2004

Distributed Uniform Sampling in Real- World Networks

Asad Awan

Ronaldo A. Ferreira

Suresh Jagannathan
Purdue University, suresh@cs.purdue.edu

Ananth Y. Grama
Purdue University, ayg@cs.purdue.edu

Report Number:
04-029

DISTRIBUTED UNIFORM SAMPLING IN REAL-WORLD NETWORKS

Asad Awan
Ronaldo A. Ferreira
Suresh Jagannathan
Ananth Y. Grama
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD TR #04-029
October 2004

Distributed Uniform Sampling in Real-World Networks

Asad Awan Ronaldo A. Ferreira Suresh Jagannathan Ananth Grama
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
Email: {awan,rf,suresh,ayg}@cs.purdue.edu

Abstract

Uniform sampling in networks is at the core of a wide variety of randomized algorithms. Random sampling can be performed by modeling the system as a graph with associated transition probabilities and defining a corresponding Markov chain (MC). A random walk of prescribed minimum length, performed on this graph, yields a stationary distribution, and the corresponding random sample. This sample, however, is not uniform when network nodes have a non-uniform degree distribution. This poses a significant practical challenge since typical large scale, real-world, unstructured networks tend to have non-uniform degree distributions, e.g., power-law degree distribution in unstructured peer-to-peer networks.

In this paper we present a distributed algorithm that enables efficient uniform sampling in large real-world networks. Specifically, we prescribe necessary conditions for uniform sampling in such networks and present distributed algorithms that satisfy these requirements. We empirically evaluate the performance of our algorithm in comparison to known algorithms. We also quantify, in context of the presented algorithms, the performance parameters in uniform sampling that are most relevant in a distributed setting – computational complexity, number of network messages, and the uniformity of the sampling. Detailed experimental results are used to support our claims relating to performance improvements of our algorithm.

1 Introduction

Uniform sampling in networks is an important substrate that provides the basis for a variety of randomized algorithms. These algorithms address problems such as leader election, duplicate elimination and controlled replication, search and routing, and group communications. The emergence of peer-to-peer (P2P) networks, where frequent node arrivals and departures make it difficult to maintain accurate network state, provide strong motivation for this class of algorithms. This paper addresses the critical problem of efficient distributed uniform sampling via random walks in large unstructured networks.

The uniform sampling problem can be formally defined as follows:

Definition 1.1.(Uniform random sampling) *An algorithm samples uniformly at random from a set of nodes in a connected network if and only if it selects a node i belonging to the network, with probability $1/n$, where n is the number of nodes in the network.*

Notice that this problem is analogous to the problem of selecting a number uniformly at random in a given range. A trivial approach to this problem would be to collect the entire set of node identifiers at each node and index randomly into this table of identifiers. This simple approach, however, does not work for our target applications because the overhead of frequently updating system state at each node (if at all possible) would be extremely high. An alternate approach this problem relies on the notion of a random walk. Starting from an initial node, a random walk (of predetermined length) transitions through a sequence of intermediate nodes with probabilities defined for each link and ends at a destination node. The likelihood of terminating a random walk at any node determines whether the walk is a uniform sampling random walk or not. Formally, we define a uniform sampling random walk as follows:

Definition 1.2 (Uniform random sampling using random walk) *A random walk of a given length samples uniformly at random from a set of nodes of a connected network if and only if the walk terminates at a node i belonging to the network, with probability $1/n$, where n is the number of nodes in the network.*

A number of researchers, over the years, have studied properties of random walks. Lovasz [11] provides an excellent survey on these techniques. The simplest random walk algorithm selects an outgoing edge at every node with

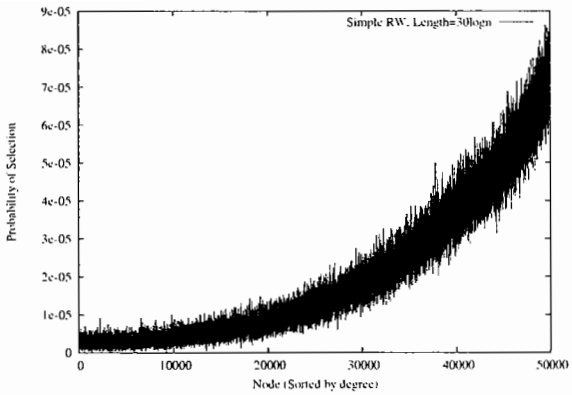

Figure 1. Random sampling using a simple random walk on a power-law graph. The resulting sample is strongly correlated with the degree distribution.

equal probability, e.g., if a node has degree four, each of the edges is traversed with a probability 0.25. It can be shown that the probability distribution associated with target nodes becomes stationary after a finite length random walk (also known as the mixing time for the corresponding Markov chain). This length can be shown to approach $\log n/(1 - SLEM)$, for a network of n nodes. Here, SLEM (second largest eigenvalue) corresponds to a network topology parameter. These concepts are discussed in greater detail in Section 2. The main drawback of the simple random walk is that, while it reaches a stationary distribution, this distribution is not uniform for typical networks. In fact, it can be shown that the probability of terminating a random walk at a node is directly proportional to the degree of the node. In the context of conventional networks, where node degrees can vary significantly, this does not correspond to an acceptable uniform sample. In Figure 1, we plot the probability of terminating at a node for a power-law graph, with 50,000 nodes. Note that the variability in sampling is close to an order of magnitude!

