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

January 30, 2006

Abstract

This paper presents RaWMS, 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 random walk (RW) sampling technique. The paper includes a formal analysis of both the RW sampling technique and RaWMS and verifies it through a detailed simulation study. In addition, RaWMS is compared both analytically and by simulations with a number of other known methods such as flooding and gossip-based techniques.

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 [10], 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, 14, 27], peer sampling services [26], location services and uniform quorums [23], random overlay constructions [31], DHTs in which the hash function is calculated on node ids [35], 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 the *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.

Notice that the uniform randomness of the view is defined here not on peer identifiers, but rather on the fact that each view should include identifiers of nodes from random parts of the network. This is because the target applications of such membership services depend on the fact that there is very little overlap in the views of any pair of neighboring nodes. Clearly, when node identifiers are picked at random, the immediate neighbors of each node form a uniform random set of identifiers. However, such sets of identifiers for any pair of neighboring nodes will typically have considerable overlap. Thus, we are interested in ensuring that each view will include identifiers of nodes from random parts of the network.

Unlike many gossip-based algorithms, our service possesses four important properties. These include 

1. proven uniform randomness of the constructed views, 
2. analytically proven bounds on the load of an individual node (view size), 
3. enabling each node to set its view size independently of other nodes without any implications on the randomness of the views’ content, 
4. a low chance of partition in the knowledge graph induced by the views, and 
5. 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. This is done by consulting its neighboring nodes, which carries an additional small communication overhead.

We provide a detailed formal analysis of our implementation of RaWMS. Additionally, we extend the generic gossip-based peer sampling framework introduced in [26] to incorporate ad hoc networks. We utilize it to compare RaWMS with other membership construction techniques, such as lpbcast [14], Shuffling [17] and flooding. Finally, we study the performance of RaWMS by simulations, evaluating its properties and comparing it to the other known gossip and flooding-based techniques mentioned above. These measurements largely confirm our theoretical analysis.

**Paper’s road-map:** Section 2 introduces the system model. In Section 3, we present the RW technique for peer sampling in ad hoc networks. Section 4 describes RaWMS and its formal analysis. Section 5 describes a generic framework used in a variety of gossiping algorithms for membership construction. Section 6 presents the simulation results for RaWMS vs. known gossip-based membership services. Section 7 discusses related work and we conclude with a discussion in Section 8.

1To clarify, in our model nodes have no geographic knowledge. However, the formal analysis and simulations show that we still obtain the desired randomness.
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 [34]. 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 [19, 20]. See Appendix A for a formal description of the model.

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

3.1 Basics of random walks

Informally, a random walk (RW) on a network, or its corresponding graph, is a process in which a token (or a special message) is repeatedly forwarded from a node that holds this token to one of its neighbors. Each such token pass is also referred to as a step.

Formally, let $G = (V, E)$ be an undirected graph, $|V| = n$. Let $d_v$ denote the degree of a vertex $v \in V$. A random walk on $G$ is specified by an $n \times n$ probability transition matrix $P$. For every $(v, u) \in E$, $P_{v,u}$ defines the probability that a token placed on vertex $v$ at time $t$ will move to a neighboring vertex $u$ at time $t + 1$ (for $(v, u) \notin E$, $P_{v,u} = 0$). Note that this process is memory-less: the transition probability is independent of the history of the random walk. In a simple random walk, the transition probability is uniform for all neighbors of $v$. That is,

$$P_{v,u} = \begin{cases} 1/d_v, & \text{if } (v, u) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

In practice, the transmission range does not behave exactly as a disk due to various physical phenomena. However, for the description of the protocol it does not matter, and on the other hand, a disk assumption greatly simplifies the formal model. At any event, our simulation results are carried on a simulator that simulates a real transmission range behavior including distortions, background noise, etc.
For every time step $t \geq 0$, $\phi_t$ 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 $\phi_0$ specifies the vertex at which the random walk is started. Typically, random walks start at a fixed node, and hence $\phi_0$ is concentrated on a single node. For every $t \geq 1$, $\phi_t = \phi_{t-1}P$, and thus by induction $\phi_t = \phi_0P^t$.

If the transition matrix is ergodic (meaning that the graph $G$ is connected and non-bipartite), then for every initial distribution $\phi_0$, the sequence of distributions $\{\phi_t\}_{t=0,1,2,...}$ is guaranteed to converge to a limit distribution. Furthermore, this limit distribution is independent of the initial distribution. That is, there is a unique limit distribution $\pi$ on $V$ s.t. for all $\phi_0$, $\phi_0P^t \to \pi$ as $t \to \infty$. It is easy to check that $\pi$ is also a stationary distribution of $P$, that is, $\pi P = \pi$, meaning that if the random walk happens to reach the distribution $\pi$, it stays there forever.

