v)$ received at least $\frac{D}{K}c\log N$ connections from new d-nodes in the interval $[t-c\log N, t]$. Among these new d-nodes no more than $|Z(v)|$ nodes enter the cache and became c-nodes during this interval. Using the bound on $|Z(v)|$ from Lemma III.2, w.h.p., there is a d-node attached to a node of $Z(v)$ at time t. ∎

C. Connectivity

The proof that at any given time the network is connected w.h.p. is based on two properties of the protocol: 1) steps 3–4 of the protocol guarantee (deterministically) that at any given time a node is connected through "preferred connections" to a cache node and 2) the random choices of new connections guarantee that w.h.p. the $O(\log N)$ neighborhoods of any two cache nodes are connected to each other. In Section IV, we show that the first property is essential for connectivity. Without it, there is a constant probability that the graph has a number of small disconnected components.

Lemma III.4: At all times, each node in the network is connected to some cache node directly or through a path in the network.

Proof: It suffices to prove the claim for c-nodes since a d-node is always connected to some c-node. A c-node v is either in the cache, or it is connected through its preferred connection to a node that was in the cache after v left the cache. By induction, the path of preferred connections must lead to a node that is currently in the cache. ∎

Lemma III.5: Consider two cache nodes v and u at time $t \geq c\log N$, for some fixed constant $c > 0$. With probability $1 - O(\log^2 N/N)$, there is a path in the network at time t connecting v and u.

Proof: Let $Z(v)$ be the set of nodes that occupied the cache v's slot during the interval $[t-c\log N, t]$. By Lemma III.2, w.h.p. $|Z(v)| = d\log N$, for some constant d.

The probability that no node in $Z(v)$ leaves the network during the interval $[t-c\log N, t]$ is

$$e^{-(cd\log^2 N)/N} \geq 1 - O\left(\frac{\log^2 N}{N}\right).$$

Note that if no node in $Z(v)$ leaves the network during this interval then all nodes in $Z(v)$ are connected to v by their chain of preferred connections.

The probability that no new node that arrives during the interval $[t-c\log N, t]$ connects to both $c(v)$ and $c(u)$ is bounded by $(1-D^2/K^2)^{c\log N} = O(1/N^{c'})$. ∎

Since there are $K = O(1)$ cache locations, we have the following theorem.

Theorem III.3: There is a constant c such that at any given time $t > c\log N$

$$Pr(G_t \text{ is connected}) \geq 1 - O\left(\frac{\log^2 N}{N}\right).$$

The above theorem does not depend on the state of the network at time $t - c\log N$. It, therefore, shows that the network rapidly recovers from network disconnection.

Corollary III.1: There is a constant c such that if the network is disconnected at time t

$$Pr(\mathrm{G_{t+c\log N}} \text{ is connected}) \geq 1 - O\left(\frac{\log^2 N}{N}\right).$$

Theorem III.4: At any given time t such that $t/N \to \infty$, if the graph is not connected then it has a connected component of size $N(1-o(1))$.

Proof: By Lemma 3.4 all nodes in the network are connected to some cache node. The $O(\log^2 N/N)$ failure probability in Theorem III.3 is the probability that some cache node is left with fewer than $d\log N$ nodes connected to it. Excluding such cache nodes all other cache nodes are connected to each either with probability $1 - K^2(1-D^2/K^2)^{c\log N} = 1 - 1/N^c$, for some $c > 0$. ∎

D. Diameter

We state our main theorem which gives a bound on the diameter of the network.

Theorem III.5: For any t, such that $t/N \to \infty$, w.h.p., the largest connected component of G_t has diameter $O(\log N)$. In particular, if the network is connected (which has probability $1 - O(\log^2 N/N)$), then w.h.p., its diameter is $O(\log N)$.

Note that the above diameter bound is the best possible for a constant degree network.

