Doubly Stochastic Converge: Uniform Sampling for Directed P2P Networks

Cyrus Hall and Antonio Carzaniga
Faculty of Informatics
University of Lugano
Lugano, Switzerland
hallc@lu.unisi.ch, antonio.carzaniga@unisi.ch

Technical Report 2009/02

Abstract

Uniformly sampling nodes from deployed peer-to-peer (P2P) networks has proven to be a difficult problem, as current techniques suffer from sample bias and limited applicability. A sampling service which randomly samples nodes from a uniform distribution across all members of a network offers a platform on which it is easy to construct unstructured search, data replication, and monitoring algorithms. We present an algorithm which allows for uniform random sampling, by the use of biased random walks, over the peers of any P2P network. Our algorithm, called *doubly stochastic converge* (DSC), iteratively adjusts the probabilities of crossing each link in the network during a random walk, such that the resulting transition matrix is doubly stochastic. DSC is fully decentralized and is designed to work on physically directed networks, allowing it to work on any P2P topology. Our simulations show that DSC converges quickly on a wide variety of topologies, and that the random walks needed for sampling are short for most topologies. In simulation studies with FreePastry [7], we show that DSC is resilient to extremely high levels of churn, while incurring a minimal sample bias.

1 Introduction

Our overarching goal is to create a generic, plug-and-play framework for sampling properties of arbitrary P2P networks. Given a topology of links, we want to sample some set of properties (bandwidth, load, etc.) of the connected peers. Ideally, the resulting measurements should give an unbiased view of the current distribution of the properties over the network, which is useful for immediate parameter tuning, and will hopefully lead to a correct understanding of network dynamics over multiple sample runs.

In this paper we focus on a fundamental component of such a framework. Specifically, we implement uniform sampling over a P2P system of an arbitrary topology. This amounts to choosing a peer at random from the uniform distribution over all peers in a

network with an unknown link distribution. In addition to serving as a basis for monitoring, uniform random sampling is a useful building block in distributed systems, where it is used to support search, maintenance, replication, and load-balancing [9, 17, 24].

Existing techniques for uniform sampling rely on biased random walks: a node is selected at the end of a sufficiently long random walk in which the probabilities of following each edge (the transition probabilities) are adjusted to obtain a uniform visitation distribution across the network. Current algorithms, such as Metropolis-Hastings and maximum-degree, use the ratio between a node and its neighbors' link-table size to compute new weights for the transition probabilities of each link, and generate doubly stochastic transition matrices [2, 6, 24]. These algorithms are able to do this because the ratio of link-table sizes is directly proportional to the nodes' visitation probabilities, which is a local property in undirected topologies.

Others have suggested using gossip-based peer-sampling techniques [13]. While often useful for implementing services associated with sampling, such as search, load-balancing, and peer membership, we find that gossip peer-sampling does not offer an appropriate interface for other sampling related tasks such as monitoring. In particular, Gossip-based services do not provide sampling that is *independent and identically distributed*, and do not handle bursts of sampling well, both properties we desire.

Our contribution is to remove the assumption of bidirectional links in the network, so called *undirected topologies*. Existing algorithms are only efficient in such an environment, restricting the design space of the underlying network, and forcing the maintenance of bidirectional links in an Internet where NAT and middle-boxes are common. P2P topologies which do not need bidirectional links for every connection, i.e. *directional topologies*, can therefore benefit from a sampling algorithm that does not need them either. An algorithm that works well in the absence of bidirectional links, yet can utilize them to improve performance when they are available, is clearly ideal.

The algorithm we propose is fully distributed. The main idea is to avoid the calculation of each node's visitation probability, and instead to adjust link transition probabilities iteratively, converging to a state in which the sum of input probabilities at each node is equal. The resulting transition matrix is said to be *doubly stochastic*, and induces uniform visitation probabilities with sufficiently long random walks. We find that the algorithm performs well on both static and dynamic topologies, and is able to keep the ratio of most and least likely to be sampled node below 1.2 for realistic churn conditions. Further, our algorithm generates link-biases that keep the expected sample walk length reasonably short, between 20-35 hops for 100 node static topologies, and around 23 hops for 1000 node Pastry networks.

In Section 2 we review the relevant background necessary to explain our algorithm and related work. Section 3 reviews previous work on P2P sampling. Section 4 presents a basic version of our *doubly stochastic converge* algorithm (*DSC*) and a proof of its convergence. A more advanced variant that reduces the length of the random walks and deals with failure is then presented. Section 5 evaluates DSC in both simulations on static topologies, and with FreePastry [7]. Finally, Section 6 concludes with a discussion of future work.

2 Background Theory

We briefly discuss Markov Chains and their properties in order to introduce concepts used in the review of related work and then the presentation and analysis of our algorithm. For an excellent introduction to both general and ergodic Markov chains, see Grinstead and Snell [10], and Norris [19].

2.1 Basic Definitions

We model a P2P topology as a labeled directed graph $G = (V, E)$ of $N = |V|$ vertices. Each edge $(u, v) \in E$ is labelled with a probability $p_{uv} \in [0, 1]$. For any given vertex u, p_{uv} is the probability of transitioning along the edge (u, v), with $p_{uv} = 0$ if $(u, v) \notin E$, and $\sum_{v \in V} p_{uv} = 1$. We form a stochastic $N \times N$ matrix $\mathbf{P}$, where $\mathbf{P}(u, v) = p_{uv}$, representing the transition matrix of the Markov chain across the vertices of G.

Finally, we define two sets that will be useful throughout the paper: the predecessors of a node u, $N(u) \triangleq \{v \in V : (v, u) \in E \wedge v \neq u\}$, and the successors, $S(u) \triangleq \{v \in V : (u, v) \in E \wedge v \neq u\}$. It will prove convenient to exclude u from being a predecessor or successor of itself.

2.2 Stationary Distribution

Given a probability distribution $\mathbf{x}$ across the states in the Markov chain described by $\mathbf{P}$, one can calculate the successive distribution after an additional step in a walk across the chain, with $\mathbf{x}_{t+1} = \mathbf{x}_t\mathbf{P}$. If the Markov chain is ergodic, the probability to be at a given node u after a sufficiently long walk stabilizes to a unique *stationary distribution* written as $\boldsymbol{\pi}$. A Markov chain is ergodic if it is both irreducible and aperiodic, such that there exists a time q for which $\forall u \forall v \in V, \forall n \geq 0 : \mathbf{P}^{q+n}(u, v) > 0$. The *mixing-time* of $\mathbf{P}$ is the minimal length of a walk that achieves the desired level of uniformity, a topic addressed in Section 5.1.

