RaWMS - Random Walk based Lightweight Membership Service for Wireless Ad Hoc Networks*

Ziv Bar-Yossef
Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa 32000, Israel
zivby@ee.technion.ac.il

Roy Friedman Gabriel Kliot
Computer Science Department
Technion - Israel Institute of Technology
Haifa 32000, Israel
{roy,gabik}@cs.technion.ac.il

ABSTRACT

RaWMS is a novel lightweight random membership service for ad hoc networks. The service provides each node with a partial uniformly chosen view of network nodes. Such a membership service is useful, e.g., in data dissemination algorithms, lookup and discovery services, peer sampling services, and complete membership construction. The design of RaWMS is based on a novel reverse random walk (RW) sampling technique. The paper includes a formal analysis of both the reverse RW sampling technique and RaWMS and verifies it through a detailed simulation study. In addition, RaWMS is compared with a number of other known methods such as flooding and gossip-based techniques.

Categories and Subject Descriptors:
C.2.1 [Comp.-Communication Networks]: Network Architecture and Design—*Wireless communication*; C.2.4 [Comp.-Communication Networks]: Distributed Systems—*Distributed applications*

General Terms: Algorithms, Design.

Keywords: Ad Hoc Networks, Membership service, Random Walk

1. INTRODUCTION

Context of this study. *Membership services* serve as essential building blocks in a variety of other services and applications in ad hoc networks. A membership service provides each node with a *view* regarding who are the other nodes in the network.

In traditional membership services [9], the view of each process approximates the entire membership. Moreover, views must be consistent, and changes to views must be coordinated among all their members. This complete and strongly consistent approach works well in wired LANs. However, generally speaking, it is not suitable for large networks and mobile ad hoc networks. This is because maintaining such membership information consumes a lot of memory and requires large message and computational overheads for each membership change. In contrast, in mobile ad hoc networks, nodes often have limited memory capacities. The dynamic nature of the system implies frequent changes to the network membership. Additionally, wireless multi-hop networks are more sensitive to high message loads than wired LANs, and the energy consumption associated with sending and receiving many messages could quickly drain the batteries of mobile devices.

Interestingly, many applications do not need complete membership information. Instead, they only require each member to hold a partial random view of the network membership. Examples of such applications are probabilistic reliable dissemination of data and events [5, 11, 21], peer sampling services [20], location services and uniform quorums [19], random overlay constructions [25], DHTs [29], etc. Therefore, it makes sense to offer an optimized membership service that indeed only provides nodes with partial random views. Such optimized services are the focus of this paper.

Contributions of this work. We start by introducing a novel reverse *random walk* (RW) technique for peer sampling with an adaptation to ad hoc networks along with a formal analysis of this technique. Next, we present the *RAndom Walk based Membership Service* (RaWMS), which provides a random uniformly chosen partial membership view based on random walks. In particular, the peers in the view of every node look like they were picked from random uniformly chosen locations in the network.

Unlike many gossip-based algorithms, our service posses five important properties. These include (i) proven uniform randomness of the constructed views, (ii) analytically proven bounds on the load of an individual node (view size), (iii) enabling each node to set its view size independently of other nodes without any implications on the randomness of the views' content, (iv) a low chance of partition in the knowledge graph induced by the views, and (v) self healing from partitions when they do occur. Another important characteristic of our algorithm is that it does not require multiple-hop routing. Also, its sole assumption about the network's topology is that the network will be connected.

In the implementation of RaWMS, we seek to obtain a good tradeoff between the communication overhead incurred by our protocol vs. its memory consumption. To deal with this issue, our protocol allows every node to choose the target size of its view, independently and without any correlation with other nodes. Moreover, a node can adjust its view size on-the-fly according to its currently available memory. In a small or medium size network, or if a node has plenty of memory, it may wish to maintain a large or even complete membership knowledge. On the other hand, in a sensor network or a very large ad hoc network, nodes may wish to save memory and only maintain a partial membership view. In case that at some point a node with a small view requires knowledge of the entire membership, e.g., due to its application's demand, our service can reactively increase its view in a fast and efficient manner.

We provide a detailed formal analysis of our implementation of

*This research is partially funded by the Israeli Science Foundation grant #44/03.

MobiHoc'06, May 22–25, 2006, Florence, Italy.

RaWMS. We then analytically compare RaWMS with other membership construction techniques, such as lpbcast [11], Shuffling [14] and flooding. Finally, we study the performance of RaWMS by simulations, evaluating its properties and comparing it to lpbcast. These measurements largely confirm our theoretical analysis.

2. SYSTEM MODEL

Consider a set of nodes spread across a geographical area and communicating by exchanging messages using a wireless medium. A node in the system is a device owning an omni-directional antenna that enables wireless communication. Each node v may send messages that can be received by all other nodes within its transmission range r_v. A node u is a *neighbor* of another node v if u is located within the transmission range of v. The transmission disk of node v is a disk centered on v with radius r_v. The combination of the nodes and the transitive closure of their transmission disks forms a wireless ad hoc network. The network described above can also be modeled as a graph $G = (V, E)$ where V is the set of network nodes and E models the one-to-one neighboring relations.

The network connectivity graph of an ad hoc network is a special case of a *Random Geometric Graph* (RGG). A d-dimensional RGG, denoted $G^d(n, r)$, is obtained by placing n nodes uniformly at random on the surface of a d-dimensional unit torus, and connecting nodes within Euclidean distance r of each other [28]. RGGs have been studied in the context of random walks, and thus we can utilize some of these results for our purposes. Specifically, the $G^2(n, r)$ graph, also known as the *Unit Disk* graph, is often used to model the network connectivity graph of 2-dimensional wireless ad hoc networks and sensor networks [16].

We assume that nodes do not know their position and we do not use any geographic knowledge in our algorithms. Each node has a unique identifier that is used for sending messages to that node. The membership knowledge of a node, defined as the *view* of this node, is a list of identifiers of other network nodes known to this node. In addition to the view structure, we assume that each node knows all of its direct neighbors, whose addresses are stored in the node's *neighbors list*. This list can be constructed, e.g., by a simple heartbeat mechanism that is present in any case in most routing algorithms for ad hoc networks. A node can communicate with its neighbors directly. Additionally, a node can communicate with other distant nodes whose address is present in its view by applying a routing algorithm to route messages to these nodes.

Nodes can physically move across the network; new nodes may join and existing nodes may leave the network at any time, either gracefully or by suffering a crash failure. Nodes that crash or leave the network may rejoin it later.

3. RANDOM WALK TECHNIQUES

Simple random walks. Let $G = (V, E)$ be an undirected graph, $n = |V|$. Let d_v denote the degree of a vertex $v \in V$. A *simple random walk* on G is a stochastic process in which a "token" is repeatedly forwarded from a node to a randomly chosen neighbor. Formally, the random walk is specified by an $n \times n$ probability transition matrix P, where $P_{v,u} = 1/d_v$, if $(v, u) \in E$, and $P_{v,u} = 0$ otherwise. For every time step $t \geq 0$, ϕ_t is a probability distribution over the vertex set V. It specifies, for each $v \in V$, the probability that the token is placed on vertex v at step t. The initial distribution ϕ_0 specifies the vertex at which the random walk is started. For every $t \geq 1$, $\phi_t = \phi_0 P^t$.

If the graph is connected and non-bipartite, then the sequence of distributions $\phi_0, \phi_1, \phi_2, \ldots$ is guaranteed to converge to a unique *limit distribution* π, which is *independent* of the initial distribution. π is also a *stationary* distribution of P, that is, $\pi P = \pi$.

A simple analysis (cf. [22]) shows that the stationary distribution of the simple random walk has a limit distribution that assigns probabilities to nodes proportionally to their degree: $\pi(v) = \frac{d_v}{2|E|}$, for every $v \in V$. Therefore, a stationary distribution of a simple random walk on a graph is uniform if and only if the graph is *regular*, i.e., all nodes have the same degree. Later in the section we will present the *Maximum Degree* random walk, whose stationary distribution is uniform even for non-regular graphs.

RW-based sampling. The following algorithm uses a random walk on G to sample nodes from the limit distribution π:

`sample` (p,T)
1) **start** a RW from p;
2) **run** the RW for T steps;
3) **return** the node in which the RW was stopped

If π happens to be the uniform distribution, then the algorithm generates uniform sample nodes. The idea of the algorithm is very simple: it starts the random walk at some start vertex p and runs it for T steps. The node reached after T steps is returned as a sample. If T is sufficiently large, then the distribution ϕ_T of the node returned is close to the limit distribution π.

Notice that this sampling technique does not require a priori knowledge of all network nodes and does not use multi-hop routing. A node must only be aware of its neighbors. This makes RW-based sampling attractive for ad hoc networks.

The main question to be addressed is how to set T to guarantee that ϕ_T is close to π. To this end, we define the *mixing time* of a random walk:

DEFINITION 1. *For every node $v \in V$, let ϕ_0^v be the initial distribution concentrated on v. For every step $t \geq 0$, the total variation distance between $\phi_0^v P^t$ and π is defined as:*

$$\Delta_v(t) = \frac{1}{2} \sum_{u \in V} |\phi_0^v(u) - \pi(u)|.$$

For every $\epsilon > 0$, the ϵ-mixing time *of the random walk is:*

$$T_{\mathrm{mix}}(\epsilon) = \max_{v \in V} \min\{t \mid \Delta_v(t') \leq \epsilon, \forall t' \geq t\}.$$

Intuitively, the mixing time of a RW is the minimum number of steps t required to guarantee that, regardless of the start vertex of the random walk, the probability distribution reached after t steps is ϵ-close to the stationary distribution. Throughout this paper, when the parameter ϵ is omitted, we refer to mixing time with $\epsilon = \Theta(\frac{1}{n})$.