Proof: Since a d-node is always connected to a c-node it is sufficient to discuss the distance between c-nodes. Thus, in the following discussion all nodes are c-nodes. For the purpose of the proof, we define a constant f and call a cache node good if during its time in cache, it receives a set of $r \geq f$ connections such that:

- the r connections are "reconnect " connections;
- the r connections are not preferred connections;
- the r connections resulted from τ different nodes leaving the network.

We color the edges of the graph using three colors: A, $B1$ and $B2$. All edges are colored A except a random f edges of the set of r "reconnect" edges that satisfied the three requirements of a good node. A random half of these f edges are colored $B1$, the rest are colored $B2$.

Since the proof of Theorem III.3 uses only preferred connection edges, and edges of new d-nodes, it is easy to verify that at any time t, the network is connected with probability $1 - O(\log^2 N/N)$ using only A edges, and that if the network is not connected, then w.h.p., the A edges define a connected component of size $N(1-o(1))$.

We rely on the "random" structure of the B edges to reduce the diameter of the network. However, we need to overcome two technical difficulties. First, although the B edges are "random," the occurrences of edges between pairs of nodes are not independent as in the standard $G_{n,p}$ random graph model ([3]). Second, the total number of B edges is relatively small; thus, the proof needs to use both the A and the B edges.

Lemma III.6: Assume that node v enters the cache at time t, where $t/N \to \infty$. Then, for a sufficiently large choice of the constant C, the probability that v leaves the cache as a good node is at least $\gamma > 1/2$. Further, the f recolored edges of a good cache node are distributed uniformly at random among the

nodes currently in the network. Furthermore, the probability that a c-node is good is independent of other c-nodes.

Proof: Consider the interval of time in which v was a cache node.

1) New nodes join the network according to a Poisson process with rate 1. Also, the expected number of connections to v from a new node is D/K.
2) Nodes also leave the network according to a Poisson process with rate 1. Also, the expected number of connections to v as a result of an old node leaving the network is
$$\sum_{u \in V} \frac{d(u)}{V} \frac{D}{d(u)} \frac{1}{K} = \frac{D}{K} < 1.$$
3) The expected number of connections from an old node u to v in unit time is $\frac{d(u)}{N} \frac{D}{d(u)} = D/N$.

From 1) and 2) above, it follows that each connection to v , while it is in the cache, has a constant probability each of being from a new or an old node. Also from 2), we have the expected number of connections to v as a result of one old node leaving the network is <1; thus, each connection has a constant probability of being triggered by a unique node leaving the network. Thus, for a sufficiently large C, the $C - D$ connections to v include, with probability $\gamma > 1/2$, $r \geq f$ reconnect edges from different nodes leaving the network.

Further, from 3) and using the fact that each node leaves the network independently and identically under the same exponential distribution it follows that each node in the network—irrespective of its degree—has an equal probability of being connected to v. Finally, it is easy to see the independence of the events for different c-nodes, since a cache node stays in the cache till it accepts C connections irrespective of other cache nodes. ∎

For the proof of the theorem, we need the following definitions. Given a node v in G_t, let $\Gamma_0(v)$ be an arbitrary cluster of $d \log N$ c-nodes, such that $v \in \Gamma_0(v)$, and this cluster has diameter $O(\log N)$ using only A edges. For $i \geq 1, i$ odd (with respect to even), let $\Gamma_i(v)$ be all the c-nodes in G_t that are connected to $\Gamma_{i-1}(v)$ and are not in $\cup_{j=0}^{i-1} \Gamma_j(v)$ using $B1$ (with respect to $B2$) edges.

We first show the following "expansion" lemma which states that each neighborhood of v starting from $\Gamma_1(v)$ is at least twice the size of the previous neighborhood.

