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

Abstract. Selecting a random peer with uniform probability across a peer-to-peer (P2P) network is a fundamental function for unstructured search, data replication, and monitoring algorithms. Such uniform sampling is supported by several techniques. However, current techniques suffer from sample bias and limited applicability. In this paper, we present a sampling algorithm that achieves a desired uniformity while making essentially no assumptions about the underlying P2P network. This 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 both directed and undirected topologies, making it suitable for virtually any P2P network. 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, we show that DSC is resilient to 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 (bandwidth, load, etc.) of interconnected peers in an arbitrary P2P network. 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 as well as for 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 a *uniform sampling function* that returns a peer chosen uniformly at random among all peers in a network. This function is applicable to networks of any topology, and requires no global knowledge. In addition to serving as a basis for monitoring, uniform random sampling is a useful building block in distributed systems in general, where it is used to support search, maintenance, replication, and load-balancing [1,2].

Existing techniques for uniform sampling are based on biased random walks or gossiping. In the first case, a node is selected at the end of a sufficiently long random walk in which the probabilities of following each link are adjusted to obtain a uniform visitation distribution across the network. Existing algorithms,

H. Sips, D. Epema, and H.-X. Lin (Eds.): Euro-Par 2009, LNCS 5704, pp. 511–522, 2009.

such as Metropolis-Hastings and maximum-degree, use local properties of nodes to compute their visitation probabilities, which they then use to bias the transition probabilities [2,3,4]. The locality of this information makes these algorithms practical and efficient. However, these algorithms also assume that all links are bidirectional, and therefore may not be universally applicable. In gossip-based sampling, nodes maintain a pool of addresses of other peers that is frequently exchanged and shuffled with other peers through a gossip protocol [5]. Gossip sampling is tunable and efficient in terms of traffic, and is effective for applications such as search, load-balancing, and topology construction. However, it is less appropriate for statistical sampling, especially with a bursty or high demand, because of the overhead of an increased gossip rate, which is necessary to provide independent and identically distributed samples (see Section 3).

Our contribution is to remove the assumption of bidirectional links in the network while maintaining desirable statistical properties (i.e., uniformity) for the sampling service. Non-bidirectional networks come in two types: networks with truly asymmetric links, and networks lacking a consistency protocol for tracking incoming links (e.g., DHTs). The algorithm we propose is fully distributed and uses only localized information. These features make it applicable to virtually any P2P system, as well as to a wide range of applications.

Our algorithm falls in the general category of random-walk sampling. The main idea is to avoid the calculation of each node's visitation probability, and instead to adjust the transition probabilities iteratively, converging to a state in which the sum of transition probabilities into each node equals 1. The resulting transition matrix is said to be *doubly stochastic*, and induces uniform visitation probabilities. In practice, the algorithm performs well on both static and dynamic topologies, keeping the ratio between the maximum and minimum sample probability below 1.2 for realistic churn conditions. Further, our algorithm generates link-biases that keep the expected sample walk length reasonably short, between 20 and 50 hops for 1000-node static (no churn) topologies of various kinds, and around 23 hops for 1000-node Pastry networks under churn.

In Section 2 we cover the relevant background necessary to explain our algorithm. Section 3 reviews previous work on P2P sampling. Section 4 presents a basic version of our *doubly stochastic converge* (DSC) algorithm, sketches a proof of its convergence, and presents a more advanced variant that reduces the length of the random walks and deals with failure. Section 5 evaluates DSC in simulations on static topologies, and within a concrete system under churn. Finally, Section 6 concludes with a discussion of future work.

2 Background Theory

We model a P2P network as a directed graph $G = (V, E)$ on $n = |V|$ vertices. For each vertex v, $N^-(v) = \{u \in V \setminus \{v\} : (u, v) \in E\}$ is the *in-neighborhood* of v, and $N^+(v) = \{u \in V \setminus \{v\} : (v, u) \in E\}$ is the *out-neighborhood* of v. Notice that N^- and N^+ do not contain v itself. For each pair of vertices u and v, p_{uv} is the *transition probability* from u to v. That is, p_{uv} is the probability of proceeding from u to v in a random walk. Transition probabilities are such that $p_{uv} = 0$ when $(u, v) \notin E$, and $\sum_{v \in V} p_{uv} = 1$ for every vertex u, and therefore form a stochastic $n \times n$ matrix $\mathbf{P}$.

