Node Sampling using Random Centrifugal Walks

Andrés Sevilla , Alberto Mozo , and Antonio Fernández Anta

Abstract. Sampling a network with a given probability distribution has been identified as a useful operation. In this paper we propose distributed algorithms for sampling networks, so that nodes are selected by a special node, called the *source*, with a given probability distribution. All these algorithms are based on a new class of random walks, that we call *Random Centrifugal Walks* (RCW). A RCW is a random walk that starts at the source and *always* moves *away* from it.
Firstly, an algorithm to sample any connected network using RCW is proposed. The algorithm assumes that each node has a weight, so that the sampling process must select a node with a probability proportional to its weight. This algorithm requires a preprocessing phase before the sampling of nodes. In particular, a minimum diameter spanning tree (MDST) is created in the network, and then nodes' weights are efficiently aggregated using the tree. The good news are that the preprocessing is done only once, regardless of the number of sources and the number of samples taken from the network. After that, every sample is done with a RCW whose length is bounded by the network diameter.
Secondly, RCW algorithms that do not require preprocessing are proposed for grids and networks with regular concentric connectivity, for the case when the probability of selecting a node is a function of its distance to the source.
The key features of the RCW algorithms (unlike previous Markovian approaches) are that (1) they do not need to warm-up (stabilize), (2) the sampling always finishes in a number of hops bounded by the network diameter, and (3) it selects a node with the *exact probability distribution*.

1 Introduction

Sampling a network with a given distribution has been identified as a useful operation in many contexts. For instance, sampling nodes with uniform probability is the building block of epidemic information spreading [13,12]. Similarly, sampling with a probability that depends on the distance to a given node [3,17] is useful to construct small world network topologies [14,7,2]. Other applications that can benefit from distance-based node sampling are landmark-less network positioning systems like NetICE9 [16], which does sampling of nodes with special properties to assign synthetic coordinates to nodes. In a different context, currently there is an increasing interest in obtaining a representative (unbiased) sample from the users of online social networks [9]. In this paper we propose a distributed algorithm for sampling networks with a desired probability distribution.

Related Work One technique to implement distributed sampling is to use gossiping between the network nodes. Jelasity et al. [12] present a general framework to implement a uniform sampling service using gossip-based epidemic algorithms. Bertier et al. [2] implement uniform sampling and DHT services using gossiping. As a side result, they sample nodes with a distribution that is close to Kleinberg's harmonic distribution (one instance of a distance-dependent distribution). Another gossip-based sampling service that gets close to Kleinberg's harmonic distribution has been proposed by Bonnet et al. [3]. However, when using gossip-based distributed sampling as a service, it has been shown by Busnel et al. [5] that only partial independence (ϵ-independence) between views (the subsets of nodes held at each node) can be guaranteed without re-executing the gossip algorithm. Gurevich and Keidar [11] give an algorithm that achieves ϵ-independence in $O(ns \log n)$ rounds (where n is the network size and s is the view size).

Another popular distributed technique to sample a network is the use of random walks [18]. Most random-walk based sampling algorithms do uniform sampling [1,9], usually having to deal with the irregularities of the network. Sampling with arbitrary probability distributions can be achieved with random walks by re-weighting the hop probabilities to correct the sampling bias caused by the non-uniform stationary distribution of the random walks. Lee et al. [15] propose two new algorithms based on Metropolis-Hastings (MH) random walks for sampling with any probability distribution. These algorithms provide an unbiased graph sampling with a small overhead, and a smaller asymptotic variance of the resulting unbiased estimators than generic MH random walks.

Sevilla et al. [17] have shown how sampling with an arbitrary probability distribution can be done without communication if a uniform sampling service is available. In that work, as in all the previous approaches, the desired probability distribution is reached when the stationary distribution of a Markov process is reached. The number of iterations (or hops of a random walk) required to reach this situation (the warm-up time) depends on the parameters of the network and the desired distribution, but it is not negligible. For instance, Zhong and Sheng [18] found by simulation that, to achieve no more than 1% error, in a torus of 4096 nodes at least 200 hops of a random walk are required for the uniform distribution, and 500 hops are required for a distribution proportional to the inverse of the distance. Similarly, Gjoka et al. [10] show that a MHRW sampler needs about 6K samples (or 1000-3000 iterations) to obtain the convergence to the uniform probability distribution. In the light of these results, Markovian approaches seem to be inefficient to implement a sampling service, specially if multiple samples are desired.

Contributions In this paper we present efficient distributed algorithms to implement a sampling service. The basic technique used for sampling is a new class of random walks that we call *Random Centrifugal Walks* (RCW). A RCW starts at a special node, called the *source*, and *always* moves *away* from it.

All the algorithms proposed here are instances of a generic algorithm that uses the RCW as basic element. This generic RCW-based algorithm works essentially as follows. A RCW always starts at the source node. When the RCW reaches a node x (the first node reached by a RCW is always the source s), the RCW stops at that node with a *stay probability*. If the RCW stops at node x, then x is the node selected by the sampling. If the RCW does not stop at x, it jumps to a neighbor of x. To do so, the RCW chooses only among neighbors that are farther from the source than the node x. (The probability of jumping to each of these neighbors is not necessarily the same.) In the rest of the paper we will call all the instances of this generic algorithm as *RCW algorithms.*

Firstly, we propose a RCW algorithm that samples *any* connected network with *any* probability distribution (given as weights assigned to the nodes). Before starting the sampling, a preprocessing phase is required. This preprocessing involves building a minimum distance spanning tree (MDST) in the network[4], and using this tree for efficiently aggregating the node's weights. As a result of the weight aggregation, each node has to maintain one numerical value per link, which will be used by the RCW later. Once the preprocessing is completed, any node in the network can be the source of a sampling process, and multiple independent samplings with the exact desired distribution can be efficiently performed. Since the RCW used for sampling follow the MDST, they take at most D hops (where D is the network diameter).

