The Convergence-Guaranteed Random Walk and Its Applications in Peer-to-Peer Networks

Ming Zhong, Kai Shen, and Joel Seiferas
Department of Computer Science, University of Rochester
Rochester, NY 14627-0226, USA
Email: {zhong, kshen, joel}@cs.rochester.edu

Abstract— **Network structure construction and global state maintenance are expensive in large-scale, dynamic peer-to-peer (p2p) networks. With inherent topology independence and low state maintenance overhead, random walk is an excellent tool in such network environments. However, the current uses are limited to unguided or heuristic random walks with no guarantee on their converged node visitation probability distribution. Such a convergence guarantee is essential for strong analytical properties and high performance of many p2p applications. In this paper, we investigate an approach for random walks to converge to application-desired node visitation probability distributions while only requiring information about direct neighbors of each peer. Our approach is guided by the Metropolis-Hastings algorithm for Monte Carlo Markov Chain sampling. Our contributions are three-fold. First, we analyze the convergence time of the random walk node visitation probability distribution on common p2p network topologies. Second, we analyze the fault tolerance of our random walks in dynamic networks with potential walker losses. Third, we present the effectiveness of random walks in assisting three realistic network applications: random membership subset management, search, and load balancing. Both search and load balancing desire random walks with biased node visitation distributions to achieve application-specific goals. Our analysis, simulations, and Internet experiment demonstrate the advantage of our random walks compared with alternative topology-independent index-free approaches.**

I. Introduction

Random walk is a way to sample network nodes — at each step, a walker randomly chooses its next hop to visit (among direct neighbors of the current node) following certain probabilistic preference for each neighbor. Random walk is particularly attractive in large-scale, dynamic peer-to-peer (p2p) networks. In these networks, nodes can join and leave dynamically without centralized control, and the network topology itself can also change over time. Random walk requires little global knowledge or state maintenance and it can function on almost all connected network topologies. In these aspects, it is superior to systems with sophisticated index states or rigid network structures, *e.g.*, distributed hash tables (DHTs) [35], [37], [43]. Compared with index-free node traversal schemes like network flooding, random walk is inherently scalable in that its network communication overhead does not increase as the network size grows.

This work was supported in part by the National Science Foundation (NSF) grants CCR-0306473, ITR/IIS-0312925, CNS-0615045, and CCF-0621472. Shen was also supported by an NSF CAREER Award CCF-0448413 and two IBM Faculty Awards.

Many current uses of random walks [6], [12], [18], [19], [26], [28] do not follow any topology-driven or application-specific guidance — at each step the walker chooses from current outgoing links with an equal probability for each link. Others make biased random walk decision at each step following some simple heuristics [10], [27], [29], [44]. In either case, the random walks do not provide any guarantee on their converged node visitation probability distributions. Such a convergence guarantee is desirable or even essential for many p2p applications utilizing random walks. For instance, a random membership subset service desires uniformly random node sampling for the maintenance of representative membership subsets [15], [16], [22], [24], [25], [38]. As another example, object search with certain bias (each node is searched with probability proportional to the square-root of its content popularity) is known to achieve low search latency among index-free searches [11], [28].

In this paper, we investigate an approach to support random walks that uniquely converge to application-desired node visitation probability distributions. At each step, the walker randomly chooses its next hop following certain probabilistic preference determined with the assistance of the Metropolis-Hastings algorithm [21], [32]. In this approach, the random walk movement at each step only requires local information and information concerning direct neighbors.

Our contribution in this paper is three-fold. First, we study the convergence time of our random walks for achieving the targeted node visitation probability distribution. This is important since random walks can take significant warm-up time to converge to the desired node visitation distribution. We provide analytical bounds for the random walk convergence time on several common p2p network topologies: k-dimensional tori, Chord topology [43], random powerlaw topology, and random regular topology. In addition to the analytical bounds, we also produce simulation results under some typical network setups.

Our second contribution is an analysis of the fault tolerance of random walk based node sampling. Random walk is inherently robust since it requires no global state maintenance. However, a walker may be lost due to node failures or departures in a dynamic network. We propose two methods to recover walker losses: one employs periodic walker callbacks to detect walker losses; the other enforces a bounded lifetime for each walker and deterministically reincarnates a new walker at the lifetime bound of the older walker. We analyze these two methods' fault tolerance abilities under given random walk convergence time and node failure model. Particularly for the second method, we derive an optimal walker lifetime to maximize its fault tolerance.

Our third contribution concerns application studies. Unstructured p2p applications often require a node sampling service that can achieve guaranteed node visitation distributions with high scalability and robustness. Specifically, some applications may desire uniform node visitation probability distribution while others may want non-uniform distributions to achieve application-specific goals. We present the effectiveness of random walks in assisting three realistic network applications: random membership subset management, search, and load balancing. Our results show that our convergence-guaranteed random walks can achieve desired application goals or high performance in a topology-independent index-free fashion.

The rest of this paper is organized as follows. Section II discusses existing study on random works in p2p networks. Section III provides the theoretical foundation for guaranteed unique convergence of random walk node visitation probability distribution. Sections IV and V analyze two important issues concerning our proposed random walks — convergence time and fault tolerance — respectively. Section VI presents the effectiveness of our approach and application-specific issues in the context of three application studies. Section VII describes a prototype implementation and Internet experiment of our proposed convergence-guaranteed random walks. Section VIII concludes the paper with a summary of our results.

II. Related Work

Random walks have been used in many p2p applications, including search [18], [19], [28], topology construction [18], [26], [31], and peer sampling [6]. For example, Lv *et al.* find that random walk based search is preferable to flooding-based search in unstructured p2p networks [28]. Gkantsidis *et al.* report that random walks are particularly better than flooding in two application scenarios, when topologies are clustered or when multiple requests are issued for the same query [18]. Their subsequent work shows that hybrid search schemes, which combine short random walks with local flooding, may lead to further performance improvement over pure flooding or random walks in clustered topologies [19]. Law and Siu propose a low-overhead distributed algorithm for constructing well-connected topologies, where a new peer is connected to existing nodes chosen by running random walks for a certain number of steps [26]. In addition, Bharambe *et al.* use a random walk based peer sampling algorithm that allows each node to estimate system-wide metrics (*e.g.*, global load distribution) based on peer samples [6]. However, random walks in these approaches do not follow any topology-driven or application-specific guidance — at each step the walker chooses from current outgoing links with equal probabilities. Such unguided random walks always visit nodes with probabilities proportional to their degrees and cannot support other application-specific node sampling distributions.

There has been some attempts to exploit the applicability of biased random walks in p2p systems [10], [12], [27], [29], [44]. Typically, such work is motivated by Adamic *et al.*'s discovery that the high-degree nodes (super-peers) in powerlaw graphs may be utilized by random walks to achieve search performance scaling sub-linearly with the network size [1]. Specifically, Gia uses random walks biased towards high-capacity peers to enhance the search performance of Gnutella [10]. Lv *et al.* also use capacity-biased random walks to speed up the search process in Gnutella [29]. Cooper improves the search performance of random walks by always forwarding walkers to the neighbors with the most documents [12]. This may increase the probability of finding matches since the walkers tend to quickly cover a large volume of data. Loguinov *et al.* suggest that unbalanced zone partitioning in DHTs may be addressed by using zone size biased random walks (split the largest zone found upon peer joining, and merge with the smallest zone discovered upon peer departure) [27]. Adaptive probabilistic search (APS) uses feedback from previous searches (maintained in local index tables) to direct random walkers [44]. In summary, the setup of these biased random walks are guided by heuristics derived from network topologies or application-level techniques. Although these empirical methods may achieve better quantitative performance than unguided random walks, there is little understanding on their analytical properties because they lack a guarantee on converged node visitation distributions.

We are aware of a recent work by Stauffer and Barbosa on probabilistic random flooding [42]. Both random walks and random partial flooding can achieve probabilistic node visitations. Like random walks, flooding is robust and requires very limited index information. However, Stauffer and Barbosa's work on probabilistic random flooding [42] only attempts to achieve uniform visitation. In comparison, our guided random walks can converge to arbitrary (potentially non-uniform) node visitation probabilities with proved convergence. Further, we would like to point out that the "node visitation probability" for our random walks differs from that of typical flooding. Our random walks are concerned with the steady-state node visitation probability of each walk step. In probabilistic flooding [42], the node visitation probability often refers to the chance for a node to be eventually visited in a flooding session. Such semantic difference makes flooding most appropriate for information dissemination while our random walks can support additional applications such as membership subset management and continuous load statistics maintenance.

III. Foundation for Guaranteed Random Walk Convergence

Let $G = (V, E)$ be an undirected connected graph. A *random walk* on G starts at a node v_0, which is either fixed or drawn according to some initial distribution π_0. If the random walk is at node v_t at time step t, then it moves to a neighbor v_{t+1} of node v_t at step $t+1$, chosen randomly with a certain probability distribution. Let π_t denote the distribution of node v_t, so that $\pi_t(i) = Prob(v_t = i)$ for each $i \in V$. Let $P = (P_{i,j}), i, j \in V$, denote the transition matrix of the random walk — $P_{i,j}$ is the probability that the random walk moves from node i to node j in one step. $P_{i,j} = 0$ if nodes i,

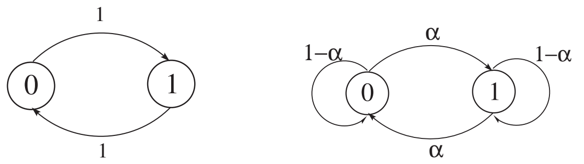

Fig. 1. An illustration on how self-loops achieve unique convergence of random walks.

j are not adjacent. The dynamics of the random walk follows $\pi_{t+1} = \pi_t P = \pi_0 P^{t+1}$.

Our random walks build on the Metropolis-Hastings algorithm [21], [32] — a standard approach to assign state transition probabilities to Monte Carlo Markov Chains such that they converge to any specified probability distributions.