A popular method for bounding the mixing time of a random walk is via the *spectral gap* of its transition matrix. Let λ_1, λ_2, $\ldots$, λ_n be the n eigenvalues of P ordered in decreasing absolute value. It can be shown that all these eigenvalues must be real and lie in the interval $[-1, 1]$, where the principal eigenvalue, $\lambda_1 = 1$. If G is connected and non-bipartite, then $|\lambda_2| < 1$. The difference $1 - |\lambda_2|$ is called the *spectral gap* of P and turns out to determine the mixing time of the random walk (cf. [17]):

THEOREM 1. *The mixing time of a random walk with transition matrix P is upper bounded as follows:*

$$T_{\mathrm{mix}}(\epsilon) \leq \frac{\ln \pi_{\min}^{-1} + \ln \epsilon^{-1}}{1 - |\lambda_2|},$$

where $\pi_{\min} = \min\{\pi(v) \mid v \in V\}$.

Note that when π is the uniform distribution then $\pi_{\min} = 1/n$.

Theorem 1 provides the means for setting the parameter T in the sampling algorithm. Given a bound on the spectral gap of P

(which is typically derived by analyzing combinatorial properties of the graph G) and given the desired accuracy parameter ϵ, we can use the above formula to calculate T.

The Maximum Degree RW. As mentioned above, the simple RW on a graph converges to a uniform limit distribution if and only if the graph is regular. Ad hoc network graphs are typically non-regular, and thus we cannot use the simple RW directly to obtain uniform sampling of network nodes. Instead, we use a different RW, called the *Maximum Degree* (MD) random walk, which has been used before in various contexts [22, 3, 6, 7] to achieve uniform sampling.

Let $G = (V, E)$ be an undirected, connected, and non-bipartite graph, which is not necessarily regular. Suppose we have an upper bound D on $d_{\max}$, the maximum degree of G. (We show how to get such a bound below.) We use D to transform G into a regular graph G'. To this end, we add to each node v of G a *weighted self loop* (i.e., multiple edges from v to itself). The weight of the self loop of v is set to be $D - d_v$. The degrees of all nodes in the resulting graph G' are the same and equal D. The Maximum Degree random walk on G is the simple random walk on G'. The transition matrix of this random walk is then the following:

$$P_{v,u} = \begin{cases} 1/D, & \text{if } (v,u) \in E, v \neq u, \\ 0, & \text{if } (v,u) \notin E, \\ 1 - \sum_{u' \neq u} P_{u',u} & \text{if } v = u. \end{cases}$$

If G is connected, then G' is connected and non-bipartite, and hence the MD random walk has a limit distribution. Furthermore, since G' is regular, this distribution is uniform.

Many of the steps performed in a MD random walk are self loop steps. In many applications, including ours, self loop steps are "free": they can be executed in zero time and require no communication. Thus, it makes sense to define the *actual mixing time* of a random walk, denoted $T_{\text{actual\_mix}}$, which is the expected number of actual steps (i.e., non-self loop steps) needed for the random walk to approach its limit distribution.

As we shall see later, an overestimate of D may increase the mixing time of the MD random walk, but typically does not affect the actual mixing time. This is because an inflated D increases the mixing time at the same rate it increases the fraction of self loop steps, leaving the number of actual steps intact.

Random walks on ad hoc networks. Wireless ad hoc and sensor networks are typically modelled as *Random Geometric Graphs* (RGG). We show that for an appropriate choice of the radius parameter r, a random geometric graph $G^2(n, r)$ is with high probability undirected and connected. Hence, the MD random walk on $G^2(n, r)$ is likely to converge to a uniform limit distribution.

Undirectedness. Recall that two nodes $u, v \in G^2(n, r)$ are connected by an edge if and only if the Euclidean distance between them is at most r. Since Euclidean distance is symmetric, $G^2(n, r)$ must be undirected. [1]

Connectivity. The connectivity of $G^2(n, r)$ was extensively studied in the context of the minimal transmission power necessary to ensure that with high probability a given ad hoc network graph is still connected as the number of nodes in the network grows to infinity. Gupta and Kumar [16] have shown that if n nodes are placed on a unit disk and each node transmits at a power level that covers an area of $\pi r^2 = \frac{\log n + c(n)}{n}$, then the resulting network is asymptotically connected with probability one, if and only if $c(n) \to \infty$ as $n \to \infty$. In [27], the authors obtain a similar result when nodes are distributed in the unit square $[0, 1]^2$.

Throughout this paper we assume a radius $r = \sqrt{\frac{C \ln n}{n}}$ for the transmission range, where C is a constant. For $C > 1/\pi$, this radius satisfies the connectivity condition of Gupta and Kumar. Thus, we can assume w.h.p. that the ad hoc network graph is connected.

For technical reasons, we also assume the radius r is not too large ($r \leq 1/2$). If the radius is greater than $1/2$, then the resulting graph is a clique or close to a clique, and thus the random walk on this graph mixes very quickly.

Setting the maximum degree bound. We now prove an upper bound on the maximum degree of the random graph $G^2(n, r)$. This bound can be useful in setting the parameter D of the MD RW.

PROPOSITION 1. *Suppose $r \leq 1/2$. Fix any $0 < \alpha_d < 1$ and let $\delta_d = \sqrt{\frac{3}{\pi r^2 (n-1)} \cdot \ln \frac{2n}{\alpha_d}}$. Let d_{avg}, $d_{\max}$, and $d_{\min}$ be, respectively, the average, maximum, and minimum degree of the random geometric graph $G^2(n, r)$. Then,*

(1) $E(d_{\text{avg}}) = \pi r^2 (n-1)$

(2) with probability at least $1 - \alpha_d$, $d_{\min} \geq (1 - \delta_d) \cdot \pi r^2 (n-1)$ and $d_{\max} \leq (1 + \delta_d) \cdot \pi r^2 (n-1)$

PROOF. Fix any $i \in \{1, \ldots, n\}$. For each $j \neq i$, let X_j be a 0-1 random variable indicating whether the j-th node of $G^2(n, r)$ is a neighbor of the i-th node of $G^2(n, r)$ or not. Since two nodes are neighbors if and only if they are at distance at most r from each other, then $E(X_j) = \Pr(X_j = 1) = \pi r^2$. (Here we use the fact $r \leq 1/2$. Otherwise, a disk of radius r centered at the i-th node "wraps around" itself, and thus contains multiple "copies" of the same points on the surface of the unit torus. In particular, this means that the probability to have the j-th node as a neighbor the i-th node is lower than πr^2.)

Let $Y_i = \sum_{j \neq i} X_j$ be the degree of the i-th node. By linearity of expectation, $E(Y_i) = \pi r^2 (n-1)$. Note that $d_{\text{avg}} = \frac{1}{n} \sum_{i=1}^{n} Y_i$. Thus, using linearity of expectation again, $E(d_{\text{avg}}) = \pi r^2 (n-1)$. By Chernoff bounds

$$\Pr(|Y_i - E(Y_i)| > \delta_d E(Y_i)) \leq 2 \cdot \exp(-\frac{\delta_d^2 E(Y_i)}{3}).$$

Substituting $E(Y_i) = \pi r^2 (n-1)$ and the value of δ_d, we have:

$$\Pr(|Y_i - \pi r^2 (n-1)| > \delta_d \cdot \pi r^2 (n-1)) \leq \frac{\alpha_d}{n}.$$

Using the union bound, the probability that there is a node whose degree is less than $\pi r^2 (n-1) \cdot (1 - \delta_d)$ or more than $\pi r^2 (n-1) \cdot (1 + \delta_d)$ is at most α_d. □

As shown by the proposition, the average degree of every node in $G^2(n, r)$ is $(n-1)\pi r^2$. For example, for $C = 1$ and $\alpha_d = 0.1$, the average degree is around $\pi \ln n$ and the maximum degree is at most a factor $(1 + \sqrt{1 + 3/\ln n}) \sim 2$ away from the average degree with probability 0.9.

Mixing time. Next, we analyze the mixing time of the Maximum Degree random walk on $G^2(n, r)$. Avin and Recal [2] and Boyd et al. [7] analyze the mixing time of the simple random walk on $G^2(n, r)$ and show it equals $\Theta(r^{-2} \log n)$. Boyd et al. [7] mention in their paper that a similar analysis can show the same bound on

[1] The symmetry assumed in the theoretical model of RGGs is not always valid in real ad hoc networks and the transmission range does not behave exactly as a disk due to various physical phenomena. In practice, it is possible that a node v receives messages sent from node u, but not vice versa. Yet, such phenomena are rare and on the other hand, those assumptions greatly simplify the formal model. At any event, our theoretical results were verified through an extensive simulation with real transmission range behavior including distortions, background noise, unidirectional links, etc.

the mixing time of the MD random walk. Yet, they do not give this analysis explicitly. Furthermore, the analysis provided in these papers is asymptotic, and does not include the exact constants.

