

A Simple Churn-Tolerant Structured Peer-to-Peer Scheme

Gopal Pandurangan^{a,*}, Suresh Jagannathan^a

^a *Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA.*

Abstract

We present a simple and general scheme to build a churn (fault)-tolerant structured Peer-to-Peer (P2P) network. Our scheme shows how to “convert” a static network into a dynamic distributed hash table(DHT)-based P2P network such that all the good properties of the static network are guaranteed with high probability. Applying our scheme to a cube-connected cycles network, for example, yields a $O(\log N)$ degree network, in which *every* search succeeds in $O(\log N)$ hops w.h.p., using $O(\log N)$ messages, where N is the expected stable network size. Our scheme has an $O(\log N)$ storage overhead (the number of nodes responsible for servicing a data item) and an $O(\log N)$ overhead (messages and time) per insertion and no overhead for deletions. All these bounds are essentially optimal.

Keywords: P2P Network; DHT Scheme; Churn; Fault-tolerance; Stochastic Analysis

1 Introduction

Peer-to-Peer (P2P) networks are highly dynamic: peers enter and leave the network and connections may be added or deleted at any time and thus the topology changes very dynamically. The independent arrival and departure by a large number of peers creates a collective effort that is called as *churn*. Measurement studies of real-world P2P networks [19, 20, 22] show that the churn rate is quite high: nearly 50% of peers in real-world networks can be

* Corresponding author.

Email addresses: gopal@cs.purdue.edu (Gopal Pandurangan),
suresh@cs.purdue.edu (Suresh Jagannathan).

URLs: <http://www.cs.purdue.edu/people/gopal> (Gopal Pandurangan),
<http://www.cs.purdue.edu/people/suresh> (Suresh Jagannathan).

replaced within an hour. However, despite a large churn rate, these studies show that the total number of peers in the network is relatively *stable*. The study by Stutzbach and Rejaie [22] also indicate that P2P networks exhibit a high degree of variance in terms of the session time (the amount of time spent by a node in the network in a session). They show that the distribution of session times appear to follow a Weibull or lognormal distribution.

P2P systems must have efficient and reliable routing in the presence of a dynamically changing network. The P2P overlay must exhibit good topological properties (e.g., connectivity, low diameter, low degree, etc.) even if the composition of the underlying physical network exhibits significant change. Because the system dynamics of these networks are also highly asymmetric with only a small number of peers persistent over significant time periods, providing churn-tolerance in the presence of mostly short-lived peers is essential [22].

A Distributed Hash Table (DHT) scheme (e.g., [15, 21, 11]) creates a fully decentralized index that maps data items to peers and allows a peer to search for a data item very efficiently (typically logarithmically in the size of the network) without flooding. Such systems have been called as *structured* P2P networks, unlike Gnutella, for example, which are *unstructured*. In unstructured networks, there is no relation between the file identifier and the peer where it resides. Structured networks are more difficult to implement than an unstructured networks, mainly due to the fact that it is not easy to maintain a DHT under a highly dynamic setting. In addition, to this, since data is stored typically in some arbitrary node, fault-tolerance to node deletion is essential. These are some of the reasons why structured P2P networks, despite their efficient search mechanism, have been somewhat less successful than unstructured networks when it comes to practical deployment. Hence, it is of both theoretical and practical interest to develop simple and efficient DHT schemes that work well under high churn rates.

In this paper, we present a simple and general scheme to build a churn-tolerant structured P2P network. The basic idea behind our scheme is simple. It is easy to design a static topology with desirable properties such as connectivity, low degree, low diameter, and an efficient (and local) routing algorithm. Indeed such topologies, e.g., hypercube, butterfly, cube connected cycles, de Bruijn graphs, etc., have been studied extensively in parallel and distributed computing literature. Our scheme shows how to “convert” a static graph into a fault-tolerant DHT scheme network such that all the good properties of the static graph are guaranteed with high probability. For example, by applying our scheme to a cube-connected cycles (CCC) graph yields a $O(\log N)$ degree P2P network that has a $O(\log N)$ latency (i.e., search time), using $O(\log N)$ messages, with $O(\log N)$ storing overhead (the number of nodes storing a data item). Here N is the expected stable network size (cf. Section 3.1). Our