Theorem 1: [3], [32] Let π be the desired probability distribution. Let d_i denote the degree of node i. For each neighbor j of node i, let

$$P_{i,j} = \begin{cases} \frac{1}{d_i} & \text{if } \frac{\pi(i)}{d_i} \le \frac{\pi(j)}{d_j}; \\ \frac{1}{d_j} \cdot \frac{\pi(j)}{\pi(i)} & \text{if } \frac{\pi(i)}{d_i} > \frac{\pi(j)}{d_j}. \end{cases}$$

and $P_{i,i} = 1 - \sum_{j \in \ neighbors(i)} P_{i,j}$. Then π is a converged probability distribution of the random walk with transition matrix P.

It is easy to show that π is a converged distribution by verifying $\pi P = \pi$. We also use a laziness factor in our random walk configuration to introduce self-loops at each node, which ensures that π is the unique converged distribution according to Doeblin [14].

Theorem 2: [14] If P is irreducible and aperiodic, then π_t converges to a unique stationary distribution π such that $\pi P = \pi$, independent of the initial distribution π_0.

Here P is *irreducible* if and only if for any i, j, there exists a t such that $(P^t)_{i,j} > 0$. P is *aperiodic* if and only if for any i, j the greatest common divisor of the set $\{t : (P^t)_{i,j} > 0\}$ is 1. Intuitively, *irreducibility* means that any two nodes are mutually reachable by random walks. *Aperiodicity* means that a random walk does not periodically commute between any two nodes. Aperiodicity can be achieved by introducing self-loop transitions of some positive probability on each node. For example, if a random walk with transition probabilities defined in the left diagram of Figure 1 starts from node 0, then it always stays at node 1 after an odd number of steps and visits node 0 after an even number of steps. Consequently, such a random walk oscillates between node 0 and 1 and does not have a unique converged distribution. However, it is easy to bring unique convergence to this random walk by just introducing a self-loop with probability $0 < \alpha < 1$ to each node as shown in the right diagram of Figure 1, after which the new random walk uniquely converges to π with $\pi(0) = \pi(1) = 0.5$.

Putting them altogether, we configure the random walk transition matrix in the following fashion. For each neighbor j of node i, we set

$$P_{i,j} = \begin{cases} \alpha \cdot \frac{1}{d_i} & \text{if } \frac{\pi(i)}{d_i} \le \frac{\pi(j)}{d_j}; \\ \alpha \cdot \frac{1}{d_j} \cdot \frac{\pi(j)}{\pi(i)} & \text{if } \frac{\pi(i)}{d_i} > \frac{\pi(j)}{d_j}. \end{cases}$$

and $P_{i,i} = 1 - \sum_{j \in \ neighbors(i)} P_{i,j}$. α is a laziness factor (between 0 and 1) to guarantee unique convergence. In our configuration, the random walk movement at each step only requires the knowledge on network degrees and desired visitation probabilities of the current node and its direct neighbors.

IV. Random Walk Convergence Time

Although our random walks are guaranteed to uniquely converge to the desired node visitation probability distribution, the convergence is not immediate after a random walk is initiated. For instance, after one step of the walk, it is not possible for the walker to go beyond the direct neighbors of the starting node. The convergence time indicates when a newly initiated random walk starts visiting nodes in the desired probability distribution. It also affects the random walk recovery from walker losses (examined later in Section V). In this section, we study the convergence time of random walk node visitation probability distribution.

A. Preliminary and Our Approach

We first introduce a metric for measuring the difference between two probability distributions.

Definition 1: The *difference* between two arbitrary probability distributions, x and y, is defined as $\|x, y\| = \frac{1}{2}\sum_i |x_i - y_i|$. The factor $\frac{1}{2}$ is to ensure that the maximum difference does not exceed 1.

Assuming that π is the desired distribution and π_t is the random walk node visitation distribution at step t, the extent to which the convergence is achieved at step t is measured by $\|\pi_t, \pi\|$. $\|\pi_t, \pi\| = 0$ obviously represents complete convergence. The factor $\frac{1}{2}$ ensures that $\|\pi_t, \pi\|$ never exceed 1. Fast convergence means that $\|\pi_t, \pi\|$ goes down quickly as t grows.

Definition 2: For $\epsilon > 0$, the *convergence time* is defined as $\tau(\epsilon) = \min\{t : \forall t' \ge t, \|\pi_{t'}, \pi\| \le \epsilon\}$.

The convergence time measures the time for π_t to converge to π. With these metrics, the convergence time of a random walk is bounded as follows.

Theorem 3: [13] Let $\pi_{\min} = \min_{\pi(i)>0} \pi(i)$, then $\tau(\epsilon) \le \Delta_P^{-1} \log((\pi_{\min}\epsilon)^{-1})$. Here Δ_P is the eigengap of the random walk transition probability matrix P.

It is known that P has $|V|$ eigenvalues $\lambda_1, \lambda_2, ..., \lambda_{|V|}$ such that $1 = \lambda_1 > |\lambda_2| \ge ... \ge |\lambda_{|V|}|$. The eigengap of P is defined as $\Delta_P = 1 - |\lambda_2|$, which provides a bound for the convergence time. A larger eigengap means shorter convergence time. However, for large-scale p2p network applications, the sizes of transition matrices are so large that it is very difficult to compute exact eigenvalues and eigengaps. Several approaches [13], [39], [40] have been proposed for establishing bounds for eigengaps of transition matrices. In this paper, we compute the eigengap bounds by using the *canonical path* approach [39].

The main idea of the canonical path approach is as follows. When there is a small cut in the random walk probability transition flow graph, it takes a long time for the probability flow to move from one side of the cut to the other (in order to reach the equilibrium). Thus the minimum cut (max-flow) in the probability transition graph provides a bound for the convergence time.

Let π be the unique converged distribution. P is the transition matrix of the random walk. $G = (V, E)$ is the random walk probability transition graph corresponding to P. For distinct nodes x, y in the graph G, a *canonical path* γ_{xy} refers to a path between x, y. Γ, a family of canonical paths, includes exactly one path for each pair of distinct nodes x, y: $\Gamma = \{\gamma_{xy} : x, y \in V, x \neq y\}$. Let $Q(e) = \pi(x)P_{x,y} = \pi(y)P_{y,x}$. From the view of probability flows between nodes, the path γ_{xy} carries a probability flow of $\pi(x)\pi(y)$ and $Q(e)$ represents the capacity of the edge e. A canonical path family Γ represents a routing scheme for every pair of distinct nodes in the network. The *congestion* of Γ is defined as

$$\rho(\Gamma) = \max_e \frac{1}{Q(e)} \sum_{\gamma_{xy} \ni e} \pi(x)\pi(y), \tag{1}$$

and the *congestion* of the graph G is defined as

$$\rho(G) = \min_\Gamma \bar{\rho}(\Gamma), \text{ where } \bar{\rho}(\Gamma) = \max_e \sum_{\gamma_{xy} \ni e} 1. \tag{2}$$

In principle, a canonical path family with low congestion means that the random walk transition probability graph does not have stringent bottlenecks for probability flows and hence the random walks can converge quickly. $\rho(\Gamma)$ measures the maximum per-edge flow-to-capacity ratio of a specific canonical path family Γ. $\bar{\rho}(\Gamma)$ is the maximum number of paths that is routed over an edge for the canonical path family Γ. It is obvious that the worst routing scheme may lead to a congestion of n^2 for a network with n nodes — when the flow between every two nodes must go through the same edge. It is also obvious that the minimum congestion on a network with n nodes and $|E|$ edges is $\frac{n^2}{|E|}$, when every edge is equally congested. $\rho(G)$ chooses the canonical path family with the minimum number of per-edge routing paths. Note that $\rho(G)$ is an inherent property of the network topology G and is independent of the random walk node visitation distribution π.

For random walks configured based on the Metropolis-Hastings algorithm (Theorem 1), we have (for $e = (i, j)$).

$$Q(e) = \pi(i)P_{i,j} = \begin{cases} \frac{\pi(i)}{d_i} & \text{if } \frac{\pi(i)}{d_i} \leq \frac{\pi(j)}{d_j}; \\ \frac{\pi(j)}{d_j} & \text{if } \frac{\pi(i)}{d_i} > \frac{\pi(j)}{d_j}. \end{cases}$$

Hence we have

$$\max_e \frac{1}{Q(e)} \leq \frac{D_{\max}}{\pi_{\min}},$$

where D_{max} is the maximum node degree.

The canonical path family with the minimum congestion on a given random walk transition probability graph provides a lower bound for the eigengap of the corresponding transition matrix P:

Theorem 4: [39] Let l represent the network diameter. Let D_{max} be the maximum node degree.

$$\Delta_P \geq \frac{1}{l \cdot \min_\Gamma \rho(\Gamma)} \geq \frac{1}{l} \cdot \frac{\pi_{\min}}{D_{\max}} \cdot \frac{1}{(\sum_{i \in \Pi_{\sqrt{\rho(G)}}} \pi(i))^2}$$

where $\Pi_{\sqrt{\rho(G)}}$ represent the $\sqrt{\rho(G)}$ nodes with the largest node visitation probabilities in the distribution π.