We follow the footsteps of Boyd et al. and provide a rigorous analysis of the mixing time of the MD RW. We show that:

THEOREM 2. *Suppose $r \leq 1/2$ and $n \geq 10$. Let $G^2(n,r)$ be a random geometric graph chosen with n nodes and radius r. Let D be any value that upper bounds the maximum degree of $G^2(n,r)$. Let $T_{\mathrm{mix}}(\epsilon)$ be the mixing time of the MD random walk on this graph, when applied with the value D. Let $T_{\mathrm{actual\_mix}}(\epsilon)$ be the actual mixing time of this random walk (i.e., excluding self loop steps). For any $C > 49$, if $r = \sqrt{\frac{C \ln n}{n}}$, then with probability at least $2/3$ (over the choice of the graph),*

$$\begin{aligned} T_{\mathrm{mix}}(\epsilon) &\leq \frac{30}{(1-\frac{7}{\sqrt{C}})^2} \cdot \frac{D}{n} \cdot \frac{1}{r^4} \cdot (\ln n + \ln \epsilon^{-1}). \\ T_{\mathrm{actual\_mix}}(\epsilon) &\leq \frac{120}{(1-\frac{7}{\sqrt{C}})^2} \cdot \frac{1}{r^2} \cdot (\ln n + \ln \epsilon^{-1}). \\ T_{\mathrm{actual\_mix}}(\epsilon) &\leq \frac{d_{\max}}{D} \cdot T_{\mathrm{mix}}(\epsilon). \end{aligned}$$

The proof of Theorem 2 is rather involved. Due to lack of space, it is omitted from this version of the paper. It appears in the full version, available as a technical report [4]. The proof relies on Sinclair's canonical paths method [31] for bounding the spectral gap of a random walk. The construction and the analysis of these canonical paths are done via partitioning of the torus into a square grid, and defining "square paths" on this grid. Several additional remarks are in order.

(1) If $d_{\max} \approx \pi r^2 n$ (as guaranteed w.h.p. by Proposition 1) and if we choose D to be close to $d_{\max}$, then the mixing time of the MD random walk is $T_{\mathrm{mix}}(\epsilon) = O(r^{-2}(\ln n + \ln \epsilon^{-1}))$. For our choice of r, if C is a constant, then this mixing time is $T_{\mathrm{mix}}(\epsilon) = O(n(1 + \frac{\ln \epsilon^{-1}}{\ln n}))$. On the other hand, if D is a gross overestimate of $d_{\max}$, T_{mix} can get higher.

(2) As opposed to the standard mixing time, which can get large if D is an overestimate of $d_{\max}$, the actual mixing time is not affected by the difference between D and $d_{\max}$. That is, $T_{\mathrm{actual\_mix}}(\epsilon) = O(r^{-2}(\ln n + \ln \epsilon^{-1}))$ always, regardless of the value of D. For constant C, we have

$$T_{\mathrm{actual\_mix}}(\epsilon) = O(n(1 + \frac{\ln \epsilon^{-1}}{\ln n})).$$

(3) The theorem exhibits a tradeoff between the mixing time and the radius r: the larger is r, the smaller is the mixing time. This is to be expected, since a large transmission range improves the connectivity of the graph, which results in a faster mixing time. On the other hand, large transmission range increases the number of transmission collisions, reducing the quality of the wireless link.

(4) The minimum network size, for which the above theorem gives a non-trivial result is obtained by setting $C = 50$, in which case $n \geq 1,060$. For smaller networks, the lower bound on r implies $r > 1/2$, which means that the graph $G^2(n,r)$ is a clique. In cliques (with self loops), the random walk mixes in a single step.

(5) The theorem shows that the asymptotic behavior of the random walk is linear. The fact that the bounds provide non-trivial results only for sufficiently large networks and that Theorem 2 is applicable only for quite large radiuses ($C > 49$) are artifacts of the involved theoretical analysis and not of the algorithm itself. We believe that in practice the RW mixes quickly for much smaller transmission ranges and for small networks as well. This is supported by our experimental results, in which we have experienced with $C = 1$ and observed almost uniform quality of the RW sampling for $T_{\mathrm{mix}}(\epsilon) = n/2$.

3.1 Reverse RW-based uniform sampling in ad hoc networks

The naïve, direct, approach for applying the MD random walk for generating uniform samples in an ad hoc network is the following. Every node v starts the sampling algorithm described above using the MD random walk, passing its own id and the random walk's mixing time as parameters. The last node reached in the random walk notifies v of its id. This id represents a uniformly sampled node from the network. The notification can be done either by using the reverse path of the RW or by applying unicast routing. Both introduce significant additional communication overhead.

To solve this problem, we propose using a *reverse sampling technique*. That is, instead of informing the source node v about a sampled destination node u, the destination u is informed about the source v. We claim that this constitutes a random sample of source nodes. Using symmetry arguments, the destination node u can use the source v as if v was sampled by u directly. This way, there is no additional routing overhead for notifying the result of the RW to its initiating node. Since every node can initiate a number of RWs with its id simultaneously, we can use this technique to construct for each node a random sample of s ($1 \leq s \leq n$) other nodes.

Below, we prove that reverse sampling indeed results in a uniform sample of nodes. (For simplicity of analysis, we assume in the proof that each RW produces a truly-uniform node and not a nearly-uniform node. The extension to deal with nearly-uniform samples is rather straightforward.)

LEMMA 1. *Suppose every node v in a network chooses (via a random walk) a random node X_v. For every u, let Z_u be the set of nodes that selected u (the RWs started by them have stopped at u): $Z_u = \{v \mid X_v = u\}$. Then, given that the size of Z_u is k, Z_u is a random subset of the vertex set of size k.*

PROOF. To prove the lemma, we need to show that $\forall u \in V$, $\forall 1 \leq k \leq n$, and for set S of k distinct nodes, $\Pr(Z_u = S \mid |Z_u| = k) = 1/\binom{n}{k}$.

Fix any u, any $k \in \{1, \ldots, n\}$, and any set $S = \{v_1, \ldots, v_k\}$ of k nodes. By Bayes rule,

$$\begin{aligned} &\Pr(Z_u = S \mid |Z_u| = k) = \\ &= \Pr(|Z_u| = k \mid Z_u = S) \cdot \frac{\Pr(Z_u = S)}{\Pr(|Z_u| = k)}. \end{aligned} \quad (1)$$

We next analyze each of the three terms on the RHS of Equation 1. For the first term, we have $\Pr(|Z_u| = k \mid Z_u = S) = 1$. Regarding $\Pr(Z_u = S)$, note that $Z_u = S = \{v_1, \ldots, v_k\}$ iff $X_{v_1} = u, X_{v_2} = u, \ldots, X_{v_k} = u$, and for every $v \notin S$, $X_v \neq u$. The events $\{X_v = u\}_{v \in V}$ are independent of each other (because the random walks are independent). Furthermore, for every v, $\Pr(X_v = u) = \frac{1}{n}$. Therefore, $\Pr(Z_u = S) = (\frac{1}{n})^k \cdot (1 - \frac{1}{n})^{n-k}$.

Regarding $\Pr(|Z_u| = k)$, $|Z_u|$ has a binomial distribution with n trials and a probability of success $\frac{1}{n}$. Therefore, $\Pr(|Z_u| = k) = \binom{n}{k} \cdot (\frac{1}{n})^k \cdot (1 - \frac{1}{n})^{n-k}$. Substituting the three terms into Equation 1, we have the desired result. □

4. RANDOM WALK BASED MEMBERSHIP SERVICE

In RaWMS, a *View* at a node v is defined as a set of node descriptors, where each descriptor consists of <*NodeIdentifier*,*LastTime*>. *NodeIdentifier* is the unique identifier of a given node u and *LastTime* is the last time that v has "heard" from u. Every node v advertises itself every Δ time units by starting a reverse sampling process, as described in Section 3.1. In other words, each Δ time units, v starts a Maximum Degree RW, whose messages carry v's

identifier. Each of these RWs traverses the network for a number of steps that is equal to the mixing time and stops at some node u. If u already has a descriptor corresponding to v in its view, u refreshes the last time it heard from v and discards the RW. Otherwise, u stores the identifier of v in its view. We propose two methods for removal of nodes from the view: *size-based* and *time-based*.

In the size-based method, a node maintains a hard limit on its view size. Each node may choose the target size of its view independently and without any correlation with other nodes. In case that node's u view exceeds its limit upon storing a new identifier, u discards a descriptor with the oldest *LastTime* from its view.

In the time-based method, every node discards nodes' descriptors according to a predefined timeout. The descriptor of node v is removed from node's u view, if u has not heard from v for *Timeout* time units. Each node may choose the value of *Timeout* independently and without any correlation to other nodes. A node can probabilistically adjust its view size by setting the *Timeout* proportionally to the mixing time and Δ. Both methods automatically deal with purging descriptors of nodes that already left the network. The difference between the methods is the probabilistic versus deterministic guarantee of the view size.

The general structure of RaWMS is presented in Figure 1. The protocol consists of two threads: an active thread that initiates a new RW every Δ time units and a passive thread waiting for incoming messages. The `discardExpiredFromView`(*View, Timeout*) function discards all descriptors from the view that the node has not heard from in the last *Timeout* period; `discardOldestFromView`(*View*) discards the oldest descriptor from the view; `refreshInView`(*View,addr*) refreshes the *LastTime* attribute of a given descriptor in the view; `storeInView`(*View,addr*) stores a new descriptor corresponding to a given address and the current time in the view. `pickNextNode` picks either one of the neighboring nodes or a self-loop (of the current node) according to the RW transition matrix probabilities. Also, in the actual implementation, if a node v does not succeed to forward a RW message to the neighbor chosen in a given step, v makes a new attempt to send this message to another random neighbor within the same step. This way, RWs never get lost.

```
do forever
    wait(Δ time units);
    // start a new RW
    ttl ← MixingTime;
    handleRW(myAddress,ttl);
    if timeoutBasedMethod then
        discardExpiredFromView(View,Timeout)
    endif;
enddo

upon receive(RW_message<addr,ttl>) from u do
    // resend the RW to the next node
    ttl ← ttl-1;
    handleRW(addr,ttl) enddo

handleRW(addr,ttl)
    while ttl> 0 do
        next ← pickNextNode();
        if next != v then
            send (RW_message<addr,ttl> to next;
            return
        else
            ttl ← ttl-1 // self-loop step, only count ttl
        endif
    enddo
    publish(addr)

publish(addr)
    if addr ∈ View then
        refreshInView(View,addr)
    else
        storeInView(View,addr)
    endif
    if sizeBasedMethod and ViewIsFull then
        discardOldestFromView(View)
    endif
```

Figure 1: RaWMS - code for node v

4.1 Formal performance analysis

For the purpose of analysis of RaWMS, we assume that all nodes start the algorithm simultaneously with initial empty views and all nodes have the same target view size, denoted $s(n)$. Notice that these assumptions are only required for the formal performance analysis of RaWMS. On the other hand, the correctness of the reverse sampling (and RaWMS) does rely on the fact that all nodes advertise themselves at the same average rate $1/\Delta$. Otherwise, a bias towards more frequently advertising nodes will be created.