Secondly, when the probability distribution is distance-based and nodes are at integral distances (measured in hops) from the source, RCW algorithms without preprocessing (and only a small amount of state data at the nodes) are proposed. In a *distance-based probability distribution* all the nodes at the same distance from the source node are selected with the same probability. (Observe that the uniform and Kleinberg's harmonic distributions are special cases of distance-based probability distributions.) In these networks, each node at distance $k > 0$ from the source has neighbors (at least) at distance $k - 1$. We can picture nodes at distance k from the source as positioned on a ring at distance k from the source. The center of all the rings is the source, and the radius of each ring is one unit larger than the previous one. Using this graphical image, we refer the networks of this family as *concentric rings networks.* [5]

[4] Using, for instance, the algorithm proposed by Bui et al. [4] whose time complexity is $O(n)$.

[5] Observe that *every* connected network can be seen as a concentric rings network. For instance, by finding the breadth-first search (BFS) tree rooted at the source, and using the number of hops in this tree to the source as distance.

The first distance-oriented RCW algorithm we propose samples with a distance-based distribution in a network with grid topology. In this network, the source node is at position $(0, 0)$ and the lattice (Manhattan) distance is used. This grid contains all the nodes that are at a distance no more than the radius R from the source (the grid has hence a diamond shape[6]). The algorithm we derive assigns a stay probability to each node, that only depends on its distance from the source. However, the hop probabilities depend on the position (i, j) of the node and the position of the neighbors to which the RCW can jump to. We formally prove that the desired distance-based sampling probability distribution is achieved. Moreover, since every hop of the RCW in the grid moves one unit of distance away from the source, the sampling is completed after at most R hops.

We have proposed a second distance-oriented RCW algorithm that samples with distance-based distributions in concentric rings networks *with uniform connectivity*. These are networks in which all the nodes in each ring k have the same number of neighbors in ring $k-1$ and the same number in ring $k+1$. Like the grid algorithm, this variant is also proved to finish with the desired distribution in at most R hops, where R is the number of rings.

Unfortunately, in general, concentric rings networks have no uniform connectivity. This case is faced by creating, on top of the concentric rings network, an overlay network that has uniform connectivity. In the resulting network, the algorithm for uniform connectivity can be used. We propose a distributed algorithm that, if it completes successfully, builds the desired overlay network. We have found via simulations that this algorithm succeeds in building the overlay network in a large number of cases.

In summary, RCW can be used to implement an efficient sampling service because, unlike previous Markovian (e.g., classical random walks and epidemic) approaches, (1) it always finishes in a number of hops bounded by the network diameter, (2) selects a node with the *exact probability distribution*, and (3) does not need warm-up (stabilization) to converge to the desired distribution. Additionally, in the case that preprocessing is needed, this only has to be executed once, independently on the number of sources and the number of samples taken from the network.

The rest of the paper is structured as follows. In Section 2 we introduce concepts and notation that will be used in the rest of the paper. In Section 3 we present the RCW algorithm for a connected network. In Sections 4 and 5 we describe the RCW algorithm on grids and concentric rings networks with uniform connectivity. In Section 6 we present the simulation based study of the algorithm for concentric rings topologies without uniform connectivity. Finally, we conclude the paper in Section 7.

2 Definitions and Model

Connected Networks In this paper we only consider connected networks. This family includes most of the potentially interesting networks we can find. In every network, we use N to denote the set of nodes and $n = |N|$ the size of that set. When convenient, we assume that there is a special node in the network, called the *source* and denoted by s. We assume that each node $x \in N$ has an associated weight $w(x) > 0$. Furthermore, each node *knows* its own weight. The weights are used to obtain the desired probability distribution p, so that the probability of selecting a node x is proportional to $w(x)$. Let us denote $\eta = \sum_{j \in N} w(j)$. Then, the probability of selecting $x \in N$ is $p(x) = w(x)/\eta$. (In the simplest case, weights are probabilities, i.e., $w(x) = p(x), \forall x$ and $\eta = 1$.)

RCW in Connected Networks As mentioned, in order to use RCW to sample connected networks, some preprocessing is done. This involves constructing a spanning tree in the network and performing a weight aggregation process. After the preprocessing, RCW is used for sampling. A RCW starts from the source. When the RCW reaches a node $x \in N$, it selects x as the sampled vertex with probability $q(x)$, which we call the *stay probability*. If x is not selected, a neighbor y of x in the tree is chosen, using for that a collection of *hop probabilities* $h(x, y)$. The values of $q(x)$ and $h(x, y)$ are computed in the preprocessing and stored at x. The probability of reaching a node $x \in N$ in a RCW is called the *visit probability*, denoted $v(x)$.

[6] A RCW algorithm for a square grid is easy to derive from the one presented.

Concentric Rings Networks We also consider a subfamily of the connected networks, which we call *concentric rings networks.* These are networks in which the nodes of N are at integral distances from s. In these networks, no node is at a distance from s larger than a radius R. For each $k \in [0, R]$, we use $\mathbb{R}_k \neq \emptyset$ to denote the set of nodes at distance k from s, and $n_k = |\mathbb{R}_k|$. (Observe that $\mathbb{R}_0 = \{s\}$ and $n_0 = 1$.) These networks can be seen as a collection of concentric rings at distances 1 to R from the source, which is the common center of all rings. For that reason, we call the set $\mathbb{R}_k$ the *ring at distance* k. For each $x \in \mathbb{R}_k$ and $k \in [1, R]$, $\gamma_k(x) > 0$ is the number of neighbors of node x at distance $k-1$ from s (which is only 1 if $k = 1$), and $\delta_k(x)$ is the number of neighbors of node x at distance $k+1$ from s (which is 0 if $k = R$).