We model a random walk across the P2P network as a Markov chain with transition matrix $\mathbf{P}$. The vector $\mathbf{x}$ denotes the probabilities of being at each vertex in the graph, and therefore in each state of the Markov chain. Given this distribution of probabilities $\mathbf{x}_t$ at time t, one step in the random walk produces the distribution $\mathbf{x}_{t+1} = \mathbf{x}_t\mathbf{P}$. If the Markov chain is ergodic, then the probability to arrive at a given node after a sufficiently long walk stabilizes, independently of the starting point, to a unique *stationary distribution* $\boldsymbol{\pi}$, where each node u has a *visitation probability* of $\boldsymbol{\pi}(u)$. The *mixing-time* of $\mathbf{P}$ is the minimal length of a walk that achieves a desired deviation from the stationary distribution.

A Markov chain is ergodic if every state (i.e., every peer) is reachable from any other state, and, beyond a certain path length, is reachable at all greater path lengths. Formally, there exists a number of steps q for which $\forall n \geq q : \mathbf{P}^n(u, v) > 0$ for all nodes u and v. In practice, a P2P network that is both strongly connected and aperiodic—reasonable assumptions—will have an ergodic Markov chain.

Since we want to sample peers uniformly starting from any node, our goal is to obtain a *uniform* stationary distribution. The theory of Markov chains provides a sufficient condition that leads to such a stationary distribution. Specifically, every ergodic Markov chain with a *doubly-stochastic* transition matrix has a uniform stationary distribution. A matrix is doubly-stochastic when every row and every column sums to 1. In other words, if the sum of the transition probabilities of all the *incoming* edges of every node is 1 (the sum of the transition probabilities of all the *outgoing* edges is 1 by definition) then the transition matrix is doubly-stochastic, and the stationary distribution is uniform.

3 Related Work

For the purpose of this paper, it is useful to distinguish two broad categories of techniques for uniform peer sampling over P2P systems. First we discuss techniques developed for specific P2P systems or topologies. Then we review more general techniques, which we further classify in two main subcategories: those based on random walks, and those based on gossip protocols.

King and Saia present a method to select a peer uniformly at random from any DHT that supports the common ID-based lookup function, where *get*(x) returns the closest peer to ID x [6]. Similarly, studies of churn dynamics in DHTs often assume that peers can be sampled by looking up randomly selected values in the address space [7]. 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, natural differences in the distribution of IDs biases toward peers that are responsible for larger portions of the ID-space. In general, any technique which samples peers in a manner correlated with session length will lead to biased results [2,8]. Beyond their statistical properties, these techniques are also limited in that they are only applicable to DHTs.

Moving away from system-specific techniques, one general way to sample peers at random is to use random walks, where a peer is sampled at the end of a sufficiently long random walk. However, since the probability of reaching each peer depends on the topology, and is generally not uniform, the walk must be biased in such a way as to obtain a uniform visitation probability. This is typically done in two steps: first, each node u computes for each neighbor v the ratio between its unbiased visitation probability and that of v. This is easily accomplished assuming that all links are bidirectional, as the ratio $\boldsymbol{\pi}(u)/\boldsymbol{\pi}(v)$ equals the ratio between the degrees of u and v, and can therefore be computed locally [2]. In the second step, an algorithm such as Metropolis-Hastings or maximum-degree [2,3,4,9] is used to bias the transition probability of each link in order to obtain a uniform stationary distribution $\boldsymbol{\pi}$.

In practice, existing random-walk techniques are effective. However, as they assume that all links are bidirectional, they may not be universally applicable. Furthermore, the use of bidirectional connections may incur additional costs, such as NAT traversal, or additional state tracking for incoming links.

Other general-purpose sampling techniques are based on gossip protocols [5], which have also been shown to be useful in topology construction and maintenance [10], and in the estimation of global properties [11]. Gossip-based sampling amounts to picking an address from a local pool of peer addresses that is filled and continually shuffled through periodic "gossip" exchanges with other peers. The main advantage of gossip techniques is that extracting one sample is an immediate and local operation. Also, gossip protocols can be tuned to function at a low overhead, as only a few gossip messages per peer can guarantee a good level of randomness in the peer pools. However, these advantages deteriorate in the presence of frequent sample requests, as multiple requests can no longer return independent and identically distributed samples. There are two ways to improve the statistical quality of multiple samples in such scenarios: (1) increase the size of the peer pools, and (2) increase the gossip frequency. Unfortunately both actions also increase the overhead of the gossip protocol, and do so uniformly across the entire network. We also note that the cost of gossip messages must be added to the communication costs of the P2P system being sampled, even in the absence of churn, as gossip messages are exchanged with peers selected from the gossip peer pool, and not among the neighbors in the sampled P2P network.