We define the *convergence time* to be the number of protocol steps required until all views reach their target size. The period from the beginning of the protocol run until the convergence time has passed is the *convergence period*. In order to evaluate the performance of RaWMS, we study the time and the communication complexity of the protocol throughout the convergence period. Obviously, the target view size, that can be picked by each node independently from other network nodes by enforcing a view size limit or by using an aging timeout, has a direct impact on the memory consumption of the node, as well as on the time and the communication complexity of the convergence process. Intuitively, the larger the target view size is, the more messages should be sent and the more time the view construction takes.

Intuitively, if each random walk started by some node v would have reached a different node, then in order to obtain a view of size $s(n)$, it would have been enough to start $s(n)$ RWs at each node during the convergence period. However, two random walks started at the same node v have a non-negligible probability of reaching the same node u. Thus, in order to obtain the target view size $s(n)$, each node should start a larger number of RWs, which we denote by $r(n)$. Once we compute $r(n)$, we can immediately compute the communication and time complexity to reach convergence.

The average value of $r(n)$. In order to calculate $r(n)$, we refer to the famous *bins and balls* probabilistic problem: how many balls should be placed randomly into n bins in order to have at least one ball in s bins. In our case, we wish to calculate the number $r(n)$ of random trials (the "balls") that are required until $s(n)$ different destination nodes (the "bins") are picked. Each random trial corresponds to a single RW. (For simplicity of analysis, we assume below that each RW chooses a truly uniform node from the network, i.e., $\epsilon = 0$). We prove the following:

LEMMA 2. *Let $1 \leq s = s(n) \leq n$ and let $r = r(n)$ be the random variable specifying the number of balls needed to be randomly placed in n bins until s of the bins are non-empty. Then,*

$$E(r) = n(H_n - H_{n-s}) \leq \begin{cases} n \ln \frac{n}{n-s}, & s < n, \\ n \ln n + O(1), & s = n. \end{cases}$$

where $H_k = \sum_{i=1}^{k} \frac{1}{i}$ is the k-th harmonic number ($H_0 = 0$).

Note that using the inequality $1 + x < e^x$, which holds for all $x > 0$, we have:

$$n \ln \frac{n}{n-s} = n \ln(1 + \frac{s}{n-s}) < \frac{ns}{n-s}.$$

This gives a tight bound on $E(r)$ for $s \ll n$.

PROOF. We view the balls as being placed in the bins sequentially, one by one. The first ball is inserted into an empty bin. The second ball is placed into an empty bin with probability $\frac{n-1}{n}$ and into a non empty bin with probability $\frac{1}{n}$. Using the independence assumption, the expected number of balls required to have a second non empty bin is a geometric random variable with parameter $p = \frac{n-1}{n}$ and mean $\frac{1}{p} = \frac{n}{n-1}$. The additional number of balls required to get the third non empty bin is a geometric random variable with parameter $p = \frac{n-2}{n}$ and mean $\frac{1}{p} = \frac{n}{n-2}$. This process goes on until s bins have at least one ball. r is the number of balls used in this process and is therefore a sum of geometric random variables. By linearity of expectation, we have:

$$\begin{aligned} E(r) &= 1 + \frac{n}{n-1} + \frac{n}{n-2} + \cdots + \frac{n}{n-s+1} \\ &= n(\frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + \cdots + \frac{1}{n-s+1}) \\ &= n(H_n - H_{n-s}). \end{aligned}$$

In order to bound the difference $H_n - H_{n-s}$, we use the following well-known bounds on the harmonic number:

$$\ln n + \gamma + \frac{1}{2(n+1)} \;\le\; H_n \;\le\; \ln n + \gamma + \frac{1}{2n},$$

where γ is a constant. The case $s = n$ immediately follows from the above bound on H_n. For $s < n$,

$$\begin{aligned} n(H_n - H_{n-s}) &\le n\left(\ln n + \frac{1}{2n} - \ln(n-s) - \frac{1}{2(n-s+1)}\right) \\ &\le n(\ln n - \ln(n-s)) \;=\; n \ln \frac{n}{n-s}. \end{aligned}$$

□

Note that nodes start new RW every Δ time units and do not have to be aware of $r(n)$ or make any use of it in RaWMS. $r(n)$ is used here only for the performance estimation of the algorithm.

However, $r(n)$ can be exploited by nodes working in the *time-based* method in order to adjust their average view size. A node that wishes to maintain an average view size of $s(n)$ can calculate the corresponding $r(n)$ independently based on its $s(n)$ and use the value $r(n) \cdot \Delta$ as the *Timeout* for purging old descriptors out of its view. According to this strategy, no identifier stays in a node's view for more than $r(n) \cdot \Delta$ time units on average without being refreshed by a new RW. Thus, an important property of RaWMS is that every view is refreshed to contain a completely new set of identifiers every $r(n) \cdot \Delta$ time units on average.

Communication and time complexity for convergence. The communication complexity during the convergence period is determined by the number of random walks each node should start, i.e., the value $r(n)$ calculated above, multiplied by the length of each random walk. Thus, the total communication complexity during the convergence period is $n \cdot r(n) \cdot T_{\text{actual\_mix}} = \Theta(n^2 \cdot r(n))$. The time complexity is $r(n) \cdot \Delta + T_{\text{actual\_mix}}$, i.e., the time to start $r(n)$ RWs and for the last RW to reach its destination.

For the special case of $s(n) = \sqrt{n}$, we get $r(n) \approx \frac{ns}{n-s} \approx \sqrt{n}$. This means that for relatively small view sizes, there is a very little chance of getting collisions. The convergence time in this case is about $\sqrt{n} \cdot \Delta + T_{\text{actual\_mix}} = \Theta(\sqrt{n} \cdot \Delta + n)$ and the total communication complexity is $n \cdot \sqrt{n} \cdot T_{\text{actual\_mix}} = \Theta(n^2\sqrt{n})$.

Join, leave, and maintenance. When a new node joins the network it starts the same algorithm as any other node, i.e., it starts advertising itself by initiating multiple RWs. After a convergence period, a new node will produce enough advertisements so that its identifier will be uniformly distributed across the network. Therefore, the time and communication complexities of a join process are $r(n) \cdot \Delta + T_{\text{actual\_mix}}$ and $r(n) \cdot T_{\text{actual\_mix}}$, respectively.

The algorithm purges the identifiers of failed or departed nodes automatically, without relying on any action on their side. In the time-based method, a failed node's identifier will be purged from the views of all other nodes precisely *Timeout* time units after its departure. In the size-based method, this will occur on average after $r(n) \cdot \Delta$ time units.

The maintenance complexity of RaWMS is constant: all nodes keep advertising themselves at an average rate of $1/\Delta$ advertisements per time unit. The value of Δ can be tuned to tradeoff communication complexity with the time it takes to react to node leaves/failures and to purge their identifiers from all views.

4.2 Service properties

In our evaluation, we consider several properties of the generated random views, that are important to the envisioned applications discussed in the introduction. The properties are best described using a graph-theoretic view [20] as follows. Define the *knowledge graph* as a directed graph, whose vertices are the network nodes, and that contains an edge from v to u if and only if u's identifier is in the view of v. If the views are truly uniform, then the graph induced by the views is actually a random graph. This framework allows us to study the *connectivity* of the knowledge graph and the *load* of an individual node (out-degree).

Uniformity of the views. A nice feature of RaWMS, compared to other probabilistic methods like [1, 20], is that the uniformity of the views (nodes in the views are picked from random uniformly chosen locations in the network) is guaranteed by construction. The regularization of the graph guarantees that the reverse RW sampling produces uniform samples. The sample accuracy is controlled by the RW length and is probabilistically guaranteed to differ by up to $\epsilon = \Theta(\frac{1}{n})$ from the uniform distribution.

Connectivity of the knowledge graph. By using the same analysis of connectivity probability applied in [11], we can deduce that since the knowledge graph induced by the views is truly uniform, it has a very low chance of partitioning. Another property exhibited by RaWMS is a *self healing* from partitions. Since in RaWMS nodes are not restricted to communicate only with nodes in their views, even if partition in the knowledge graph does occur at some time, it will be fixed by itself after a short period of time.

Load-balancing the view sizes. As we have already shown, average view size can be set independently by every node. We now take a closer look at the view size of a given node at the end of the convergence period when using the *time-based* method. Fix some node v out of the n nodes. Let X_v be the random variable specifying the size of v's view at the end of the convergence period. Each of the n nodes advertises itself to $s(n)$ uniformly chosen nodes. Thus, each node has a probability of $s(n)/n$ to advertise itself to v. Since advertisements of different nodes are independent of each other, X_v has a binomial distribution with parameters n and $s(n)/n$.