bounds are essentially optimal since in our model (cf. Section 3.1) if we want all nodes to access all data items with high probability, then it is necessary that the degree be at least $\Omega(\log N)$ and that every data item be stored by at least $\Omega(\log N)$ nodes; otherwise there will be a non-negligible probability that there would be nodes disconnected from the system. In a dynamic network, there is the additional challenge of quantifying the work done by the algorithm to *maintain* the desired properties. An important advantage of the above protocol is that it takes $O(\log N)$ overhead (messages and time) per insertion and *no* overhead for deletions. This is optimal since, Liben-Nowell et al. [10] show that $\Omega(\log N)$ work is required to maintain (even) connectivity in this stochastic model.

Our scheme is an improvement in the degree size and message complexity over the network of Saia et al. [18]. The structured P2P network described by Saia *et al.* [18] has a $O(\log N)$ latency for search, using $O(\log^2 N)$ messages, and $O(\log^3 N)$ degree. Their fault-tolerant overlay is a butterfly-based expander topology. Their work guarantees that a large number of data items are available even if a large fraction of *arbitrary* peers are deleted (hence their scheme can tolerate even adversarial deletions), under the assumption that, in each time step, the number of peers deleted by an adversary must be smaller than the number of peers joining. In contrast, our scheme constructs a $O(\log N)$ latency and $O(\log N)$ degree P2P network that guarantees that *every* search succeeds with high probability (whp)¹ at any time, rather than just a large fraction, under a natural and general stochastic model — the $M/G/\infty$ model [16]. In a $M/G/\infty$ model the holding (session) times of nodes can have an *arbitrary* distribution, while arrival of nodes is assumed to be Poisson. (Real-world P2P network measurement studies [19, 22] have shown that this is a reasonable statistical model.) The construction of our overlay is also much simpler compared to [4, 18]. Our scheme also improves significantly on the Warp scheme [7]. Warp guarantees $O(\log N)$ search time, but has a degree of $O(\log^3 N)$. Our scheme has low maintenance overhead. In particular, node deletions does not incur any overhead. Multiple nodes can join and leave at the same time (in particular, up to a constant fraction of the total) without any need to change the protocol, and hence our protocol can operate in a highly dynamic setting.

Other Related Work. There has been other works on building fault-tolerant DHTs under different deletion models — adversarial deletions and stochastic deletions. Fiat and Saia [4] proposed a DHT network that is robust against adversarial deletions (i.e., an adversary can choose which nodes to fail). In this model some small fraction of the non-failed nodes would be denied from accessing some of the data items. While this solution is more general than our model it has some disadvantages: (1) It is not clear whether the system can

¹ Throughout, “whp” means “with probability at least $1 - N^{-\Omega(1)}$ ”.

guarantee its bounds when nodes leave and join dynamically; (2) the message complexity is large — $O(\log^3 N)$ and so is the network degree. Moreover their construction is very complicated which can increase the likelihood of error in implementation and decrease the possibility of practical use. In a subsequent paper Saia et al. [18] address the first problem and give a scheme with $O(\log N)$ time for search, using $O(\log^2 N)$ messages, and $O(\log^3 N)$ degree. Datar [3] gives a scheme based on the multibutterfly network that improves on the scheme of Fiat and Saia [4] under the adversarial deletion model. Naor and Weider [13] describe a simple DHT scheme that is robust under the following simple random deletion model — each node can fail independently with probability p . They show that their scheme can guarantee logarithmic degree, search time, and message complexity if p is sufficiently small. In contrast, our scheme is simpler than [13] and works under a more realistic stochastic deletion model (even a large constant fraction of nodes can get deleted in our model) and guarantees the same (essentially optimal) performance bounds. Also our scheme requires no maintenance overhead under deletions unlike the scheme of [13]. Hildrum and Kubiawicz [6] describe how to modify two popular DHTs, Pastry [17] and Tapestry [23] to tolerate random deletions. Finally, we point out that several DHT schemes (e.g., [21, 15, 8]) have been shown to be robust under the simple random deletion model mentioned above.