By combining Theorem 3 and Theorem 4, we have:

Theorem 5: The convergence time is bounded by

$$\tau(\epsilon) \leq l \cdot \frac{D_{max}}{\pi_{\min}} \cdot \left(\sum_{i \in \Pi_{\sqrt{\rho(G)}}} \pi(i) \right)^2 \cdot \log((\pi_{\min}\epsilon)^{-1})$$

The above result reveals that the random walk convergence time bound depends on network topology properties such as graph diameter, maximum node degree, and the congestion of the graph. It also relies on the skewness of π, the targeted node visitation probability distribution — the more skewed π is, the larger the time bound is (since the total probability of $\Pi_{\sqrt{\rho(G)}}$ becomes larger). For example, if π is a uniform distribution, then the convergence time bound is

$$\begin{aligned} \tau(\epsilon) &\leq l \cdot \frac{D_{max}}{\pi_{\min}} \cdot \left(\sum_{i \in \Pi_{\sqrt{\rho(G)}}} \pi(i) \right)^2 \cdot \log((\pi_{\min}\epsilon)^{-1}) \\ &= l \cdot \frac{D_{max}}{\frac{1}{n}} \cdot \frac{\rho(G)}{n^2} \cdot \log((\pi_{\min}\epsilon)^{-1}) \\ &= l \cdot D_{max} \cdot \frac{\rho(G)}{n} \cdot \log((\pi_{\min}\epsilon)^{-1}) \end{aligned}$$

As another example, if π is a highly skewed distribution (*e.g.*, Zipf's distribution) with the total probability of $\Pi_{\sqrt{\rho(G)}}$ close to 1, then the convergence time can be bounded as follows (larger than previous example)

$$\begin{aligned} \tau(\epsilon) &\leq l \cdot \frac{D_{max}}{\pi_{\min}} \cdot \left(\sum_{i \in \Pi_{\sqrt{\rho(G)}}} \pi(i) \right)^2 \cdot \log((\pi_{\min}\epsilon)^{-1}) \\ &\leq l \cdot \frac{D_{max}}{\pi_{\min}} \cdot \log((\pi_{\min}\epsilon)^{-1}) \end{aligned}$$

B. Asymptotic Convergence Time Bounds for Common P2P Topologies

Using the canonical path approach explained above, here we derive bounds for the convergence time of two commonly used convergence-guaranteed random walks in various p2p topologies. Specifically, we examine four kinds of network topologies as listed below.

- *Tori.* Tori-like structures have been used for self-organizing Content Addressable Networks (CAN) [35]. Structures like 2-d tori are also common in geographically-constrained networks (where nodes' transmission ranges are limited by geographical distances) such as wireless ad hoc networks.
- *Chord topologies* [43]. An n-node ring-like network with each node also connected to (besides its direct neighbors on the ring) its 2-hop neighbors, 4-hop neighbors, ..., $\frac{n}{2}$-hop neighbors on the ring.
- *Random powerlaw graphs.* Measurement results on many existing p2p systems [41] observed powerlaw node degree distributions. It is known that powerlaw node degree distributions may occur when each network node is

connected to some other nodes chosen randomly with probability biased towards their degrees [5], [9].
- *Random graphs*. Random graphs model those network applications in which each node is connected to some random nodes chosen uniformly at random [38].

Table I shows the network diameters (l) and congestions ($\rho(G)$ as defined in Equation 2) of these network topologies. Based on Theorem 5 and Table I, the convergence time of random walks with two commonly targeted node visitation distributions (one is uniform while the other is Zipf's distribution) can be bounded as shown in Table II.

In general, tori have the largest diameter and congestion values and hence the slowest convergence. The other three topologies are known to possess low diameters and low congestion properties, which leads to quick convergence. In addition to the network topology, the desired sampling distribution also affects the random walk convergence time in that edges with small transition probabilities (associated with nodes with small visitation probabilities) may slow down random walks. The problem may become more severe when these edges happen to be bottlenecks for random walk movements as revealed by edge congestion. Table II shows that the convergence time grows as the targeted distribution becomes more skewed, *i.e.*, $\frac{(\sum_{i\in\Pi} \sqrt{\rho(G)} \pi(i))^2}{\pi_{min}}$ becomes higher.

C. Simulation Results

As a complement to the analytical results, we provide simulation results to quantitatively assess the convergence time under common network topologies and node visitation distributions. For our simulation, we measure the convergence time of a random walk as $\tau(0.01)$, the number of walk steps needed before $\|\pi_t, \pi\|$ drops below 0.01.

Figure 2 presents the random walk convergence time under different network sizes, topologies, and node visitation distributions. The results show that the convergence time of different network topologies follows the order of "2-d tori" $>$ "random powerlaw" $>$ "random regular" $>$ "Chord". This mostly matches the order of their network diameters and congestion properties shown in Table I. However, it is worth noting that Chord networks unexpectedly outperform random powerlaw and random regular topologies. This is because the average node degree of Chord topologies ($\log n$) is higher than that of these random topologies (a constant). More per-node links decrease congestion properties and reduce diameters.

Figure 2 also shows that the convergence time tends to grow linearly with the network size, which falls within the analytical convergence time bounds derived in Table II. In particular, the quantitatively measured convergence time grows significantly slower than the analytical bounds on 2-d tori and random powerlaw graphs. This is mainly because current bounding techniques may not be able to achieve tight convergence time bounds for random walks on all topologies and node visitation distributions.

D. Summary

We summarize our results on random walk convergence time as follows. First, the convergence time depends on the network topology and the targeted node visitation probability distribution. In principle, topologies with higher connectivity or target visitations with more uniform distribution allow faster convergence. Second, we derived analytical bounds for random walk convergence time on common p2p network topologies and target visitation probability distributions. Our quantitative simulation validates the analytical bounds although, due to the limitations of current bounding techniques, the bounds are not tight in some cases.

Note that the derived convergence time is represented in the number of walk steps. In practice, the convergence delay in absolute time may be of more direct interests. Given the convergence delay in random walk steps, we can achieve desired absolute convergence time by adjusting the time between adjacent walk steps. Therefore faster convergence time can be achieved with faster-paced walks (and consequently more processing and network overhead).

V. Random Walk Fault Tolerance

Random walks in p2p networks must tolerate network faults and dynamic network changes. These include node or link failures, dynamic node arrival/departure, and network topology changes. Compared to network management with sophisticated index states or rigid network structures (*e.g.*, DHTs [35], [37], [43]), random walks are inherently more fault tolerant since it requires little state maintenance. For our convergence-guaranteed random walks, the only required state at each node consists of the network degree and visitation probability of its direct neighbors. Consequently, our random walks can tolerate network faults and changes as long as the required state at each node (information concerning direct neighbors only) can be properly updated. However, one problem due to dynamic network changes warrants attention — a walker may be lost if the node it currently resides at abruptly departs from the network (or simply fails). To maintain continuous random walks, walker losses must be promptly discovered and new walkers must be initiated.

Note that we assume fail-stop node failure model in this study. Under the fail-stop model, a node fails by simply stopping its function. We do not consider other failure models such as Byzantine node failures (in which a failed node may do arbitrary things) or malicious nodes.

A. Walker Loss Recovery Methods

We describe two methods to recover from walker losses:

- *Callback.* Each walker makes periodic callbacks to the originating node. If a sufficient number of callbacks are not received in a row, the walker is considered lost and a new walker will be initiated.
- *Expiration.* Each random walker is associated with a certain Time-To-Live (or lifetime). The walker will stop propagating (or expire) when the lifetime ends. The walker originating node keeps a timer that alerts at the end of walker lifetime. A new walker will be initiated at such time. If a walker is lost before its lifetime ends, its replacement is not initiated until that time.

	k-d tori	Chord	random powerlaw	random graphs
Diameter	$n^{\frac{1}{k}}$	$\log n$ [27]	$O(\log n)$ with high prob. [8]	$O(\log n)$ with high prob. [7]
Congestion $\rho(G)$	$O(n^{1+\frac{1}{k}})$	$O(n \cdot \log n)$	$O(n \cdot (\log n)^2)$ with high prob. [17]	$O(n \cdot \log n)$ with high prob. [17]

TABLE I

THE DIAMETERS AND CONGESTION PROPERTIES OF SEVERAL COMMON NETWORK TOPOLOGIES. n IS THE NUMBER OF NODES IN THE NETWORK.

	k-d tori	Chord	random powerlaw	random graphs
Random walks for uniform distributions	$O(n^{\frac{2}{k}} \cdot \log \frac{n}{\epsilon})$	$O((\log n)^3 \cdot \log \frac{n}{\epsilon})$	$O((\log n)^3 \cdot n^{\frac{1}{\alpha}} \cdot \log \frac{n}{\epsilon})$	$O((\log n)^3 \cdot \log \frac{n}{\epsilon})$
Zipf-biased random walks where $\pi(i) \propto \frac{1}{i}$	$O(n^{1+\frac{1}{k}} \cdot \log n \cdot \log \frac{n}{\epsilon})$	$O(n \cdot (\log n)^3 \cdot \log \frac{n}{\epsilon})$	$O(n^{1+\frac{1}{\alpha}} \cdot (\log n)^2 \cdot \log \frac{n}{\epsilon})$	$O(n \cdot (\log n)^3 \cdot \log \frac{n}{\epsilon})$

TABLE II

THE CONVERGENCE TIME, $\tau(\epsilon)$, OF TWO RANDOM WALKS ON COMMON P2P TOPOLOGIES. WE ALSO ASSUME THAT THE RANDOM POWER-LAW GRAPH FOLLOWS A DEGREE DISTRIBUTION OF $P(k) \propto k^{-\alpha}$, FOR WHICH THE MAXIMUM NODE DEGREE IS $O(n^{\frac{1}{\alpha}})$ WITH HIGH PROBABILITY FOR LARGE n'S. IT IS KNOWN THAT α RANGES FROM 2 TO 3 IN MANY REAL-WORLD APPLICATIONS [4].