We conclude that the expectation of X_v is $s(n)$, as expected by our construction. In order to investigate the possible deviation of X_v from its mean, we use Chernoff bounds ([26]). We view X_v as the sum of n independent Bernoulli random variables $Y_1, \ldots, Y_n$, where Y_i is 1 if and only if the i-th network node advertises itself to v. By Chernoff bounds, for any $0 < \delta < 1$,

$$\Pr\left[|X_v - s(n)| > \delta s(n)\right] \;<\; 2 \cdot \exp(-s(n)\delta^2/3).$$

For example, for a value of $\delta = 0.5$, the probability for a given node to have a view size larger than $1.5 \cdot s(n)$ or smaller than $0.5 \cdot s(n)$ is less than $2/e^{s(n)/12}$.

By the union bound, the probability for any node to have a view size that differs from the average size by a factor of δ is:

$$\Pr\left[\exists v : |X_v - s(n)| > \delta s(n)\right] \; < \; 2 \cdot \exp(-s(n)\delta^2/3).$$

Conclusion. The view constructed by RaWMS in every node contains a random sample of nodes. Moreover, the probability that any view will deviate from the mean view size is very low (exponentially small with the average view size).

4.3 Reactive extension of the view

It is possible that a node will wish to extend its local view to a larger one upon its application's demand. Increasing the desired view size, $s(n)$, is a good long term solution, since it does not incur any additional communication and relies on existing advertisements. The drawback is that it may take a significant amount of time until the new target size is reached (increasing a view size by $s(n)$ will take $r(n) \cdot \Delta$ time units). On the other hand, maintaining a large view size all the time may be wasteful in case such a large view is typically not required or may even be impossible, if the node's memory is limited.

We propose an on demand RW-based method for extending the views, which allows to control the exact number of nodes that must be contacted in order to construct a larger view without knowing any network property. A node v requesting to extend its view up to a size of $es(n)$, starts a RW including its current view and the target view size, $es(n)$. Every node u that receives this message adds its view to the message while removing duplicates. If the combined view is smaller than $es(n)$, u sends the combined view to one of its neighbors picked at random. Once a combined view reaches the target size, it is sent back to v on the reverse path of the RW.

4.4 Network size estimation

RaWMS assumes that the number of nodes in the network n is known. This is required in order to determine the length of the RW in the reverse sampling procedure (the mixing time). There are a number of methods for obtaining a loose upper bound on the network size, e.g., [12, 30]. Once we have such a loose upper bound, we can use the birthday paradox principle to obtain a much tighter bound in the following manner. We have shown that according to the reverse sampling technique, every time RW stops at node u, it has the effect of having u pick uniformly at random a node identifier out of all n nodes. According to the famous birthday paradox, it is well-known that after $m = \sqrt{2n}$ random trials such that each trial picks uniformly one of n distinct values, the probability to pick m distinct values is at most $\frac{1}{e}$ and it drops rapidly as m increases ([26]). Therefore, every node can calculate the first time it receives the same advertisement again (denoted by m) and use this number to estimate n according to $n = \frac{m^2}{2}$. This process is repeated constantly and averaged across a number of measurements. In order to deal with accumulating errors, the loose upper bound should be re-used periodically and the tight bound estimation be restarted.

5. GOSSIP-BASED MEMBERSHIP

lpbcast. In each round of lpbcast [11], every node v sends its view to F (fanout) nodes, chosen randomly from v's view. In order to send such a message, some routing algorithm must be employed. The number of rounds is logarithmic in the network size. The main drawbacks of lpbcast is that it relies on costly routing, and according to [20], lpbcast may fail to provide uniform views.

Shuffling. Shuffling was first introduced in the context of sensor networks and originally used for information dissemination [14]. Yet, shuffling can also be used for construction of random views, by disseminating nodes identifiers. In shuffling, a node communicates only with its direct neighbors. Every round each node randomly picks X identifiers out of its view and shuffles them with its randomly chosen neighbor. Unlike other gossiping algorithms, Shuffling avoids loss of data during items exchange. This is accomplished by having the peers agree on which data items will be kept by each of them after the exchange takes place. Any two nodes that engage in a shuffle essentially swap a number of items. In doing so, they "move" this data around in a seemingly random fashion. We provide a performance analysis of shuffling by adapting some RW techniques to it in the full version of this paper [4].

Flooding. Flooding can be used to implement a membership service by having each node flood the network with its identifier. An efficient implementation of flooding requires memory which is linear in the number of nodes in the system. That is, in order to prevent a node from delivering (and retransmitting) the same message more than once, a node should remember the identifiers of the last few broadcast messages initiated by every other node. Since the implementation of flooding itself requires linear memory space, there is no point in limiting the view to include fewer than n identifiers.

5.1 Probabilistic starvation

One of the main usages of partial membership services is in gossip-based probabilistic multicast algorithms. Specifically, these algorithms attempt to deliver every message to almost every node with high probability. The percent of nodes that receive a message is called the *reliability factor*. However, those algorithms usually make no attempt to provide reliability for a single node. When the views are not truly random, there is a possibility that while most nodes receive all messages, a small number of nodes do not receive messages at all or receive only a small fraction of all messages. In particular, if there are some nodes (e.g., low degree nodes) that are not uniformly distributed among other nodes' views, those nodes will be constantly denied messages and thus suffer from *probabilistic starvation*. On the other hand, views constructed by RaWMS are proven to be uniform and therefore any probabilistic multicast algorithm built a top of it will not suffer from such a phenomenon.

5.2 Comparison

An asymptotic comparison of all the methods mentioned above appears in Table 1. Note that when nodes are mobile, there is an additional cost due to routing. In particular, lpbcast is highly affected by mobility since it relies heavily on unicast routing. When nodes move, routes break and must then be reestablished or repaired. In contrast, neither RaWMS nor shuffling suffer due to mobility, since they do not use multi-hop routing. In fact, in these two approaches, nodes' mobility can actually facilitate faster and more random dissemination of membership information.

6. SIMULATIONS

Model. The simulations were performed on the JiST/SWANS simulator [32] from Cornell university. Nodes use two-ray ground radio propagation model with IEEE 802.11 MAC protocol and 1Mb/sec throughput. The multi-hop routing protocol used by lpbcast is AODV (recall that RaWMS does not use routing at all). The mobility pattern was the Random Waypoint model with the speed of movement ranging from 0.5-2 m/s, which corresponds to walking and running speeds, and an average pause time of 30s. All simulations were run on networks of 10, 50, 100, 200, 400 and 800 nodes. We have used

	#rounds	time of a round	total time	msgs per round	total msgs sent	msg size	com. overhead	mem. overhead	additional overhead
RaWMS	$r(n)$	$T_{\text{actual\_mix}} = n$	$n + r(n)$	n^2	$n^2 \cdot r(n)$	1	$n^2 \cdot r(n)$	view size s	
lpbcast	$\log n$	Av. path length $\sqrt{\frac{n}{\log n}}$	$\sqrt{n \log n}$	$n \cdot \sqrt{\frac{n}{\log n}} \cdot F$	$n \cdot \sqrt{\frac{n}{\log n}} \cdot F \cdot \log n$	view size s	$n \cdot \sqrt{n \log n} \cdot F \cdot s$	memory for routing	Unicast routing
Shuffle	$\frac{s}{X} T_{\text{mix}} = \frac{s}{X} n$	1	$n \frac{s}{X}$	n	$n^2 \frac{s}{X}$	X	$n^2 s$	view size s	
Flooding	1	Av. path length $\sqrt{\frac{n}{\log n}}$	$\sqrt{\frac{n}{\log n}}$	n^2 broadcasts	n^2 broadcasts	1	n^2	linear memory for flooding.	Memory for flooding

Table 1: Comparing RaWMS with gossip-based membership and flooding

the default Java pseudo random number generator, initialized with the current system time in milliseconds as a seed.

The nodes were placed at uniformly random locations in a square universe. The transmission range was fixed for all network sizes and all simulations at 200m. The size of the simulation area was scaled in order to comply with the analytical results of Gupta and Kumar [16] regarding the critical transmission range. For a square area a^2 the radius of the transmission range is $r = a\sqrt{\frac{C \ln n}{n}}$, $r \in [0, a]$. The average number of nodes in the transmission range of any node was set to $d_{\text{avg}} = 3 \ln n$. This means that the simulation area a^2 for n nodes was picked such that $d_{\text{avg}} = \frac{\pi r^2 n}{a^2} = \frac{\pi a^2 \frac{C \ln n}{n} n}{a^2} = \pi C \ln n = 3 \ln n$, resulting in $a^2 = \frac{\pi 200^2 n}{3 \ln n}$ and $C \approx 1$. By proposition 1, for such a radius $d_{\max} \approx 2 d_{\text{avg}}$.

Each simulation lasted 1,000 seconds (of simulation time) and each data point was generated as an average of 10 simulation runs. Simulations started after a 60 seconds initialization period, which was enough to construct one hop neighborhood information. The neighbors discovery protocol was running throughout the entire simulation period in all scenarios. RaWMS was run with a *time-based* method; the node's descriptor timeout in the view was set so that the average view size will be $\sqrt{n}$. In each scenario of RaWMS, each node started $r(n)$ RWs, calculated out of the expected view size of $\sqrt{n}$ as described in Section 4.1. These advertisements were spread over the whole simulation period.

Uniformness of RaWMS. The first measure we used to evaluate RaWMS is the uniformness of the locations of nodes appearing in the views. To this end, we used a χ^2 statistical test to compare the distribution of nodes in the view of every node at the end of the convergence period with the desired uniform distribution. Namely, we have partitioned all nodes into a number of bins according to their distance from the tested node. For every node v we have calculated the following score: $Score_v = \sum_{j=1}^{\#bins} \frac{(Actual_{v,j} - Expected_{v,j})^2}{Expected_{v,j}}$, $Actual_{v,j}$ being the actual number of nodes from distance j found in the view of node v and $Expected_{v,j}$ is the number of nodes from distance j that are expected to be found in the view of node v. The total network score $TotalScore$ corresponds to the average of all $Score_v$s. Thus $TotalScore$ is a statistical test for the difference between the distribution of paths lengths obtained by simulations and the assumed uniform distribution.