2 The Scheme

We will show how to build a DHT-based P2P network $G = (V_G, E_G)$ of expected stable size $N = |V_G|$ (defined precisely in Section 3.1). Let $H = (V_H, E_H)$ be the static (“template”) graph that will be used to build G . We will use the term *node* to denote a node (peer) of G and the term *vertex* to denote a vertex of H .

Although, in principle, any graph can be taken as a template graph, for the purposes of constructing efficient P2P networks it is desirable that H has certain properties such as connectivity, regularity, recursive structure, constant degree, logarithmic diameter, and a simple and efficient (local) routing scheme. Good candidates for H are hypercube network and its variants (butterfly, Beneš network and cube connected cycles), de Bruijn graph etc. Henceforth, the following assumptions will be made with respect to H :

- (1) H has diameter D and maximum degree Δ .
- (2) H has a local and efficient routing scheme \mathcal{R} that can route between any two nodes in $O(D)$ steps using $O(D)$ messages, where D is the diameter of H . Specifically it will be required that H has a vertex labeling scheme that enables shortest path routing with low memory overhead (see e.g., [5] for a survey on such routing schemes). In such a routing scheme, vertex

labels are assigned in such a way that every source vertex v , and given the destination address u , one can decide locally (based solely on the address of u) the output port of the outgoing edge of v that leads to u by using only a routing table of size at most $O(\Delta)$ entries per node. (Each entry of the routing table will specify which outgoing edge to take for a given destination u .) A well-known example of such a scheme is the *bit-fixing routing scheme* in a hypercube (and its variants) [9].

Given the above assumptions, our scheme builds a DHT-based P2P network (with expected stable size N) with the following properties:

- The degree of a node and its routing table size is bounded by $O(\Delta \log N)$ w.h.p. (cf. Theorem 2)
- At any time, the network is connected and has a diameter of $O(D)$ w.h.p. (cf. Theorem 2)
- *Every* search will succeed in $O(D)$ steps w.h.p and will use $O(D)$ messages. (cf. Theorem 3)
- The time and message overheads for a node to join the network is $O(D)$ and $O(D + \Delta \log N)$ w.h.p. (cf. Theorem 4)
- Number of nodes responsible for servicing a data item is $O(\log N)$ w.h.p.

Throughout, we will illustrate by taking H to be a *cube connected cycle (CCC)* network. Our scheme can be adapted to other similar types of graphs. The r -dimensional CCC is constructed from the r -dimensional hypercube by replacing each vertex of the hypercube with a cycle of r vertices in the CCC. The i th dimension edge incident to a vertex of the hypercube is then connected to the i th vertex of the corresponding cycle of the CCC [9]. In a CCC, the label of a vertex is represented by a pair $\langle w, i \rangle$ where i is the position of the node within its cycle and w is the label of the vertex in the hypercube that corresponds to the cycle. Two vertices $\langle w, i \rangle$ and $\langle w', i' \rangle$ are linked by an edge in the CCC if and only if either (1) $w = w'$ and $i - i' \equiv \pm 1 \pmod{r}$ or (2) $i = i'$ and w differs from w' in precisely the i th bit. Edges of the first type are called cycle edges, while edges of the second type are referred to as hypercube edges. A CCC graph of N nodes has diameter $O(\log N)$ and each node has degree 3. A CCC has an efficient routing scheme, namely the bit-fixing routing scheme [9] that can route in $O(\log N)$ steps using $O(\log N)$ messages using only routing tables of size $O(1)$. In this scheme, to route a message between two vertices with vertex labels $\langle u, i \rangle$ and $\langle v, j \rangle$, the bits of u are successively transformed (say, from the first to the last) to match v . The message is routed between one dimension to the next using the hypercube edges, while the cycle edges are used to bring the message to the vertex of the cycle with the appropriate dimension.