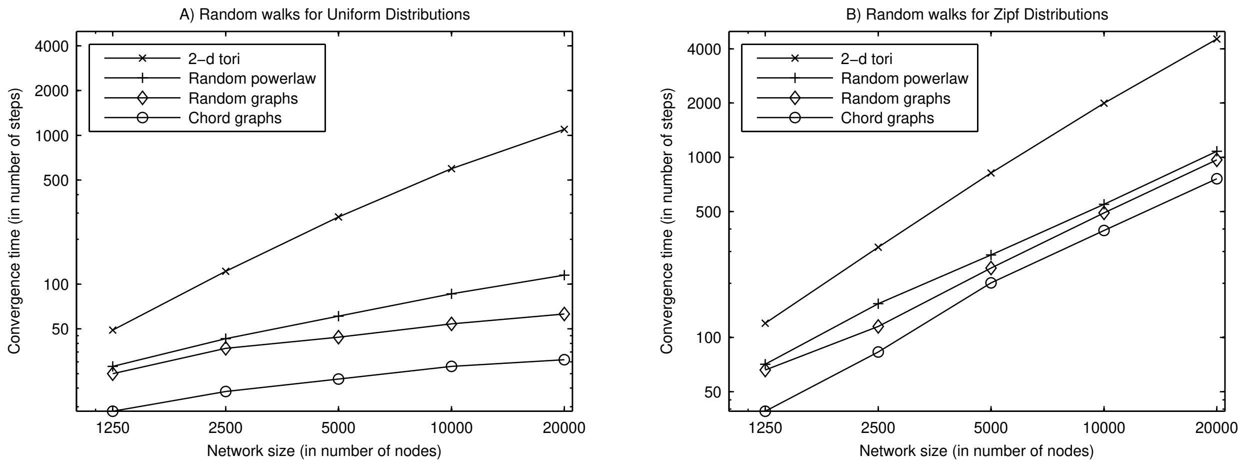

Fig. 2. Random walk convergence time on different network sizes, topologies and node visitation distributions. We use topologies with an average node degree of 4. In the simulation, the random powerlaw graph is generated by linking each new node to existing nodes chosen randomly with probability proportional to their degrees. Such a preferential link creation process is known to generate topologies with powerlaw degree distributions [5], [9]. We generate random graphs by a simple process of linking each node to some other nodes chosen uniformly at random.

The main weakness with the Callback method lies in the overhead of callbacks. Note that some random walk based applications require callback messages as part of the application semantics (such as membership subset management and load balancing as described later in Section VI). For these applications, periodic walker callbacks for loss detection can be piggybacked in application callback messages and consequently they are almost free.

Unlike Callback, the Expiration method requires no additional network overhead. However, its recovery of walker losses may not be prompt — if a walker is lost soon after it leaves the originating node, a replacement walker will not be initiated until the full expiration timer. At the other end, the Expiration method forces walker re-initiation even if the previous walker has not been lost, thus requiring additional random walk convergence time to reach desired node sampling distribution. An additional problem with this scheme is that it may be hard for the walker to track the elapsed time since its initiation. The difficulty arises in networks where nodes do not have synchronized clocks and network latencies between nodes are unknown.

B. Analysis and Quantitative Results

We analyze the availability of converged random walks under dynamic network conditions with potential walker losses. We define a metric of *availability* as the proportion of time during which the random walk is existent and has already converged to the desired node visitation probability distribution. As defined in Section IV-A, a random walk converges when its current node visitation probability distribution differs from the desired distribution by no more than a given error ϵ. We analyze the availability for the two walker loss recovery methods — Callback and Expiration.

The availability under walker losses certainly depends on how long a walker is lost since its initiation. We use $f(t)$ to denote the probability density function for a walker lost after t unit of time since its launch. Our analysis contains three parts. In part one, we derive the availability result for the two walker loss recovery methods with no assumption on $f(t)$. In part two, we will refine the result with the assumption that $f(t)$ follows an exponential distribution (assuming the failure model is memory-less). Part three is motivated by the difficulty of choosing the walker lifetime for the Expiration method. We will follow up the result in part two to derive the availability-maximizing walker lifetime for this method.

1) We first define some notations. Let the walker convergence delay be C unit of time (assumed to be a constant). For the Callback method, let the walker loss detection delay be D unit of time (assumed to be a constant). For the Expiration method, let the walker lifetime be T unit of time.
   We call a round as a duration of time between two adjacent walker initiations. The availability of converged random walks is calculated as the mean available time during a round divided by the mean time of a round. For Callback, the availability is:
$$\frac{\int_C^\infty f(t)\cdot(t-C)\,\mathrm{d}t}{\int_0^\infty f(t)\cdot(t+D)\,\mathrm{d}t} \tag{3}$$
   For Expiration, the availability is 0.0 if $T \leq C$. Otherwise, the availability is:
$$\frac{\int_C^T f(t)\cdot(t-C)\,\mathrm{d}t + \int_T^\infty f(t)\cdot(T-C)\,\mathrm{d}t}{T} \tag{4}$$
2) In this part we assume each node follows a memory-less failure model with an exponential distribution for the time to next failure. The probability density function for this distribution is $\frac{e^{-t/\lambda}}{\lambda}$ where λ is the average time to next failure (or MTTF). We also assume nodes fail independently of each other. Then the walker loss time follows the same distribution as node failure — *i.e.*, $f(t) = \frac{e^{-t/\lambda}}{\lambda}$.
   In this case, for Callback, the availability in Equation (3) is refined to:
$$\frac{e^{-C/\lambda}}{1+D/\lambda} \tag{5}$$
   In this case, for Expiration, the availability in Equation (4) is refined to:
$$\frac{\left(e^{-C/\lambda}-e^{-T/\lambda}\right)}{T/\lambda} \tag{6}$$
3) For the Expiration method, the choice of T (walker lifetime) may affect the availability of converged random walks. Too short a walk lifetime is undesirable because a random walk may not have converged before it expires. Too long a walker lifetime is also undesirable because a lost walker may take a long time to recover. We want to find an optimal T that maximizes the availability of converged random walks. We follow the assumption of part two that $f(t)$ follows an exponential distribution (assuming the failure model is memory-less). The optimal T (called $\hat{T}$) is the point at which the first derivative of Equation (6) equals to 0. This means:
$$1+\frac{\hat{T}}{\lambda} = e^{\frac{\hat{T}}{\lambda}-\frac{C}{\lambda}} \tag{7}$$
   To derive a closed-form solution, we use the approximation of $e^x \approx 1+x+\frac{x^2}{2!}$, the first three items of the Taylor series of e^x. Given the approximation, we can simplify Equation (7) and provide a closed-form solution:
$$\frac{\hat{T}}{\lambda} \approx \frac{C}{\lambda}+\sqrt{2\cdot\frac{C}{\lambda}} \tag{8}$$

Under the assumption of independent memory-less node failure model, Figure 3 quantitatively shows the availability of two walker loss recovery methods with a varying range of C/λ. The availability of the Callback method is also affected by the walker loss detection delay (represented by D). We show three availability curves with $D/\lambda = 0.0$, $D/\lambda = 0.01$, and $D/\lambda = 0.1$ respectively. In practice, the walker loss detection delay is mostly affected by the frequency of callback messages. If we assume the callback messages are immediate and perfectly reliable, a walker loss is detected after a single callback message fails to arrive. In this case, the walker loss detection delay D is bounded by the interval length between two consecutive callback messages. For the Expiration method, we show its availability under the optimal walker lifetime setup derived in Equation (8).

Results in Figure 3 indicate that, when $C/\lambda = 0.0001$, the availability of converged random walks under the Callback method are 0.9999, 0.9900, and 0.9090 for $D/\lambda = 0.0$, $D/\lambda = 0.01$, and $D/\lambda = 0.1$ respectively while the availability for the Expiration method is 0.9859. When $C/\lambda = 0.01$, the availability under the two methods are 0.9000–0.9900 and 0.8623 respectively. When $C = \lambda$, the availability under the two methods are 0.3344–0.3679 and 0.1153 respectively. Overall, Callback achieves better availability than Expiration does (except for very small C and large D) but it incurs more overhead due to callback messages.

C. Summary

Random walks are inherently robust since it requires little state maintenance. However, node failures in a dynamic network may lead to walker losses. We consider two methods to recover from walker losses — Callback and Expiration. Callback detects walker losses more quickly at the cost of additional callback messages. Under given random walk convergence speed and node failure model, we analyze the availability of converged random walks under these two methods. Particularly for the Expiration method, we derive an optimal walker lifetime to maximize its fault tolerance. Our quantitative results show that the availability is high ($\geq 99\%$ for Callback and $\geq 86\%$ for Expiration) when the node

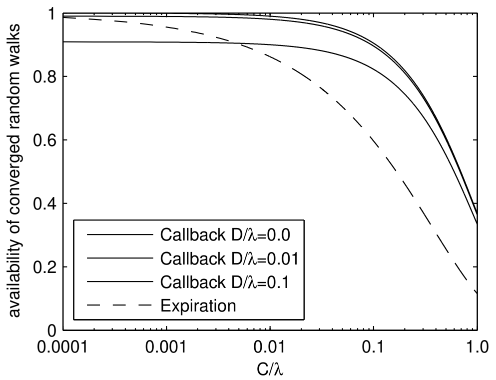

Fig. 3. The availability of two walker loss recovery methods when walker may be lost due to node failures. Note that the X-axis is in the logarithmic scale. We assume nodes fail independently of each other and we also assume each network node follows a memory-less failure model with an exponential distribution for the time to next failure. λ represents MTTF of node failures. C represents the convergence time for a newly initiated random walk. For the Callback method, we show three availability curves with $D/\lambda = 0.0$, $D/\lambda = 0.01$, and $D/\lambda = 0.1$ respectively (where D represents the walker loss detection delay). A curve with a smaller D is higher in the figure.

failure MTTF is at least two orders of magnitude larger than the random walk convergence time. When the node failure MTTF is only one order of magnitude larger than the random walk convergence time, Callback can still achieve over 90% availability.

VI. Application Studies

In this section, we show how convergence-guaranteed random walks can assist realistic applications in unstructured p2p networks. Specifically, our random walks provide a distributed node sampling service with high scalability, robustness, and guaranteed node visitation distribution. Section VI-A describes random membership subset management, which desires a uniform random walk node visitation probability to acquire representative subsets of the whole network members. Section VI-B presents the results for random walk based object search. Based on a known result [11], this application desires random walks with a biased node visitation probability distribution — each node is probed with probability proportional to the square-root of its content popularity. Section VI-C presents the results for random walk based load balancing. We show that our scheme (based on linearly load-biased node sampling) achieves better load balancing than conventional alternatives. We provide simulation results to quantitatively measure the performance gain of convergence-guaranteed random walks over alternative topology-independent index-free approaches.