Many applications of random walks are extremely sensitive to the quality of uniform sampling. Biases in sampling may result in poor performance of randomized algorithms, congestion in underlying networks, or sub-optimal utilization of storage resources. This provides the underlying motivation for our work. In addition to the quality of uniform sampling, a key performance parameter is the length of the random walk. Since the length of the random walk directly corresponds to the number of network messages, it is highly desirable to minimize the length of the walk. Consequently, the focus of this paper is on random walk techniques capable of uniform sampling, while minimizing the length of the walk. The paper makes the following specific contributions:

- It identifies sufficient and necessary conditions on a transition matrix for a uniform sampling random walk over the corresponding network.
- It presents an algorithm, called Random Weight Distribution (RWD), for achieving these conditions while reducing the length of the random walk.
- It provides detailed empirical evaluation of the performance characteristics of RWD in comparison to existing methods, namely Metropolis-Hastings (MH) and Maximum-Degree (MD).

The structure of this paper is as follows – in Section 2, we provide the background and foundations for our algorithm. In Section 3, we present necessary and sufficient conditions for uniform sampling. We also present two known algorithms and our new algorithm, which enables uniform sampling in irregular networks. In Section 4 we empirically evaluate the performance of our algorithm, and present a simulation based comparison of the presented algorithms. In Section 5, we present related work, followed by conclusions, in Section 6.

2 Background

In this section, we provide necessary background on random sampling using random walks in large networks. The abstraction of random walks as Markov chains is used to set up the notation and concepts that are used in the rest of the paper. Using a Markov chain model, we show, based on known results, that a simple random walk cannot achieve uniform sampling unless each node in the network has an identical number of connections. We also discuss various parameters that determine the length of the random walk required to achieve a stationary sample distribution.

Let $G(V,E)$ be a simple connected undirected graph representing a distributed system with $|V| = n$ nodes and $|E| = m$ links. The degree, or number of links, of a node i, $1 \leq i \leq n$, is given by d_i and $d_{max} = max_{1 \leq i \leq n}\{d_i\}$ denotes the maximum degree. The set of neighbors of a node i is given by $\Gamma(i)$, where edge $(i,j) \in E, \forall j \in \Gamma(i)$. The $n \times n$ adjacency matrix of G is given by $A = \{a_{ij}\}$, where $1 \leq i,j \leq n$, $a_{ij} = 1$ if the edge $(i,j) \in E$, and $a_{ij} = 0$ otherwise. The corresponding $n \times n$ transition probability matrix is given by $P = \{p_{ij}\}$, where $0 \leq p_{ij} \leq 1$ is the probability of moving from node i to node j in one message hop (or time step). Furthermore, it is easy to see that $\sum_j p_{ij}$ should equal 1, which implies that P is a row-stochastic matrix.

Random walks: A *simple random walk* on G is a sequence of nodes visited at each step of the walk. The transition from node i to its neighbor is governed by the transition probability matrix P, where $\forall j \in \Gamma(i)$, $p_{ij} = 1/d_i$; $p_{ij} = 0, \forall j \notin \Gamma(i)$.

The sequence of nodes can be denoted as $\{X_t, X_{t+1}, ...\}$, where $X_t = i$ implies that at step t the walk is at node i.

If we consider nodes in G as states in a finite state space, then the random walk represents a discrete-time stochastic process, $\{X_t\}_{t \geq 0}$. For this stochastic process we have,

$$\begin{aligned} Pr(X_{t+1} = j | X_0 = i_0, ..., X_{t-1} = i_{t-1}, X_t = i) \\ = Pr(X_{t+1} = j | X_t = i) = p_{ij} \end{aligned} \quad (1)$$

Equation (1) simply implies that during a random walk the probability of moving to node j from node i in one step only depends on node i and is independent of t. This is known as the *memoryless* or *Markov* property. A random walk can be conveniently modeled as a Markov chain, more specifically a homogeneous Markov chain, since the right hand side of Equation (1) is independent of t. Such a Markov chain has the following properties: it is irreducible if the graph G is connected and is aperiodic if G is aperiodic. A graph G is aperiodic if the greatest common divisor of the length of all cycles in the graph is 1. In particular, an undirected aperiodic graph cannot be bipartite, which is a reasonable assumption for real networks in which connections are established randomly.

Equation (1) can be written more generally as $\pi(t+1)^T = \pi(t)^T P$, where $\pi(t)^T$ is the transpose of the vector of probability distribution of states at time t. Let, P^t be the *t-step probability transition matrix*. Therefore, we have:

$$\pi(t)^T = \pi(0)^T P^t. \quad (2)$$

It is well known that an irreducible and aperiodic Markov chain has a stationary distribution $\pi^T = \pi^T P$, and $\pi^T = \pi^T P^t$ follows. It is easy to show ([15], page 132) that π_i, the component corresponding to node i, $1 \leq i \leq n$, is $\pi_i = d_i/2m$.

Eigenvalues of P: From $\pi^T = \pi^T P$, we see that π is a left eigenvector of P with eigenvalue 1. Also, $P\mathbf{1} = \mathbf{1}$ (P is row-stochastic, and $\mathbf{1}$ is a vector with all entries equal to 1) implies that $\mathbf{1}$ is a right eigenvector with eigenvalue 1. It follows that $P^\infty = \mathbf{1}\pi^T$. This implies that a very long walk converges to the stationary distribution π irrespective of the initial distribution. Since P is a non-negative primitive $n \times n$ matrix (i.e., irreducible and aperiodic), from basic linear algebra, we also know that P has n distinct eigenvalues $1 = \lambda_1 > |\lambda_2| \geq \cdots \geq |\lambda_n|$ [3].