4 The DSC Algorithm

One can uniformly sample a network by first assigning transition probabilities to edges such that the stationary distribution is uniform, and then performing random walks of 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, $\mathbf{P}$ is stochastic by definition, as the outgoing edges of every node have a total probability of 1. Our task is to assign outgoing probabilities

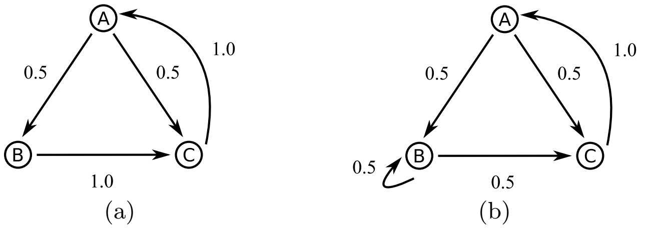


Fig. 1. An ergodic graph (a) Labeled with initial transition probabilities. (b) Balanced using a self-loop.

so that, for each node, the probabilities of the incoming edges also add up to 1, thereby making **P** doubly stochastic. We refer to this 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 (possibly 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 node 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 exemplifies the difficulties of 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 can not redirect the excess probability elsewhere. Setting $p_{AC} = 0$ balances all in- and out-probabilities, but renders the graph periodic, and hence no longer ergodic. The solution, shown in Figure 1(b), is to use the self-loop (B, B) to increase the in-probability of B to 1, which reduces the in-probability of C to 1, balances **P**, and keeps 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 node v as $In(v) \triangleq p_{vv} + \sum_{u \in N^-(v)} p_{uv}$ and $Out(v) \triangleq p_{vv} + \sum_{u \in N^+(v)} p_{vu}$, where p_{vv} is the self loop. Both $In(v)$ and $Out(v)$ must sum to 1 in order for **P** to be doubly stochastic. Clearly, increasing or decreasing the self-loop forces a decrease or increase, respectively, of the sum of probability across both $N^-(v)$ and $N^+(v)$. At any given time, a node is in one of three states: it has an in-probability surplus, so it is in $V^+ \triangleq \{v \in V : In(v) > 1\}$; it has an in-probability deficit, so it is in $V^- \triangleq \{v \in V : In(v) < 1\}$; or it is balanced, in $V^= \triangleq \{v \in V : In(v) = 1\}$.

The mean node in-probability is 1, as the sum of all in-probabilities equals the sum of all out-probabilities, which equals n since $Out(v) = 1$ for all v. Therefore, the total surplus of in-probability in V^+ equals the total deficit of in-probability in V^-. It follows that if we move nodes from V^- to $V^=$, nodes in V^+ will eventually be forced into $V^=$. Once all nodes are in $V^=$, $\mathbf{P}$ becomes doubly stochastic. Fortunately, moving a node v from V^- to $V^=$ is simple: increase the self-loop p_{vv} by $1 - In(v)$, bringing $In(v)$ to 1. This is the basic strategy of DSC. In particular, every node executes two 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 $u \in N^+(v)$, notifying them of their current transition probability, $(1 - p_{vv})/|N^+(v)|$. 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, network latency, and churn in mind (see Section 5). A short t leads to quicker convergence, but also increasing the chance to miss updates if latency is high. The frequency of updates f can be set to 1 if on-time delivery is assured, or higher in the presence of packet losses or high latency. Step 2 of the basic version of DSC evenly balances and transmits the outgoing probabilities. The uniformity of transition probabilities across outgoing links is non-optimal for most topologies. In Section 4.3 we discuss a variant of this assignment strategy which can assign outgoing probabilities in a more effective way.

In a technical report [12], we prove that, as long as G is ergodic, basic DSC iteratively updates $\mathbf{P}$ so that, in the limit, $\mathbf{P}$ converges to a doubly stochastic matrix. The main intuition behind the proof is that the ergodicity of the graph assures that any reduction of out-probability at a node v in V^- (caused by the node increasing its self-loop) will eventually reach a set of nodes in V^+, the members of which will each absorb some fraction of the change, leading to a global reduction of in-probability deficit. A key point of our proof is that neither the propagation time (number of updates), nor the amount of in-probability absorbed by u, depend on time. As a result, the global in-probability deficit converges exponentially to zero, leading to a doubly-stochastic transition matrix.