A node x in G has a label called the *node-id* which corresponds to a vertex label of H . We will choose the size of H to be $S = |V_H| = N/\log N$. For

example, if H is a CCC network, then $S = |V_H| = (N/\log^2 N) \log(N/\log^2 N)$. (Throughout we will assume logarithm to the base 2. We will omit floors and ceilings, assuming that quantity in question is rounded to the nearest integer.) Node-ids of vertices of H are assigned randomly by sampling from all possible vertex labels of H . Specifically, if H is a CCC, the node-id of a vertex is obtained as follows: toss a fair coin (has a equal probability of getting a 0 or 1) $r = \log(N/\log^2 N)$ times independently and obtain a r -bit random bit string σ_r (r is the dimension of the CCC). Also sample a random number from 1 to r and call it i . Then the node-id of the vertex is $\langle \sigma_r, i \rangle$. We say that the node *covers* the vertex having the label corresponding to its node-id. There is an edge between two nodes with node-ids x and y if there is an edge in H between x and y or if $x = y$. We call a vertex in H to be *occupied* if there is a node in the network (i.e., a live peer) which covers this vertex; otherwise we call it to be a *hole*.

Joining and Leaving the Network. A node (say v) that wants to join the network chooses its node-id as explained above. We assume that N (the expected stable network size) or an estimate of N (a constant factor estimate is sufficient) is known to all joining nodes. Because of the numbering scheme, v can locally determine the node-ids of its (potential) neighbors without any global knowledge. v 's neighbors in the P2P network are the nodes that cover the above determined node-ids. To join the network, v contacts any one of the nodes in the network (such entry points are provided by an external mechanism). v can then make use an efficient routing scheme \mathcal{R} of H to find its neighbors (i.e., their IP addresses) and joins by connecting to them. If H is a CCC, then \mathcal{R} can be the standard *bit-fixing routing scheme* mentioned earlier.

For the vertex in its node-id, data is copied from any other node which shares this vertex. Thus this node will also be responsible for servicing requests for this data item.

A node can simply leave the network at any time; the node's data need not be transferred.

Data Placement and Search Scheme. Data placement and search are handled by a DHT scheme, similar to other DHT schemes such as Chord [2]. The data (or its key) is hashed to a random vertex label in the same fashion as was done for choosing the node-id of a vertex. Data is inserted to *all nodes* having this label as its node-id. Search for this data is thus directed to a node having its node-id equal to the data's hashed value. Since all nodes sharing a node-id are connected to each other (forming a clique), search will succeed even if only one node covering this vertex is live in the network. Search is performed by invoking the routing scheme \mathcal{R} on H . If H is a CCC, then the bit-fixing routing scheme is used as illustrated below by an example. Suppose a node with node-id x wants to search for a data item hashed to a number t (

$1 \leq t \leq S$). Let the route (in H) given by the bit-fixing routing scheme from x to t be $\langle x, u_1, u_2, \dots, t \rangle$. Then x will send a message to a neighbor node which covers u_1 which in turn will forward to its neighbor node which covers u_2 and so on. We will show that the search will succeed in $O(D)$ steps w.h.p (cf. Theorem 3).

3 Analysis

We analyze various network parameters — degree, connectivity and diameter, maintenance overhead for joins, and the complexity for doing search. We first describe the stochastic model used in our analysis.

3.1 Model

In evaluating the performance of our protocol we focus on the long term behavior of the system in which nodes arrive and depart in an uncoordinated, and unpredictable fashion. We model this setting by a stochastic continuous-time process: 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 determined by an *arbitrary* distribution G with mean $1/\mu$. This is also called the $M/G/\infty$ model in queuing theory. (This is more realistic than the less general $M/M/\infty$ used in [14] to model P2P networks.) Measurement studies of real P2P systems [19, 20, 22] indicate that the above model approximates real-life data reasonably well, especially since the holding time distribution is arbitrary (in particular the study in [22] actually indicates that the holding times may follow Weibull or lognormal distributions).

Let G_t be the network at time t (G_0 has no vertices). We are interested in analyzing the evolution in time of the stochastic process $\mathcal{G} = (G_t)_{t \geq 0}$. Since the evolution of \mathcal{G} depends only 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 by showing that the number of nodes in the network rapidly converges to N which we call the *expected stable network size* (or simply, stable network size). We use the notation $G_t = (V_t, E_t)$ to denote the network at time t .