Random sampling: The above results indicate that a long enough random walk converges to a random sample irrespective of where the walk started. Thus, random walk is a good candidate for random sampling in a network. However, we also know that the resulting sample distribution is dependent on the degree of the node: $\pi_i = d_i/2m$. This last result implies that the random sample is uniform ($\pi_{uniform} = (1/n)\mathbf{1}$) only if the graph G is regular (i.e., the degrees of all nodes are equal). Since typical large scale, real-world, unstructured networks tend to have non-uniform degree distributions (e.g., power-law degree distribution of unstructured P2P networks [19], and irregular degrees due to irregular placement of sensors in a sensor network [4]), uniform sampling in practical scenarios poses a significant challenge.

Length of walk for random sampling: The sample distribution at step t of the walk depends on P^t, which in turn depends on the eigenstructure of P. From the Perron-Frobenius theorem, we have $P^t = \lambda_1^t v_1 u_1^T + O(t^{m_2-1}|\lambda_2|^t)$, where v_1 is the right eigenvector corresponding to eigenvalue λ_1 and u_1 is the left eigenvector, and m_2 is the algebraic multiplicity of λ_2 (see, [3] Chapter 6). Rewriting the above equation, we have $P^t = P^\infty + O(t^{m_2-1}|\lambda_2|^t)$. These results simply imply that

$$P^t = \mathbf{1}\pi^T + O(t^{m_2-1}|\lambda_2|^t). \quad (3)$$

As $|\lambda_2| < 1$, when t is large, $|\lambda_2|^t \approx 0$. Therefore, the smaller the second largest eigenvalue modulus (SLEM)[1], the faster the convergence to stationary distribution. As a result, a walk of smaller length is required for random sampling. The number of steps required to converge to the stationary distribution is called the *mixing time* of the Markov chain.

3 Distributed Uniform Sampling Algorithms

In this section, we describe necessary and sufficient conditions for uniform sampling using random walks. We present known distributed algorithms, which change transition probabilities between neighboring nodes such that a random walk of a given minimum length can be used for uniform sampling. One of the shortcomings of these algorithms is that the minimum length of the random walk required to reach stationary distribution is significant. We present a new distributed algorithm, called Random Weight Distribution (RWD) that allows uniform sampling, while shortening the required minimum length of the random walk. We analytically show that our algorithm outperforms known distributed algorithms in terms of the minimum length of the random walk, and that the setup overhead of the algorithm is minimal.

3.1 Uniform Sampling via Random Walks

As mentioned in Section 2, a random walk of a given minimum length converges to a stationary distribution π. If the stationary distribution $\pi_{uniform}$ is such that $\pi_{uniform} = (1/n)\mathbf{1}$, the random walk will terminate at any node in the network with equal probability (c.f. Definition 1.2).

[1] Intuitively, a small SLEM is an indicator of good global connectivity of the network.

To achieve a uniform stationary distribution in an irregular graph, we need to modify its probability transition matrix. Recall that, if the graph is not regular the probability transition matrix introduced for simple random walks will not suffice. As we shall see, it is straightforward to define probability transition matrices that have a stationary distribution $\pi_{uniform}$.

Let P be a probability transition matrix of a Markov chain, then $\pi^T_{uniform} = \pi^T_{uniform}P$. Rewriting this as $(1/n)\mathbf{1}^T = (1/n)\mathbf{1}^T P$, shows that $\mathbf{1}^T = \mathbf{1}^T P$ for such a matrix. This means that the sum of column vectors of P is equal to 1 ($\sum_i(1.Pij) = 1$) for each column j of the matrix, i.e., P is *column stochastic*. A probability transition matrix which is column stochastic in addition to being row stochastic is called *doubly stochastic*. Therefore, we can state that a doubly stochastic matrix has a stationary distribution $\pi_{uniform}$. The following observation will be used to prove that the algorithms presented next result in uniform sampling random walks.

Observation 3.1 *Symmetric probability transition matrices ($P^T = P$) are doubly stochastic. P is row stochastic because it is a probability transition matrix, and P is column stochastic by virtue of symmetry. Therefore, a Markov chain defined over a symmetric probability transition matrix has a stationary distribution $\pi_{uniform}$.*

3.1.1 Random Walk Implementation

Random walks are easy to implement in real-world distributed systems. In its simplest form the walk can be implemented as a message that is forwarded from one node to another node, selected based on the transition probability matrix. Such a message should contain a time-to-live (TTL) field that is set by the origin node to be the length of the walk. The TTL is decremented at every transition (which may include self-transitions, depending on the set up of the probability transition matrix). A node that receives the random walk message with $TTL = 0$ is selected as the random sample.

3.2 Existing Distributed Random Walk Algorithms

We present two known algorithms that modify the transition probabilities between nodes to produce a probability transition matrix that has a stationary distribution $\pi_{uniform}$. A random walk on a network with node transition probabilities defined using these algorithms will, therefore, result in a uniform sample. The algorithms are traditionally presented in the context of Markov chains, for example in [2], however, their adaptations to a distributed network are straightforward.

3.2.1 Maximum-Degree Algorithm (MD)

In the distributed adaptation of this algorithm, each node can perform a local computation to set up its transition probabilities. The main problem with the algorithm is that it requires the knowledge of the maximum degree, d_{max}, among all nodes in the network. The maximum degree is a dynamic and global parameter of the network, and its dissemination to all nodes at run-time in a large distributed system is difficult. The algorithm sets up the transition matrix P^{mh} as follows:

$$p^{md}_{ij} = \begin{cases} 1/d_{max} & \text{if } i \neq j \text{and} j \in \Gamma(i) \\ 1 - d_i/d_{max} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Note that the self-transition probability maintains the doubly-stochastic property of the matrix. In this algorithm, $p^{md}_{ij} = p^{md}_{ji} = 1/d_max$, therefore, the resulting probability transition matrix is symmetric and doubly stochastic. It follows from Observation 3.1 that a random walk using these transition probabilities will select a node uniformly at random if the random walk is long enough.