A. Application I: Random Membership Subset Management

A membership service provides the list of members in a dynamic network and it is an important building block for distributed applications. When the overhead of maintaining the full list of members is too high, random membership subset is a viable alternative that can satisfy the membership service needs of many applications [25]. For random membership subset management, each node maintains a small, dynamically changing, random membership subset with uniform representation over network members.

Many existing random membership management algorithms, such as lpbcast [15], SCAMP [16], Saxons [38], and Jelasity *et al.* [22], provide analytical and experimental results on the membership information propagation speed. However, no theoretical guarantee is given for the uniformity of their membership subsets. Kostić *et al.* proposed a random membership subset service for tree-shaped network topologies [25]. However, this algorithm cannot be applied to more general mesh-like network structures. King and Saia proposed a distributed algorithm which, with high probability, always chooses a node uniformly at random from the set of nodes in distributed hash tables [24]. However, their algorithm only works for ring topologies.

We propose the first random membership subset management algorithm with topology independence and proved uniformity. Our algorithm maintains random membership subsets using random walk samplers that converge to a uniform distribution on arbitrary connected topologies. According to our approach described in Section III, we can guarantee unique convergence to a uniform distribution by configuring the random walk in the following way. If the random walk is at node x at time step t, then for each neighbor y of x, it moves to y with probability $P_{x,y}$, where

$$P_{x,y} = \begin{cases} \frac{1}{2} \cdot \frac{1}{d_x} & \text{if } d_x \geq d_y; \\ \frac{1}{2} \cdot \frac{1}{d_y} & \text{if } d_x < d_y. \end{cases}$$

and $P_{x,x} = 1 - \sum_{z \in \ neighbor(x)} P_{x,z}$. Here d_x, d_y denote the number of neighbors of node x and y, respectively.

A full service may function in the following way. For a node (called i) requiring a random membership subset with size k_i, it initiates k_i random walks $R_{i,1}, R_{i,2}, ..., R_{i,k_i}$, each of which converges to a uniform node visitation probability distribution. When visited by a random walk $R_{i,l}$, node j sends its membership information (*e.g.*, IP address) to i. Upon receiving j's membership information, i updates the lth element of it local membership set with j if j is not yet in set.

Simulation Results: We run simulations to validate the convergence of our random membership management algorithm on two common p2p topologies: random graphs and random powerlaw graphs. Random graphs represent those p2p topologies where new links are made independent of existing node degrees. We generate random graphs by connecting each new nodes to some nodes selected uniformly at random from existing nodes. Random powerlaw graphs represent those networks where new links are more likely attached to nodes with large degrees. In our simulation, the random powerlaw graphs are generated by using the PLRG algorithm [20]. We use the random powerlaw graphs with exponent $\beta = 0.8$, following Lv *et al.*'s simulation setup [28].

We compare our degree-guided random walks against unguided random walks — at each step the walker chooses from current outgoing links with equal probabilities. Figures 4 and 5 show the convergence results on random graphs and random powerlaw graphs respectively. For both topologies, our

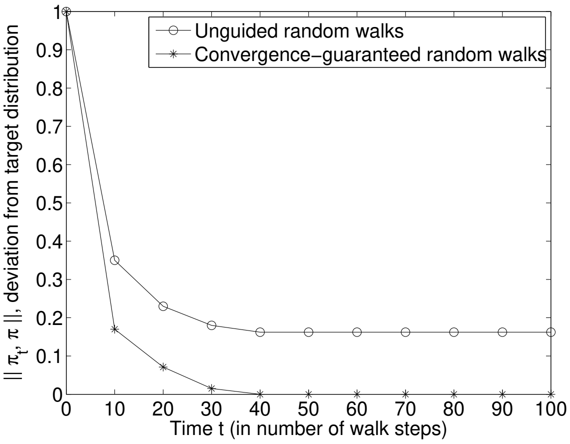

Fig. 4. The convergence of random walk node visitation probability distribution on random graphs (20000 nodes).

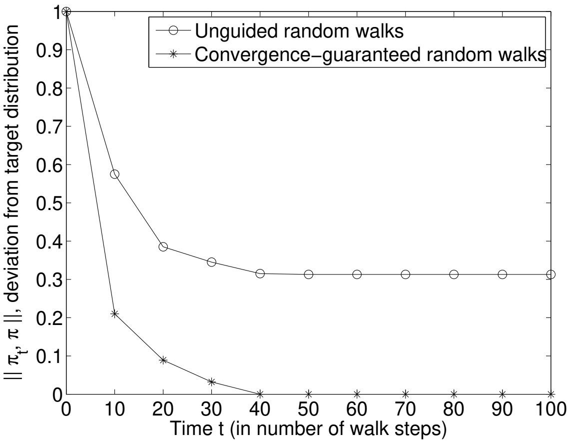

Fig. 5. The convergence of random walk node visitation probability distribution on random powerlaw graphs (20000 nodes).

degree-guided random walks quickly converge to the desired uniform node sampling distribution, *e.g.*, $\|\pi, \pi_t\|$ drops below 0.01 within 40 walk steps. In comparison, unguided random walks do not converge to the uniform node sampling. This is because unguided random walks are more likely to visit nodes with higher degrees while most network topologies (including the two experimented topologies) have skewed node degree distributions.

B. Application II: Index-Free Object Search

Many p2p search techniques utilize pre-constructed query routing indices about data locations to speed up the search process. The indices range from simple routing hints [36], [44], [46] in unstructured p2p networks to exact object locations used in distributed hash tables [35], [37], [43], which may be too expensive to maintain. In comparison, index-free search methods like query flooding and random walks are easier to deploy and maintain. Without any guidance, however, these approaches often suffer from long search latency caused by having to probe a large number of network nodes.

Cohen and Shenker showed that index-free searches guided by the *square-root principle* can achieve low search latency [11]. Under this principle, each object is probed with probability proportional to the square root of its query popularity. The square-root principle can be realized through data replication or topology adjustment. Specifically, data replication adjusts peer content popularities [11], [28] while topology adjustment changes peer visitation probabilities [12] under unguided random walks or flooding. However, these techniques may not be feasible in p2p applications with large, dynamic datasets, where the maintenance of up-to-date topologies or data replication copies often incur considerable overhead.

We seek to support efficient index-free search using popularity-biased random walks rather than biased replication or topology adjustments. Our goal is to achieve search time comparable with alternative search methods but at no cost of data movement or topology changes. Based on the framework in Section III, each query issues a random walker configured as follows. Let d_i denote the number of network neighbors of peer i. Let p_i denote the content popularity of peer i. If a random walker is at peer i at a certain time step, then for each neighbor j of i it moves to j with probability $P_{i,j}$ after next step, where:

$$P_{i,j} = \begin{cases} \frac{1}{2} \cdot \frac{1}{d_i} & \text{if } \frac{\sqrt{p_i}}{d_i} \leq \frac{\sqrt{p_j}}{d_j}; \\ \frac{1}{2} \cdot \frac{1}{d_j} \cdot \frac{\sqrt{p_j}}{\sqrt{p_i}} & \text{if } \frac{\sqrt{p_i}}{d_i} > \frac{\sqrt{p_j}}{d_j}. \end{cases} \tag{9}$$

and the probability for the random walker does not move at the step $P_{i,i} = 1 - \sum_{k \in \ neighbor(i)} P_{i,k}$. The peer content popularity p_i can be estimated as the number of queries satisfied at peer i divided by the total number of queries received by i [12]. Hence $P_{i,j}$ is locally computable.

It is easy to see that the above random walk converges to π with $\pi(i) \propto \sqrt{p_i}$. After convergence, our random walks achieve the minimum expected search time for the known popularity distribution p [11]. Note that the convergence time is typically short compared with the expected object search time (after convergence) on common p2p topologies (*e.g.*, random graphs and random powerlaw graphs), which are known to support fast random walk convergence due to their high expansions and low diameters.

To speed up the search, multiple independent random walkers can be used, with the expectation that k independent random walkers after T steps tend to cover nearly equal number of nodes as one random walker after $k \cdot T$ steps [28]. Hence the search time can be reduced by roughly k times with no extra communication overhead.

Simulation Results: We compare the performance of our popularity-biased random walks with two existing approaches to achieve the square-root principle. Below are the specific approaches we consider in our simulation study.

- *Square-root replication*. Each object is replicated randomly over the network with the number of replication copies proportional to the square-root of its popularity. One unguided random walker is used for searching the network while we set the average number of replication copies as the number of random walkers used in the three approaches. This is intended to make a fair comparison since the expected search time for square-root replication is inversely proportional to the average number of replication copies. For example, making replication copies at every node obviously leads to 1-step search time.

- *Square-root topology*. Unguided random walkers are used to search the network in this scheme. The degree of each node is proportional to the square root of its content popularity. To transform the original topology into this square-root topology, we compute the node degree sequence and use the PLRG algorithm [20] to generate the new randomized topology with the desired node degree sequence.
- *Square-root biased walks*. Each query issues a number of random walkers that travel according to Equation (9). Similar to unguided random walks, the random walkers coordinate with each other and terminate if others have found the target.
- *Unguided random walks*. Each query issues a number of random walkers that, at each step, travel along each outgoing link of the current node with equal probabilities. The random walkers coordinate with each other by periodically calling back the source to know whether other walkers have found the target. If so, the remaining walkers terminate themselves.

We simulate a system that contains 1,000,000 objects. The number of queries is 100,000. In our simulation, query popularities follow Zipf-like distributions (the frequency of the ith most popular query is proportional to $\frac{1}{i^{\beta}}$). Specifically, we choose the exponent $\beta = 0.6$ and $\beta = 1.2$ based on Sripanidkulchai's measurement results on Gnutella traces [41]. We use random graphs and random powerlaw graphs as network topologies in our simulation. Their generation methods are the same as described in Section VI-A.