The concentric rings networks considered must satisfy the additional property that the probability distribution is *distance based.* This means that, for all $k \in [0, R]$, every node $x \in \mathbb{R}_k$ has the same probability p_k to be selected. We assume that each node $x \in \mathbb{R}_k$ knows its own p_k. These properties allow, in the subfamilies defined below, to avoid the preprocessing required for connected networks.

Grids A first subfamily of concentric rings networks considered is the grid with lattice distances. In this network, the source is at position $(0, 0)$ of the grid, and it contains all the nodes (i, j) so that $i, j \in [-R, R]$ and $|i| + |j| \leq R$. For each $k \in [0, R]$, the set of nodes in ring k is $\mathbb{R}_k = \{(i, j) : |i| + |j| = k\}$. The neighbors of a node (i, j) are the nodes $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, and $(i, j+1)$ (that belong to the grid).

Uniform Connectivity The second subfamily considered is formed by the concentric rings networks with *uniform connectivity.* These networks satisfy that

$$\forall k \in [1, R], \forall x, y \in \mathbb{R}_k, \delta_k(x) = \delta_k(y) \wedge \gamma_k(x) = \gamma_k(y). \tag{1}$$

In other words, all nodes of ring k have the same number of neighbors δ_k in ring $k+1$ and the same number of neighbors γ_k in ring $k-1$.

RCW in Concentric Rings Networks The behavior of a generic RCW was already described. In the algorithm that we will present in this paper for concentric rings networks we guarantee that, for each k, all the nodes in $\mathbb{R}_k$ have the same visit probability v_k and the same stay probability q_k. A RCW starts from the source. When it reaches a node $x \in \mathbb{R}_k$, it selects x as the sampled vertex with stay probability q_k. If x is not selected, a neighbor $y \in \mathbb{R}_{k+1}$ of x is chosen.

The desired *distance-based probability distribution* is given by the values p_k, $k \in [0, R]$, where it must hold that $\sum_{k=0}^{R} n_k \times p_k = 1$. The problem to be solved is to define the stay and hop probabilities so that the probability of a node $x \in \mathbb{R}_k$ is p_k.

Observation 1 *If for all $k \in [0, R]$ the visit v_k and stay q_k probabilities are the same for all the nodes in $\mathbb{R}_k$, the RCW samples with the desired probability iff $p_k = v_k \cdot q_k$.*

3 Sampling in a Connected Network

In this section, we present a RCW algorithm that can be used to sample any connected network. As mentioned, in addition to connectivity, it is required that each node knows its own weight. A node will be selected with probability proportional to its weight.

Preprocessing for the RCW Algorithm The RCW algorithm for connected networks requires some preprocessing which will be described now. This preprocessing has to be done only once for the whole network, independently of which nodes act as sources and how many samples are taken.

Building a spanning tree. Initially, the algorithm builds a spanning tree of the network. A feature of the algorithm is that, if several nodes want to act as sources for RCW, they can all share the same spanning tree. Hence only one tree for the whole network has to be built. The algorithm used for the tree construction is not important for the correctness of the RCW algorithm, but the diameter of the tree will be an upper bound on the length of the RCW (and hence possibly the sampling latency). There are several well known distributed algorithms (see, e.g., [6] and the references therein) that can be used to build the spanning tree.

```
1  task Weight_Aggregation(i)
2  if i is a leaf then
3    send WEIGHT(w(i)) to neighbor x
4    receive WEIGHT(p) from neighbor x
5    T_i(x) ← p
6  else
7    repeat
8      receive WEIGHT(p) from x ∈ neighbors(i)
9      T_i(x) ← p
10     foreach y ∈ neighbors(i) \ {x} do
11       if received WEIGHT(·) from neighbors(i) \ {y} then
12         send WEIGHT(w(i) + Σ_{z∈neighbors(i)\{y}} T_i(z)) to y
13     end foreach
14   until received WEIGHT(·) from every x ∈ neighbors(i)
```

Fig. 1. Weight aggregation algorithm. Code for node i.

In particular, it is interesting to build a minimum diameter spanning tree (MDST) because, as mentioned, the length of the RCW is upper bounded by the tree diameter. There are few algorithms in the literature to build a MDST. One possible candidate to be used in our context is the one proposed by Bui et al. [4]. Additionally, if link failures are expected, the variation of the former algorithm proposed by Gfeller et al. [8] can be used.

Weight aggregation. Once the spanning tree is in place, the nodes compute and store aggregated weights using the algorithm of Figure 1. The algorithm executes at each node $i \in N$, and it computes in a distributed way the aggregated weight of each subtree that can be reached following one of the links of i. In particular, for each node x that is in the set of neighbors of i in the tree, $neighbors(i)$, the algorithm computes a value $T_i(x)$ and stores it at i. Let (i, x) be a link of the spanning tree, then by removing the link (i, x) from the spanning tree there are two subtrees. We denote by $stree(x, i)$ the subtree out of them that contains node x.