A P2P topology that is both strongly connected and aperiodic—reasonable assumptions—will have an ergodic Markov chain. In such a topology, the value of the stationary distribution for a given node u, $\boldsymbol{\pi}(u)$, gives the probability to be at u after a random walk of a sufficient length, independent of the starting node. Our goal is to achieve a *uniform* stationary distribution.

2.3 Uniform Stationary Distributions

One way to assure that an ergodic Markov chain has a uniform stationary distribution is to set the p_{uv} values in its transition matrix $\mathbf{P}$ in such a way that the sum of transition probabilities over the incoming edges is 1 for all vertices. Thus, $\sum_{u \in V} p_{uv} = 1$ for all v. In this case $\mathbf{P}$ is *doubly-stochastic*, and the Markov chain has a uniform stationary distribution. Defining $\mathbf{1}$ as the row vector $[1, \ldots, 1]$ of dimension N:

Theorem 2.1. *Let $\boldsymbol{P}$ be a doubly stochastic transition matrix, and $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_t, \ldots$ be a sequence where $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t\boldsymbol{P}$. Then as $t \to \infty$, $\boldsymbol{x}_t \to \boldsymbol{\pi} = \frac{1}{N}\boldsymbol{1}$.*

Proof. Assume $\mathbf{x}_t = \frac{1}{N}\mathbf{1}$ is a probability distribution across the vertices of G. Since $\mathbf{x}_{t+1} = \mathbf{x}_t\mathbf{P}$, the following holds $\forall v \in V$:

$$\mathbf{x}_{t+1}(v) = \sum_{u \in V} \mathbf{x}_t(u) p_{uv} = \frac{1}{N} \sum_{u \in V} p_{uv} = \frac{1}{N}$$

The last reduction comes from the fact that each column sums to 1. Therefore, the value at each index of $\mathbf{x}_{t+1}$ equals $1/N$, and the distribution is stationary. Since the Fundamental Limit Theorem of Markov Chains states there is a unique stationary distribution for any given matrix [19], $\boldsymbol{\pi} = \frac{1}{N}\mathbf{1}$ is the sole stationary distribution for all doubly stochastic matrices, and any initial $\mathbf{x}_0$ must converge to $\boldsymbol{\pi}$. □

Snell offers another proof of this theorem [22]. From Theorem 2.1, we set the goal of DSC to bias the transition probability of edges in an ergodic graph G such that the transition matrix $\mathbf{P}'$ representing the transformed graph G' is doubly stochastic.

We note that the properties discussed in this section assume a *static* topology, or at least a topology that changes so infrequently that the random walks used for sampling are never affected by churn. However, such conditions are not likely to exist in real P2P networks. Therefore, we make the simplifying assumption that churn events affect the network in a fairly localized manner, and that the stationary distribution does not change radically after most churn events. Indeed, we show in Section 5.2 that DSC quickly corrects for high rates of churn and that sampling is largely unaffected.

3 Related Work

P2P researchers have recently looked at sampling generic undirected structures, but the suggested techniques do not work for directed topologies. Several topology specific methods for directed networks have been suggested, but we are interested in a *generic* solution to the problem.

3.1 Undirected Topologies

There are two broad classes of undirected sampling techniques. The first uses random walks to sample nodes, while the second uses local caches of node identifiers that are exchanged and shuffled via a gossip protocol.

Most previous work on uniform sampling in P2P networks has focused on using random walks in undirected topologies. While many P2P networks do not actively maintain bidirectional connections, the assumption that such connections exist, or can be created on demand, simplifies biasing the transition probabilities on links. This is because the probability of selecting a node u, after a significantly long walk, in the Markov chain representing an undirected topology is a local topological property, $\boldsymbol{\pi}(u) = \frac{\mathcal{D}(u)}{2|E|}$, where $\mathcal{D}(u) = |\{v \in V : (u, v) \in E\}|$.

Most existing techniques utilize this property and work in two steps. First, a node u distributes it's edge count $d(u)$ to all neighbors. Then, after receiving edge counts from all adjacent neighbors, each node biases all edges appropriately [2, 6, 24, 25]. The most common bias technique uses the Metropolis-Hastings algorithm. Assuming each

node initially assigns a uniform transition probability to adjacent neighbors, Metropolis-Hastings compares the degree of adjacent nodes and then reduces the probability of transition across (u, v) if $\frac{\mathcal{D}(u)}{\mathcal{D}(v)} < 1$, with $p'_{uv} = \frac{\mathcal{D}(u)}{\mathcal{D}(v)} p_{uv}$. Any lowering of transition probability to neighbors is balanced by adding the remaining probability to a *self-loop*, (u, u), assuring in- and out-probabilities sum to 1.

Another related set of techniques are gossip protocols, particularly peer-sampling and membership-management protocols [13]. Gossip techniques have proven useful in topology construction and maintenance [12], and maintaining estimates of global properties [14]. Gossip-based peer-sampling works by mixing sets of peer references randomly via periodic exchange between neighboring peers, and returning nodes from the local peer cache when a sample is requested. However, the interface offered by such protocols is inappropriate for a general sampling service. For example, when monitoring a P2P network, we desire to understand the current state of the network, a task which can occasionally require large numbers of near-simultaneous samples. In order to support such sampling, a gossip-based scheme would need to keep a large local peer cache, yet could not significantly lower the exchange interval unless churn was low. Further, only the first sample operation from the gossip-algorithm peer cache is independent and identically distributed, often a desirable or necessary property. Further samples are returned from distributions that deviate further and further from the expected.

3.2 Directed Topologies

While we know of no other general algorithm for sampling directed P2P networks, there are algorithms to sample classes of topologies. King and Saia present a method to select an uniformly random peer from any DHT that supports the standard lookup interface, where $get(x)$ returns the closest peer to id x [15]. Similarly, there are various general techniques to sample web-sites during web-crawls, and from search-engines via random queries [3, 11]. Such techniques typically transform the directed graph of web-links into an undirected graph by adding reverse links, and then use techniques such as Metropolis-Hastings to sample uniformly.

Studies of churn dynamics in DHTs often make the assumption that random peers can be sampled by looking up randomly selected ids in the address space [4]. However, many DHTs show a preference toward storing long-lived nodes in routing tables in order to increase reliability, which biases the selection toward such nodes. Further, random differences in the distribution of IDs biases toward peers responsible for larger amounts of ID-space. In general, any technique which samples peers in a manner correlated with session length will lead to biased results [23, 24].