Throughout our analysis we use the Chernoff bounds for the binomial and the Poisson distributions. Let the random variable X denote the sum of n independent and identically distributed Bernoulli random variables each having a probability p of success. Then, X is binomially distributed with $\mu = np$. We have the following Chernoff bounds [1]: For $0 < \delta < 1$: $\Pr(X > (1 + \delta)\mu) \leq$

$e^{-\mu\delta^2/3}$ and $\Pr(X < (1 - \delta)\mu) \leq e^{-\mu\delta^2/2}$. We have identical bounds even when X is a Poisson random variable with parameter μ [1].

3.2 Network Size

The following theorem characterizes the network size and is a consequence of the fact that the number of nodes at any time t is a Poisson distribution (this is true even if the holding times follow an arbitrary distribution) [16, pages 18-19]; applying the Chernoff bound for the Poisson distribution gives the high probability result.

Theorem 1 (Network Size) *If $\frac{t}{N} \rightarrow \infty$ then $E[|V_t|] = N$, and 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 $1 - G(t - \tau)$. 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 $[0, t]$),

$$p(t) = \frac{1}{t} \int_0^t (1 - G(t - x)) dx = \frac{1}{t} \int_0^t (1 - G(x)) dx.$$

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 [16, pages 18-19]).

When $t/N \rightarrow \infty$, $E[|V_t|] = N$.

We can now use a tail bound for the Poisson distribution [1, page 239] to show that

$$\Pr\left(\left||V_t| - E[|V_t|]\right| \leq \sqrt{bN \log N}\right) \geq 1 - 1/N^c$$

for some constants b and $c > 1$. \square

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

Corollary 1 *Suppose that the ratio of 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 $\frac{t-\tau}{N'} \rightarrow \infty$ w.h.p. G_t has $N' + o(N')$ nodes.

3.3 Network Degree

Theorem 2 [Degree] *At any time t such that $t/N \rightarrow \infty$, the degree of a node and the routing table size is bounded by $O(\Delta \log N)$ w.h.p., where Δ is the maximum degree of H . (If H is a CCC, then the degree is $O(\log N)$.)*

Proof. We first show that the number of node-ids covering a given vertex is $\Theta(\log N)$ w.h.p. When the network is stable, the number of live nodes is at least $N - o(N)$ w.h.p, i.e., with probability at least $1 - 1/N^{\Omega(1)}$. Let Y_j be the indicator random variable for the event that some node j covers a given vertex v . Then

$$\Pr(Y_j = 1) \geq (1 - (1 - 1/S))^{N-o(N)}(1 - 1/N^{\Omega(1)})$$

where $S = \Theta(N/\log N)$, is the size of H . Thus, by linearity of expectation, the expected number of nodes covering a given vertex is

$$(N \pm o(N))(1 - (1 - 1/S))^{N-o(N)}(1 - 1/N^{\Omega(1)}) = \Theta(\log N).$$

Applying the Chernoff bound gives the high probability result.

Since the maximum degree of a vertex is Δ the number of edges incident on a node is $\Theta(\Delta \log N)$ since each node has edges to nodes that cover a constant number of vertices and each vertex covered by $\Theta(\log N)$ nodes w.h.p. The bound on the routing table size follows from the fact that H admits an routing scheme \mathcal{R} that has a routing table size of $O(\Delta)$ entries per node. \square

3.4 Fault-tolerance and Search

We show that every query succeeds w.h.p at any time (after a short initial period). We show this by first proving that every vertex is occupied w.h.p which ensures that a search that maps (i.e., hashes) to this vertex value can be serviced by some (live) node. This fact along with the way edges are constructed in the P2P network will show that every search will succeed w.h.p.