Theorem 1. *After the completion of the Weight Aggregation algorithm (of Figure 1), each node $i \in N$ will store, for each node $x \in neighbors(i)$, in $T_i(x)$ the value $\sum_{y \in stree(x,i)} w(y)$.*

Proof. Consider $stree(x, i)$ a tree rooted at x. We prove the claim by induction in the depth of this tree. The base case is when the tree has depth 1. In this case x is a leaf and, from the algorithm, it sends to i its weight $w(x)$, which is stored at i as $T_i(x)$. If the depth is $k > 1$, by induction hypothesis x ends up having in $T_x(y)$ the sum of the weight of the subtree $stree(x, y)$, for each $y \in neighbors(x) \setminus \{i\}$. These values plus $w(x)$ are added up and sent to i, which stores the resulting value as $T_i(x)$.

The values $T_i(x)$ computed in this preprocessing phase will later be used by the RCW algorithm to perform the sampling. We can bound now the complexity of this process in terms of messages exchanged and time to complete. We assume that all nodes start running the Weight Aggregation algorithm simultaneously, that the transmission of messages takes one step, and that computation time is negligible.

Theorem 2. *The Weight Aggregation algorithm (of Figure 1) requires $2(n-1)$ messages to be exchanged, and completes after D steps, where D is the diameter of the tree.*

Proof. It is easy to observe in the algorithm that one message is sent across each link in each direction. Since all spanning trees have $n-1$ links, the first part of the claim follows.

The second claim can be shown as follows. Let us consider any node i as the root of the spanning tree. Let d be the largest distance in the tree of any node from i. We show by induction on k that all nodes at distance $d-k$ from i have received the aggregated weight of their corresponding subtrees by step k. The base case is $k = 1$, which follows since the leaves at distance d send their weights to their parents in the first step. Consider now any (non-leaf) node j at distance $d-k+1$ from i. Assume that y is the parent (at

1 **task** $RCW(i)$
2 **when** $RCW\_MSG(s)$ **received from** x
3 $candidates \leftarrow neighbors(i) \setminus \{x\}$
4 **with** *probability* $q(i) = \frac{w(i)}{w(i)+\sum_{z \in candidates} T_i(z)}$ **do**
5 *select node* i *and report to source* s
6 **otherwise**
7 *choose a node* $y \in candidates$ **with** *probability* $h(i,y) = \frac{T_i(y)}{\sum_{z \in candidates} T_i(z)}$
8 **send** $RCW\_MSG(s)$ *to* y

Fig. 2. RCW algorithm for connected networks. Code for node i.

distance $d-k$) of j in the tree rooted at i. By induction hypothesis j has received all the aggregated weights of the subtrees by step $k-1$. Then, when the latest such value was received from a neighbor x (Line 8), the foreach loop (Lines 10-13) if executed. In this execution, the condition of the if statement at Line 11 is satisfied for y. Then, the aggregated weight $w(j) + \sum_{z \in neighbors(j) \setminus \{y\}} T_j(z)$ is sent to y by step $k-1$. This value reaches y in one step, by step k. Then, i receives all the aggregated weights by step d. Since the largest value of d is D, the proof is complete.

RCW Sampling Algorithm In this RCW algorithm (Figure 2) any node can be the source. The spanning tree and the precomputed aggregated weights are used by any node to perform the samplings (as many as needed). The sampling process in the RCW algorithm works as follows. To start the process, the source s sends a message $RCW\_MSG(s)$ to itself. When the $RCW\_MSG(s)$ message is received by a node i from a node x, it computes a set of candidates for next hop in the RCW, which are all the neighbors of i except x. Then, the RCW stops and selects that node with a *stay probability* $q(i) = \frac{w(i)}{w(i)+\sum_{z \in candidates} T_i(z)}$ (Line 4). If the RCW does not select i, it jumps to a neighbor of i different from x. To do so, the RCW chooses only among nodes y in the set of candidates (that move away from s) using $h(i,y) = \frac{T_i(y)}{\sum_{z \in candidates} T_i(z)}$ as hop probability (Line 7).

Analysis We show now that the algorithm proposed performs sampling with the desired probability distribution.

Theorem 3. *Each node* $i \in N$ *is selected by the RCW algorithm with probability* $p(i) = \frac{w(i)}{\eta}$.

Proof. If a node i receives the $RCW\_MSG(s)$ from x, let us define $candidates = neighbors(i) \setminus \{x\}$, and $T(i) = w(i) + \sum_{z \in candidates} T_i(z)$. We prove the following stronger claim: Each node $i \in N$ is visited by the RCW with probability $v(i) = \frac{T(i)}{\eta}$ and selected by the RCW algorithm with probability $p(i) = \frac{w(i)}{\eta}$.

We prove this claim by induction on the number of hops from the source s to node i in the spanning tree. The base case is when the node i is the source s. In this case x is also s, $candidates = neighbors(s)$, and $T(s) = \eta$. Hence, $v(s) = \frac{T(s)}{\eta} = 1$ and $q(s) = \frac{w(s)}{\eta}$, yielding $p(s) = \frac{w(s)}{\eta}$.

The induction hypothesis assumes the claim true for a node x at distance k from s, and proves the claim for i which is at distance $k+1$. We have that $\Pr[\text{visit } i] = v(x)\,(1-q(x))\,\frac{T(i)}{T(x)-w(x)}$, where $1-q(x)$ is the probability of not selecting node x when visiting it, and $\frac{T(i)}{T(x)-w(x)}$ is the probability of choosing the node i in the next hop of the RCW. The stay probability of x and i are $q(x) = w(x)/T(x)$ and $q(i) = w(i)/T(i)$, respectively (Line 4). Then, $v(i) = \frac{T(x)}{\eta}\left(1 - \frac{w(x)}{T(x)}\right)\frac{T(i)}{T(x)-w(x)} = \frac{T(x)}{\eta}\left(\frac{T(x)-w(x)}{T(x)}\right)\frac{T(i)}{T(x)-w(x)} = \frac{T(i)}{\eta}$ and $\Pr[\text{select } i] = v(i)q(i) = \frac{T(i)}{\eta}\frac{w(i)}{T(i)} = \frac{w(i)}{\eta}$.