In general, our vision is to create a *topology-agnostic* solution for sampling directed P2P networks. This would make studying and maintaining current and new networks simple and less error prone. Further, the basic version of our algorithm only assumes it can communicate in the same direction as the logical links in the host P2P topology, making it ideal for situations when reverse packet flows are expensive or undesirable. When reverse flows are available, we use them to optimize performance. Although not covered in this paper, We have also designed DSC so that it can be used in a *opportunistic* manner, such that by piggy-backing on existing traffic it generates no or minimal extra packets.

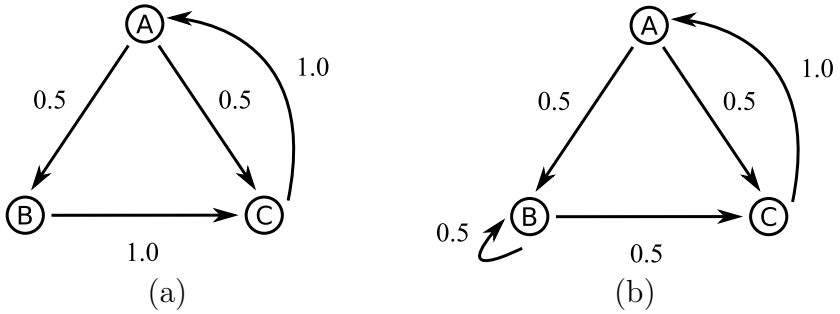

Figure 1: An example ergodic graph. (a) Labeled with initial transition probabilities. (b) Balanced using a self-loop.

4 The DSC Algorithm

One can uniformly sample a network by first assigning transition probabilities to edges so that the stationary distribution is uniform, and then performing random walks of a sufficient lengths, bounded by the mixing-time. In this section we discuss an algorithm that solves the first of these requirements.

4.1 Basic DSC

As stated in Section 2.1, $\mathbf{P}$ is stochastic by definition, as each node's outgoing edges have a total probability of 1. Our task is to assign those probabilities so that, for each node, the probabilities of the incoming edges also add up to 1, thereby transforming $\mathbf{P}$ to be doubly stochastic. We introduce several constraints on the transformation of $\mathbf{P}$. First, the transformation from $\mathbf{P}$ to $\mathbf{P}'$ should be non-destructive, such that if an edge existed in $\mathbf{P}$, it will also exist in $\mathbf{P}'$. Second, we limit the creation of new edges to self-loops, ensuring that any existing constraints on network packet flows are not violated. We refer to the entire process as *balancing*.

In a fully-decentralized balancing algorithm, a node v can not directly control the probabilities assigned to its incoming edges, which are controlled by v's predecessors. In fact, v may only know about a subset or even none of its predecessors. Even if v could control the probability of some of its incoming edges, changing one of these probabilities would affect the balance of probabilities for other nodes. To gain some degrees of freedom for the purpose of balancing, we introduce an extra edge, the *self-loop*, linking each vertex back to itself. The self-loop is both an incoming and outgoing edge, and therefore can be used by a node to make up deficits in its own incoming probability.

Figure 1(a) exemplifies the difficulties in balancing, and the use of the self-loop. Neither node B or C can be balanced without removing edge (A, C). B has a deficit of incoming probability, yet A cannot rebalance its out-probability to satisfy B without removing all probability from (A, C). Conversely, C has too much incoming probability, and B has no other edge on which to place excess probability. Setting $p_{AC} = 0$ balances all in- and out-probabilities, but the graph becomes periodic, and hence no longer ergodic.

In Figure 1(b), we use the self-loop (B, B) to increase the in-probability of B to 1, inadvertently reducing the in-probability of C to 1, and balancing $\mathbf{P}$ and keeping G ergodic. This leads directly to the core intuition of DSC: *increasing* the self-loop for nodes

with in-probability deficits, and therefore reducing the probability across their outgoing edges, *decreases* the excess in-probability of other nodes.

More formally, we define the sum of in- and out-probability at a vertex v as

$$In(v) \triangleq p_{vv} + \sum_{u \in N(v)} p_{uv}; \quad Out(v) \triangleq p_{vv} + \sum_{u \in S(v)} p_{vu}$$

where p_{vv} is the self loop. Both $In(v)$ and $Out(v)$ must sum to 1 in order for $\mathbf{P}$ to be doubly stochastic. Clearly, increasing or decreasing the self-loop forces v to decrease or increase, respectively, the sum of probability across both $N(v)$ and $S(v)$.

At any given time t, vertices are in one of three states:

$$V^+ \triangleq \{v \in V : In(v) > 1\}$$
$$V^= \triangleq \{v \in V : In(v) = 1\}$$
$$V^- \triangleq \{v \in V : In(v) < 1\}$$

The total amount of surplus in-probability of $v \in V^+$ equals the total deficit of in-probability of $v \in V^-$. This stems from the fact that $\forall v \in V$, $Out(v) = 1$, and therefore the mean of $In(v)$ must equal 1. Any in-probability over 1 at a particular vertex v must result in a deficit at some other vertex u, and vice-versa.

If we can move vertices from V^- to $V^=$, we will also force vertices from V^+ into $V^=$, leading to $V^= = V$ and $\mathbf{P}$ to become doubly stochastic. Promoting vertices from V^- to $V^=$ is extremely simple: increase the self-loop p_{vv} by $1 - In(v)$, bringing $In(v) = 1$. This is the strategy of the basic version of DSC, with each node $v \in V$ executing the following steps:

1. Every t seconds, v updates its self-loop p_{vv}. When $In(v) < 1$, the self-loop is increased by $1 - In(v)$. If $In(v) \geq 1$ no action is taken.

2. Every t/f seconds, v sends updates to all $u \in S(v)$, notifying them of their current transition probability, $(1 - p_{vv})/|S(v)|$. Successor nodes store this value, using it to calculate $In(v)$ in step 1.

When step 1 is executed we say there has been an *update*. The time t should be selected with bandwidth and network latency in mind, with a short t leading to faster convergence, but leading to missed updates if latency is high. The frequency of updates, f, can be set higher when packet loss or latency is a problem, or to 1 if on-time delivery is assured.

This basic version of DSC sends evenly balanced probabilities in step 2. In most situations this is not optimal, and we discuss a change to the algorithm which generally performs better in Section 4.3. However, in situations where receiving feedback from nodes in $S(v)$ is not possible, or it is inconvenient to do so in a timely manner, the basic version of DSC will still converge quickly in practice, if to a less optimal state.