Within our proof, we also establish a lower bound for the reduction factor of the overall deficit. However, this bound is very loose. Therefore, we do not use it to evaluate DSC, and instead rely on simulation (see Section 5). In practice, DSC is quite fast in converging toward a doubly-stochastic bias, although various factors can influence its rate of convergence. One such factor is the ordering of node updates. 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.

4.2 Failure and Relaxation

Churn events (i.e., nodes joining or leaving the network) have the effect of increasing self-loops. A node joining the network may move its successors from $V^=$ or V^- into V^+. Such successors may find themselves in the peculiar state of having a surplus of in-probability *and* a self-loop greater than 0. This is never the case in the absence of 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, a leaving node u can force some $v \in N^+(q) : q \in N^-(u)$ back into V^+ with a non-zero self-loop.

Taken together, these two phenomena lead to ever increasing self-loops, which is problematic as high self-loops tend to increase the mixing time. If we intend 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, a node $v \in V^+$ with a self loop $p_{vv} > 0$ decreases its self loop by $\min(p_{vv}, 1 - \mathit{In}(v))$, setting it to zero, or as close to zero as possible. In the same operation, v must also raise the transition probabilities to its successors accordingly. In turn, this can cause $q \in N^+(u)$ to also lower their self-loop, and so on. However, the 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 functioning of DSC in a network with churn. Without it, self-loops will approach 1, leading to higher and higher sample walk lengths. Importantly, relaxation does not require a channel of communication in the opposite direction of the network links, allowing it to work with basic DSC.

4.3 Feedback

Basic DSC does not assume bidirectional connectivity, and therefore does not use information about the state of a node's successors. In particular, basic DSC spreads the outgoing probabilities uniformly across successors, without considering their individual state. This can slow convergence and, more seriously, increase the mixing-time due to high self-loops probabilities. Whenever bidirectional links exist, feedback from successors can be used to weight outgoing probabilities. Specifically, in addition to storing the transition probability sent by a predecessor v, a node u may reply with an acknowledgment containing $\mathit{In}(u)$. v then uses this value to first bias p_{vu}, and then to renormalize all its other out-probabilities:

1. $p_{vu} \leftarrow p_{vu}/\mathit{In}(u)$ (bias)
2. $\forall q \in N^+(v) : (p_{vq} \leftarrow (p_{vq}/\sum_{q \in N^+(v)} p_{vq})(1 - p_{vv}))$ (normalize)

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.

5 Simulation Analysis

Our analysis of DSC considers two main questions: (1) How does DSC perform across a wide range of topologies, both in terms of convergence speed, and expected walk length? (2) Is DSC resilient under churn conditions? To characterize convergence and walk lengths, we use a discrete-event simulation of DSC. To analyze the effects of churn, we run a concrete implementation of DSC integrated within the FreePastry framework[1] under various levels of churn. A more extensive analysis of these experiments is available in a technical report [12].

---

[1] http://freepastry.rice.edu; an open-source implementation of Pastry [13].