3.2.2 Metropolis-Hastings Algorithm (MH)

This algorithm is an adaptation for uniform sampling of the classical Metropolis-Hastings algorithm [14, 6, 2]. In this distributed algorithm each node i sends a message, stating its degree, d_i, to each of its neighbors $j \in \Gamma(i)$. Once this information is received from each of the neighbors, the transition probability matrix P^{mh} is set up as follows:

$$p^{mh}_{ij} = \begin{cases} 1/max(d_i, d_j) & \text{if } i \neq j \text{and} j \in \Gamma(i) \\ 1 - \sum_{j \in \Gamma(i)}(p^{mh}_{ij}) & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

As in the previous algorithm, self-transition probability maintains the row-stochastic property. Clearly, P^{mh} is symmetric and hence will enable uniform sampling via random walks (Observation 3.1).

3.2.3 Performance

Conditioned on the non-uniformity of the number of links per node (i.e., high variance in degree distribution), both MD and MH might have high self-transition probabilities for nodes with low degrees. Comparing MD and MH algorithms, we can observe that the self-transition probability of the MH algorithm is lower-bounded by the self-transition probability of the MD algorithm.

Intuitively, high self-transition probability implies that the length of the walk must be higher to attain sufficient mixing (a comprehensive experimental evaluation is provided in Section 4), otherwise the walk might be biased

towards low degree nodes. This bias is certainly not desired because often times low degree has a correlation with low importance of the node in the network (for e.g., in P2P systems low degree nodes generally stay in the network for smaller periods of time and have fewer resources than higher degree nodes).

3.3 Random Weight Distribution Algorithm

In this section, we present our distributed algorithm, referred to as the Random Weight Distribution (RWD) algorithm. RWD is a completely decentralized algorithm that sets up transition probabilities in a connected network to enable efficient uniform sampling via random walks.

The algorithm proceeds as follows. In the initialization phase each node, locally, sets transitions probability as:

$$p_{ij}^{rwd} = \begin{cases} 1/\rho & \text{if } i \neq j \text{and} j \in \Gamma(i), \text{ where } \rho \geq d_{max} \\ 1 - d_i/\rho & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Here, ρ is a static system parameter with the constraint that it should be greater than d_{max}. This parameter is static because we can sufficiently overestimate d_{max} knowing system properties (e.g., popular P2P clients have a maximum connection limit [10]). Furthermore, as shown subsequently, overestimating d_{max} does not affect the performance of our algorithm. Note that this phase results in a high self-transition probability for low degree node. Also note that the resulting transition probability matrix is symmetric.

After the initialization is complete, each node attempts to distribute its self-transition probability randomly and symmetrically to its neighbors. The algorithm runs at each node in the network. In the following sections the terminology *weight of a node* refers to the self-transition probability of the node at any given time during the execution of the algorithm. At a node i, the algorithm terminates when either the weight of the node becomes zero or the weight of all nodes $j \in \Gamma(i)$ becomes zero. The pseudo code for the complete RWD algorithm is shown in Figure 2.

3.3.1 Discussion

A node i keeps a set N of its neighbors that have non-zero weights (self-transitions). It selects a neighbor j with equal probability from this set and sends j an `INCREASE` message. Selecting neighbors uniformly at random is an important characteristic of this algorithm and will be used later in its analysis. Node j, on receiving the `INCREASE` message, checks to see if its weight is greater than or equal to δ, $\delta < 1$ is a global quantum parameter. If the weight of j is greater than or equal to δ, it accepts the `INCREASE` request, reduces its self-transition probability by δ, and increases the transition probability of the (j,i) link by δ. Node i is notified

At each node i:

Initialization
1. $N := \Gamma(i)$
2. $\delta :=$ `Quantum`
3. $p_{ii} = 1 - d_i/\rho$
4. **foreach** $j \in \Gamma(i)$ **repeat**
5. $\quad p_{ij} = 1/\rho$
6. **end foreach**

Random Weight Distribution
1. **while** $p_{ii} \geq \delta$ **and** $N \neq \{\varnothing\}$
2. $\quad j := random(N)$
3. $\quad reply := send\_mesg(j, INCREASE)$
4. $\quad$ **if** $reply = ACK$ **then**
5. $\quad\quad p_{ij} := p_{ij} + \delta$
6. $\quad\quad p_{ii} := p_{ii} - \delta$
7. $\quad$ **else**
8. $\quad\quad N := N - j$
9. $\quad$ **end if**
10. **end while**

Receive Message Handler
1. $mesg := receive()$
2. $j := get\_sender(mesg)$
3. $type := get\_type(mesg)$
2. **if** $p_{ii} \geq \delta$ **and** $type = INCREASE$ **then**
3. $\quad p_{ij} := p_{ij} + \delta$
4. $\quad p_{ii} := p_{ii} - \delta$
5. $\quad reply := ACK$
6. **else**
7. $\quad reply := NACK$
8. **end if**

Figure 2. The Random Weight Distribution algorithm.

of the success by an `ACK` message. On receiving the `ACK` message, node i reduces its self-transition probability and increases the transition probability on the (i,j) link. Observe that after this operation the sums of transition probabilities at nodes i and j remain equal to one. Conversely, if the weight of node j is less than δ, it replies with a `NACK` message. On receiving the `NACK` message, node i removes node j from its set N, and does not change its weight. Note that both operations preserve the symmetry of the transition probability on the link between i and j. The following remark follows:

Remark 3.1 *Each step in the RWD algorithm maintains symmetry in the global transition probability matrix P^{rwd}. Therefore, the transition probability matrix remains symmetric when the algorithm terminates. Using Observation 3.1, we see that a random walk based on P^{rwd} will have stationary distribution $\pi_{uniform}$.*

The termination condition of our algorithm, restated below, is used in the analysis in the subsequent section.
Termination condition: the algorithm terminates if either the self-transition probability of the node becomes zero, i.e., it has no more weight to distribute, or if the set N becomes empty, i.e., the weight of all of its neighbors is zero.