4.2 Convergence of Basic DSC

We now prove that basic DSC changes the transition probabilities in any network topology such that they converge to a uniform stationary distribution. It accomplishes this by iteratively reducing the distance between $\mathbf{P}$ and a doubly stochastic matrix.

Theorem 4.1. *If G is an ergodic graph, DSC updates $\boldsymbol{P}$ iteratively in such a way that, in the limit, $\boldsymbol{P}$ converges toward doubly stochastic.*

Proof. Let $D(\mathbf{P})$ measure a distance form $\mathbf{P}$ to a doubly stochastic matrix as follows:

$$D(\mathbf{P}) \triangleq \sum_{v \in V} d(v), \text{ where } d(v) = |1 - In(v)|$$

We show that, after i iterations of the DSC algorithm, $D(\mathbf{P})$ must decrease by a factor of $(1-\epsilon)^i$ where $\epsilon > 0$ and ϵ does not depend on i. Therefore, $\lim_{i \to \infty} D(\mathbf{P}_i) = 0$ and $\mathbf{P}$ becomes doubly stochastic.

Let $D(\mathbf{P}_0) > 0$ be the initial distance, and therefore $|V^+| \geq 1 \wedge |V^-| \geq 1$. Let $u \in V^-$ be the node with the maximal in-probability deficit $d(u) = 1 - In(u)$, and consider the execution of DSC on u. The effect of u updating its self-loop is to spread u's deficit onto its successors' in-probabilities. A successor node $q \in S(u)$ can be in one of the following states:

- $q \in V^=$ or $q \in V^-$: in these cases, q has to completely relay u's deficit (along with its own) to its successors. There is no net reduction in $D(\mathbf{P})$.
- $q \in V^+$: in this case some or all of the deficit passed from u to q will be absorbed by q's surplus, with a corresponding net reduction in $D(\mathbf{P})$.

As DSC updates continue, and u's deficit is relayed two and more hops away from u, more of that deficit may be absorbed by other nodes in V^+. However, in order to find a minimum reduction in deficit (and therefore distance) in a bounded amount of steps, we focus on the node $w \in V^+$ with the maximal in-probability surplus $s(w)$, and we determine the minimum amount of $d(u)$ that gets absorbed by $s(w)$, and in how many steps. To do that, we focus on the shortest path from u to w.

Since G is ergodic, the length of the shortest path from u to w is at most $N-1$. At each hop along that path, the deficit is evenly divided among the all successors, and therefore is progressively reduced. Since at each hop there are at most $N-2$ successors, the deficit that reaches w must be at least $d(u)/(N-2)^{N-1}$, which can be absorbed completely or up to $s(w)$. Therefore, after $N-1$ iterations, the overall distance $D(\mathbf{P})$ is reduced by at least $\min(d(u)/(N-2)^{N-1}, s(w))$.

Now, since $d(u)$ is the maximal deficit, and since there are at most $N-1$ nodes in V^-, it must be that $d(u) \geq D(\mathbf{P}_0)\frac{1}{2(N-1)}$. For the same reason, $s(w) \geq D(\mathbf{P}_0)\frac{1}{2(N-1)}$. This gives us a lower bound for the reduction of $D(\mathbf{P})$ after $N-1$ iterations

$$D(\mathbf{P}_{N-1}) \leq D(\mathbf{P}_0)\,(1-\epsilon) \quad \text{where}$$
$$\epsilon = \min\left(\frac{1}{2(N-1)(N-2)^{N-1}}, \frac{1}{2(N-1)}\right)$$

This means that $D(\mathbf{P}_i)$ decreases after a fixed number of steps $(N-1)$ by at least a factor $1-\epsilon$ that does not depend on time. Therefore $D(\mathbf{P}_i)$ converges exponentially to 0. □

This is an extremely loose bound, and convergence is much faster in practice. Further, the ordering of node updates can play a role in determining the rate of convergence. For

example, in Figure 1, updating A and C first is clearly not optimal. We have observed that different update ordering may change the rate of convergence by a factor of 2 for large topologies.

While the basic version of the algorithm works, it has several undesirable properties. We work on fixing these by introducing feedback and relaxation.

4.3 Feedback

Basic DSC does not use information about the state of a node's successors. This can have the effect of slowing convergence and, more seriously, increasing the mixing-time. Figure 1(b) provides an example of the second of these effects, as the naive balancing leads to a large self-loop for node B. If A was aware that C was in V^+, it could weight edges (A, B) and (A, C) more appropriately, for example, $p_{AB} = 2/3$, $p_{AC} = 1/3$. Convergence can be slowed for similar reasons. For example, a node in V^+ can wait for reductions of in-probability that could be made immediately if their predecessors were aware of their state.

As the name suggests, feedback attempts to intelligently weight the probability on edges using feedback sent by successors. In addition to storing the transition probability sent by a predecessor v, a node u now replies with an acknowledgement containing $In(u)$. v then uses this value to bias p_{vu} and rebalance all transition probabilities. This happens in two steps:

(1) $p'_{vu} = p_{vu}/In(u)$
(2) $\forall q \in S(v) : p'_{vq} = (p_{vq}/\sum_{q \in S(v)} p_{vq})(1 - p_{vv})$

The weighting is only applied once every t seconds, no matter how high f may be, so that lossy links are not inappropriately biased against.

4.4 Failure and Relaxation

Nodes joining or leaving the network both have the effect of increasing self-loops. Any node in $V^=$ or in V^- can be driven back into V^+ by the addition of a new predecessor. Such nodes find themselves in the peculiar state of having a self-loop greater than 0, but also an in-probability greater than 1, an impossible state without churn.

Nodes leaving the network have a similar effect. Predecessors of a leaving node u will increase their probabilities across remaining out-edges, while successors will have to increase their self-loop to make up for lost in-probability. Just as when a node joins the network, the node u leaving can force some $v \in S(q) : q \in N(u)$ back into V^+ while having a non-zero self-loop.

Taken together, these two phenomena paint a picture of ever increasing self-loops. Clearly, if we aim to keep a reasonable mixing-time in a network with churn, we cannot rely on an algorithm that only increases self-loops. We therefore allow DSC to also lower self-loops. In particular, when a node $v \in V^+ \wedge p_{vv} > 0$, p_{vv} is decreased by $\max(-p_{vv}, 1 - In(v))$. In other words, the value of the self-loop is lowered to 0 if possible, or as close to 0 if not.