A simple analysis (cf. [28]) shows that the stationary distribution of the simple random walk has a limit distribution that assigns probabilities to nodes proportionally to their degree:

**Proposition 3.1.** The stationary limit distribution of a simple random walk over an undirected, connected, and non-bipartite graph $G = (V,E)$ is $\pi(v) = \frac{d_v}{\sum_{u \in V} d_u}$, for every $v \in V$.

The above proposition implies that 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 a different random walk, the Maximum Degree random walk, whose stationary distribution is uniform even for non-regular graphs.

### 3.1.1 RW-based sampling

Suppose we have a random walk on a network graph $G$ and that the stationary distribution of this random walk is $\pi$. Figure 1 shows an algorithm that uses the random walk to sample nodes from $G$ according to the distribution $\pi$. In particular, if $\pi$ happens to be the uniform distribution on the network’s nodes, 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 $\phi_T$ of the node returned is close to the limit distribution $\pi$.

```plaintext
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
```

**Figure 1: A Generic RW-based Sampling Algorithm**

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, as is done in Section 3.3.

The main question to be addressed is how to set $T$ to guarantee that $\phi_T$ is close to $\pi$. In other words, we would like to study the convergence rate of the random walk. To this end, we introduce a notion of distance used to characterize convergence, and then define the mixing time of a random walk:

**Definition 3.2.** For every node $v \in V$, let $\phi_0^v$ be the initial distribution concentrated on $v$. For every step $t \geq 0$, the total variation distance between $\phi_0^vP^t$ and $\pi$ is defined as:

$$\Delta_v(t) = \frac{1}{2} \sum_{u \in V} |\phi_0^v(u) - \pi(u)|.$$
For every $\epsilon > 0$, the $\epsilon$-mixing time of the random walk is:

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

Intuitively, the mixing time of a random walk 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 $\epsilon$-close to the stationary distribution. Throughout this paper, when the parameter $\epsilon$ is omitted, we refer to mixing time with $\epsilon = \Theta\left(\frac{1}{n}\right)$.

Probably the most popular method for bounding the mixing time of a random walk is via the spectral gap of its transition matrix. Let $\lambda_1(P) \geq \lambda_2(P) \geq \cdots \geq \lambda_n(P)$ be the $n$ eigenvalues of $P$. It can be shown that all these eigenvalues must be real and lie in the interval $[-1, 1]$. The principal eigenvalue, $\lambda_1(P)$, must equal 1, because $P$ is a stochastic matrix (a non-negative matrix, all of whose rows sum to 1). We define $\lambda_{\max}(P) = \max\{\lambda_2(P), -\lambda_n(P)\}$. When the graph $G$ is connected then $\lambda_2(P) < 1$ and when it is non-bipartite, $\lambda_n(P) > -1$. Hence, for a connected non-bipartite graph, $1 - \lambda_{\max}(P) > 0$. The difference $1 - \lambda_{\max}(P)$ is called the spectral gap of $P$ and turns out to determine the mixing time of the random walk (cf. [21]):

**Theorem 3.3.** The mixing time of a random walk with transition matrix $P$ is upper bounded as follows:

$$T_{\text{mix}}(\epsilon) \leq \frac{\ln \pi_{\min}^{-1} + \ln \epsilon^{-1}}{1 - \lambda_{\max}(P)},$$

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

When $\pi$ is the uniform distribution then $\pi_{\min} = 1/n$, implying the following upper bound on the mixing time:

$$T_{\text{mix}}(\epsilon) \leq \frac{\ln n + \ln \epsilon^{-1}}{1 - \lambda_{\max}(P)}.$$ 

Theorem 3.3 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 $\epsilon$, we can use the above formula to calculate $T$.

### 3.1.2 The Maximum Degree RW

As mentioned above, the simple random walk 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 random walk directly to obtain uniform sampling of network nodes. Instead, we use a different random walk, called the Maximum Degree (MD) random walk, which has been used before in various contexts [28, 4, 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 $d_{\text{max}}$, the maximum degree of $G$.

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:

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

\(^3\)We show a way of doing this in Section 3.2 below. For now, we simply assume that such a bound exists.
If \( G \) is connected, then so is \( G' \). \( G' \) is non-bipartite (even if \( G \) is), because every node has a self loop. Therefore, by Proposition 3.1, the MD random walk has a limit distribution. Furthermore, since \( G' \) is regular, this distribution is uniform:

**Proposition 3.4.** If \( G \) is an undirected and connected graph, then the MD random walk on \( G \) converges to the uniform limit distribution.