Bound on the number setup of messages: At each step in the algorithm the weight is reduced by δ. This parameter determines how many INCREASE messages it takes for the algorithm to reach the final transition probabilities. The following lemma provides a bound on the number of messages.

Lemma 3.1 *The number of* INCREASE *messages is strictly less than* $(1-(d_i/\rho))/\delta + d_i$.

Proof: Note that irrespective of ρ, we can provide a strict bound on the number of messages as follows: $p_{ii} < 1$ for the node to be connected (because, there must be a non-zero transition probability to a neighboring node). Therefore, the number of INCREASE messages per node is strictly less than $1/\delta + d_i$. Note that d_i is added because INCREASE messages which result in NACK messages also need to be counted. There can be at most d_i such messages. Now using the knowledge that $p_{ii} = 1 - d_i/\rho$, we can see that the number of INCREASE messages is strictly less than $(1-(d_i/\rho))/\delta + d_i$. The inequality is strict because the algorithm stops when either a node reduces its weight to zero, in which case there will be no NACKs and the d_i term can be removed, or the node is not able to reduce its weight to zero and receives NACKs from all its neighbors. In the second case the number of messages is exactly $(1-(d_i/\rho)-p_{ii})/\delta + d_i$, which is smaller than the given inequality. Observe that the total number of reply messages (ACK+NACK) is the same as the number of INCREASE messages. □

Note that setting the value of quantum to be as low as 0.025 results in strictly less than $40 + d_i$ INCREASE messages and a corresponding number of ACK+NACK messages per node. This can be considered a small constant overhead. Furthermore, the following optimization can be applied. We know that the neighboring nodes in real-world distributed systems frequently exchange messages, e.g., search queries, heartbeats, and ping-pong messages [10]. All messages from a node are always routed through its neighbors. By reserving a few bits in each message, the communication required for our algorithm can be piggybacked on existing messages. This minimizes the network overhead of our algorithm.

4 Experimental Results

In this section, we study the performance of the Random Weight Distribution (RWD) algorithm in comparison with the distributed adaptations of the Maximum-Degree (MD) and Metropolis-Hastings (MH) algorithms. First, using exact calculation of SLEM (second-largest eigenvalue magnitude) and $t - step$ transition probability matrix evaluation (using matrix multiplication), we quantify that P^{rwd} has a lower SLEM and hence shorter mixing time compared to P^{mh} and P^{md}. We also characterize the performance of the RWD algorithm, with varying values of system parameters δ (increment quantum) and ρ (inverse of initial edge transition weight). Next, we perform random walks using transition probability matrices computed by the three algorithms. In the rest of this section, we abbreviate the previous statement as random walks using RWD, MH, or MD algorithm. By varying the length of the random walks, we show that our algorithm achieves uniform sampling with low standard deviation, while using walks of significantly smaller lengths. Finally, we analyze the network messages (i.e., non-self transition component of the walks) generated by each of the algorithms when they attain uniform stationary distributions. These experimental results show that our algorithm also outperforms MH and MD in terms of the number of messages sent in the network.

4.1 Experimental Setup

Our experiments are based on power-law topologies. In a power-law random graph the node degree distribution follows a power-law distribution, i.e., if the nodes are sorted in descending order of degree then the i^{th} node has degree D/i^{α}, where D is a constant. Such graphs are often used in literature to model large non-uniform network topologies. For example, it is believed [20] that P2P networks conform to such power-law topologies. The parameter $\alpha = 0.8$ is used for our results, unless stated otherwise. This value of α is popularly used in evaluation studies of P2P networks [12]. The topology is constructed by first selecting the degree for each node using the power-law distribution and then connecting them randomly. Motivated by real-world systems [10], we limit the maximum degree of any node in the network to 100. In typical P2P clients such as Limewire [10], such restrictions are often applied to restrict the number of connections of a given node in order to limit the load on the node. For our simulation of random walks, we use 50,000 nodes in the network. To study the properties of the transition probability matrix, which involves the exact calculation via matrix multiplication of SLEM (μ) and number of steps to convergence, we use a topology with 5,000 nodes.

4.2 Convergence to Stationarity

In this section we present our study of the characterization of the transition probability matrices, P^{rwd}, P^{mh}, and

P^{md}. For this experiment, we use a 5,000 node power-law random graph represented as an adjacency matrix. We generate the transition probability matrix by running the three algorithms on the same adjacency matrix. For each of the transition probability matrices, we evaluate the second largest eigenvalue using Matlab. The length of a walk t required for P to converge to the uniform stationary distribution, $|P^t - (1/n)\mathbf{1}\mathbf{1}^T| \leq \varepsilon$, is evaluated using matrix multiplication. We generate multiple transition probability matrices using our algorithm by: (1) varying the quantum as $0.001, 0.005, 0.01, 0.05$, and 0.1, while keeping ρ constant at 200; and (2) varying the value of ρ as $150, 170, 200, 250$, and 300, while keeping the quantum constant at 0.025. Observe that all values of ρ are higher than the maximum degree of the network. As the following results indicate, after a certain threshold, higher values of ρ do not affect the length of the random walk.