Lemma III.7: If $|\Gamma_{i-1}(v)| = o(N)$
$$Pr\{|\Gamma_i(v)| \geq 2|\Gamma_{i-1}(v)|\} \geq 1 - \frac{1}{N^5}.$$

Proof: Let $W = \Gamma_{i-1}(v)$, $w = |W|$, and let $z \notin W \cup (\cup_{j=0}^{i-1} \Gamma_j(v))$. W.l.o.g., assume that $i - 1$ is even. Partition W into W_0, consisting of nodes in W that are older than z, and W_1, consisting of nodes in W that arrived after z. The probability that z is connected to W_0 using $B1$ edges is $(1/4)(f|W_0|/N)(1 - o(1))$ using Lemma III.6. Similarly, each node in W_1 has probability $(f/4N)(1 - o(1))$ of being connected to z by $B1$ edges. Thus, the probability that z is connected to W by $B1$ edges is at least $\frac{1}{4} \frac{fw}{N}(1 - o(1))$.

Let $Y = |\Gamma_i(v)|$ be the number of c-nodes outside W that are connected to W by $B1$ edges. $E[Y] = \frac{f}{4}(w(1 - o(1)))$. Let $w_1, w_2, \ldots,$ be an enumeration of the nodes in W, and let $N(w_i)$ be the set of neighbors of w_i outside W using $B1$ edges. Define an exposure martingale $Z_0, Z_1, \ldots,$ such that $Z_0 = E[Y], Z_i = E[Y \mid N(w_1), \ldots, N(w_i)], Z_w = Y$. Since the degree of all nodes is bounded by C, a node w_i can connect to no more than C nodes outside W. Thus, $|Z_i - Z_{i-1}| < C$.

Using Azuma's inequality [2], it follows that that for sufficiently large constant d
$$Pr\left\{|Y - E[Y]| \geq \frac{f}{8} \frac{\sqrt{w}}{C} C\sqrt{w}\right\} \leq 2e^{-(f^2/128C^2)w} \leq \frac{1}{N^5}.$$
∎

Now, we complete the proof of Theorem III.5. Our goal is to show that w.h.p. the distance between any two c-nodes is $O(\log N)$. Consider any two c-nodes v and u. By applying Lemma III.7 repeatedly $O(\log N)$ times, we have with probability $1 - O(\log N/N^5)$, for some k_v, $k_u = O(\log N)$, $|\Gamma_{k_v}(v)| \geq \sqrt{N} \log N$ and $|\Gamma_{k_u}(u)| \geq \sqrt{N} \log N$. The probability that $\Gamma_{k_v}(v)$ and $\Gamma_{k_u}(u)$ are disjoint and not connected by an edge is bounded by $(1 - f/2N)^{N \log^2 N}$; thus, with probability $1 - O(\log N/N^5)$, an arbitrary pair of nodes u and v are connected by a path of length $O(\log N)$ in G_t. Summing the failure probability over all $\binom{n}{2}$ pairs, it follows that w.h.p. any pair of nodes in G_t is connected by a path of length $O(\log N)$. ∎

IV. Why Preferred Connections?

In this section, we show that the preferred connection component in our protocol is essential: running the protocol without it leads to the formation of many small disconnected components. A similar argument would work for other fully decentralized protocols that maintain a minimum and maximum node degree and treat all edges equally, i.e., do not have preferred connections. Observe that a protocol cannot replace all the lost connections of nodes with degree higher than the minimum degree. Indeed, if all lost connections are replaced and new nodes add new connections, then the total number of connections in the network is monotonically increasing while the number of nodes is stable; thus, the network cannot maintain a maximum degree bound.

To analyze our protocol without preferred nodes, define a type H subgraph as a complete bipartite network between D d-nodes and D c-nodes, as shown in Fig. 1.

Lemma IV.1: At any time $t \geq c$, where c is a sufficiently large fixed constant, there is a constant probability (i.e., independent of N) that there exists a subgraph of type H in G_t.

Proof: A subgraph of type H arises when D incoming d-nodes choose the same set of D nodes in cache. A type H subgraph is present in the network at time t when all the following four events happen.