Figures 6 and 7 present the search time and communication overhead on different network topologies (random graphs and random powerlaw graphs), query popularity distributions, and the number of random walkers (k). We observe that unguided random walks have much lower performance than other three approaches guided by the square-root principle. Furthermore, the three methods have similar performance with small variations (average 14% difference for random graphs and 19% for random powerlaw graphs). Compared to the other two approaches, it is important to note that the random walk approach has the advantage of requiring no data movements or link changes.

Figures 6 and 7 also show the impact of query popularity distributions on the performance of the three search methods guided by the square-root principle. We find that the search performance for high-skewness popularity distributions ($\beta = 1.2$) is higher than that for low-skewness distributions ($\beta = 0.6$). This is because highly skewed query popularity distributions contain more heterogeneity to be exploited.

Our simulation supports the following results: 1) At no cost of topology maintenance and/or movement, our popularity-biased random walks achieve search performance comparable with other approaches guided by the square-root principle. 2) Using multiple random walkers can significantly reduce the search time, with slight increase in the communication overhead. Such increase is due to the convergence overhead

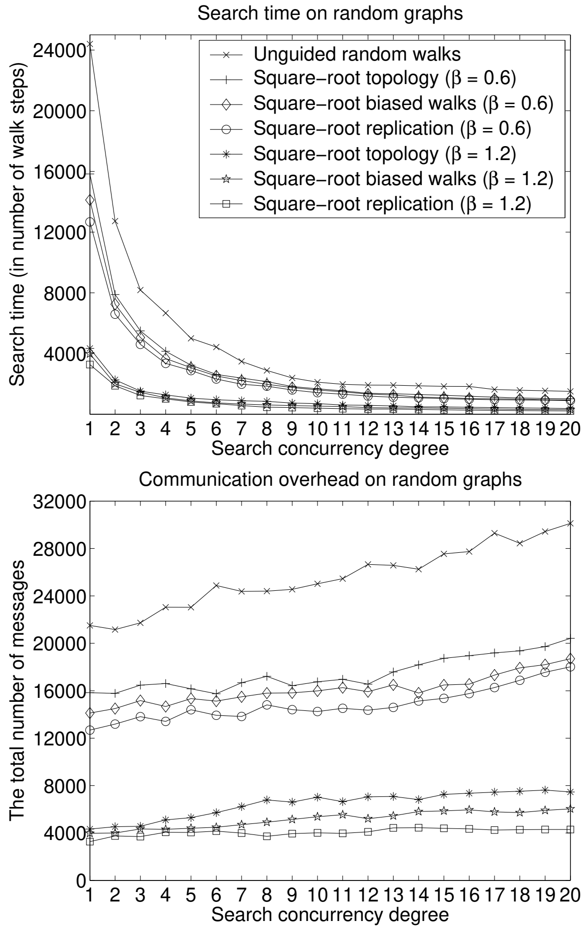

Fig. 6. The search time and communication overhead on random graphs (20000 nodes). The search concurrency degree represents the number of random walkers (or the average number of replication copies for square-root replication).

associated with more random walkers — each walker incurs certain overhead during its convergence process and more walkers require more overhead.

C. Application III: Load Balancing

In p2p networks, load imbalance may be caused by factors such as uneven distribution of data objects among nodes, heterogeneity in node capacities and data object sizes, as well as network structure changes. Many existing p2p load balancing algorithms (*e.g.*, virtual servers [23], [43] and dynamic zone balancing [2], [30], [34], [35], [45]) require an underlying distributed hash table (DHT) infrastructure. Hence they are not applicable to load balancing in unstructured p2p networks, where the overlay topology can be formed arbitrarily and query mechanisms such as DHT may not be available.

Without the assistance of a structured network or DHT, Karger and Ruhl [23] suggest to use uniform node sampling to support pair-wise load sharing. In each round, they let each node periodically balance its load with another node sampled with uniform probability over all network nodes. They show that their approach can reduce the maximum per-node load to a constant times the average per-node load within $O(\log n)$ load

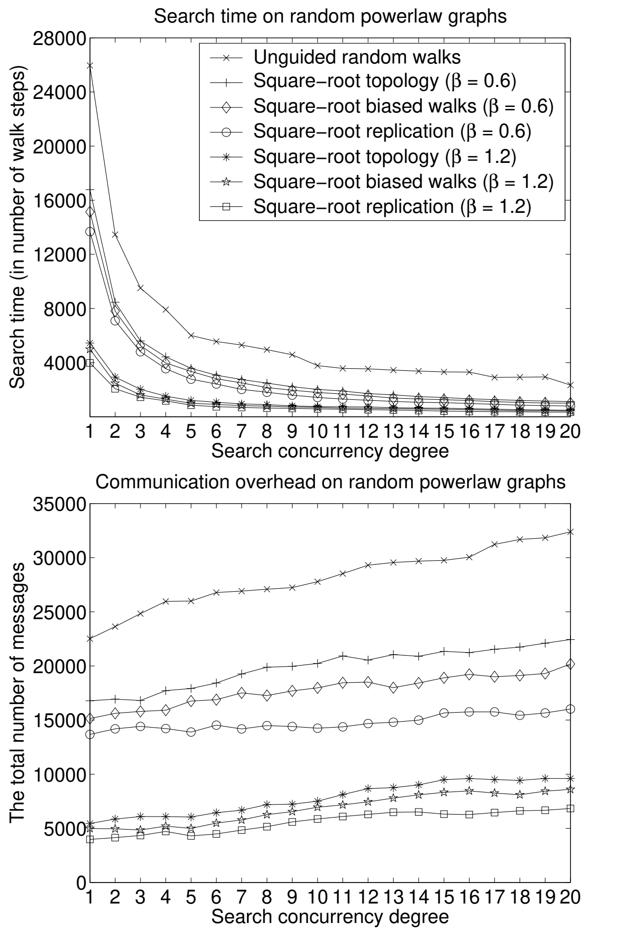

Fig. 7. The search time and communication overhead on random powerlaw graphs (20000 nodes). The search concurrency degree represents the number of random walkers (or the average number of replication copies for square-root replication).

balancing rounds (n is the network size). Furthermore, their result is asymptotically optimal because each load balancing operation between two nodes can at most reduce the per-node load by a half, and the maximum per-node load could be n times higher than the average.

Intuitively, sampling with probabilities biased towards highly loaded nodes are more likely to bring them into load balancing operations than uniform sampling. We propose a new load balancing algorithm that uses load-biased random walks to sample nodes and subsequently moves their load to lightly loaded nodes. Although at the same asymptotic level (which is optimal), our approach leads to *four* times smaller load balancing time upper-bound than that of uniform node sampling in Karger and Ruhl's approach [23]. For the detailed analysis, please refer to [47].

The basic framework for pair-wise load sharing is as follows:

> *In each round, every node i samples a random node $j \neq i$. They perform a load movement to balance their loads if the load of one node is at least γ times larger than that of the other.*

where the threshold γ avoids the load movement between two nodes with similar loads. We choose $\gamma = 2$ in our study.

The key variant of pair-wise load sharing is how the random node j is sampled. We sample nodes with probabilities proportional to their loads and perform load balancing operations accordingly. Such a linearly load-biased sampling has a unique property that the probability to choose a node with load $k \cdot L$ (assuming L is the average per-node load) is always $\frac{k}{n}$ regardless of the network-wide load distribution. Consequently, this sampling scheme leads to faster load balancing than uniform sampling because it tends to discover overloaded nodes more often. More importantly, this sampling distribution is superior to other biased distributions (*e.g.*, quadratically load-biased) in that it always favors nodes with above-average loads independent of global load distribution. We implement linearly load-biased node sampling based on the framework in Section III, with the random walk configured as follows.

Let $d(i)$ denote the number of neighbors of node i. If the random walk is at node i, then for each neighbor j of i, it moves to j with probability $P_{i,j}$ after next step, where:

$$P_{i,j} = \begin{cases} \frac{1}{2} \cdot \frac{1}{d(i)} & \text{if } \frac{Load(i)}{d(i)} \leq \frac{Load(j)}{d(j)}; \\ \frac{1}{2} \cdot \frac{1}{d(j)} \cdot \frac{Load(j)}{Load(i)} & \text{if } \frac{Load(i)}{d(i)} > \frac{Load(j)}{d(j)}. \end{cases}$$

and $P_{i,i} = 1 - \sum_{j \in \ neighbor(i)} P_{i,j}$.

Each node issues a random walker that persistently runs over the network as a node sampler. At each round, the random walker reports its currently visited node as a node sample to its source node, which then performs a load balancing operation to balance the load between itself and the sampled node, *i.e.*, an operation moves $\frac{A-B}{2}$ load from a node with load A to a node with load B ($A > B$). Here the time period for each sampling round needs to be long enough to avoid too frequent load sharing operations since load movement typically incurs substantial network bandwidth consumption. In addition, the random walkers from different nodes are independent and no synchronization is needed between them.

Simulation Results: We run simulations to compare the load balancing performance of our linearly load-biased balancing (based on random walks) against that of uniform balancing proposed by Karger and Ruhl [23]. We consider random graph and random powerlaw graph network topologies in our study. Their generation methods are the same as described in Section VI-A. The comparison is done in terms of two metrics.

- The *maximum load imbalance factor*, defined as the maximum per-node load divided by the average per-node load.
- The *number of hotspots*, defined as the number of nodes with load at least four times the average per-node load.

We assume that the initial load at a node (before load balancing algorithms are employed) is chosen with a powerlaw distribution with exponent -3, mean value 2, and minimum value 1. The heavy-tail feature of powerlaw distribution means that there may exist some highly loaded nodes in the network. Let T be the length of a load balancing round (or the time interval between two consecutive load balancing operations initiated from one node). Each node starts running sampling and load balancing algorithms at a time point chosen uniformly at random from the range $[0, T]$ in our simulation.