In Figure 3, we show the plots of the resulting eigenvalues and steps to convergence when using different values of the quantum. The values observed for the MD and MH algorithm are also plotted as horizontal lines. We see that our algorithm performs better, for all but one value of quantum, in terms of the required number of steps to reach convergence. Furthermore, note that the trend for the required length of walk follows the trend of the SLEM plot. The value of quantum for which our algorithm is outperformed by the other two algorithms is 0.1. The reason for this failure is that such a high quantum leads our algorithm to local maximas when the self-transition probabilities are distributed. Secondly, because the value of quantum is fixed at 0.1, self-transition probabilities, which are just slightly lower than 0.1 remain as such. Note that by choosing quantum as 0.1 versus, for example, 0.025 saves at most 30 messages per node irrespective of ρ. This is a small saving, specially if the messages are piggybacked on existing system messages, in which case the saving is negligible. Thus, a quantum set to 0.025 represents a good trade-off between number of messages and efficiency of the algorithm.

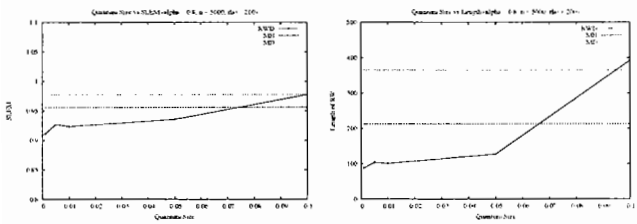

Figure 3. Characterization of P^{rwd} with varying values of quantum, while ρ is fixed at 200. The values corresponding to the MH and MD algorithms are also shown.

In Figure 4, we plot the results for our algorithm using

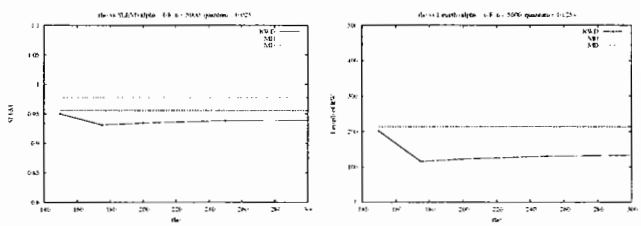

Figure 4. Characterization of P^{rwd} with varying values of ρ, while quantum is fixed at 0.025. The values corresponding to the MH and MD algorithms are also shown.

different values of ρ. using our algorithm. The values observed for the MD and MH algorithms are also plotted as horizontal lines. We see that our algorithm performs better for all values of ρ. However, $\rho = 150$ yields poorer performance compared to that with higher values of ρ. Note that it is required that $\rho > d_{max}$. If the value of ρ is close to d_{max} the algorithm has less degree of freedom in the weight distribution phase. Therefore, simply overestimating ρ is a good heuristic for using our algorithm.

4.3 Length of Random Walks and Uniform Sampling

In this section, we evaluate the effect of the length of random walks and the resulting uniformity of the samples. A practical hindrance for uniform sampling via random walks is that it is difficult to estimate the minimum required TTL or the length of the walk. As stated in [11], the random walk length necessary to achieve stationary distribution has order $O(logn)$. However, the constant in this bound is dependent on the SLEM of the network. As shown in Section 4.2, our algorithm achieves a lower SLEM and hence requires a shorter walk. For a 50,000 node network, we evaluate the constant associated with the required length of the walk for the transition probability matrix generated using RWD, MH and MD algorithms. In [8], Horowitz and Malkhi propose an algorithm to estimate the network size using only local information. This algorithm provides a good estimation of the logarithm of the size of the network. Convergence to stationarity can be found using coupling methods [16]. Once, the length of the random walk of a given random graph is known, it is not expected to change drastically for stable network topologies.

We evaluate the uniformity of the samples by running random walks of increasing length for each algorithm. As a measure of uniformity, we calculate the standard deviation from the expected probability ($1/n = 1/50000 = 0.00002$) (c.f. Definition 1.2). As a comparison, we also present corresponding results for a uniform sample generated using the C function `drand48()`. We use random walks of lengths

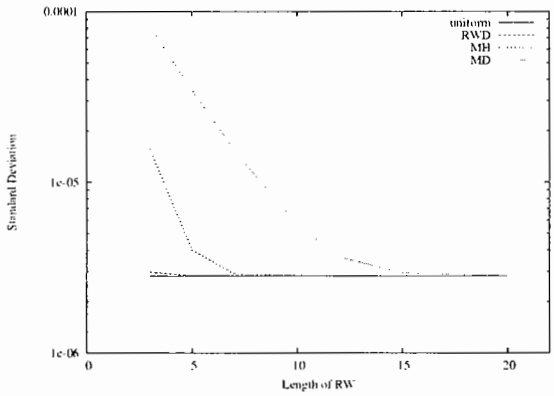

Figure 5. The standard deviation vs length of walk for RWD, MH and MD over a 50,000 node graph. The standard deviation of exact uniform sampling is also provided as a reference.

$3 \times \log n, 5 \times \log n, 7 \times \log n, 10 \times \log n, 15 \times \log n$, and $20 \times \log n$. The number of different random walks is set to $50n$. These results are shown in Figure 5 as the standard deviation vs. the length of the walk for each algorithm (note that the y-axis has a logscale.) The main observation is that the RWD algorithm has a low standard deviation even with a random walk of length $3 \times \log n$. On the other hand, MD has a very high deviation followed by MH. MH takes double the length of the walk required for RWD to converge to stationarity. Similarly, MD requires a walk that is four times longer. Note that once the results converge to a stationary distribution, subsequent hops do not produce a better result.

We now present, in detail, the random sampling achieved by each of the walks using plots of selection probability for each of the nodes in the network. These plots give an important insight into why a longer walk is required for MH and MD algorithms. Figure 6 shows the random sampling achieved by each of RWD, MH, and MD. It is evident by the graphs that some nodes have an extremely high probability of being selected by MH and MD. In fact these nodes are low degree nodes, and hence have high self-transitions. As stated earlier, a bias to low degree nodes during random sampling is not desirable. On the other hand RWD is clearly better, with lower variability. Note that the scales of the plots are different.