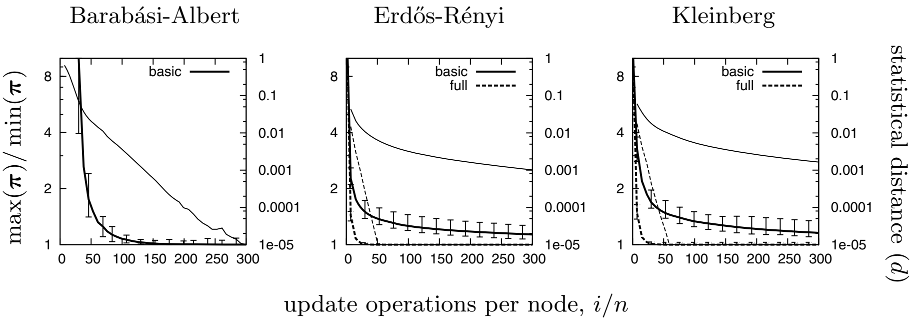


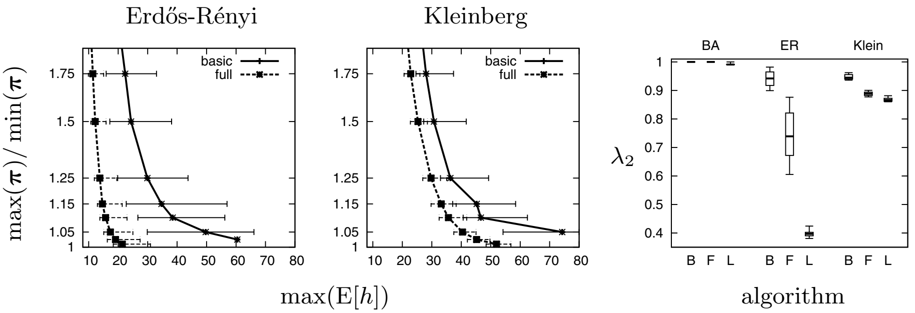
Fig. 2. Uniformity ratio r (thick) and statistical distance d (thin) after i/n updates

We simulate DSC over a diverse set of generated graphs. In particular, we use three types of generative models commonly used in P2P research:

- *Kleinberg:* nodes are connected in a grid, along with q long distance links chosen with probability inversely proportional to the squared distance [14].
- *Erdős-Rényi:* edges are selected independently with a fixed probability [15].
- *Barabási-Albert:* nodes are connected to high-degree nodes with preferential-attachment factor 0.5, creating a power-law degree distribution [15].

We use variants of these standard models to obtain directed graphs. All graphs have the same number of vertices $n = 1000$ and, with minor differences, the same number $|E| \approx n \log n$ of edges, which corresponds to a node–edge ratio found in many deployed systems. We run two versions of DSC: basic and full, where the latter includes feedback and relaxation. To measure the quality of the resulting sampling service, we compute two values, the ratio $r = \max(\boldsymbol{\pi})/\min(\boldsymbol{\pi})$, which highlights the worst-case difference in sample probability, and therefore gives a very conservative characterization of the uniformity achieved by DSC, and the statistical distance $d(\frac{1}{n}, \boldsymbol{\pi}) = \frac{1}{2}\sum_{s \in S} |\frac{1}{n} - \boldsymbol{\pi}(s)|$, which measures the normalized absolute difference between $\boldsymbol{\pi}$ and the uniform distribution.

Figure 2 shows the ratio r and statistical distance d as a function of the number of updates per peer, i/n, where i is the total number of updates over the network. Notice that updates execute in parallel, so i/n represents the number of rounds of update of the network as a whole. We simulate DSC on 50 topologies of each type, and plot the median value of r and d, with error bars for r showing the 10th and 90th percentile. In all situations DSC converges exponentially fast to the optimal ratio of $r = 1$. For Erdős-Rényi and Kleinberg topologies, full DSC performs better, reaching a tight ratio $r \leq 1.01$ in just 30 rounds. Reasonable values of d are achieved in less than 10 rounds for full DSC, and 50 rounds without feedback. Barabási-Albert topologies exhibit different behavior: basic DSC converges quickly, reaching $r = 1.01$ in 160 rounds, but full DSC experiences severe oscillations in r, and converges much more slowly. Further analysis reveals that this is due to in-probability oscillating between V^+ and V^- at low in-degree nodes, behavior which is caused by feedback. This problem could be eliminated by making feedback sensitive to successor in-degree.



(a)

(b)

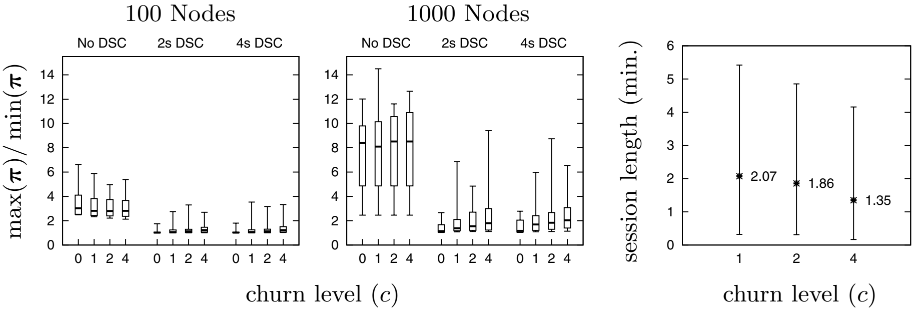
Fig. 3. (a) Relation between uniformity ratio r and maximum expected walk length. (b) Second eigenvalue for basic DSC (B), full DSC (F), and linear-programming (L), for each topology type: Barabási-Albert (BA), Erdős-Rényi (ER), and Kleinberg (Klein).