Clearly the transition probability across (v, u), $u \in S(v)$ must be raised proportionally. In turn, this can cause $q \in S(u)$ to also lower their self-loop, and so on. However, the

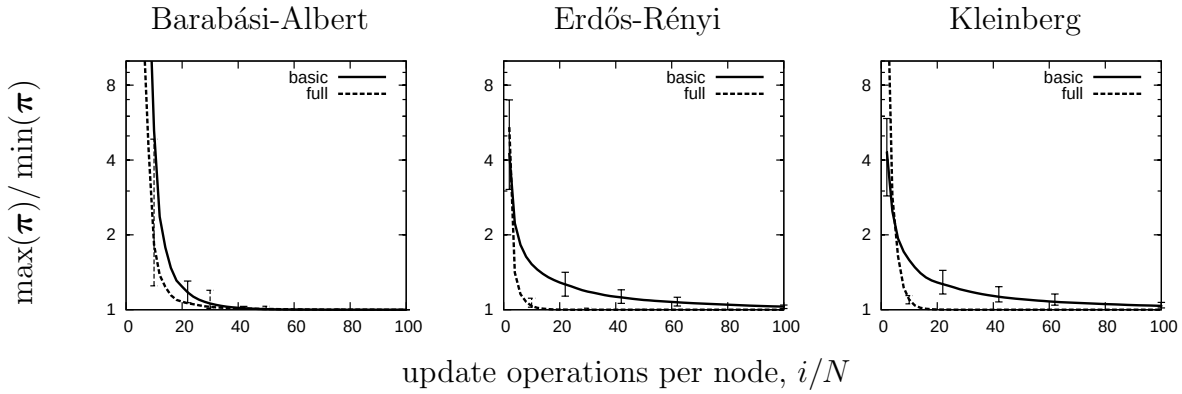

Figure 2: Maximum ratio in visitation probability r after i/N updates. The median, 10th, and 90th percentile are given.

effect is dampened each successive step away from v, and is unlikely to propagate to the entire network, as small increases in probability are absorbed by nodes in V^- or $V^=$ with non-zero self-loops.

Relaxation is essential to the current function of DSC in a network experiencing churn, with or without feedback. Without it, self-loops will approach 1, and the probability to transition to other nodes will approach 0, leading sample walk lengths to approach ∞. Unlike feedback, relaxation does not necessitate a channel of communication in the opposite direction of the network links, allowing it to work with either basic or full DSC.

5 Simulation Analysis

In order to both ensure correctness, and better understand its properties under ideal conditions, we ran DSC across a wide range of static topologies commonly used to model different types of P2P networks. We then studied the dynamics of DSC under churn using a concrete implementation of DSC. This implementation, which includes feedback and relaxation, is done in Java and is integrated with FreePastry [7], a widely used open source implementation of Pastry [20]. We find that DSC improves the uniformity of the stationary distribution in FreePastry networks under churn often by several orders of magnitude, using as little as 1 packet per link per second.

When studying both the static and dynamic topologies, we focus on how peers are sampled, and claim that this is equivalent to studying how a given property is sampled. To sample properties instead of peers, one would perform one extra action beyond sampling the peer-id, and sample the property at the same time. Importantly, we allow for the same node to be sampled multiple times, allowing for an accurate description of properties. For a discussion of accurate property sampling, see Stutzbach et. al. [24].

5.1 Static Topologies

In order to confirm that DSC works on a large range of possible network topologies, we simulate it over a diverse set of statically generated graphs. In particular, we look at three types of topologies:

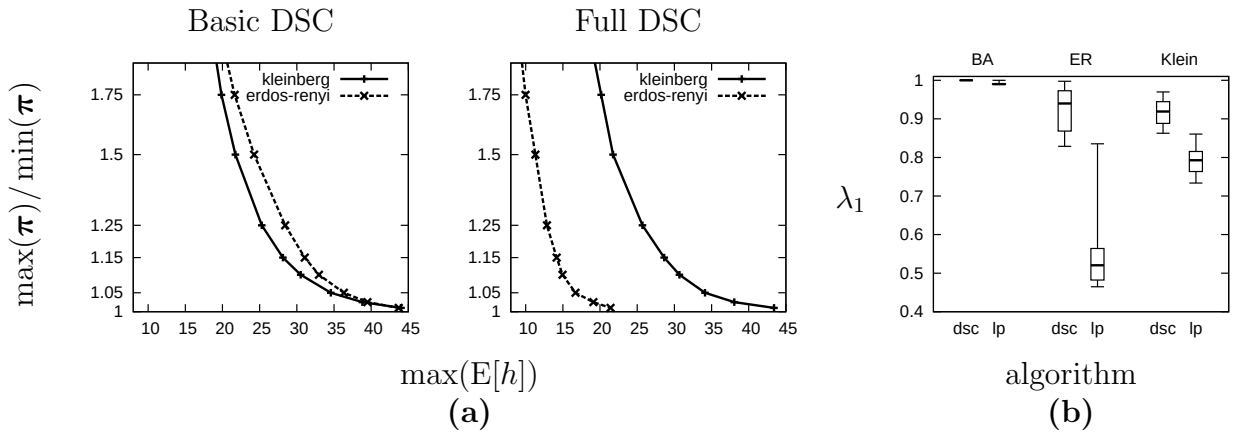

Figure 3: **(a)** Maximum expected hop count for different convergence levels. **(b)** Second eigenvalue for two different algorithms: DSC, and a linear programming model (lp), given for each topology type (BA = Barabási-Albert, ER = Erdős-Rényi, and Klein = Kleinberg). The minimum, 10th percentile, median, 90th percentile, and maximum are plotted.

- **Kleinberg**: Each node is connected in a grid to neighbors at lattice distance $p = 1$, along with q long distance links, each chosen with probability inversely proportional to the squared distance.
- **Erdős-Rényi**: Edges are selected independently with a given probability p.
- **Barabási-Albert**: Vertices are connected preferentially with high-degree vertices, creating topologies with power-law degree distributions. We use a preferential attachment factor of 1/2.

Both the Kleinberg and Barabási-Albert fall under the category of small-world topologies, which are often cited as modeling current P2P networks, as well as offering a promising source for future topological structures. For more information on these models, see the excellent survey by Albert and Barabási [1] and Kleinberg's description of his model [16]. We have used directed variations of the standard constructions, and confirmed that all graphs are strongly connected. All generated graphs have (as close as possible) the same number of vertices $N = 100$ and edges $|E| = N \log N$. This is a similar node-edge ratio to what is seen in many deployed systems. We run two versions of the algorithm: basic DSC and full DSC, which includes feedback and relaxation. We analyze the state of the network after every round of N updates.