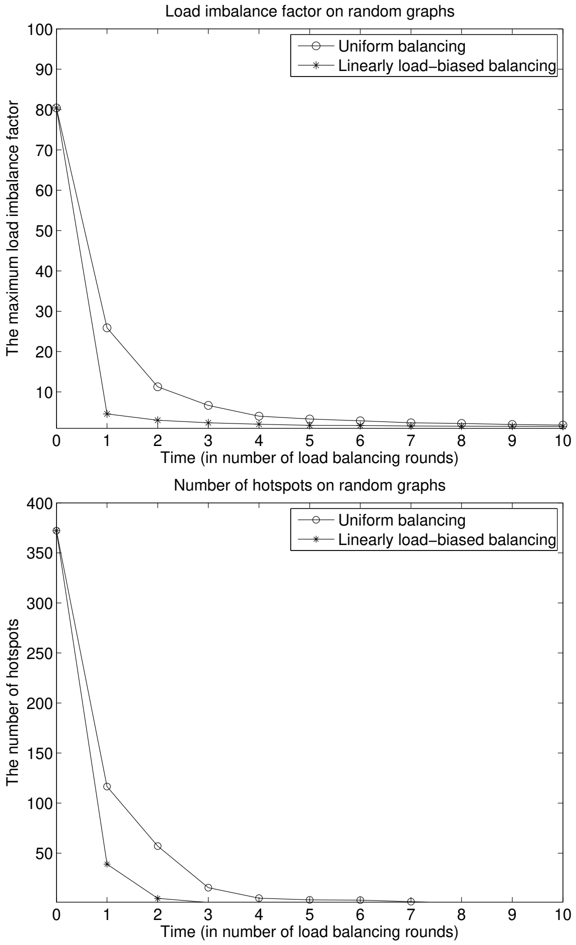

Fig. 8. The maximum load imbalance factor and the number of hotspots on random graphs (20000 nodes).

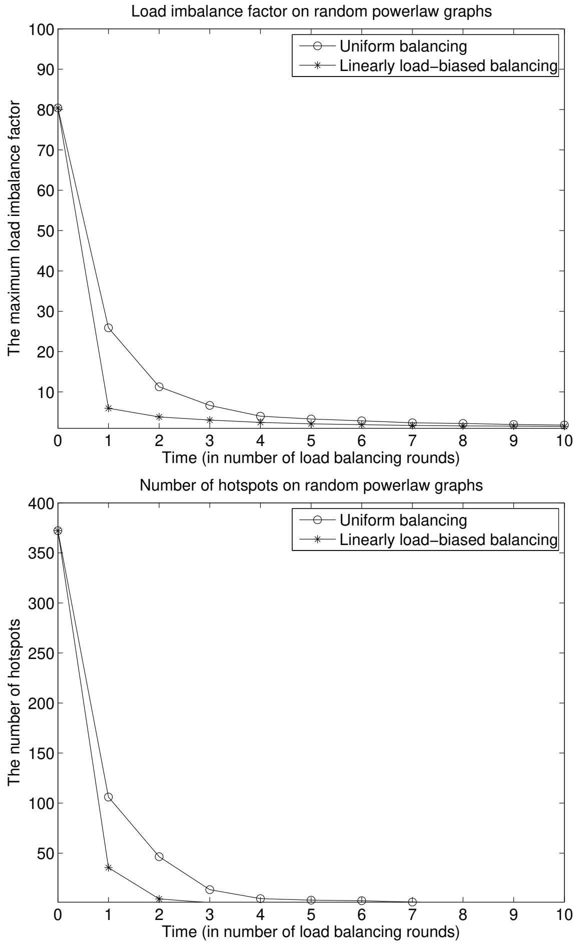

Fig. 9. The maximum load imbalance factor and the number of hotspots on random powerlaw graphs (20000 nodes).

Figures 8 and 9 compare the load imbalance reduction speeds of the two load balancing approaches. Results show that our load-biased random walks reduce load imbalance significantly faster than uniform sampling based balancing. For example, our load-biased random walks eliminate all hotspots within 3 load balancing rounds while uniform balancing takes 7 rounds. The performance difference is mainly due to the different speeds at which the three schemes discover and offload hotspots.

VII. Prototype Implementation and Internet Experiment

We have made a prototype implementation of convergence-guaranteed random walks over Internet overlay networks. The implementation is encapsulated in an event-driven random walk daemon at each node of the network. The daemon receives propagated random walkers and passes them to next hop (or keep them unmoved) at each step according to the appropriate transitional probabilities determined in Section III. A walker pauses for a certain time period after each step. We call this period the per-step interval length and it is employed to control the walker propagation speed and associated network overhead. The only non-local state that our random walk maintains at each node is the network degrees of its neighbors. Such state can be robustly maintained through soft state based communications between network neighbors. More specifically, neighboring nodes periodically update each other with up-to-date network degrees. Dynamic state changes or message losses can be simply recovered by later updates.

We consider the network overhead of walker movements. Figure 10 illustrates the random walker data structure in our implementation and it is 28-byte large. Counting the additional 28-byte UDP/IP headers, a random walker with 10 seconds per-step interval length will incur a small 45 bits/second walker movement overhead.

We conduct a simple experiment to demonstrate that our prototype implementation functions as expected. Our experiment uses 66 Planetlab [33] nodes over the Internet. The nodes form a random-topology network with an average node degree of 4 and a maximum node degree of 12. Our experiment attempts to achieve uniform node visitation probability distribution. Figure 11 illustrate the node visitation probability distributions of our convergence-guaranteed random walk. For each test, we let an arbitrarily chosen source send out 660 (ten times the node count) random walkers and we then track their movements at each step. We show the node visitation probability distributions at different stages of walker movements (steps 0–19, steps 20–39, and steps 40–59). Results suggest

```
struct RandomWalker {
    int type;            /* type of walker: unguided, convergence-guaranteed, etc. */
    struct in_addr source;     /* the walker originating node */
    int id;              /* walker id (distinguishing from other walkers from the same source) */
    double interval;     /* per-step interval length */
    unsigned int step;   /* number of steps since the beginning */
    unsigned int TTL;    /* time-to-live in steps to expiration */
};
```

Fig. 10. The random walker data structure in our implementation.

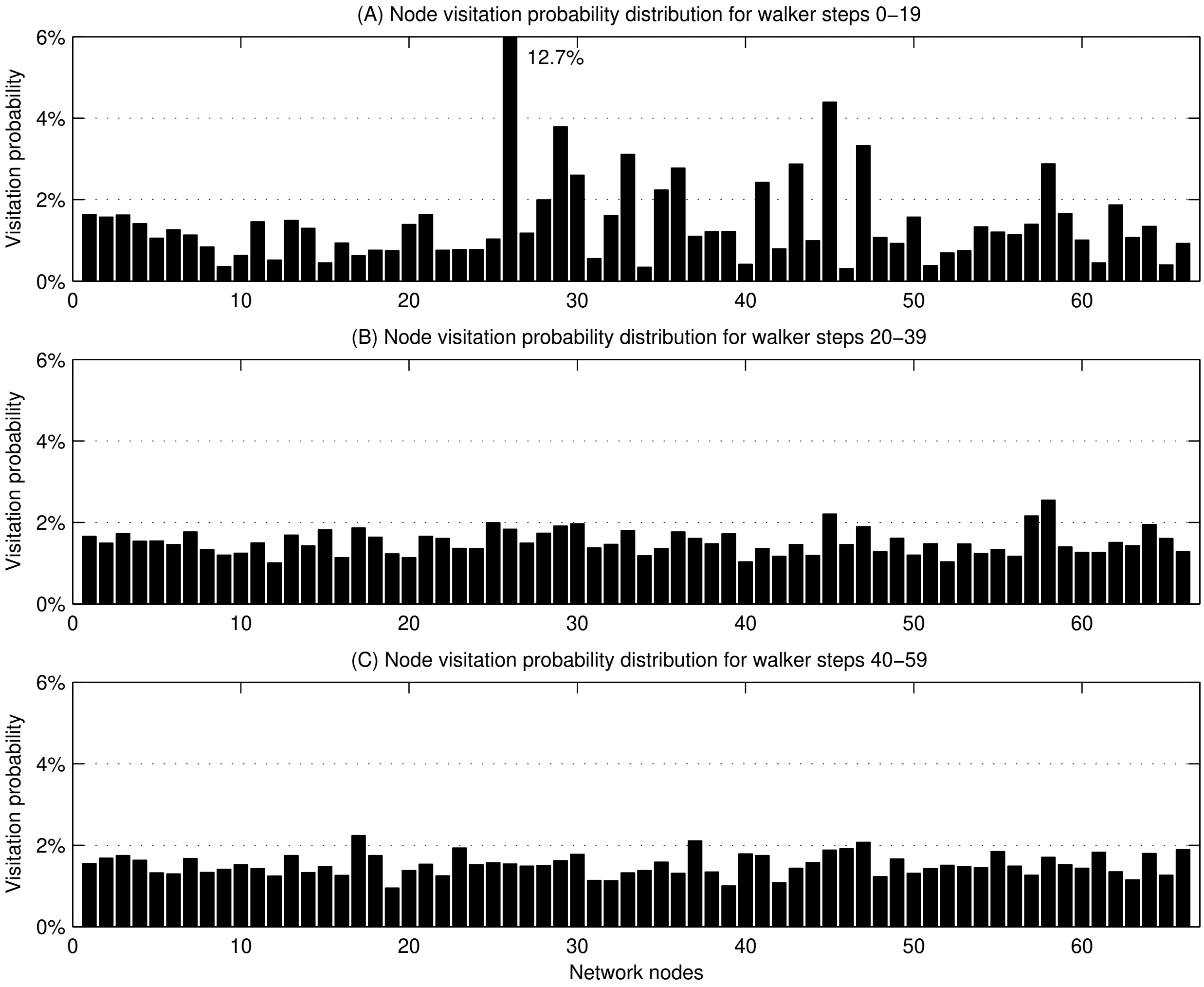

Fig. 11. Node visitation probability distributions of convergence-guaranteed random walks on a random topology with 66 Planetlab nodes. The converged distribution is the uniform distribution.

that our convergence-guaranteed random walks can quickly achieve approximately uniform node visitation probabilities (after only 20 walk steps).