Quick convergence indicates that the cost of balancing is low, but does not say anything about the costs of sampling. This latter cost is determined by the mixing-time, that is, the number of hops after which a random walk becomes independent of its starting node. We analyze this cost using basic results from Markov-chain theory [12]. After the convergence of each topology, we calculate the maximum expected walk-length max(E[h]) for different target values of r.

Figure 3(a) shows the results of this analysis, graphing the median value, with error bars representing the 10th and 90th percentiles. Reasonable values of r are achievable in less than 25 hops for Erdős-Rényi and 45 hops for Kleinberg with full DSC. However, feedback in full DSC can not significantly improve walk-lengths for Kleinberg topologies. This is because feedback takes advantage of irregularity in link structure, and Kleinberg topologies are highly regular. Barabási-Albert topologies are not shown, as max(E[h]) can be as high as 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 less than 10^{-6}) which develop extremely high self-loops (often greater than 0.995).

In evaluating walk lengths in DSC, we would like to compare them with the best achievable walk lengths for each topology. Unfortunately, we are not aware of a suitable optimization algorithm for topologies of the type and scale we considered. In order to obtain an estimate of the minimal walk length, we try to balance the transition probabilities using a linear-programming optimization. In particular, we use an objective function that minimizes self-loops while trying to evenly balance out-probability [12]. We then compare DSC with the linear-programming solution using the second-largest eigenvalue modulus λ_2 of the transition matrix $\mathbf{P}$ (lower values of λ_2 indicate better walk lengths [16]).

The results, shown in Figure 3(b), demonstrate that there is room for improvement in the way DSC assigns probability across edges. The linear-programming method minimizes λ_2 to a greater extent thanks to its global view of the network, whereas DSC operates in a completely decentralized manner. We are considering ways to improve DSC's weightings by integrating more information into



(a) (b)

Fig. 4. (a) Uniformity ratio r under churn. (b) Session lengths.

feedback. The end goal is to move transition probability to those successors with low in-probability [3].

In the second part of our evaluation, we study the behavior of DSC under churn by running it within FreePastry, an implementation of the fault tolerant, locality-aware Pastry DHT [13]. Our churn model follows the findings of Stutzbach and Rejaie and their study of a DHT based on Kademlia [8,12]. We use FreePastry in simulation mode over 100 and 1000 nodes, with the churn process starting as soon as the last node has joined. After the initial join period is complete, we measure r for 10 minutes.

The graphs in Figure 4(a) show the ratio r (median, min, max, 10th, and 90th percentile) for different levels of churn. $c = 1$ indicates churn dynamics very similar to those measured by Stutzbach and Rejaie, while $c = 2, 4$ means twice and four times those levels [8,12]. Figure 4(b) shows the session lengths (median, first, and third quartile) for each value of c. We first measure the ratio r without DSC, then with DSC with updates every 2 and 4 seconds ($t = 2$ or 4). The results show that DSC achieves high uniformity of sampling, keeping the ratio r under 2 for more than 50% of the topologies under all churn levels, and does even better at $c = 1$.

We study the mixing-times in the Pastry experiments with the same methodology used for the static topologies.

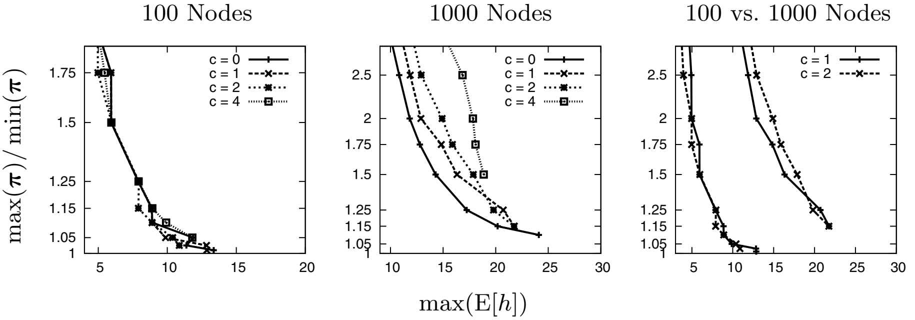


Fig. 5. Uniformity ratio r and maximum expected walk length under churn

Figure 5 displays the relation between the ratio r and the walk lengths under varying levels of churn, and shows that DSC achieves good uniformity with short sampling walks. In fact, 1000-node Pastry networks, with or without churn, have shorter or comparable walk lengths than any of the static topologies we studied. Furthermore, the graph comparing the 100 and 1000-node topologies suggests a sub-linear growth of the length of the walks as a function of the size of the network.