5.1.1 Convergence

Figure 2 shows the convergence, measured by the *maximum ratio in* $\boldsymbol{\pi}$, $r = \max(\boldsymbol{\pi})/\min(\boldsymbol{\pi})$. This metric does not fully characterize the stationary distribution $\boldsymbol{\pi}$. Rather, it is intended to characterize the worst-case difference in sampling probability independently of the network size.

We display i/N on the x-axis, where i is the number of updates completed, emphasizing that updates happen in parallel. We ran simulations over 50 topologies of each type.

Figure 2 shows the median maximum ratio r along with its 10^{th} and 90^{th} percentile values (error-bars). In all situations, DSC converges exponentially fast to a uniform distribution, with a maximum ratio of $r = 1$.

For Erdős-Rényi and Kleinberg topologies, full DSC outperforms basic DSC in terms of speed of convergence, reaching an extremely tight ratio in $\boldsymbol{\pi}$ of less than 1.00001 in slightly under 3600 total updates. Since updates are performed in parallel, this is equivalent to each node updating 36 times, easy to accomplish in under a minute if each node in a P2P network updates once every second.

Barabási-Albert topologies have rather different behavior. Full DSC converges faster for the first 46 update rounds, but then basic DSC then moves ahead, until finally converging somewhat higher than full DSC. We are not completely satisfied with our understanding of this behavior, but at least some of the effect is from in-probabilities oscillating on low in-degree nodes when using full DSC: a node u with low in-degree may see its total in-probability bounce between $In(u) < 1$ and $In(u) > 1$ at each update round, causing its own out-probability to also oscillate, which in turn may cause downstream peers to also suffer from the same behavior. This effect is dampened by the feedback function each update round, but the behavior clearly indicates there is improvement to be made for topologies that exhibit small in-degrees. One possibility is to include the estimated in-degree during feedback, suppressing the rebalancing if successors are found to be vulnerable to oscillations.

5.1.2 Expected Walk Length

While a uniform stationary distribution allows one to sample uniformly, it does not guarantee that such sampling will be cheap. This is a property of the mixing-time, the speed at which a random walk becomes independent of its starting location. We studied the mixing-time by randomly choosing a starting node and multiplying the initial state distribution $\mathbf{x}_0$ repeatedly with a converged $\mathbf{P}$ generated with DSC. This simulates running an infinite number of walks starting at the randomly selected node. After each multiplication, $\mathbf{x}_t = \mathbf{x}_{t-1}\mathbf{P}$, we compute r and the expected hop count, $\mathrm{E}[h] = t(1 - y)$, where t is the current step and y is the relative weight of all the self-loops in the topology, $y = \sum_{v \in V} p_{vv}/N$.

We use twenty random starting locations for each topology, and measure the maximum expected walk length $\max(\mathrm{E}[h])$ over the ten runs. We do this for each of the 50 topologies used to study convergence, and for a set of target average values r. Specifically, we create a set of target average values $\{r_0, \ldots, r_n\}$, and then iterate the matrix multiplication until the distribution satisfies each target.

Figure 3(a) shows the results of this analysis. Full DSC performs as well or better than basic DSC for both Erdős-Rényi and Kleinberg topologies, with Erdős-Rényi performing significantly better. Reasonable values of r are achievable in less than 28 hops for either topology for full DSC. We discuss ways to improve this length below, and note that it is topology dependant, with full DSC unable to reduce self-loops in the highly regular Kleinberg topologies.

Barabási-Albert topologies are not graphed. Basic DSC can see $\max(\mathrm{E}[h])$ approach over 10^7 for $r = 1.05$. All of our Barabási-Albert topologies have several extremely weakly connected nodes with very low initial visitation probability, typically $< 10^{-6}$, which then

develop extremely high self-loops (often $p_{uu} > 0.995$).

5.1.3 Walk Length Optimality

To try and understand how far from optimal DSC is in terms of walk length, we cast the approximate optimal solution as a linear programming problem. Our objective function minimizes the self-loops, while trying to balance out-going probability evenly across out-going edges:

$$\begin{aligned} \textbf{Minimize} \quad & w_0 \sum_{u \in V} p_{uu} + w_1 \sum_{u \in V} \max(|p_{uv} - p_{uw}|) \\ \textbf{s.t.} \quad & \forall u \in V : \sum_{v \in V} p_{uv} = 1 \\ & \forall u \in V : \sum_{v \in V} p_{vu} = 1 \end{aligned}$$

The second term in the objective function is linearized with an auxiliary variable. We do not list other, less important constraints due to lack of space. w_0 and w_1 were varied so as to obtain the best solutions over all topologies, with $w_0 = 1.0, w_1 = 0.5$ proving to be a good weighting. Since minimizing the mixing time is a non-linear problem, we expect the linear-programming solution to be sub-optimal. However, for many of the Erdős-Rényi, and a few of the Kleinberg topologies, the linear-programming solver found solutions with no self-loops, leading us to believe they are likely close to optimal.

Figure 3(b) compares DSC's ability to minimize walk length with the linear-programming solution, and graphs the second-largest eigenvalue λ_1 of $\mathbf{P}$. The second-largest eigenvalue modulus bounds the mixing time by $O(\frac{1}{1-\lambda_1})$, and therefore the walk length, of an ergodic graph [21].

It is clear from Figure 3(b) that there is room for improvement in the way DSC assigns probability across edges. This isn't particularly surprising, as we have not dedicated much time to minimizing self-loops. It should be easy to iteratively improve DSC's weightings by integrating more information into feedback, with the goal of moving transition probability from high to low in-probability neighbors [2].

5.2 FreePastry Simulations

Pastry [20] is a fault tolerant, locality aware DHT. Each node in a Pastry network random chooses a id in an 128-bit identifier space when joining the network, and then proceeds to bootstrap both a *leaf-set* of the closest l neighbors in both directions of the id-space, and a *route table* containing $\lceil \log_{2^b} N \rceil$ rows with $(2^b - 1)$ entries each, where b is the base of the network. Each successive row in the table halves the distance in the id-space, allowing for greedy routing of node lookups in $O(\log_{2^b} N)$ hops.