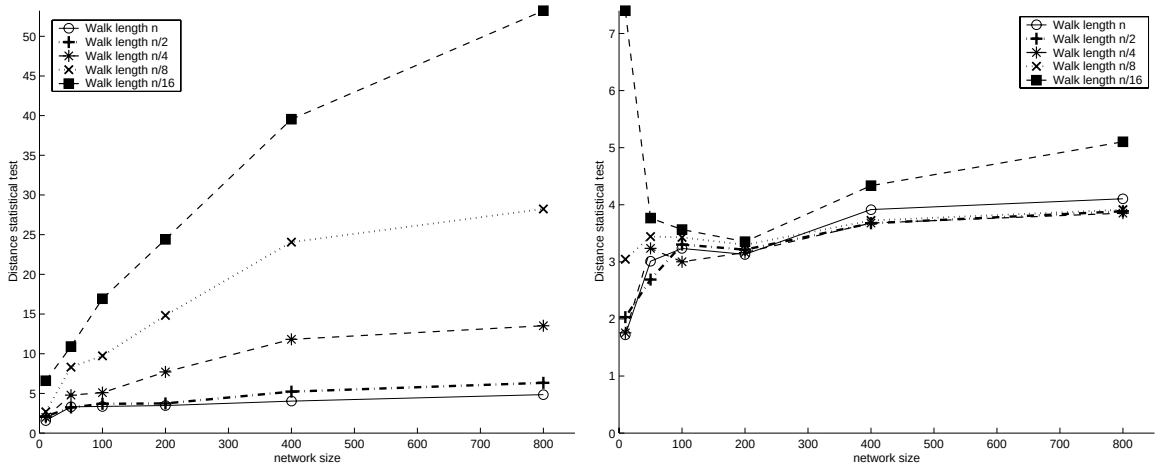
The results of the path length distribution test for static networks are depicted in Figure 2(a). The simulations were run with 5 different lengths of the random walk, corresponding to 5 different candidates for the mixing time, T_{mix}. Clearly, the longer the walk is, the closer is the distribution reached by the RW to the uniform stationary distribution, since a long walk has a "better" chance to reach a random node. We can see that for lengths of n and $n/2$ the $TotalScore$ is relatively low and almost does not change as the number of nodes grows. This means that walks of $n/2$ steps are long enough to correspond to the mixing time of those networks. Shorter walks exhibit a dramatic degradation in the test's score. The larger the network is, the worst are the results of these short RWs, since they do not get a chance to move far away from the originating node. As a result, every node ends up with relatively more nodes in its view that are geographically closer to it and with fewer nodes that are geographically far from it.

Figure 2(b) presents the results of our simulations with mobility. Interestingly, the random dissemination of membership information is actually improved by nodes movements, and even RWs of length $n/8$ get the same results as with length n. Nodes that used to be close to some node in the initial stage of the algorithm may end up in a completely different location in the network after some time, helping the "mixing" effect of the RW.

View size distribution. Recall that the size of the view is a binomial distributed random variable with probability $\frac{s(n)}{n}$, mean value $s(n)$ and variance $s(n)(1 - \frac{s(n)}{n})$. We have compared those expected values with the actual mean and variance values of view sizes at the end of the convergence process.

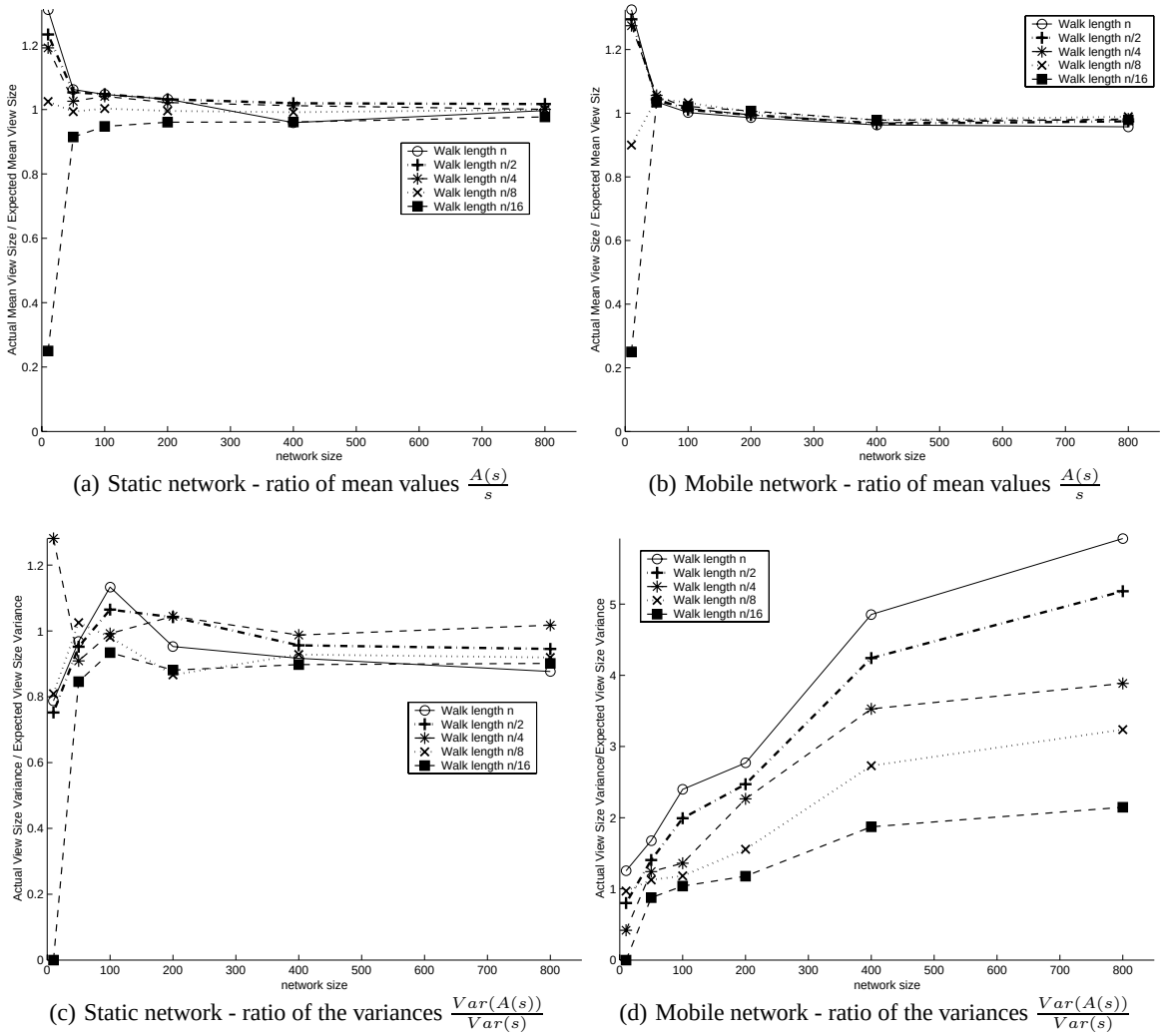
Figures 3 presents the graphs for $\frac{A(s)}{s}$ and $\frac{Var(A(s))}{Var(s)}$, with s representing the expected view size, $A(s)$ the actual mean view size, $Var(s)$ the expected variance, and $Var(A(s))$ the actual variance, calculated as $\frac{\sum_{i=1}^{n} (A(s) - view_i)^2}{n}$. For all network sizes and for all walk lengths, in both static and mobile networks, the average size of the view is almost equal (typically up to 90%) to the ideal expected mean size. Only for small networks the mean view size is a bit larger than expected, due to the fact that for s of the order of n, $\frac{ns}{n-s}$ is not a tight bound of $r(n)$ (see Lemma 2). In these cases, nodes simply start too many RWs. The variance in the view sizes is also very close to an expected one in the static networks, presenting another evidence to the fact that the view size is a binomial distributed random variable. The only exception is a small network of 10 nodes. For very short walks ($n/16$), the RWs did not get a chance to walk even a single step and the resulting view includes only the node itself. The variance is zero in such a case.

Notice that in mobile networks the variance is larger than in static networks. Surprisingly, in mobile networks, the variance is even larger for long RWs than for short RWs. The reason for this is as follows. A fast moving mobile node v has a lower chance of getting a RW message, because if v passes next to a node u that has the RW message, v disappears from the transmission range of u before



(a) Static network (b) Mobile network

Figure 2: RaWMS - the path length distribution test ($TotalScore$ versus n)



(a) Static network - ratio of mean values $\frac{A(s)}{s}$ (b) Mobile network - ratio of mean values $\frac{A(s)}{s}$

(c) Static network - ratio of the variances $\frac{Var(A(s))}{Var(s)}$ (d) Mobile network - ratio of the variances $\frac{Var(A(s))}{Var(s)}$

Figure 3: View size distribution - the difference between actual and expected mean and variance values

the neighbors discovery protocol at u detects v. The result is that static and slow moving nodes have a much larger view, at any given point in time, then fast moving nodes. This phenomenon becomes even worse in long RWs, since the longer the RW, the greater is the chance that it will be "stuck" at a static or slow moving node.