Lemma 1 (Occupancy of Vertices) *At any time t , such that $t/N \rightarrow \infty$, w.h.p. every vertex of H is occupied.*

Proof. When $t/N \rightarrow \infty$ nodes depart the network according to a Poisson process with rate 1. Also from Theorem 1, w.h.p., the number of nodes in the

network is at least $N - o(N)$. Since, every hole has an equal probability of getting filled at any time step t , the probability that a vertex is not covered is at most

$$\left(1 - \frac{1}{S}\right)^{(N-o(N))} (1 - o(1/N)) \leq e^{\Theta(1-o(1))} = o(1/N).$$

Applying the union bound ([12][Lemma 1.2]), the probability that no vertex is unoccupied is at most $1/N$. \square

The following theorem on the success probability of a search query is a consequence of the previous theorem and the way nodes link to each other.

Theorem 3 (Search) *For any time t , such that $t/N \rightarrow \infty$, w.h.p. every search will be successful. The number of steps needed is $O(D)$ w.h.p., where D is the diameter of H .*

Proof. Consider a search query emanating from the node with node-id x for a node covering a node-id t (the hash value of the data). This search will be successful if there is a path in G to one of the nodes covering t . In terms of the template graph H , consider the path from x to t given by the routing scheme \mathcal{R} of length $O(D)$. (If H is a CCC, then the \mathcal{R} is the bit-fixing scheme and D is $O(\log N)$.) This path goes through a sequence of vertices in H . From Lemma 1, since every vertex of H is occupied w.h.p., every vertex in G is covered by some (live) node in the network. From our construction of G there is an edge between any node covering a vertex to any node covering the neighbor of the vertex. Thus, w.h.p the query will succeed and will take $O(D)$ steps. \square

The above theorem also implies the following result on the connectivity and diameter of the network.

Corollary 2 (Connectivity and Diameter) *For any time t , such that $t/N \rightarrow \infty$, the network is connected and has a diameter of $O(D)$ w.h.p.*

Theorem 4 (Overhead of Joining) *For any time t , such that $t/N \rightarrow \infty$, the time and message overheads for a node to join the network are respectively $O(D)$ and $O(D + \Delta \log N)$ w.h.p.*

Proof. An incoming node has to locate a node in the network with the same node-id; then it can find all of its neighbors in $O(1)$ time and $O(\log N)$ messages. Finding such a node (starting from some entry point node) takes $O(D)$ time (Theorem 3). Hence the total time needed to find all neighbors is $O(D)$. The total number of messages needed is $O(D + \Delta \log N)$ w.h.p., since, w.h.p., D messages are needed for routing (to find a node of same id) and routing table update of size $O(\Delta \log N)$ has to be done in total (for the new node as

well as the neighbors of the new node). \square

4 Handling Change in Stable Network Size

The performance of our scheme depends on the stability of the network. It is easy to see that our scheme can easily tolerate changes up to constant factors (thus, as mentioned earlier, it is enough to have an estimate of N up to some constant factor). However, bad events, such as the network size drastically getting reduced, possibly even leading to the network getting disconnected, can happen, but with minuscule probability in our model. In case such events happen (which will eventually happen with probability 1 if the system runs forever) remedial measures can be taken such as resorting to an external mechanism to connect the network again (if the network gets disconnected) or rejecting new connections (if the size exceeds very much) till the situation self-corrects itself. Our analysis can be extended to handle such situations.

We now discuss how the scheme can be modified to accommodate gradual changes in stable network size. As shown in Corollary 1, if the ratio between the arrival and departure rates in the network change, then this leads to a new expected stable network size. Suppose the new stable size is one-half of the original network size. How can the network adapt to this changed size? Assume that H is a hypercube (similar argument will work for CCC and other related networks). All that is required is to reduce S (the size of H) by a factor of 2. This can be done easily in a local manner. Each node will simply reduce its dimension by 1. This can be accomplished by dropping the last bit in the node-id. The hash values of data are also altered in the same way. It is easy to see that because of the recursive nature of construction of the hypercube (i.e., a hypercube of dimension r can be constructed from two hypercubes of dimension $r - 1$), reducing the dimension will require only $O(1)$ overhead per node. To illustrate, consider two nodes with node-ids $\langle x_1, \dots, x_r, 0 \rangle$ and $\langle x_1, \dots, x_r, 1 \rangle$. Dropping the last bit, will make *both* these nodes to cover the vertex with label $\langle x_1, \dots, x_r \rangle$. Data that were originally serviced by either of these will now be serviced by both of them. On the other hand, if the stable network size increases by a factor of two, then each node will increase its dimension by one, by adding one more random bit to its node-id (cf. Section 2). To illustrate, consider the set of nodes with the same node-id $\langle x_1, \dots, x_r \rangle$. Randomly adding one more bit (last bit), will make on the average half of the nodes in this set to cover the vertex with label $\langle x_1, \dots, x_r, 0 \rangle$ and the other half to cover $\langle x_1, \dots, x_r, 1 \rangle$. The data that are serviced by these nodes also get hashed to the same node-ids. It is not difficult to show that the above transformation preserves all the properties of the scheme, namely network