Due to processing and time constraints, we use $N = 100, 1000$ node Pastry networks for our simulations. In each simulation run there are two phases: first, nodes join the network as fast as possible. As soon as the last node has joined, a churn process is started. Our churn model closely follows the findings of Stutzbach and Rejaie and their study of a DHT based on Kademlia [23, 18]. In particular, we model node session time as random variates of a Weibull distribution, with the shape parameter $k_s = 0.4$ and the scale

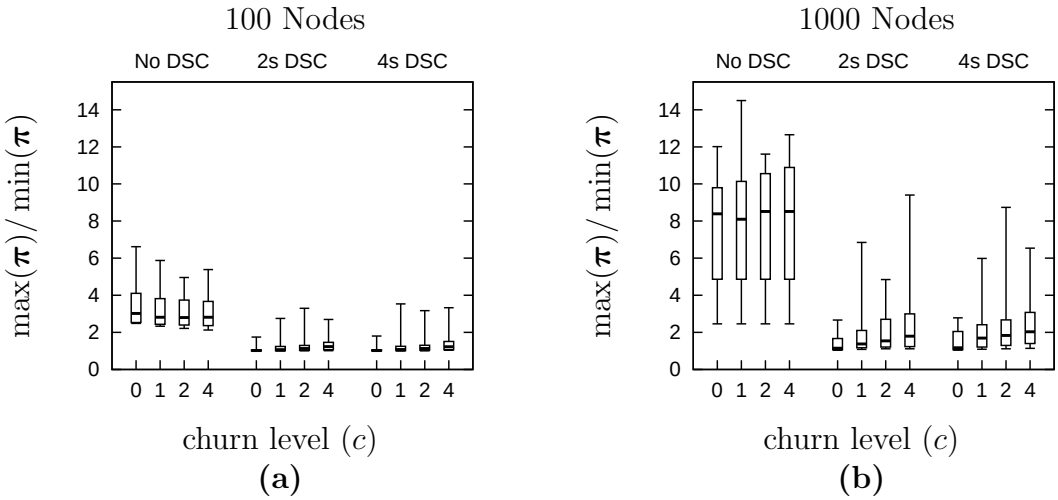

Figure 4: Maximal relative difference in visitation probability: min, 10th percentile, median, 90th percentile, and max. The four columns in each setting of DSC represent different churn levels, 0, 1, 2, and 4, respectively, from left-to-right.

$\lambda_s = 30/c$. We refer to c as the *churn parameter*; the higher c, the shorter the average session length. Node inter-arrival time is modeled by a second Weibull distribution, with $k_a = 0.65, \lambda_a = \mu_s/(N\Gamma(1 + \frac{1}{k_a}))$, where μ_s is the mean session time, and $\Gamma(x)$ is the Gamma function. Both distributions model time in minutes.

5.2.1 Convergence with Pastry

The graphs in Figure 4 show the maximum ratio $\max(\boldsymbol{\pi})/\min(\boldsymbol{\pi})$ for different levels of churn of FreePastry topologies. Different values of c control the scale parameter of the inter-arrival time and mean session length, creating various levels of churn. Networks with value $c = 1.0$ have very similar dynamics to those measured by Stutzbach and Rejaie [23]. Samples of r are only collected after the fast join period is complete, as during this period r is much higher than would be seen in a normal network.

Our networks are quite small compared to the size of some currently deployed DHTs. This small size means that there is significant overlap between the FreePastry leaf-set and routing table at each node, particularly for $N = 100$. Our implementation of DSC only counts each logical directed link once, but most nodes have similar in-degrees because of the overlap. Nevertheless, we find in Figure 4(a) that 100-node Pastry networks exhibit a maximum sampling ratio in $\boldsymbol{\pi}$ of up to $r = 6$, with medians around $r = 2.8$. 1000-node networks are even worse, with $r > 14$ in the worst case, and medians in the range of $8 < r < 8.5$. Clearly, naively sampling such a network without biased links would lead to a highly uneven sample of any property.

The largest contributor to the large value of r in Pastry are small perturbations in the distribution of nodes in the id-space. While the distribution of ids is uniformly random, within small id-ranges non-uniformities can be dominant. Often one finds the largest differences in $\boldsymbol{\pi}$ are between nodes that are extremely close together in the id-space, with one node receiving many links from id-space distant peers, while the other is only in the routing tables of id-close neighbors and has a significantly lower in-degree. Other contributors to differentiation of in-degree include proximity neighbor selection, and the

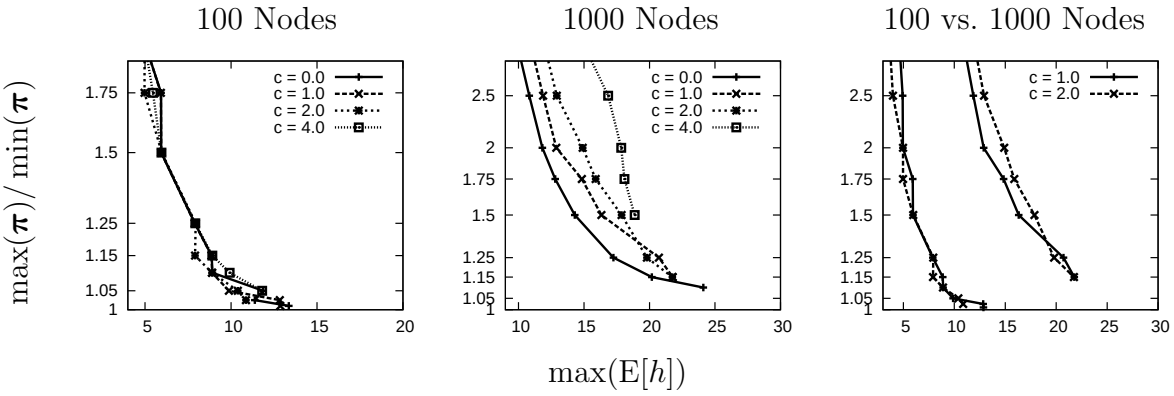

Figure 5: Maximum of r as a function of the maximum expected necessary walk length.

length of time a node has been present in the network [8].

Two different DSC cycle times over the churn workloads are shown in both graphs in Figure 4. For all workloads we ran DSC with the frequency parameter f set to 2, so the 2- and 4-second-cycle runs send 1 and 0.5 messages per second, respectively. Figure 4 clearly show that DSC significantly shrinks the gap between the least and most likely to be sampled nodes. DSC keeps r under 2 for both 100 and 1000 node networks for more than 50% of the topologies under all churn levels, and does even better a realistic level of churn.