4 Sampling in a Grid

If the algorithm for connected networks is applied to a grid, given its regular structure, the construction of the spanning tree could be done without any communication among nodes, but the weight aggregation

process has to be done as before. However, we show in this section that all preprocessing and the state data stored in each node can be avoided if the probability distribution is based on the distance. RCW sampling process was described in Section 2, and we only redefine stay and hop probabilities. From Observation 1, the key for correctness is to assign stay and hop probabilities that guarantee visit and stay probabilities that are homogenous for all the nodes at the same distance from the source.

Stay probability For $k \in [0, R]$, the stay probability of every node $(i, j) \in \mathbb{R}_k$ is defined as

$$q_k = \frac{n_k \cdot p_k}{\sum_{j=k}^{R} n_j \cdot p_j} = \frac{n_k \cdot p_k}{1 - \sum_{j=0}^{k-1} n_j \cdot p_j}. \tag{2}$$

As required by Observation 1, all nodes in $\mathbb{R}_k$ have the same q_k. Note that $q_0 = p_0$ and $q_R = 1$, as one may expect. Since the value of p_k is known at $(i, j) \in \mathbb{R}_k$, n_k can be readily computed[7], and the value of $\sum_{j=0}^{k-1} n_j \cdot p_j$ can be piggybacked in the RCW, the value of q_k can be computed and used at (i, j) without requiring precomputation nor state data.

Hop probability In the grid, the hops of a RCW increase the distance from the source by one unit. We want to guarantee that the visiting probability is the same for each node at the same distance, to use Observation 1. To do so, we need to observe that nodes (i, j) over the axes (i.e., with $i = 0$ or $j = 0$) have to be treated as a special case, because they can only be reached via a single path, while the others nodes can be reached via several paths. To simplify the presentation, and since the grid is symmetric, we give the hop probabilities for one quadrant only (the one in which nodes have both coordinates non-negative). The hop probabilities in the other three quadrants are similar. The first hop of each RCW chooses one of the four links of the source node with the same probability 1/4. We have three cases when calculating the hop probabilities from a node (i, j) at distance k, $0 < k < R$, to node (i', j').

- *Case A:* The edge from (i, j) to (i', j') is in one axis (i.e., $i = i' = 0$ or $j = j' = 0$). The hop probability of this link is set to $h_k((i, j), (i', j')) = \frac{i+j}{i+j+1} = \frac{k}{k+1}$.
- *Case B:* The edge from (i, j) to (i', j') is not in the axes, $i' = i + 1$, and $j' = j$. The hop probability of this link is set to $h_k((i, j), (i + 1, j)) = \frac{2i+1}{2(i+j+1)} = \frac{2i+1}{2(k+1)}$.
- *Case C:* The edge from (i, j) to (i', j') is not in the axes, $i' = i$, and $j' = j + 1$. The hop probability of this link is set to $h_k((i, j), (i, j + 1)) = \frac{2j+1}{2(i+j+1)} = \frac{2j+1}{2(k+1)}$.

It is easy to check that the hop probabilities of a node add up to one.

Analysis In the following we prove that the RCW that uses the above stay and hop probabilities selects nodes with the desired sample probability.

Lemma 1. *All nodes at the same distance $k \geq 0$ to the source have the same visit probability v_k.*

Proof. The proof uses induction. The base case is $k = 0$, and obviously $v_k = 1$. When $k = 1$, the probability of visiting each of the four nodes at distance 1 from the source s is $v_i = \frac{1-q_0}{4}$, where $1 - q_0$ is the probability of not staying at source node. Assuming that all nodes at distance $k > 0$ have the same visit probability v_k, we prove the case of distance $k + 1$. Recall that the stay probability is the same q_k for all nodes at distance k.

The probability to visit a node $x = (i', j')$ at distance $k + 1$ depends on whether x is on an axis or not. If it is in one axis it can only be reached from its only neighbor (i, j) at distance k. This happens with probability (case A) $\Pr[\text{visit } x] = v_k(1 - q_k)\frac{i+j}{i+j+1} = v_k(1 - q_k)\frac{k}{k+1}$. If x is not on an axis, it can be reached from two nodes, $(i' - 1, j')$ and $(i', j' - 1)$, at distance k (Cases B and C). Hence, the probability of reaching x is then $\Pr[\text{visit } x] = v_k(1 - q_k)\frac{2(i'-1)+1}{2(i'+j')} + v_k(1 - q_k)\frac{2(j'-1)+1}{2(i'+j')} = v_k(1 - q_k)\frac{k}{k+1}$. Hence, in both cases the visit probability of a node x at distance $k + 1$ is $v_{k+1} = v_k(1 - q_k)\frac{k}{k+1}$. This proves the induction and the claim.

[7] $n_0 = 1$, while $n_k = 4k$ for $k \in [1, R]$.

```
1  task RCW(x, k, δ_k, γ_k, p_k)
2  when RCW_MSG(s, v_{k-1}, p_{k-1}, n_{k-1}, δ_{k-1}) received:
3     n_k ← n_{k-1} δ_{k-1}/γ_k;   v_k ← n_{k-1} (v_{k-1} − p_{k-1})/n_k;   q_k ← p_k/v_k
4     with probability q_k do select node x and report to s
5     otherwise
6        choose a neighbor y in ring k + 1 with uniform probability
7        send RCW_MSG(s, v_k, p_k, n_k, δ_k) to y
```

Fig. 3. RCW algorithm for concentric rings with uniform connectivity (code for node $x \in \mathbb{R}_k$, $k > 0$).