VIII. Conclusion

As far as we know, this is the first work on the effectiveness and challenges in using convergence-guaranteed random walks to provide p2p systems with application-specific probabilistic node sampling service. In particular, we focus on two important issues that concern the usage of random walks in practical p2p systems: the convergence time and fault tolerance of convergence-guaranteed random walks. We present both analytical and simulation results on the random walk convergence time for different network sizes, common p2p network topologies, and various targeted node visitation probability distributions. We also derive results on random walk availability under dynamic network conditions with possible walker losses.

We evaluate the benefits of convergence-guaranteed random walks configured according to our analytical results via three p2p applications: random membership management, index-free search, and load balancing. We also implement a prototype of convergence-guaranteed random walks to assess its overhead and convergence via a Planetlab-based Internet experiment. Our results make the case that the usage of convergence-guaranteed random walks can achieve stronger analytical properties or higher system performance compared with previous topology-independent index-free approaches. Furthermore, our results and experiences (*e.g.*, choosing proper node sampling distributions, random walker configuration, bounding the convergence time, the prototype implementation) in these case studies can serve as guidance for the usage of convergence-guaranteed random walks in other p2p applications.

REFERENCES

[1] L. Adamic, B. Huberman, R. Lukose, and A. Puniyani. Search in Power Law Networks. *Physical Review*, (64):46135–46143, 2001.

[2] M. Adler, E. Halperin, R.M. Karp, and V. Vazirani. A Stochastic Process on the Hypercube with Applications to Peer-to-Peer Networks. In *Proc. of the 25th ACM Symp. on Theory of Computing (STOC)*, pages 575–584, San Diego, CA, June 2003.

[3] Y. Azar, A. Broder, A. Karlin, N. Linial, and S. Phillips. Biased Random Walks. In *Proc. of the 24th ACM Symp. on the Theory of Computing*, pages 1–9, 1992.

[4] A. Barabasi. *Linked: How Everything Is Connected to Everything Else and What It Means*. Plume, 2003.

[5] A. Barabási and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286:509–512, 1999.

[6] A.R. Bharambe, M. Agrawal, and S. Seshan. Mercury: Supporting Scalable Multi-attribute Range Queries. In *Proc. of the ACM SIGCOMM*, pages 353–366, Portland, OR, August 2004.

[7] B. Bollobás. *Random Graphs*. Academic Press, London, UK, 1985.

[8] B. Bollobás and O. Riordan. The Diameter of a Scale-free Random Graph. *Combinatorica*, 24(1):5–34, 2004.

[9] B. Bollobás, O. Riordan, J. Spencer, and G. Tusnady. The Degree Sequence of a Scale-free Random Graph Process. *Random Structures and Algorithms*, 18(3):279–290, 2001.

[10] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P Systems Scalable. In *Proc. of the ACM SIGCOMM*, August 2003.

[11] E. Cohen and S. Shenker. Replication Strategies in Unstructured Peer-to-Peer Networks. In *Proc. of the ACM SIGCOMM*, Pittsburgh, PA, August 2002.

[12] B.F. Cooper. Quickly Routing Searches Without Having to Move Content. In *Proc. of the 4th International Workshop on Peer-to-Peer Systems (IPTPS)*, Ithaca, NY, February 2005.

[13] P. Diaconis and D. Stroock. Geometric Bounds for Eigenvalues of Markov Chains. *Annals of Applied Probability*, 1:36–61, 1991.

[14] W. Doeblin. Exposé de la théorie des chaînes simples constantes de Markov á un nombre fini d'états. *Mathématique de l'Union Interbalkanique*, 2:77–105, 1938.

[15] P.Th. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight Probabilistic Broadcast. *ACM Trans. on Computer Systems*, 21(4):341–374, November 2003.

[16] A.J. Ganesh, A. Kermarrec, and L. Massoulié. SCAMP: Peer-to-peer Lightweight Membership Service for Large-scale Group Communication. In *Proc. of the 3rd International Workshop on Networked Group Communicatio*, pages 44–55, London, UK, November 2001.

[17] C. Gkantsidis, M. Mihail, and A. Saberi. Conductance and Congestion in Power Law Graphs. In *Proc. of ACM SIGMETRICS*, pages 148–159, San Diego, CA, June 2003.

[18] C. Gkantsidis, M. Mihail, and A. Saberi. Random Walks in Peer-to-peer Networks. In *Proc. of the IEEE INFOCOM*, pages 120–130, Hong Kong, China, March 2004.

[19] C. Gkantsidis, M. Mihail, and A. Saberi. Hybrid Search Schemes for Unstructured Peer-to-peer Networks. In *Proc. of the IEEE INFOCOM*, pages 1526–1537, Miami, FL, March 2005.

[20] C. Gkantsidis, M. Mihail, and E. Zegura. The Markov Chain Simulation Method for Generating Connected Power Law Random Graphs. In *Proc. 5th Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2003.

[21] W.K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57:97–109, 1970.

[22] M. Jelasity, R. Guerraoui, A. Kermarrec, and M.V. Steen. The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations. In *Proc. of the 5th International Middleware Conference*, 2004.

[23] D. Karger and M. Ruhl. Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems. In *Proc. of the 16th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 36–43, Barcelona, Spain, June 2004.

[24] V. King and J. Saia. Choosing A Random Peer. In *Proc. of the 23rd ACM Symp. on Principles of Distributed Computing*, pages 125–130, 2004.

[25] D. Kostić, A. Rodriguez, J. Albrecht, A. Bhirud, and A. Vahdat. Using Random Subsets to Build Scalable Network Services. In *Proc. of the 4th USENIX Symp. on Internet Technologies and Systems (USITS)*, Seattle, WA, March 2003.

[26] C. Law and K. Siu. Distributed Construction of Random Expander Networks. In *Proc. of the IEEE INFOCOM*, pages 2133–2143, San Francisco, CA, March 2003.

[27] D. Loguinov, A. Kumar, V. Rai, and S. Ganesh. Graph-Theoretic Analysis of Structured Peer-to-Peer Systems: Routing Distances and Fault Resilience. In *Proc. of the ACM SIGCOMM*, pages 395–406, Karlsruhe, Germany, August 2003.

[28] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *Proc. of the ACM International Conference on Supercomputing (ICS)*, pages 84–95, New York, NY, June 2002.

[29] Q. Lv, S. Ratnasamy, and S. Shenker. Can Heterogeneity Make Gnutella Scalable? In *Proc. of the First International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[30] G.S. Manku. Balanced Binary Trees for ID Management and Load Balance in Distributed Hash Tables. In *Proc. of the 23rd ACM Symp. on Principles of Distributed Computing (PODC)*, pages 197–205, St. John's, Canada, July 2004.

[31] G.S. Manku, M. Naor, and U. Wieder. Know Thy Neighbor's Neighbor: The Power of Lookahead in Randomized P2P Networks. In *Proc. of the ACM Symp. on Theory of Computing (STOC)*, 2004.

[32] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[33] Planetlab. http://www.planet-lab.org.

[34] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica. Load Balancing in Structured P2P Systems. In *Proc. of the Second Int'l Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, CA, February 2003.

[35] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proc. of the ACM SIGCOMM*, pages 161–172, San Diego, CA, August 2001.

[36] S. Rhea and J. Kubiatowicz. Probabilistic Location and Routing. In *Proc. of the IEEE INFOCOM*, pages 1248–1257, 2002.

[37] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems. In *Proc. of IFIP/ACM Middleware Conference*, pages 329–350, Heidelberg, Germany, November 2001.

[38] K. Shen. Structure Management for Scalable Overlay Service Construction. In *Proc. of the First USENIX/ACM Symp. on Networked Systems Design and Implementation (NSDI)*, pages 281–294, San Francisco, CA, March 2004.

[39] A. Sinclair. Improved Bounds for Mixing Rates of Markov Chains and Multicommodity Flow. *Combinatorics, Probability and Computing*, 1:351–370, 1992.

[40] A. Sinclair and M. Jerrum. Approximate Counting, Uniform Generation and Rapidly Mixing Markov Chains. *Information and Comput.*, 82:93–133, 1989.

[41] K. Sripanidkulchai. The popularity of Gnutella Queries and Its Implications on Scalability. In *The O'Reilly Peer-to-Peer and Web Services Conference*, 2001.

[42] A.O. Stauffer and V.C. Barbosa. Probabilistic Heuristics for Disseminating Information in Networks. *IEEE/ACM Trans. on Networking*, 15(2):425–435, April 2007.

[43] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of the ACM SIGCOMM*, pages 149–160, San Diego, CA, August 2001.

[44] D. Tsoumakos and N. Roussopoulos. Adaptive Probabilistic Search for Peer-to-Peer Networks. In *Proc. of the 3rd IEEE International Conference on P2P Computing*, 2003.

[45] X. Wang, Y. Zhang, X. Li, and D. Loguinov. On Zone-Balancing of Peer-to-Peer Networks: Analysis of Random Node Join. In *Proc. of ACM SIGMETRICS*, June 2004.
[46] B. Yang and H. Garcia-Molina. Improving Search in Peer-to-Peer Networks. In *Proc. of the 22nd ICDCS*, pages 5–14, Vienna, Austria, July 2002.
[47] M. Zhong, K. Shen, and J. Seiferas. Dynamic load balancing in unstructured peer-to-peer networks: Finding hotspots, eliminating them. Unpublished manuscript, May 2007. http://www.cs.rochester.edu/u/zhong/papers/hotspots.pdf.