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.

### 3.2 Random walks on ad hoc networks

In this section we show that the MD random walk can be used to sample nodes uniformly from ad hoc networks. We then formally analyze the mixing time of this random walk.

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, by Proposition 3.4, 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.

The symmetry assumed in the theoretical model of RGGs is not always valid in real ad hoc networks, because, in practice, it is possible that a node \( v \) receives messages sent from node \( u \), but not vice versa. Yet, this phenomenon is rare. At any event, our theoretical results were verified through extensive simulation with real transmission range behavior including distortions, background noise, etc.

**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 [19] 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 [33], 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 of nodes, where \( C \) is a constant. For \( C > 1/\pi \), this radius satisfies the connectivity condition of Gupta and Kumar. Thus, we can assume with high probability that the ad hoc network graph is connected.

**Remark:** Since the Euclidean distance between any two points on the surface of a 2-dimensional torus is at most \( 1/\sqrt{2} \), then a radius \( r \geq 1/\sqrt{2} \) implies that \( G^2(n, r) \) must be a clique (i.e., every two nodes are connected by an edge). Furthermore, if \( r > 1/2 \), then a disk of radius \( r \) centered at any point on the torus “wraps around” itself. In order to avoid such anomalies, throughout the paper we postulate that \( r \leq 1/2 \). When \( r > 1/2 \), the connectivity of \( G^2(n, r) \) (and also the fast mixing of the MD random walk on \( G^2(n, r) \)) are achieved trivially since the graph is a clique or close to a clique.
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 random walk.

**Proposition 3.5.** 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 \( d_{\text{avg}}, d_{\text{max}}, \) and \( d_{\text{min}} \) be, respectively, the average, maximum, and minimum degree of the random geometric graph \( G^2(n, r) \). Then,

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

and

\[
\Pr \left[ (1 - \delta_d) \cdot \pi r^2(n - 1) \leq d_{\text{min}} \leq d_{\text{max}} \leq (1 + \delta_d) \cdot \pi r^2(n - 1) \right] > 1 - \alpha_d.
\]

**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 \( \pi r^2 \).

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 \( d_{\text{avg}} = \frac{1}{n} \sum_{i=1}^{n} Y_i \). Hence, using linearity of expectation again, \( E(d_{\text{avg}}) = \pi r^2(n - 1) \).

By Chernoff bounds (see Appendix B),

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

Then,

\[
\Pr(|Y_i - \pi r^2(n - 1)| > \delta_d \cdot \pi r^2(n - 1)) \leq 2 \cdot \exp(-\frac{\delta_d^2 \pi r^2(n - 1)}{3}) = \frac{\alpha_d}{n}.
\]

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 \( \alpha_d \). \( \square \)

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 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 random walk. We are able to show the following:

**Theorem 3.6.** 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 \( T_{\text{mix}}(\epsilon) \) be the mixing time of the MD random walk on this graph, when applied with the value \( D \). Let
$T_{\text{actual, mix}}(\epsilon)$ 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),

$$T_{\text{mix}}(\epsilon) \leq \frac{30}{(1 - \frac{\epsilon}{\sqrt{C}})^2} \cdot \frac{D}{n} \cdot \frac{1}{r^4} \cdot (\ln n + \ln \epsilon^{-1}).$$

$$T_{\text{actual, mix}}(\epsilon) \leq \frac{120}{(1 - \frac{\epsilon}{\sqrt{C}})^2} \cdot \frac{1}{r^4} \cdot (\ln n + \ln \epsilon^{-1}).$$

The proof of Theorem 3.6 is rather involved, and is therefore deferred to Appendix C. Several remarks are in order.

1. If $d_{\text{max}} \approx \pi r^2 n$ (as guaranteed w.h.p. by Proposition 3.5) and if we choose $D$ to be close to $d_{\text{max}}$, then the mixing time of the MD random walk is $T_{\text{mix}}(\epsilon) = O(r^{-2}(\ln n + \ln \epsilon^{-1}))$. For our choice of $r$, if $C$ is a constant, then this mixing time is $T_{\text{mix}}(\epsilon) = O(n(1 + \frac{\ln \epsilon^{-1}}{\ln n}))$. On the other hand, if $D$ is a gross overestimate of $d_{\text{max}}$, then the mixing time can get higher.