Intersection between views of neighboring nodes. We have compared the measured average size of the intersection between the views of all pairs of neighboring nodes with an expected intersection. For ideal uniformly chosen views there should not be any special correlation between the views of neighbor nodes. Since the average view size is $\sqrt{n}$, the expected intersection is $\sqrt{n}\frac{\sqrt{n}}{n} = 1$, for all network sizes. It can be seen from Figure 4(a) that indeed in static networks for long enough RWs (walks of length n and $n/2$) the average intersection size is very close to an expected one. However, walks shorter than the mixing time do not have enough steps to get far away from the originating node and tend to stop at its proximity instead of at a random node. As a result the neighbors of an originating node have a greater chance to have it in their views.

In mobile networks intersection between views of neighboring nodes is greatly reduced. Here, even short RWs can get a chance to escape the proximity of its originating node, due to mobile nodes carrying the RW message.

Correlation between node degree and view size. Additional tests were conducted to measure the correlation between nodes' degrees (the number of neighbors in the ad hoc network) and view sizes. We have omitted those figures due to space limitations, but they can be found in the full version of this paper [4]. The results have shown that RWs without self loops introduce a significant bias towards high degree nodes - much more RWs stop at these nodes than at lower degree nodes, resulting in unbalanced view sizes. Maximum Degree RW balances the node degree with self loops, generating a regular graph on which the RW has a uniform stationary distribution. This annuls the bias towards high degree nodes so all views have almost the same average size.

lpbcast. In our measurements, we have separated the routing communication overhead from the application communication overhead. This highlights why lpbcast is considered a very good protocol for peer-to-peer networks, but does not do so well in ad hoc networks. lpbcast was tested with a varying number of rounds: $\log n$, $2\log n$, $4\log n$, $8\log n$, $16\log n$. The fanout was set to 3 for all simulations and the view size limit was set to $\sqrt{n}$, to establish the same conditions as with RaWMS. As can be seen in Figure 5(a), in static networks, when the number of gossip rounds is $2\log(n)$ or less, the resulting view is not uniform according to the path length distribution test. As for a view size of lpbcast, since it was limited to $\sqrt{n}$ and since nodes gossip their entire view, in almost all cases the view was full. Here too, as can be seen in Figure 5(b), the uniformity of the views is dramatically improved when nodes are mobile.

RaWMS versus lpbcast - communication overhead. Figure 6 depicts the number of messages sent by a single node during the entire simulation period, in both RaWMS and lpbcast. We have separated the number of application messages (messages directly generated by RaWMS and lpbcast) from the total number of network messages, which include the cost of routing and the neighbor discovery protocol messages. We have chosen to present RaWMS with a walk length of $n/2$ and lpbcast with $4\log n$ rounds, as these give optimal results, respectively. That is, these are the most efficient versions of both protocols, which still guarantee a fairly uniform distribution of views at the lowest possible cost.

We can see that the results generally follow our theoretical discussion in Section 5.2. In RaWMS, each node starts roughly $\sqrt{n}+2$ RWs, each walk sending $T_{\text{actual\_mix}} \leq T_{\text{mix}}\frac{d_{\max}}{D}$ messages. T_{mix} was approximated by $n/2$ according to the results in Figure 2 and D was set large enough to bound $d_{\max}$. The measured $T_{\text{actual\_mix}}$ was about $T_{\text{mix}}/2$. Thus, each node sends a total number of $\frac{n\sqrt{n}}{4}$ messages. In lpbcast, every node starts $4\log n$ rounds with fanout 3 and each message traverses the network over an average path of $\sqrt{\frac{n}{\log n}}$. Therefore, each node sends $12\sqrt{n\log n}$ messages in total.

As is evident from Figure 6(a), lpbcast generates fewer application messages than RaWMS, as expected by our previous analysis. Yet, recall that in lpbcast each message contains the whole view, while in RaWMS messages carry only a single node identifier. Therefore, the total bit communication overhead of lpbcast is $12n\sqrt{\log n}$. In addition, lpbcast has a significant message overhead due to routing. When adding the cost of routing, RaWMS becomes considerably more efficient than lpbcast.

Figure 6(b) illustrates the communication costs of RaWMS with a walk length of $n/8$ and lpbcast with $2\log n$ rounds in mobile scenarios. Again, those parameters guarantee a uniform distribution of views at the lowest possible cost. Here, the cost of RaWMS is significantly lower than lpbcast. This is due to a decreased walk length, yet without compromising the uniformness of the views. In this scenario, each node sends about $\frac{n\sqrt{n}}{16}$ messages. lpbcast sends approximately the same number of application messages as in the static case. However, with mobility, the cost of routing becomes considerable, which accounts for the dramatic affect on the overall performance of lpbcast in terms of network messages.

7. RELATED WORK

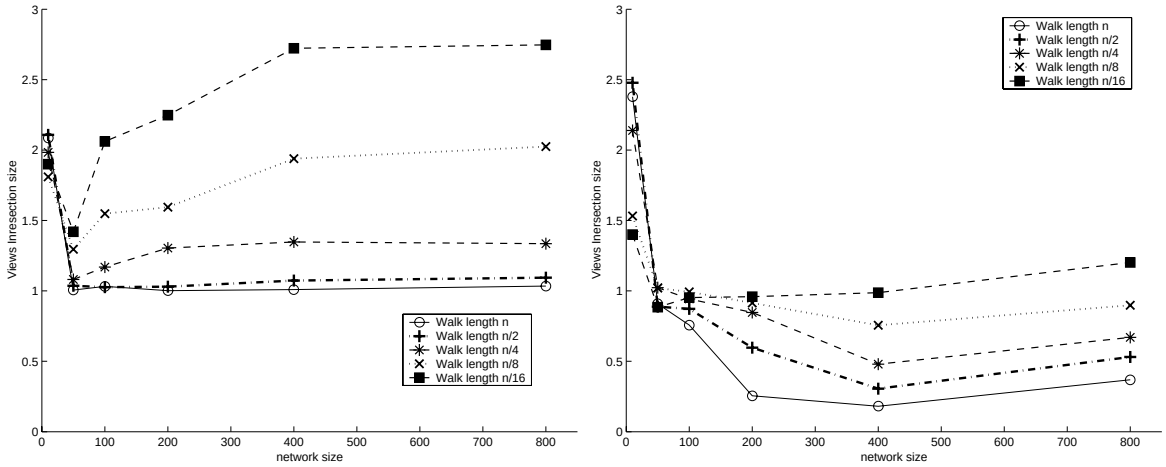
Random walks. Comprehensive surveys of random walk techniques and their analysis appear in [22] and [17]. The idea of using a "maximum-degree" RW to reach a uniform limit distribution on the state space has been used before in a number of contexts [3, 7].

Lv et al. [24] propose to use simulated RWs for searching in unstructured peer-to-peer networks. They report that such a search is preferable to searching by flooding, due to RWs' adaptiveness to termination conditions and a fine-grain control of the search space. This work reported attractive empirical results, but does not provide any analytical evaluation of the RW properties.

The work in [15] explore the performance of RWs for searching and sampling in peer-to-peer networks and show that it is possible to simulate the selection of a uniform sample of elements from the network by performing a RW with an adequate length. We use a similar sampling technique, but on a completely different communication graph. Peer-to-peer networks graphs are usually assumed to be expanders. On the other hand, ad hoc network graphs are random geometric graphs [28], which are not expanders.

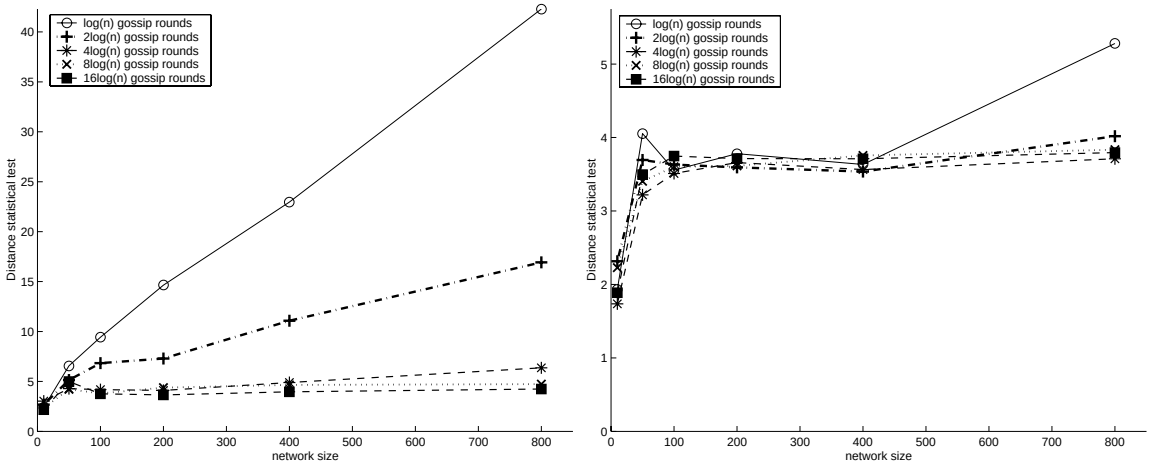
Various properties of RWs on random geometric graphs, including the mixing time and the partial cover time, have been investigated by [7] and [2]. We rely on these results in our work.