Theorem 4. *Every node at distance $k \in [0, R]$ from the source is selected with probability p_k.*

Proof. If a node is visited at distance k, it is because no node was selected at distance less than k, since a RCW always moves away from the source. Hence, $\Pr[\exists x \in \mathbb{R}_k \text{ visited}] = 1 - \sum_{j=0}^{k-1} n_j p_j$. Since all the n_k nodes in $\mathbb{R}_k$ have the same probability to be visited (from the previous lemma), we have that $v_k = \frac{1-\sum_{j=0}^{k-1} n_j p_j}{n_k}$. Now, since all the n_k nodes in $\mathbb{R}_k$ have the same stay probability is q_k, the probability of selecting a particular node x at distance k from the source is $\Pr[\text{select } x] = v_k q_k = \frac{1-\sum_{j=0}^{k-1} n_j p_j}{n_k} \frac{n_k p_k}{\sum_{j=k}^{R} n_j p_j} = p_k$, where it has been used that $(1 - \sum_{j=0}^{k-1} n_j p_j) = \sum_{j=k}^{R} n_j p_j$.

5 Sampling in a Concentric Rings Network with Uniform Connectivity

In this section we derive a RCW algorithm to sample a concentric rings network with uniform connectivity, where all preprocessing is avoided, and only a small (and constant) amount of data is stored in each node. Recall that uniform connectivity means that all nodes of ring k have the same number of neighbors δ_k in ring $k+1$ and the same number of neighbors γ_k in ring $k-1$.

Distributed algorithm The general behavior of the RCW algorithm for these networks was described in Section 2. In order to guarantee that the algorithm is fully distributed, and to reduce the amount of data a node must know a priori, a node at distance k that sends the RCW to a node in ring $k+1$ piggybacks some information. More in detail, when a node in ring k receives the RCW from a node of ring $k-1$, it also receives the probability v_{k-1} of the previous step, and the values p_{k-1}, n_{k-1}, and δ_{k-1}. Then, it calculates the values of n_k, v_k, and q_k. After that, the RCW algorithm uses the stay probability q_k to decide whether to select the node or not. If it decides not to select it, it chooses a neighbor in ring $k+1$ with uniform probability. Then, it sends to this node the probability v_k and the values p_k, n_k, and δ_k, piggybacked in the RCW.

The RCW algorithm works as follows. The source s selects itself with probability $q_0 = p_0$. If it does not do so, it chooses one node in ring 1 with uniform probability, and sends it the RCW message with values $v_0 = 1$, $n_0 = 1$, p_0, and δ_0. Figure 3 shows the code of the RCW algorithm for nodes in rings $\mathbb{R}_k$ for $k > 0$. Each node in ring k must only know initially the values δ_k, γ_k and p_k. Observe that n_k (number of nodes in ring k) can be locally calculated as $n_k = n_{k-1}\delta_{k-1}/\gamma_k$. The correctness of this computation follows from the uniform connectivity assumption (Eq. 1).

Analysis The uniform connectivity property can be used to prove by induction that all nodes in the same ring k have the same probability v_k to be reached. The stay probability q_k is defined as $q_k = p_k/v_k$. Then, from Observation 1, the probability of selecting a node x of ring k is $p_k = v_k q_k$. What is left to prove is that the value v_k computed in Figure 3 is in fact the visit probability of a node in ring k.

Lemma 2. *The values v_k computed in Figure 3 are the correct visit probabilities.*

Proof. Let us use induction. For $k = 1$ the visit probability of a node x in ring $\mathbb{R}_1$ is $\frac{1-q_0}{n_1} = \frac{1-p_0}{n_1}$. On the other hand, when a message *RCW_MSG* reaches x, it carries $v_0 = 1$, $n_0 = 1$, p_0, and δ_0 (Line 2). Then,

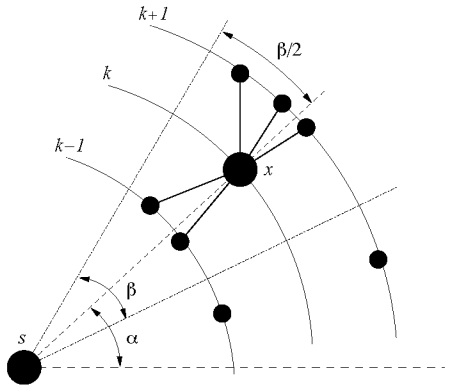

Fig. 4. Node deployment and connectivity used in the simulations.

v_1 is computed as $v_1 = n_0 \frac{v_0 - p_0}{n_1} = \frac{1-p_0}{n_1}$ (Line 3). For a general $k > 1$, assume the value v_{k-1} is the correct visit probability of a node in ring $k-1$. The visit probability of a node in ring k is $v_{k-1} n_{k-1}(1 - q_{k-1})/n_k$, which replacing $q_{k-1} = p_{k-1}/v_{k-1}$ yields the expression used in Figure 3 to compute v_k (Line 3).

The above lemma, together with the previous reasoning, proves the following.