Figure 7 shows results for random walk lengths of $5 \times \log n$. We can see that RWD provides a uniform sample with very low variability. This variability is very close to the variability observed for `drand48()`. We also notice that MH and MD samples are still biased towards nodes with high weights (self-transitions).

In Figure 8, we show the uniform sampling with low

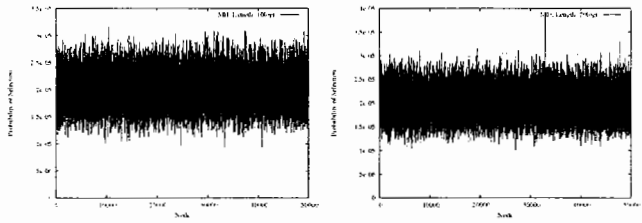

Figure 8. MH converges to uniform distribution when the length of the walk is $10 \times \log n$ and MD converges when the length is $20 \times \log n$. As a comparison, note that RWD converges to uniform distribution when the length of the random walk is only $5 \times \log n$.

variability achieved by MH and MD, which requires $10 log n$ and $20 log n$ steps, respectively.

4.4 Number of Network Messages

While the minimum length of the walk, i.e., the TTL necessary for uniform sampling is a good indicator of the performance of RWD in a distributed setting, it is not obvious that it directly implies low network message overhead in comparison with the other two algorithms. Recall that in a random walk, TTL is decremented even if a self-transition is made. We evaluate the network message overhead of the random walks that achieve uniform sampling for RWD, MH, and MD. The results are summarized in Table 1. We note that RWD requires only 38 network messages out of a walk of total length 54, MH algorithm requires 51 messages out of a walk of length 109, while MD requires 57 messages out of 216. The key observation is that the absolute value of the network messages involved in uniform sampling in MH and MD is much higher than for RWD, despite the fact that network messages constitute only a small portion of their walks. Although, the total length of the random walk does not translate exactly to network messages, it has associated computational overhead, which may be significant if the nodes that have low processing capability, e.g., in sensor networks.

We repeat the above experiments using network topologies with varying levels of non-uniformity. This is done by changing the value of α to 0.7, 0.9, 1.1, and 1.2. The results follow the same trend as discussed above but are not included here due to limitations of space.

As a final remark, the experimental results show the efficacy of RWD algorithm in terms of its non-biased sampling, shorter length of the walk, and fewer network messages required for sampling.

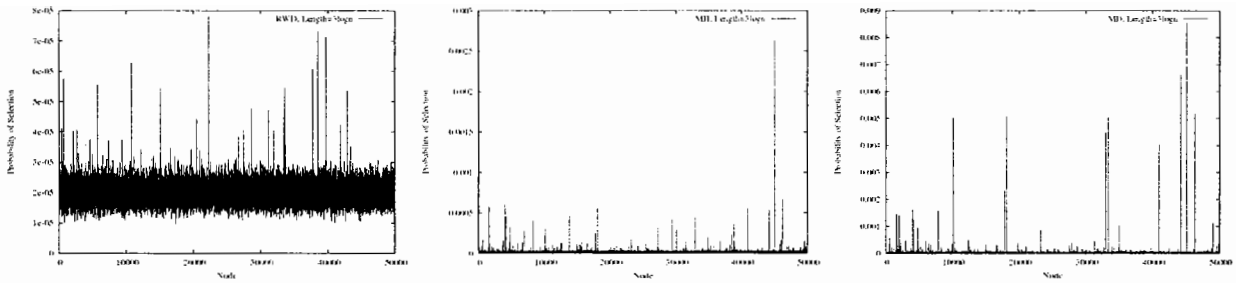

Figure 6. Random sampling achieved with a walk of length $3 \times \log n$ for RWD, MH and MD.

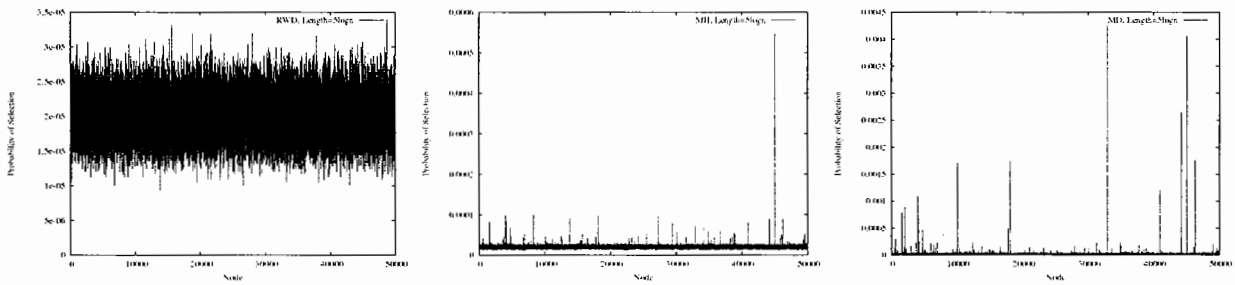

Figure 7. Random sampling with a walk of length $5 \times \log n$ for RWD, MH and MD. Note that RWD achieves a random sample with probability close to uniform ($\pi_{uniform} = 1/50000$).

Algorithm	Network Msgs	Percentage of total walk
RWD	38	70.4
MH	51	46.8
MD	57	21.6

Table 1. Network messages corresponding to random walk lengths used for uniform sampling.