Dolev et al [10] propose a randomized self-stabilizing group membership service for ad hoc networks. The group membership list is collected by a single random walk agent traversing the network. However, [10] only constructs a full membership while RaWMS can be used to construct partial membership views. Moreover, they apply a single RW that covers the whole network and runs for a period of time that is equal to the cover time. We use multiple RWs simultaneously each running for a period that is equal to the mixing time. Thus, the time and communication complexities of the algorithm in [10] are $O(n^3)$, while in RaWMS each RW runs for only $O(n)$. The communication complexity of RaWMS depends on the desired view size. For example, to construct a view of size $O(\sqrt{n})$



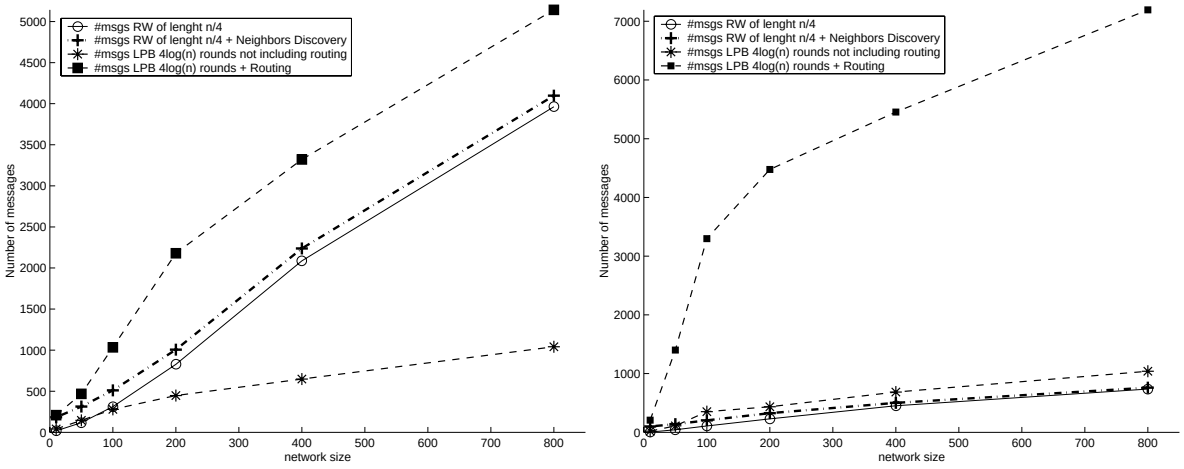
(a) Static network (b) Mobile network

Figure 4: Intersection between views of neighboring nodes



(a) Static network (b) Mobile network

Figure 5: lpbcast - path length distribution test ($TotalScore$ versus n)



(a) Static network (b) Mobile network

Figure 6: RaWMS versus lpbcast - comparing the number of messages

at every node, RaWMS sends a total of $O(n^2\sqrt{n})$ messages. A full membership can be constructed with RaWMS by an additional short RW that collects partial random views from different nodes. The total communication complexity in this case is $O(n^2\sqrt{n})$.

In [30] RWs are used for routing in large-scale sensor networks. They assume a static network and only consider a grid topology. On the other hand, we also support mobility, and do not restrict the topology except for being connected.

Gossiping. Gossiping is another well-known scheme to establish a random sample. Recently, gossip-based dissemination of membership information was proposed in order to design scalable implementations of a peer sampling service [20]. Other examples of gossip-based lightweight membership services are reported in [1, 11, 13] and are discussed in more details in Section 5.

SCAMP [13] introduced a generic random membership service that is used for probabilistic reliable dissemination of data and events in peer-to-peer networks. The appealing property of SCAMP is that the partial view obtained by a node adapts automatically to the system's size, without any a priori knowledge of the total network size. However, [13] only proves that the mean value of the sum of all views of all nodes is $\Theta(n \log n)$ and that the actual sum of all view sizes is not far from the mean. No proof is provided about the view size of a single node, which may be far from the mean by orders of magnitude. In our work, we do bound the minimal and maximal view sizes of all nodes.

RDG [23] is an adaptation of [11] to ad hoc networks. It reduces the cost of routing compared to [11] by utilizing routes created by other applications running in the same wireless node or by using proactive periodical flooding in order to establish those routes. Although RDG relies only on partial views for correct implementation of probabilistic multicast, in practice the views constructed by RDG are not necessarily partial and may even be almost full views. In addition, those views are not constructed by gossiping, but by the same flooding that establishes the routes. Gossiping is only used in RDG for data dissemination and for removal of nodes that left the network from the views. The usage of flooding results in a linear memory consumption, so there is no point in using it for constructing partial views.

Haas et al. have investigated various approaches for disseminating data using several gossip functions in ad hoc networks [18]. They investigate the impact of gossip on the message delivery ratio of broadcast messages. The *anonymous gossip* work has explored the use of gossip with direct neighbors in an ad hoc network to increase the reliability of broadcast and multicast protocols [8]. Both these works, however, do not address membership maintenance.

8. DISCUSSION

In this paper we have presented RaWMS, a random walk based lightweight membership service for ad hoc networks. We have presented a formal analysis of RaWMS, backed by simulations and have also compared RaWMS with gossip-based approaches for building such membership services. Overall, the results of the simulations confirm the formal analysis. They show that RWs present an attractive paradigm for implementing partial view based membership services in ad hoc networks. This is due to the fact that RWs do not require multi-hop routing and avoid flooding altogether. Moreover, when the network is mobile, RWs reach their target uniformity even faster than in static networks. In these cases, the mobility helps to disseminate messages to random places in the network

Finally, we believe that our analysis of RW's complexity for ad hoc networks can serve as a starting points for many additional RW-based algorithms in ad hoc networks.

9. REFERENCES

[1] A. Allavena, A. Demers, and J. E. Hopcroft. Correctness of a gossip based membership protocol. In *Proc. of the 24th PODC*, pages 292–301, 2005.

[2] C. Avin and G. Ercal. Bounds on the mixing time and partial cover of ad-hoc and sensor networks. Technical Report 040028, UCLA, June 2004.

[3] Z. Bar-Yossef, A. Berg, S. Chien, J. Fakcharoenphol, and D. Weitz. Approximating Aggregate Queries about Web Pages via Random Walks. In *Proc. of the 26th VLDB*, pages 535–544, 2000.

[4] Z. Bar-Yossef, R. Friedman, and G. Kliot. RaWMS - Random Walk based Lightweight Membership Service for Wireless Ad Hoc Networks. Technical Report CS-2006-05, Technion, Haifa, Israel, January 2006.

[5] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM TOCS*, 17(2):41–88, 1999.

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

[7] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Mixing times for random walks on geometric random graphs. In *The 2nd Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, January 2005.

[8] R. Chandra, V. Ramasubramanian, and K.P. Birman. Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks. In *Proc. of the 21st ICDCS*, page 275, 2001.

[9] G. Chockler, I. Keidar, and R. Vitenberg. Group communication specifications: a comprehensive study. *ACM Computing Surveys*, 33(4):427–469, 2001.

[10] S. Dolev, E. Schiller, and J. Welch. Random walk for self-stabilizing group communication in ad hoc networks. In *Proc. of the 21st PODC*, pages 259–259, 2002.

[11] P. Th. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems (TOCS)*, 21(4):341–374, 2003.

[12] U. Feige. A fast randomized LOGSPACE algorithm for graph connectivity. *Theoretical Computer Science*, 169(2):147–160, 1996.

[13] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulie. SCAMP: Peer-to-Peer Lightweight Membership Service for Large-Scale Group Communication. In *Networked Group Communication*, pages 44–55, 2001.

[14] D. Gavidia, S. Voulgaris, and M. van Steen. Epidemic-style Monitoring in Large-Scale Sensor Networks. Technical Report IR-CS-012, Vrije Universiteit, Amsterdam, Netherlands, March 2005.

[15] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *Proc. of the 23rd INFOCOM*, pages 259–259, 2004.

[16] P. Gupta and P. Kumar. Critical Power for Asymptotic Connectivity in Wireless Networks. In *Stochastic Analysis, Control, Optimization and Applications, Birkhauser, Boston*, pages 547–566, 1998.

[17] V. Guruswami. Rapidly mixing markov chains: a comparison of techniques. May 2000, Available at `http://www.cs.washington.edu/homes/venkat/pubs/pubs.html`.

[18] Z. Haas, J. Halpern, and L. Li. Gossip-Based Ad Hoc Routing. In *Proc. of the 21st INFOCOM*, pages 1707–1716, June 2002.

[19] Z. Haas and B. Liang. Ad Hoc mobility management with randomized database groups. In *Proc. of ICC*, volume 3, pages 1756 – 1762, June 1999.

[20] M. Jelasity, R.Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In *Proc. of the 5th Middleware*, pages 79–98, 2004.

[21] A.-M. Kermarrec, L. Massoulie, and A. J. Ganesh. Probabilistic Reliable Dissemination in Large-Scale Systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3):248–258, March 2003.

[22] L. Lovász. Random Walks on Graphs: A Survey. *Combinatorics*, 2:1–46, 1993.

[23] J. Luo, P.Th. Eugster, and J.-P. Hubaux. Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks. In *Proc. of 23rd INFOCOM*, 2003.

[24] C. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proc. of 16th ICS*, pages 84–95, 2002.

[25] R. Melamed and I. Keidar. Araneola: A Scalable Reliable Multicast System for Dynamic Environments. In *Proc. of the 3rd IEEE NCA*, pages 5–14, 2004.

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

[27] P. Panchapakesan and D. Manjunath. On the Transmission Range in Dense Ad Hoc Radio Networks. In *Proc. of IEEE SPCOM*, 2001.

[28] M. D. Penrose. *Random Geometric Graphs*. Oxford Press, 2003.

[29] H. Pucha, S.M. Das, and Y. C. Hu. Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks. In *Proc. of the 6th WMCSA*, pages 163–173, 2004.

[30] S. Servetto and G. Barrenechea. Constrained Random Walks on Random Graphs: Routing Algorithms for Large Scale Wireless Sensor Networks. In *Proc. of the 1st ACM WSNA*, 2002.

[31] A. Sinclair. Improved bounds for mixing rates of Markov chains and multicommodity flow. *Combinatorics, Probability & Computing*, 1:351–370, 1992.

[32] Cornell University. JiST/SWANS. `http://jist.ece.cornell.edu/`.