Theorem 5. *Every node at distance k of the source is selected with probability p_k.*

6 Concentric Rings Networks without Uniform Connectivity

Finally, we are interested in evaluating, by means of simulations, the performance of the RCW algorithm for concentric rings with uniform connectivity when it is used on a more realistic topology: a concentric rings network *without uniform connectivity.* The experiment has been done in a concentric rings topology of 100 rings with 100 nodes per ring, and it places the nodes of each ring uniformly *at random* on each ring. This deployment does not guarantee uniform connectivity. Instead, the nodes' degrees follow roughly a normal probability distribution. In order to establish the connectivity of nodes, we do a geometric deployment. A node x in ring k is assigned a position in the ring. This position can be given by an angle α. Then, each network studied will have associated a connectivity angle β, the same for all nodes. This means that x will be connected to all the nodes in rings $k-1$ and $k+1$ whose position (angle) is in the interval $[\alpha - \beta/2, \alpha + \beta/2]$. (See Figure 4.) Observe that the bigger the angle β is, the more neighbors x has in rings $k-1$ and $k+1$. We compare the relative error of the RCW algorithm when sampling with two distributions: the uniform distribution (UNI) and a distribution proportional to the inverse of the distance (PID). We define the relative error e_i for a node x in a collection C of s samples as $e_i = \frac{|fsim_x - f_x|}{f_x}$, where $fsim_x$ is the number of instances of x in collection C obtained by the simulator, and $f_x = p_x \cdot s$ is the expected number of instances of x with the ideal probability distribution (UNI or PID). We compare the error of the RCW algorithm with the error of a generator of pseudorandom numbers. For each configuration, a collection of 10^7 samples has been done.

Figure 5 presents the results obtained in the UNI and PID scenarios. In both cases, we can see that the RCW algorithm performs much worse than the UNI and PID simulators. The simulation results show a biased behavior of RCW algorithm because the condition of Eq. 1 is not fulfilled in this experiment (i.e. a node has no neighbors, or there are two nodes in a ring k that have different number of neighbors in rings $k-1$ or $k+1$).

Assignment Attachment Points (AAP) Algorithm To eliminate the errors observed when there is no uniform connectivity, we propose a simple algorithm to transform the concentric rings network without uniform connectivity into an overlay network with uniform connectivity.

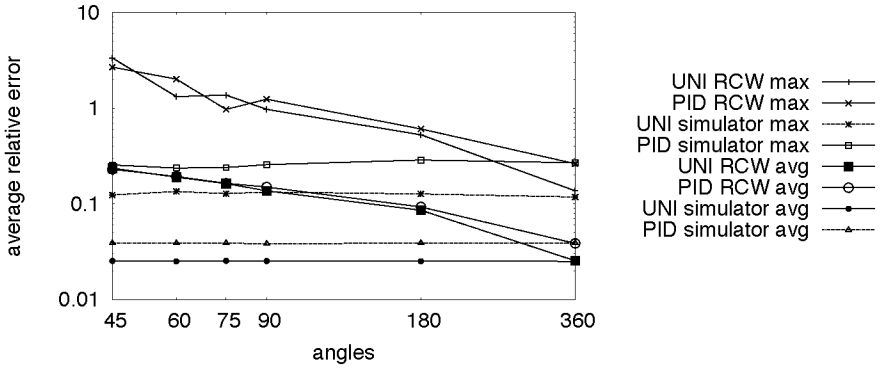

Fig. 5. UNI and PID scenarios without uniform connectivity.

```
1  function AssignAttachmentPoints(x, k)
2  ap ← LCM(n_k, n_{k+1}) / n_k
3  C ← N_{k+1}(x) /* neighbors of x in ring k + 1 */
4  A_x ← ∅ /* A_x is a multiset */
5  loop
6    choose c from C
7    send ATTACH_MSG to c
8    receive RESPONSE_MSG from c
9    if RESPONSE_MSG = OK then
10     ap ← ap − 1
11     add c to A_x /* c can be in A_x several times */
12   else C ← C \ {c}
13 until (ap = 0) ∨ (C = ∅)
14 if (ap = 0) then return A_x
15 else return FAILURE
```

Angle	% success
15°	0%
30°	0%
45°	3%
60°	82%
75°	99%
90°	100%
150°	100%
180°	100%
360°	100%

Fig. 6. Assignment Attachment Points (AAP) Function (left side). Success rate of the AAP algorithm as a function of the connectivity angle (right side).

To preserve the property that the visit probability is the same for all the nodes in a ring, nodes will use different probabilities for different neighbors. Instead of explicitly computing the probability for each neighbor, we will use the following process. Consider rings k and $k+1$. Let $r = LCM(n_k, n_{k+1})$, where LCM is the *least common multiple* function. We assign $\frac{r}{n_k}$ *attachment points* to each node in ring k, and $\frac{r}{n_{k+1}}$ attachment points to each node in ring $k+1$. Now, the problem is to connect each attachment point in ring k to a different attachment point in ring $k+1$ (not necessarily in different nodes). If this can be done, we can use the algorithm of Figure 3, but when a RCW is sent to the next ring, an attachment point (instead of a neighbor) is chosen uniformly. Since the number of attachments points is the same in all nodes of ring k and in all nodes of ring $k+1$, the impact in the visit probability is that it is again the same for all nodes of a ring.

The connection between attachment points can be done with the simple algorithm presented in Figure 6, in which a node x in ring k contacts its neighbors to request available attachment points. If a neighbor that is contacted has some free attachment point, it replies with a response message *RESPONSE_MSG* with value *OK*, accepting the connection. Otherwise it replies to x notifying that all its attachment points have been connected. The node x continues trying until its $\frac{r}{n_k}$ attachment points have been connected or none of its neighbors has available attachment points. If this latter situation arises, then the process failed. The algorithm finishes in $O(\max_k\{n_k\})$ communication rounds. (Note that $r \leq n_k \cdot n_{k+1}$ and $|C| \leq n_{k+1}$). Combining these results with the analysis of Section 5, we can conclude with the following theorem.