2. As we show in the proof of this theorem (see appendix), the actual mixing time relates to the standard mixing time as follows:

$$T_{\text{actual, mix}}(\epsilon) \leq \frac{d_{\text{max}}}{D} \cdot T_{\text{mix}}(\epsilon).$$

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

$$T_{\text{actual, mix}}(\epsilon) = O(n(1 + \frac{\ln \epsilon^{-1}}{\ln n})).$$

3. The theorem exhibits a tradeoff between the mixing time and the radius $r$: the larger is the radius $r$, the smaller is the mixing time. For example, if a radius $r = \sqrt{\frac{C \ln n}{n}}$ with $C \geq 324$, the actual mixing time is $n(1 + \frac{\ln \epsilon^{-1}}{\ln n})$. 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. For small networks, the lower bound on the radius $r$ required to guarantee fast mixing may be at odds with the upper bound $r \leq 1/2$. For example, if we wish to set $C = 324$ as above, then we need $n \geq 9,300$, in order for $r$ to be at most $1/2$. For smaller networks, the lower bound on $r$ simply means that the graph $G^2(n, r)$ is a clique. In cliques (with self loops), the random walk mixes in a single step. The minimum network size, for which the above theorem gives a non-trivial result is obtained by setting $C = 50$, in which case $n \geq 1,060$.

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 3.6 is applicable only for quite large radii ($C > 49$) are artifacts of the involved theoretical analysis and not of the algorithm itself. We believe that in practice the random walk 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 $T_{\text{mix}}(\epsilon) = n/2$.

### 3.3 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 of Figure 1 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. In either case, this introduces 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.

**Lemma 3.7.** 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 every \( k \) distinct nodes \( v_1, \ldots, v_k \),

\[
\Pr(Z_u = \{v_1, \ldots, v_k\} \mid |Z_u| = k) = \frac{1}{\binom{n}{k}}.
\]

Fix any \( u \), any \( k \in \{1, \ldots, n\} \), and any \( v_1, \ldots, v_k \). By Bayes rule,

\[
\Pr(Z_u = \{v_1, \ldots, v_k\} \mid |Z_u| = k) = \frac{\Pr(|Z_u| = k \mid Z_u = \{v_1, \ldots, v_k\}) \cdot \Pr(Z_u = \{v_1, \ldots, v_k\})}{\Pr(|Z_u| = k)}.
\]

We next analyze each of the three terms on the RHS of Equation 1. For the first term, we have \( \Pr(|Z_u| = k \mid Z_u = \{v_1, \ldots, v_k\}) = 1 \).

Regarding \( \Pr(Z_u = \{v_1, \ldots, v_k\}) \), note that \( Z_u = \{v_1, \ldots, v_k\} \) iff \( X_{v_1} = u, X_{v_2} = u, \ldots, X_{v_k} = u \), and for every \( v \notin \{v_1, \ldots, v_k\}, X_v \neq u \). 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, \( \Pr(Z_u = \{v_1, \ldots, v_k\}) = \left(\frac{1}{n}\right)^k \cdot (1 - \frac{1}{n})^{n-k} \).

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

\( \square \)

## 4 Random walk based membership service

As discussed above, the goal of the membership service is to provide every node with a partial view of network nodes, chosen uniformly at random out of all network nodes. In the implementation of RaWMS, a View at a node \( v \) is defined as a set of node descriptors, where each descriptor consists of \(<\text{NodeId},\text{LastTime}>\). 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 \( \Delta \) time units by starting a reverse sampling process, as described in Section 3.3. In other words, each \( \Delta \) time units, \( v \) starts a Maximum Degree random walk, whose messages carry \( v \)'s identifier. Each of these random walks traverses the network for a number of steps that is equal to the mixing time and stops at some destination node \( u \). If \( u \) already has a descriptor corresponding to \( v \) in its view, then \( u \) refreshes the last time it heard from \( v \) and discards the RW.
do forever
    wait(Δ time units);
    // start a new RW
    ttl ← MixingTime;
    handleRW(myAddress,ttl);
    if timeoutBasedMethod then
        discardExpiredFromView(View,Timeout)
    endif;
endo;

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

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

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

Figure 2: RaWMS - code for node v

Otherwise, u may decide whether it wants to store the identifier of v in its view or not. We propose two methods for such a decision: 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. When a RW message including an identifier of a new node v stops at a given node u, u adds v’s NodeIdentifier to its view. If the view includes now more descriptors than the size limit, then u discards a descriptor with the oldest LastTime from its view. This way, purging descriptors of nodes that already left the network is dealt with automatically.