DSC's cycle-time is designed to compensate for increased churn load. Figure 4(b) clearly shows DSC using a 2-second cycle under strain, even at $c = 1$. Lowering the cycle-time should fix this problem. DSC has to work faster to keep up because the diameter of the topology is larger, and updates have to propagate further, both during normal convergence, and during relaxation. These effects can be mitigated (or exacerbated) by the topology: if a topology has moderate or high local clustering, local changes will have less effect on nodes further away, with the inverse holding as well.

5.2.2 Expected Walk Lengths

We studied mixing-time in Pastry topologies using the same methodology described in Section 5.1.2. Figure 5 displays the data for both $N = 100$ and $N = 1000$ Pastry topologies, with varying levels of churn.

Overall, Figure 5 show that DSC biases Pastry in a manner consistent with short sampling walks. Pastry at either size, with or without churn, has shorter or comparable walk lengths than any of the static topologies at $N = 100$. The graph comparing the 100 and 1000 node topologies appears to show sub-linear growth in the length of the walk, even though cycle-length of 2000 is slightly to low (see Section 5.2.1). Unfortunately, running larger topologies to check this trend is currently beyond our computational capacity.

6 Conclusion

We have presented DSC, a distributed algorithm which balances the transition probabilities of peers in directed P2P topology such that random walks uniformly sample the network. We gave a proof of convergence for DSC, and showed, through simulations,

that the rate of convergence is usable in many concrete scenarios, and remains so under realistic levels of churn. Further, we found the sample-length of DSC-biased topologies to be acceptable, and that churn had minimal affects on sampling.

Active development continues on DSC. First, we are continuing to simulate DSC on networks of increasing sizes to study its scaling properties. In particular, we want to determine how scalable relaxation is for different topology types. Further, we would like to better understand the dynamics of the stationary distribution under churn.

Second, DSC has been designed such that it can be used in an *opportunistic* manner, so as to use little or no overhead by piggy-backing on existing P2P traffic. We are working to integrate such piggy-backing with FreePastry.

We would like to return to a more theoretical analysis of DSC, first considering the effects of feedback and relaxation on convergence, and then trying to tighten the bounds on convergence. We are also very interested in trying to improve the biasing. Compared to approximate linear programming solutions, DSC biasing produce longer walks, and a method similar to the one presented by Awan et. al could be appropriate [2]. Another option to is to explore distributed approximations of various iterative minimizations of the mixing time [5].

Finally, we want to explore a merger of Metropolis-Hastings and DSC. Most P2P topologies are largely undirected, and such sub-topologies could use Metropolis-Hastings, while peers or sub-topologies of directed nodes could use DSC. A merger of the two should prove interesting and useful.

References

[1] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.

[2] A. Awan, R. A. Ferreira, S. Jagannathan, and A. Grama. Distributed uniform sampling in unstructured peer-to-peer networks. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS '06)*. IEEE Computer Society, 2006.

[3] Z. Bar-Yossef, A. Berg, S. Chien, J. Fakcharoenphol, and D. Weitz. Approximating aggregate queries about web pages via random walks. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00)*, pages 535–544, 2000.

[4] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. In *Proceedings of the 2nd International Workshop on Peer-To-Peer Systems (IPTPS '03)*, Feb. 2003.

[5] S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing markov chain on a graph. *SIAM Review*, 46(4):667–689, 2004.

[6] S. Datta and H. Kargupta. Uniform data sampling from a peer-to-peer network. In *Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS '07)*, Washington, DC, USA, 2007. IEEE Computer Society.

[7] P. Druschel, A. Rowstron, and et. al. Freepastry, http://freepastry.rice.edu.

[8] J. Falkner, M. Piatek, J. P. John, A. Krishnamurthy, and T. Anderson. Profiling a million user DHT. In *Proceedings of the 7th Internet Measurement Conference (IMC '07)*, pages 129–134, New York, NY, USA, 2007. ACM Press.

[9] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *Proceedings of the 23rd Conference of the IEEE Communications Society (INFOCOM '04)*. IEEE Computer Society, Mar. 2004.

[10] C. M. Grinstead and J. L. Snell. *Introduction to Probability*. American Mathematical Society, 2nd edition, 1997.

[11] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. On near-uniform url sampling. *Computer Networks*, 33(1-6):295–308, 2000.

[12] M. Jelasity and O. Babaoglu. T-man: Gossip-based overlay topology management. In *Proceedings of the 3rd International Workshop on Engineering Self-Organising Applications (ESOA '05)*, pages 1–15, London, UK, 2005. Springer-Verlag.

[13] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *Transactions on Computing Systems*, 25(3), 2007.

[14] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS '03)*, pages 482–491. IEEE Computer Society, Oct. 2003.

[15] V. King and J. Saia. Choosing a random peer. In *Proceedings of the 23rd Symposium on Principles of Distributed Computing (PODC '04)*, pages 125–130, New York, NY, USA, 2004. ACM Press.

[16] J. Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proceedings of the 32nd Symposium on Theory of Computing (STOC '00)*, pages 163–170, 2000.

[17] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th International Conference on Supercomputing (ICS '02)*, pages 84–95, June 2002.

[18] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proceedings of the 1st International Workshop on Peer-To-Peer Systems (IPTPS '02)*, pages 53–65, London, UK, 2001. Springer-Verlag.

[19] J. Norris. *Markov Chains*. Cambridge University Press, 1997.

[20] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the 2nd International Middleware Conference (Middleware '01)*, pages 329–350, London, UK, 2001. Springer-Verlag.

[21] A. Sinclair. Improved bounds for mixing rates of marked chains and multicommodity flow. In *Proceedings of the 1st Latin American Symposium on Theoretical Informatics (LATIN '92)*, pages 474–487, London, UK, 1992. Springer-Verlag.

[22] J. L. Snell. *Topics in Contemporary Probability and its Applications*. CRC Press, 1st edition, 1995.

[23] D. Stutzbach and R. Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th Internet Measurement Conference (IMC '06)*, pages 189–202, New York, NY, USA, 2006. ACM Press.

[24] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger. On unbiased sampling for unstructured peer-to-peer networks. In *Proceedings of the 6th Internet Measurement Conference (IMC '06)*, pages 27–40, New York, NY, USA, 2006. ACM Press.

[25] M. Zhong, K. Shen, and J. Seiferas. The convergence-guaranteed random walk and its applications in peer-to-peer networks. *Transactions on Computers*, 57(5):619–633, 2008.