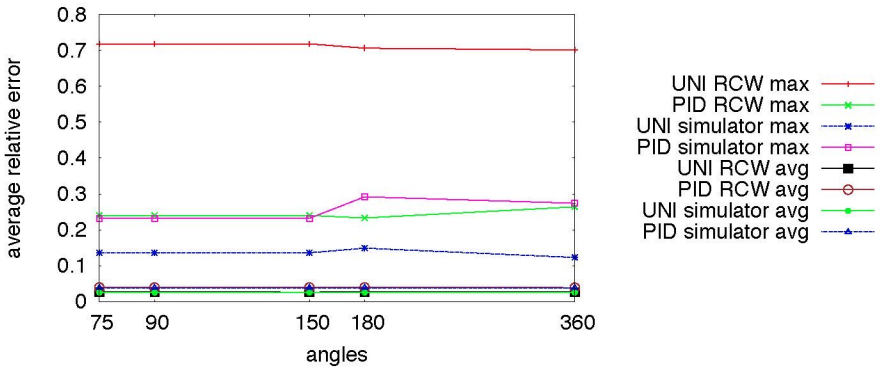

Fig. 7. UNI and PID scenarios without uniform connectivity, using the AAP algorithm.

Theorem 6. *Using attachment points instead of links and the distributed RCW-based algorithm of Figure 3, it is possible to sample a concentric rings network without uniform connectivity with any desired distance-based probability distribution p_k, provided that the algorithm of Figure 6 completes (is successful) in all the nodes.*

Figure 7 shows the results when using the AAP algorithm. As we can see, the differences have disappeared. The conclusion is that, when nodes are placed uniformly at random and AAP is used to attach neighbors to each node, RCW performs as good as perfect UNI or PID simulators.

In general, the algorithm of Figure 6 may not be succesful. It is shown in the table of Figure 6 (right side) the success rate of the algorithm for different connectivity angles. It can be observed that the success rate is large as long as the connectivity angles are not very small (at least 60°). (For an angle of 60° the expected number of neighbors in the next ring for each node is less than 17.) For small angles, like 15° and 30°, the AAP algorithm is never successful. For these cases, the algorithm for connected network presented in Section 3 can be used.

7 Conclusions

In this paper we propose distributed algorithms for node sampling in networks. All the proposed algorithms are based on a new class of random walks called centrifugal random walks. These algorithms guarantee that the sampling end after a number of hops upper bounded by the diameter of the network, and it samples with the exact probability distribution. As future works we want to explore sampling in dynamic networks using random centrifugal walks. Additionally, we will investigate a more general algorithm that would also concern distributions that do not only depend on the distance from the source.

References

1. Asad Awan, R. A. Ferreira, Suresh Jagannathan, and Ananth Grama. Distributed uniform sampling in unstructured peer-to-peer networks. In *HICSS*. IEEE CS, 2006.
2. M. Bertier, F. Bonnet, A.M. Kermarrec, V. Leroy, S. Peri, and M Raynal. D2HT: The best of both worlds, integrating RPS and DHT. In *EDCC*. IEEE CS, 2010.
3. F. Bonnet, A.-M. Kermarrec, and M. Raynal. Small-world networks: From theoretical bounds to practical systems. *Proc. OPODIS, LNCS*. Springer, 2007.
4. M. Bui, F. Butelle, and C. Lavault. A distributed algorithm for constructing a minimum diameter spanning tree. *J. Parallel Distrib. Comput.*, May 2004.
5. Y. Busnel, R. Beraldi, and R. Baldoni. On the uniformity of peer sampling based on view shuffling. *Journal of Parallel and Distributed Computing*, 2011.

6. Michael Elkin. A faster distributed protocol for constructing a minimum spanning tree. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '04, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.
7. P. Fraigniaud and G. Giakkoupis. On the searchability of small-world networks with arbitrary underlying structure. In L. J. Schulman, editor, *STOC*. ACM, 2010.
8. B. Gfeller, N. Santoro, and P. Widmayer. A distributed algorithm for finding all best swap edges of a minimum-diameter spanning tree. *IEEE Trans. Dependable Secur. Comput.*, January 2011.
9. Minas Gjoka, Maciej Kurant, Carter T. Butts, and Athina Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *INFOCOM*, pages 2498–2506. IEEE, 2010.
10. M. Gjoka, M. Kurant, Carter T. Butts, and A. Markopoulou. Practical recommendations on crawling online social networks. *Selected Areas in Communications, IEEE Journal on*, october 2011.
11. M. Gurevich and I. Keidar. Correctness of gossip-based membership under message loss. *SIAM J. Comput.*, 39(8), December 2010.
12. M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM Trans. Comput. Syst.*, 25(3), 2007.
13. D. Kempe, Jon M. Kleinberg, and Alan J. Demers. Spatial gossip and resource location protocols. *J. ACM*, 51(6):943–967, 2004.
14. J. M. Kleinberg. Navigation in a small world. *Nature*, 406(6798), August 2000.
15. Chul-Ho Lee, Xin Xu, and Do Young Eun. Beyond random walk and metropolis-hastings samplers: why you should not backtrack for unbiased graph sampling. SIGMETRICS '2012. ACM.
16. D. Milić and and T. Braun. Netice9: A stable landmark-less network positioning system. In *Local Computer Networks, 2010 IEEE 35th Conference on*, oct. 2010.
17. A. Sevilla, A. Mozo, M. Lorenzo, J.L. López-Presa, P Manzano, and A. Fernández Anta. Biased selection for building small-world networks. OPODIS 2010.
18. Ming Zhong and Kai Shen. Random walk based node sampling in self-organizing networks. *SIGOPS Oper. Syst. Rev.*, 40:49–55, July 2006.