In the time-based method, every node discards nodes’ descriptors according to some 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 parameter proportionally to the mixing time and Δ. This also facilitates purging descriptors of nodes that already left the network.

The general structure of RaWMS is presented in Figure 2. 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; finally, storeInView(View,addr) stores in the view a new descriptor corresponding to a given address and the current time. pickNextNode picks either one of the neighboring nodes or a self-loop (of the current node) according to the RW transition matrix probabilities.

Notice that RaWMS can also support construction of different views for different groups. Nodes periodically advertise themselves to all groups they belong to (every RW advertises the source node to
all groups simultaneously. When RW stops, destination nodes can filter the source node according to the groups it is in.

4.1 Formal performance analysis

For the performance analysis of RaWMS, we assume that all nodes start the algorithm simultaneously with initial empty views. Also, for the purpose of analysis, we assume that 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) only relies 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 our membership service, we study the time and the communication complexity of the protocol throughout the convergence period. As already mentioned, each node may pick the target size of its view independently from other network nodes by enforcing a view size limit or by using an aging timeout. Obviously, the target view size 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.

We start the analysis by computing the convergence time. We continue by calculating the communication overhead of each node and for the entire network during the convergence period. We then conclude by evaluating the performance of regular maintenance, supporting joins by new nodes (or “late-comers”), and leaves.

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 at each node \( s(n) \) RWs during the convergence period. However, RWs may “collide”, in the sense that two random walks that start 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 \( r(n) \). Clearly, once we compute \( r(n) \), we can immediately compute the communication and time complexity to reach convergence.

4.1.1 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. The famous coupon collector problem is a special case of the bins and balls problem with \( s = n \) (what is the expected number of coupons that must be drawn uniformly at random from the pack until all different \( n \) coupon types are collected). 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 4.1.** 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 (and define \( H_0 = 0 \)).
Note that using the inequality $1 + x < e^x$, which holds for all $x > 0$, we have:

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

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 nonempty bin with probability $\frac{1}{n}$. Using the independence assumption, the expected number of balls required to have a second nonempty 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 nonempty 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. $r$ is therefore a sum of geometric random variables. By linearity of expectation, we have:

$$E(r) = 1 + \frac{n}{n-1} + \frac{n}{n-2} + \cdots + \frac{n}{n-s+1}$$

$$= n\left(\frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + \cdots + \frac{1}{n-s+1}\right)$$

$$= n(H_n - H_{n-s}).$$

In order to bound the difference $H_n - H_{n-s}$, we use the following well-known bounds on the harmonic number (see, e.g., [13]):

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

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

$$n(H_n - H_{n-s}) \leq n\left(\ln n + \gamma + \frac{1}{2n} - \ln(n-s) - \gamma - \frac{1}{2(n-s+1)}\right)$$

$$\leq n(\ln n - \ln(n-s)) = n \ln \frac{n}{n-s}. \quad \square$$

Note that nodes start new RW every $\Delta$ 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, there is one usage of $r(n)$ that can be exploited by nodes working in the *time-based* method - it can be used for adjusting 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.

### 4.1.2 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, we get that the 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 about \( n \cdot \sqrt{n} \cdot T_{\text{actual mix}} = \Theta(n^2 \sqrt{n}) \).

### 4.1.3 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 period of time that is equal to the convergence time, the 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.

In order to speed up the uniform dissemination of new nodes in the network, a new node may initially advertise itself more frequently than \( 1/\Delta \). A new node can start the first \( r(n) \) RWs at a fast rate, or even simultaneously. It is important, however, for the correctness of the reverse sampling, that after this initial phase, the joining node will return to advertising itself only once every \( \Delta \) time units.

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 \( \text{Timeout} \) time units after its departure. In the case of 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 \( \Delta \) 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 order to evaluate the quality of the membership service, we compare it with a service that has access to an ideal independent uniform random sample of network nodes. We remind the reader that we refer to sampling nodes uniformly from all node locations rather than sampling a uniform set of node identifiers.

In our evaluation, we consider several properties of the generated random views. These properties are important to the envisioned applications discussed in the introduction. The properties are best described using a graph-theoretic view [26] 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).

#### 4.2.1 Uniformity

A nice feature of RaWMS, compared to other probabilistic methods like [1, 26], is that the uniformity of the views is guaranteed by construction. The regularization of the graph guarantees that the resulting 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.