5. Related Work

Structured peer-to-peer (P2P) networks [21, 22, 18, 13, 17] provide strong guarantees for search by imposing a well defined topology on the network. Peers and objects are assigned hash-based identifiers and objects are assigned to peers based on these identifiers. Objects in the network are found by performing efficient routing protocols that lead to the peer responsible for storing pointers to the objects. Node identifiers, generated via a hash function, are assumed to be uniformly distributed in the identifier space. A simple algorithm for choosing a random peer in these networks would be to select a random number u in the identifier space and route to the node responsible for u. King and Saia [9] show that this simple algorithm leads to biased samples. To solve this problem, they propose a more robust algorithm that always chooses each peer with probability exactly $1/n$ and that has $O(\log n)$ expected message complexity.

In [5], Gkantsidis *et al.* perform an extensive study of random walks in P2P networks. The authors explore the performance of random walks for searching and uniform sampling. For searching, the authors show that random walks perform better than flooding when the length of the random walk is the same as the number of peers covered by flooding with bounded TTL. Another important result in the paper is that it is possible to simulate the selection of a uniform sample of elements by performing a random walk of required length on a particular class of network topologies. The results presented, however, apply only to expander graphs, where the degree of the nodes is constant. This is in contrast with our work, since our focus is on general topologies, including power-law graphs.

Boyd *et al.* [2] formulate the problem of finding the fastest mixing Markov chain on a graph as a convex optimization. The problem is expressed as a semidefinite program (SDP), and the solution of the SDP yields the global optimal probabilities in the transition matrix. They also show that the Metropolis-Hasting and the maximum degree algorithms are substantially slower than the optimum. While this technique is useful for finding the optimal mixing time, it cannot be directly used in a distributed setting because of the overheads associted with solving the SDP.

Uniform sampling has also been investigated in different

large-scale applications. In [7], for example, Henzinger *et al.* propose a method for sampling web pages with near uniform distribution. The algorithm uses a random walk with the edge weights changed according to the page rank of the pages. Page rank is a measure of the popularity of a web page and is used by search engines to rank search results. The sample produced by the algorithm is not truly uniform and appears biased toward web pages with high numbers of inbound links. Sampling from web pages, however, poses additional challenges when compared with sampling in a network, since the connections form a directed graph with a nonsymmetric adjacency matrix. In [1], Bash *et al.* investigate the problem of approximating a uniform random sample in sensor networks. Their algorithm uses geographic routing and Voronoi diagrams. While this approach is applicable to sensor networks, it does not directly apply to unstructured peer-to-peer networks, where geographic routing is not feasible.

6 Conclusion

In this paper we address key challenges in uniform sampling using random walks. Based on a Markov chain abstraction we describe the necessary conditions for uniform sampling in non-uniform networks, and examine two known algorithms that result in uniform random sampling via random walks. We present a new algorithm called Random Weight Distribution (RWD), which results in uniform sampling while minimizing the length of the random walk. Using a comprehensive simulation study we support claims of superior performance of our algorithm compared to existing algorithms, using parameters most relevant in a distributed setting.

References

[1] B. A. Bash, J. W. Byers, and J. Considine. Approximately uniform random sampling in sensor networks. In *Proceedings of the International Workshop on Data Management for Sensor Networks (DMSN 2004)*, Toronto, Canada, August 2004.

[2] S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing Markov chain on a graph. In *Submitted to SIAM Review, problems and techniques section.*, February 2003.

[3] P. Brémaud. *Markov Chains Gibs Fields, Monte Carlo Simulation, and Queues*. Springer-Verlag, 1999.

[4] D. Ganesan, S. Ratnasamy, H. Wang, and D. Estrin. Coping with irregular spatio-temporal sampling in sensor networks. In *Proceedings of HotNets-II*, November 2003.

[5] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *Proceedings of the 23rd IEEE Infocom, 2004*, Hong Kong, March 2004.

[6] W. Hastings. Monte carlo sampling methods using Markov chains and their applications. In *Biometrika*, volume 57, pages 97–109. 1970.

[7] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. On near-uniform URL sampling. In *Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications networking*, pages 295–308, Amsterdam, May 2000.

[8] K. Horowitz and D. Malkhi. Estimating network size from local information. In *The Information Processing Letters journal 88(5)*, pages 237–243, December 2003.

[9] V. King and J. Saia. Choosing a Random Peer. In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC'04)*, pages 125 – 130, Newfoundland, Canada, July 2004.

[10] Limewire. http://www.limewire.com/english/content/glossary.shtml.

[11] L. Lovasz. Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty (Volume 2)*, pages 1–46, 1993.

[12] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *ACM ICS'02 Conference*, New York, NY, USA, June 2002.

[13] D. Malkhi, M. Naor, and et al. Viceroy: A Scalable and Dynamic Emulation of the Butterfly. In *Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing (PODC'02)*, Monterey, CA, July 2002.

[14] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations byfast computing machines. In *J. Chem. Phys.*, volume 21, pages 1087–101. 1953.

[15] R. Motwani and P. Raghavan. In *Randomized Algorithms*. Cambridge University Press, 1995.

[16] J. Propp and D. Wilson. Exact sampling with coupled markov chains and applications to statistical mechanics. In *Proceedings of the seventh international conference on Random structures and algorithms*, pages 223–252. John Wiley & Sons, Inc., 1996.

[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proceedings of the 2001 ACM SIGCOMM*, pages 247–254, San Diego, CA, August 2001.

[18] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the 2001 ACM SIGCOMM*, pages 247–254, San Diego, CA, August 2001.

[19] S. Saroiu, K. P. Gummadi, and S. D. Gribble. Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts. *Multimedia Systems*, 9(2):170 – 184, 2003.

[20] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, January 2002.

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

[22] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing. Technical Report UCB/CSD-0101141, UC Berkeley, Computer Science Division, April 2001.