1) There is a set S of D nodes in the cache each having degree D (i.e., these are the new nodes in the cache and are yet to accept connections) at time $t - D$.
2) There are no deletions in the network during the interval $[t - D, t]$.

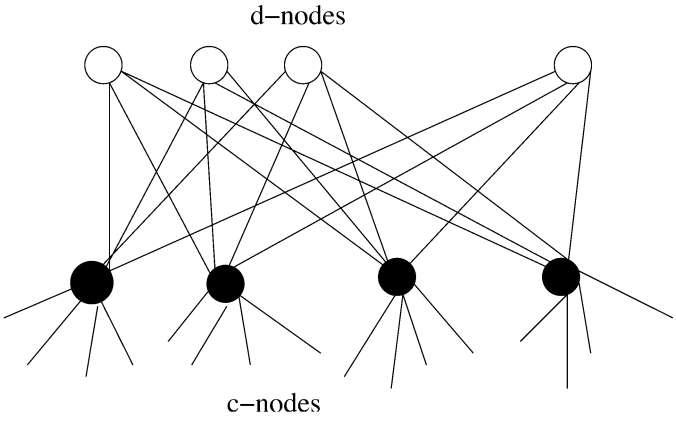


Fig. 1. Subgraph H used in proof of Lemma IV.2 . Note that $D = 4$ in this example. All the four d-nodes are connected to the same set of four c-nodes (shown in black).

3) A set T of D new nodes arrive in the network during the interval $[t - D, t]$.
4) All the incoming nodes of set T choose to connect to the D cache nodes in set S.

Since each of the above events can happen with constant probability, the lemma follows. ∎

Lemma IV.2: Consider the network G_t, for $t > N$. There is a constant probability that there exists a small (i.e., constant size) isolated component.

Proof: By Lemma IV.1 with constant probability there is a subgraph (call it F) of type H in the network at time $t - N$. We calculate the probability that the above subgraph F becomes an isolated component in G_t. This will happen if all $2D$ nodes in F survive till t and all the neighbors of the nodes in F (at most $C(C - D)$ of them connected to the D c-nodes) leave the network and there are no reconnections. The probability that the $2D$ subgraph nodes survived the interval $[t-N, t]$ is e^{-2D}. The probability that all neighbors of the subgraph leave the network with no new connections is at least $(1-e)^{-C(C-D)}(1-(D/D+1))^{C(C-D)}$. Thus, the probability that F becomes isolated is at least

$$e^{-2D}(1-e)^{-C(C-D)}\left(1-\frac{D}{D+1}\right)^{C(C-D)} = \Theta(1).$$

∎

Theorem IV.1: The expected number of small isolated components in the network at any time $t > N$ is $\Omega(N)$, when there are no preferred connections.

Proof: Let S be the set of nodes which arrived during the interval $[t - N, t - N/2]$. Let $v \in S$ be a node which arrived at at t'. From the proof of Lemma IV.2, it is easy to show that v has a constant probability of belonging to a subgraph of type H at t'. Also, by the same lemma, H has a constant probability of being isolated at t. Let the indicator variable X_v, $v \in S$ denote the probability that v belongs to a isolated subgraph at time t. Then, $E[\sum_{v \in S} X_v] \geq \Omega(N)$, by linearity of expectation. Since the isolated subgraph is of constant size, the theorem follows. ∎

V. Related Work

We briefly discuss related work in P2P systems most relevant to our work. Two important systems proposed recently are Chord [18] and CAN [13]. These are content-addressable protocols, i.e., they solve the problem of efficiently locating a node storing a given data item. There are two components for the above protocols: the first specifies how and where a particular data item should be stored in the network, and the second specifies a routing protocol to retrieve a given data item efficiently.