#### 4.2.2 Connectivity

Our resulting graph has a very low possibility of sustaining a partition. Since the graph is truly uniform, we can easily calculate the partition probability by following the same partitioning analysis as in lpb-cast [14]. The probability \( \Psi(i, n, s) \) of creating a partition of size \( i \) in a network of \( n \) nodes with a view
size of $s$ is given by:

$$\Psi(i, n, s) = \binom{n}{i} \left( \frac{(i-1)}{s} \right)^i \left( \frac{(n-s)}{n-i} \right)^{n-i}$$

In the lpcast work, they have shown that $\Psi(i, n, s)$ monotonically decreases when $s$ and $n$ increase, and becomes vanishingly small with even moderate values of $n$, such as 50. When the distribution of the membership information in a certain round does not depend on the distribution in the previous round, the authors show that it takes $\approx 10^{12}$ rounds to end up with a partitioned system for $n = 50$ and $s = 3$. Indeed, our RW-based construction possesses such view independence over time. The content of the view at time $t + r(n) \cdot \Delta$ has absolutely no correlation with the view at time $t$ (we have shown in Section 4.1.1 that the view is being completely refreshed every $r(n) \cdot \Delta$ on average) and as we have already shown, both are guaranteed to be uniform.

RaWMS exhibits another very important property regarding partitioning, which is self healing from partitions. In RaWMS, nodes are not restricted to communicate only with nodes that are in their views. Therefore, even if a partition occurs at some time, it will be immediately fixed by itself after a short period of time.

Of course, the above results are conditioned on the connectivity of the underlying physical topology, i.e., the actual ad hoc network itself. Its connectivity is guaranteed by having a sufficiently large transmission range, as we have already pointed out.

### 4.2.3 Load-balancing the view sizes

As we have already shown, view sizes of different nodes can be set independently by the nodes. We have calculated the average amount of time and communication required until the desired view size, $s(n)$, is reached at every node.

Let us now take a closer look at the average 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 $n$ network nodes. Let $X_v$ be the random variable specifying the size of the view of this node at the end of the convergence period. Each of the $n$ nodes advertises itself to $s(n)$ uniformly chosen nodes. Therefore, each such node has a probability of $s(n)/n$ to advertise itself to $v$. Since advertisements of different nodes are independent of each other, then $X_v$ 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 (see Appendix B). 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 $\delta$ 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

As we have already stated, a node may wish to extend its local view to a larger one upon its application’s demand. One possible way to do this is by increasing the desired view size, $s(n)$. This solution is good for the long term, since it does not incur any additional communication complexity, as it 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. Sometimes it may even be impossible, if the node’s memory is limited. Therefore, a method to extend the view on demand is required.

We propose two methods for reactive extension of the view by combining views of other nodes into a view of a requesting node: flooding-based and RW-based. In a flooding-based solution, a node floods the network requesting all nodes to send their views to it and then combines the obtained partial views into its own view up to the requested size. This approach could be quite wasteful if too many nodes send their views back to a requesting node. Therefore the requesting node $v$ may wish to limit the amount of replies it gets by limiting the flooding scope with a $ttl$. Only nodes whose distance is up to $ttl$ hops from $v$ will send their views. The difficulty of implementing this approach is the need to calculate the value of $ttl$ from the desired view size. This calculation requires knowing the network density, which may not be available to a node.

The RW-based method we propose allows to rigorously 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 view while removing duplicates. If the combined view is smaller than the required view, then $u$ 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.

Since in this method we remember the RW path inside the message, we can further optimize the RW by preventing it from revisiting the same nodes more than once. Studying the potential performance gain of this optimization is left for future work.

Let us note that in a mobile network, there is a chance that some link of the reverse path of the RW may not exist by the time it is used for sending the reply back to the originator. To overcome this problem, a unicast routing protocol should be used. Practically, this happens very rarely due to a short time proximity between the RW and the reply.

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., [15, 36]. Once we have such a loose upper bound, we can use the birthday paradox principle [32] to obtain a much tighter bound. This is by keeping track of the average time between publications of the same identifier.

We have shown that according to the reverse sampling technique, every time some RW stops at node $u$, it has the effect of having $u$ pick uniformly at random a node identifier out of all $n$ network 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 ([32]). Therefore, every node can calculate the first time it receives the same advertisement again (denote this number by $m$) and use this number to estimate $n$ according to $n = \frac{m^2}{2}$. This process should be repeated constantly and averaged across a number of measurements.