6 Conclusion

We have presented DSC, a distributed algorithm that balances the transition probabilities of peers in a directed P2P topology such that random walks of sufficient can uniformly sample the network. We gave a proof sketch 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 has minimal effects on sample probability.

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 both the stationary distribution and random walks under churn. Second, DSC has been designed to function in an opportunistic manner, so as to incur little or no traffic overhead, by piggy-backing on existing P2P traffic. We are working to integrate such piggy-backing with FreePastry using an aggregation technique we are developing. Finally, to make DSC usable in a concrete deployment, we are evaluating several efficient algorithms to estimate the minimal necessary walk lengths.

We would also 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 produces longer walks, and a method similar to the one presented by Awan et. al could be appropriate [3]. Another option to is to explore distributed approximations of various iterative minimizations of the mixing time [17].

Acknowledgments. This research was supported in part by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS) under Swiss National Science Foundation grant number 5005-67322. The authors are grateful to Shane Legg, Mark Carman, Thomas Shrimpton, and the anonymous reviewers for their useful comments and suggestions.

References

1. Gkantsidis, C., Mihail, M., Saberi, A.: Random walks in peer-to-peer networks. In: Proceedings of the 23rd Conference of the IEEE Communications Society, INFOCOM (2004)
2. Stutzbach, D., Rejaie, R., Duffield, N., Sen, S., Willinger, W.: On unbiased sampling for unstructured peer-to-peer networks. In: Proceedings of the 6th Internet Measurement Conference, IMC (2006)

3. Awan, A., Ferreira, R.A., Jagannathan, S., Grama, A.: Distributed uniform sampling in unstructured peer-to-peer networks. In: Proceedings of the 39th Annual Hawaii International Conference on System Sciences, HICSS (2006)
4. Datta, S., Kargupta, H.: Uniform data sampling from a peer-to-peer network. In: Proceedings of the 27th International Conference on Distributed Computing Systems, ICDCS (2007)
5. Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.M., van Steen, M.: Gossip-based peer sampling. ACM Transactions on Computing Systems 25(3) (2007)
6. King, V., Saia, J.: Choosing a random peer. In: Proceedings of the 23rd Symposium on Principles of Distributed Computing, PODC (2004)
7. Bhagwan, R., Savage, S., Voelker, G.: Understanding availability. In: Proceedings of the 2nd International Workshop on Peer-To-Peer Systems, IPTPS (2003)
8. Stutzbach, D., Rejaie, R.: Understanding churn in peer-to-peer networks. In: Proceedings of the 6th Internet Measurement Conference, IMC (2006)
9. Zhong, M., Shen, K., Seiferas, J.: The convergence-guaranteed random walk and its applications in peer-to-peer networks. ACM Transactions on Computers 57(5) (2008)
10. Jelasity, M., Babaoglu, O.: T-man: Gossip-based overlay topology management. In: Brueckner, S.A., Di Marzo Serugendo, G., Hales, D., Zambonelli, F. (eds.) ESOA 2005. LNCS, vol. 3910, pp. 1–15. Springer, Heidelberg (2006)
11. Kempe, D., Dobra, A., Gehrke, J.: Gossip-based computation of aggregate information. In: Proceedings of the 44th Symposium on Foundations of Computer Science, FOCS (2003)
12. Hall, C., Carzaniga, A.: Doubly stochastic converge: Uniform sampling for directed P2P networks. Technical report, University of Lugano, Lugano, Switzerland (2009)
13. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, p. 329. Springer, Heidelberg (2001)
14. Kleinberg, J.: The small-world phenomenon: an algorithm perspective. In: Proceedings of the 32nd Symposium on Theory of Computing, STOC (2000)
15. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Reviews of Modern Physics 74(1) (2002)
16. Sinclair, A.: Improved bounds for mixing rates of marked chains and multicommodity flow. In: Simon, I. (ed.) LATIN 1992. LNCS, vol. 583. Springer, Heidelberg (1992)
17. Boyd, S., Diaconis, P., Xiao, L.: Fastest mixing markov chain on a graph. SIAM Review 46(4) (2004)