degree, number of hops needed for search, fault-tolerance, connectivity, and diameter.

5 Concluding Remarks

We presented a simple and general scheme for building a structured P2P network. We analyzed our scheme under a realistic churn model and provably show that it gives essentially optimal bounds with respect to search time, degree, message complexity and maintenance overhead. The simplicity and generality of our scheme makes it a suitable candidate for real-world implementation.

References

- [1] N. Alon and J. Spencer. *The Probabilistic Method*. John-Wiley, 1992.
- [2] H. Balakrishnan, F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking Up Data in P2P Systems. *Communications of the ACM*, pages 43–48, February 2003.
- [3] M. Datar. Butterflies and p2p networks. In *Proceedings of the 10th European Symposium on Algorithms (ESA)*, 2002.
- [4] A. Fiat and J. Saia. Censorship Resistant Peer-to-Peer Content Addressable Networks. In *Proceedings of SODA*, 2002. Journal version in *Theory of Computing*, vol. 3, 2007, 1-23.
- [5] C. Gavoille and D. Peleg. Compact and localized distributed data structures. *Distributed Computing*, 16(2-3):111–120, 2003.
- [6] K. Hildrum and J. Kubiawicz. Asymptotically efficient approaches to fault-tolerance in p2p networks. In *17th International Symposium on Distributed Computing (DISC)*, 2003.
- [7] S. Jagannathan, G. Pandurangan, and S. Srinivasan. Query protocols for highly resilient peer-to-peer networks. In *19th International Conference on Parallel and Distributed Computing Systems*, 2006.
- [8] M. Kashoek and D. Karger. Koorde: A simple degree optimal distributed hash table. In *IPTPS*, 2003.
- [9] F. Leighton. *Introduction to Parallel Algorithms and Architectures*. Morgan Kaufmann, 1992.
- [10] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the Evolution of Peer-to-Peer Systems. In *Proceedings of ACM Principles of Distributed Computing*, 2002.
- [11] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: A Scalable and Dynamic Emulation of the Butterfly. In *ACM Principles of Distributed Computing*, 2002.

- [12] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.
- [13] M. Naor and U. Weider. A simple fault-tolerant distributed hash table. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.
- [14] G. Pandurangan, P. Raghavan, and E. Upfal. Building Low-Diameter P2P Networks. *IEEE Journal on Selected Areas in Communications*, 21(6):995–1002, 2003 (Preliminary version in FOCS 2001).
- [15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content Addressable Network. In *Proceedings of ACM SIGCOMM 2001*, 2001.
- [16] Sheldon Ross. *Applied Probability Models with Optimization Applications*. Dover Press, 1970.
- [17] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-scale Peer-to-Peer Systems . In *Proc. of the IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350, 2001.
- [18] J. Saia, A. Fiat, S. Gribble, A. Karlin, and S. Saroiu. Dynamically Fault-Tolerant Content Addressable Networks. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, March 2002.
- [19] S. Saroiu, P. Gummadi, and S. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, January 2002.
- [20] S. Sen and J. Wang. Analyzing Peer-to-Peer Traffic Across Large Networks. *IEEE/ACM Transactions on Networking*, 12(2):219–232, 2004.
- [21] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.
- [22] D. Stutzbach and R. Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet Measurement*, 2006.
- [23] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.