The focus of our work is building P2P networks with good topological properties and not the problem of searching or routing—which is an orthogonal issue for us; for example a Gnutella—like [8] or a Freenet-like [7] search/routing mechanism can be easily incorporated in our protocol. Thus, although we cannot directly compare our protocol with content-addressable networks such as Chord or CAN, we can compare them with respect to their topological properties and guarantees. CAN uses a d-dimensional Cartesian coordinate space (for some fixed d) to implement a distributed hash table that maps keys onto values. Chord, on the other hand, uses a scheme called *consistent hashing* to map keys to nodes. Although the degree (the number of entries in the routing table of a node) of CAN is a fixed constant d (the number of entries in its routing table), the diameter (the maximum distance between any two nodes in the virtual network) can be as large as $O(dn^{1/d})$. In the case of Chord, the diameter is $O(\log N)$, while the degree of every node is $O(\log N)$. If $(d = \log N)$, CAN matches the bounds of Chord). This is in comparison to the constant degree and logarithmic diameter of our protocol. However, the most important contrast is that their protocols provide no provable guarantees in a realistic dynamic setting, unlike ours. Chord gives guarantees only under a simplistic assumption that every node can fail (or drop out) with probability 1/2.

Another interesting P2P system is the dynamically fault-tolerant network of [16]. This is again a content-addressable network based on a butterfly topology. The diameter of the network is $O(\log N)$ and the degree is $O(\log^3 N)$. Peer insertion takes $O(\log N)$ time. The system is robust to fault tolerance in the sense that at any time, an arbitrarily large fraction of the peers can reach an arbitrarily large fraction of the data items. They show the above property under a somewhat artificial assumption that in any time interval during which an adversary deletes some number of peers, some larger number of peers join the network. Also, they assume that each of the new peers joining the network knows one *random* peer currently in the network. To compare with our work, we show that our protocol is naturally fault-tolerant (in the sense it recovers fairly rapidly from fragmentation and high diameter w.h.p.) under a natural dynamic model, where each node operates with no global knowledge.

VI. Conclusion and Further Work

We give a distributed protocol to construct networks with good topological properties—namely, constant degree, connectivity, and low diameter. An attractive feature of the protocol is that it is simple to implement. We analyze our protocol under a realistic dynamic setting and prove rigorously that it results in

the above properties with large probability. We also proved that our protocol is naturally robust to failures and that it has nice self-correcting properties such as rapid recovery from network fragmentation. We now discuss possible extensions and future work.

It is important to point out our protocol is concerned with building a good *virtual* network topology which may not match the underlying Internet topology (this may not be a big issue for enterprise P2P). In fact, evidence [14] suggests that these two topologies do not match well. It will be of practical interest [14] to construct topologies that respects the underlying physical topology (e.g., locality)—this is an area for further research.

In our protocol, we implicitly assume that all nodes have equal capabilities (i.e., storage and number of connections supported), and all links have equal bandwidth. In enterprises with homogeneous systems, this is closer to reality, however, this is not the case in the Internet. It will be nice to extend our protocol to incorporate heterogeneous nodes and links.

REFERENCES

[1] N. Alon and J. Spencer, *The Probabilistic Method*. New York: Wiley, 1992.

[2] K. Azuma, "Weighted sums of certain dependent random variables," *Tohoku Math. J.*, vol. 19, pp. 357–367, 1967.

[3] B. Bollobas, *Random Graphs*. New York: Academic, 1985.

[4] (2001) Gnutella Measurement Project. Clip2. [Online]. Available: http://www.clip2.com

[5] D. Clark, "Face-to-face with peer-to-peer networking," *Computer*, vol. 34, no. 1, 2001.

[6] I. Clarke, "A Distributed decentralized information storage and retrieval system," Div. Inform., Univ. Edinburgh, Edinburgh, U.K., 1999.

[7] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. presented at Proceedings of the Workshop on Design Issues in Anonymity and Unobservability. [Online]. Available: http://freenet.sourceforge.net

[8] The Gnutella Protocol Specification v0.4. [Online]. Available: http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

[9] Gnutella. [Online]. Available: http://gnutella.wego.com/

[10] S. Karlin and H. Taylor, *A First Course in Stochastic Processes*, 2nd ed. NewYork: Academic, 1997.

[11] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 1995.

[12] Napster. [Online]. Available: http://www.napster.com

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proc. ACM SIGCOMM*, 2001.

[14] M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the gnutella network: Properties of large scale peer-to-peer systems and implications for system design," *IEEE Internet Comput. J. Special Issue: Peer-to-Peer Networking*, vol. 6, no. 1, 2002.

[15] S.M. Ross, *Applied Probability Models With Optimization Applications*. San Francisco, CA: Holden-Day, 1970.

[16] J. Saia, A. Fiat, S. Gribble, A. Karlin, and S. Saroiu, "Dynamically fault-tolerant content addressable networks," in *Proc. 1st Int. Workshop Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, Mar. 2002.

[17] S. Saroiu, P.K. Gummadi, and S.D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proc. Multimedia Computing Networking (MMCN)*, San Jose, CA, 2002.

[18] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," in *Proc. ACM SIGCOMM*, 2001.

Gopal Pandurangan (M'97) received the B.Tech. degree in computer science from the Indian Institute of Technology, Madras, India, in 1994, the M.S. degree in computer science from the State University of New York, Albany, in 1997, and the Ph.D.degree in computer science from Brown University, Providence, RI, in 2002.

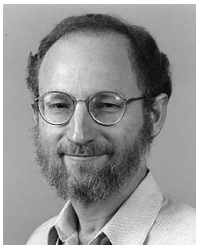
He is an Assistant Professor of computer science at Purdue University, West Lafayette, IN. His research interests are broadly in the design and analysis of algorithms. He is especially interested in stochastic analysis of computer phenomena and in probabilistic analysis of algorithms with applications to communication networks, Peer-to-Peer computing, Web algorithmics, and computational biology. He was a Visiting Scientist at the DIMACS Center at Rutgers University, Piscataway, NJ, as part of the 2000–2003 special focus on Next-Generation Networks Technologies and Applications.

Dr. Pandurangan is a Member of the Sigma Xi and ACM.

Prabhakar Raghavan (SM'94–F'00) received the Ph.D. degree from the University of California at Berkeley.

He is Vice President and Chief Technology Officer of Verity, Sunnyvale, CA, where he is responsible for the technical strategy of the corporation. Prior to joining Verity, he worked for 14 years in various technical and managerial positions at IBM T. J. Watson Center, NY and IBM Almaden Research Center, CA. He is Consulting Professor of computer science at Stanford University, Stanford, CA, and is Editor in Chief of the *Journal of the Association for Computing Machinery* (ACM).

Dr. Raghavan is a Fellow of the ACM.



Eli Upfal (SM'95–F'02) received the Ph.D. degree in computer science from The Hebrew University, Jerusalem, Israel, in 1983.

He was a Research Fellow at the University of California at Berkeley, from 1982 to 1983, and from 1983 to 1984, he was a Postdoctoral Fellow at Stanford University, Stanford, CA. He is a Professor and Chair of the Computer Science Department, Brown University, Providence, RI. From 1985 to 1997, he was a Research Staff Member at the IBM Almaden Research Center, San Jose, CA, and from 1988 to 1997, he was also a Faculty Associate Professor and later Professor in the Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot, Israel. Since January 1998, he has been a Professor of computer science at Brown University. His main research areas are randomized computation, probabilistic analysis of algorithms, theory of communication networks, parallel and distributed computing, and computational biology. His current work focuses on applications of probability theory and combinatorics to the design and analysis of computer algorithms and computer networks. Recently, he has been working on developing probabilistic techniques for studying the long-term behavior of dynamic computer processes such as communication, load balancing, cashing, and paging, and on combinatorial problems related to efficient DNA sequencing.