In order to deal with accumulating errors in this process, the loose upper bound should be re-used periodically and the process of estimating the tight bound according to the birthday paradox be restarted.
do forever
  wait \( \Delta \) time units;
  if push then
    // 0 is the initial hop count
    myDescriptor \leftarrow (myAddress,0,NewFlag);
    sendBuff \leftarrow selectItemsToSend
          (view,myDescriptor,\{\})
  else
    // empty view to trigger response
    sendBuff \leftarrow \{\}
  endif
repeat
  v \leftarrow selectPeer();
  send(sendBuff) to v
  if pull then
    receive(recvBuff) from u
    recvBuff \leftarrow increaseHopCount(recvBuff);
    view \leftarrow selectItemsToKeep(view(recvBuff))
  endif
for \( F \) times // \( F \) is the fanout parameter
enddo

Upon receive(gossip_message,recvBuff) from u do
  recvBuff \leftarrow increaseHopCount(recvBuff);
  if pull then
    // 0 is the initial hop count
    myDescriptor \leftarrow (myAddress,0,NewFlag);
    sendBuff \leftarrow selectItemsToSend
          (view,myDescriptor,recvBuff);
    send(sendBuff) to v
  endif
  view \leftarrow selectItemsToKeep(view,recvBuff)
enddo

Figure 3: A Generic Gossip Framework

5 Gossip-based membership

5.1 The generic gossip framework

As discussed in Section 7, gossiping has been studied in the past as a way to implement partial view membership services. A generic framework for such gossip-based protocols in peer-to-peer networks has been presented in [26] and adapted to sensor networks in [17]. We have combined these two frameworks into a unified framework that is adapted to both static and mobile ad hoc networks.

The view in gossip-based membership algorithms is a set of \( s \) node descriptors, each descriptor consisting of \(<\text{NodeIdentifier},\text{HopCount},\text{NewFlag}>\). In existing gossip-based membership protocols, the size of the view is usually assumed to be the same for all nodes, whether it is a constant or a function of \( n \) (the number of nodes in the network).

We assume that each node executes the same protocol whose skeleton is shown in Figure 3. As in RaWMS, the protocol consists of two threads: an active thread initiating communication with other nodes, and a passive thread waiting for incoming messages. The skeleton code is parameterized with three boolean parameters, namely push, pull and NewFlag, the desired fanout \( F \), and three functions, namely selectPeer, selectItemsToSend and selectItemsToKeep.

Periodically, each node gossips with one of its neighbors to exchange the items in their views. A view is organized as a list of descriptors, ordered according to increasing hop counts. Entries with the same hop count are ordered in a random manner. We can thus meaningfully refer to the first or last \( k \) elements of a particular view. Notice that in the protocol’s code, a call to increaseHopCount(view) increments the hop count of every element in a view.

The above skeleton enables us to evaluate within the same framework the important policies involved in gossip-based protocols along four dimensions: (i) peer selection, (ii) view propagation, (iii) keep selection, and (iv) send selection. By combining the possible values of each of these attributes, one can obtain many variations of gossip protocols, some of which have already been explored.
Peer selection: Periodically, each node \( v \) selects a peer in order to exchange membership information with it. This selection is implemented by the function \texttt{selectPeer()} that returns the address of a live node either in \( v \)'s current view or in \( v \)'s neighbors list. Below we list a few representative policies that have been mentioned in the literature.

<table>
<thead>
<tr>
<th>Policy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{rand}</td>
<td>Uniform randomly select an available node from the view</td>
</tr>
<tr>
<td>\texttt{head}</td>
<td>Select the first node from the view (the one with the lowest hop count)</td>
</tr>
<tr>
<td>\texttt{tail}</td>
<td>Select the last node from the view (the one with the highest hop count)</td>
</tr>
<tr>
<td>\texttt{neighbor}</td>
<td>Randomly select an available node from the neighbors list</td>
</tr>
<tr>
<td>\texttt{broadcast}</td>
<td>Select all nodes in the neighbors list to send a broadcast message to them</td>
</tr>
</tbody>
</table>

View propagation: Once a peer has been chosen, there are several alternatives to exchanging information with that peer, as listed below.

<table>
<thead>
<tr>
<th>Policy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{push}</td>
<td>The node sends its view to the selected peer</td>
</tr>
<tr>
<td>\texttt{pull}</td>
<td>The node requests the view from the selected peer</td>
</tr>
<tr>
<td>\texttt{pushpull}</td>
<td>Both the node and the selected peer exchange their respective views</td>
</tr>
</tbody>
</table>

Send selection: Once the peer and the way to contact it have been chosen, the sender must decide what information to send. The options that have been discussed in the literature are listed below:

<table>
<thead>
<tr>
<th>Policy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{rand}</td>
<td>Randomly select up to ( X ) descriptors from the view</td>
</tr>
<tr>
<td>\texttt{head}</td>
<td>Select the first ( X ) descriptors from the view (the ones with the lowest hop count)</td>
</tr>
<tr>
<td>\texttt{tail}</td>
<td>Select the last ( X ) descriptors from the view (the ones with the highest hop count)</td>
</tr>
<tr>
<td>\texttt{new}</td>
<td>Pick all descriptors that have been received for the first time</td>
</tr>
<tr>
<td>\texttt{full}</td>
<td>Send all ( s ) descriptors of the view</td>
</tr>
</tbody>
</table>

Keep selection: Once the membership information has been exchanged between peers, the received descriptors should be integrated into the node’s view. The integration procedure must adhere to the target size limit of \( s \) descriptors by choosing only the subset of all available descriptors. In the protocol above, this is done by the \texttt{merge(view,recv_buff)} procedure, which merges the received view with the current one. In case a descriptor appears in both views, the merged view takes the version with the most up to date timestamp. The function \texttt{selectItemsToKeep(view,recv_buff)} selects a subset of at most \( s \) elements from merged views according to one of the policies listed below:

<table>
<thead>
<tr>
<th>Policy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{rand}</td>
<td>Merge and randomly select ( s ) elements without replacement from the merged view</td>
</tr>
<tr>
<td>\texttt{head}</td>
<td>Merge and select the first ( s ) elements from the merged view</td>
</tr>
<tr>
<td>\texttt{tail}</td>
<td>Merge and select the last ( s ) elements from the merged view</td>
</tr>
<tr>
<td>\texttt{shuffle}</td>
<td>Merge and remove the elements that were sent in this data exchange to the other node until only ( s ) elements remain in the view</td>
</tr>
</tbody>
</table>

It is possible to obtain a large selection of gossip protocols by simply plugging any of the above policies in the skeleton protocol of Figure 3. Each combination is expressed by means of a 4-tuple (\textit{peer selection, view propagation, send selection, keep selection}). In particular, various combinations of the above policies were investigated in [26]. One of the conclusion of [26] is that no gossiping algorithm succeeds in constructing views that form a truly random knowledge graph. Typically, the resulting knowledge graph induced by the view’s edges has great resemblance to a small-world graph.
Speeding up the joining process with NewFlag: We can use the same technique here as in RaWMS in order to speedup the joining process of a new node. That is, a new node increases the rate of gossiping until it has managed, with high probability, to distribute its identifier to enough random nodes in the network. Note that in gossiping algorithms, other nodes must also gossip the identifiers of newly joined nodes more frequently than the standard gossip frequency. For that purpose, during a fixed period of NewTimeout time units, a new node \( v \) turns on the NewFlag flag of its descriptor each time \( v \) gossips its descriptor. When a node receives a gossip descriptor with the NewFlag flag turned on, it increases its own gossip rate for a duration of NewTimeout time units. As a result, when a new node joins the network, the gossip rate at all “infected” nodes is increased and the new identifier is gossiped faster. After NewTimeout time units elapse, the gossip rate returns to \( 1/\Delta \) in order to reduce the communication overhead.

5.2 Specific gossip methods

5.2.1 lpbcast

lpbcast [14] corresponds to (rand, push, full, rand). In each round every node \( v \) sends its view to \( F \) (the fanout parameter) nodes, chosen randomly from \( v \)’s view. As stated in [14], the number of rounds is logarithmic in the network size. In order to establish a communication path between two nodes in an ad hoc network, some routing algorithm must be employed. Since destinations are chosen randomly among the network nodes, the number of network level messages required to send a single gossip message is equal to the average path length of the network.\(^4\) The average path length in an ad hoc network is in the order of the diameter of the network divided by the transmission range. In our case, this amounts to \( O(\sqrt{n \log n}) \). Also, in each gossip message, the entire view is sent. Therefore, the total communication overhead of lpbcast for a view of size \( s \) is \( n \cdot \sqrt{n \log n} \cdot F \cdot s \).

The main drawback of lpbcast, which makes it unsuitable for ad hoc networks, is the extensive usage of unicast routing. Since each node sends messages to random network nodes, lpbcast uses \( F \cdot \log(n) \) routes in the initial convergence stage and keeps utilizing more routes afterwards. Notice that the establishment of a unicast route if often obtained through flooding, which is costly in an ad hoc network. Since the potential number of source destination pairs is quadratic, lpbcast’s traffic pattern virtually establishes all-to-all routing paths over time, which are created merely for lpbcast’s usage and are not necessarily used by the application. Those routes break over time due to nodes mobility, which adds the cost of repairing them.

Another drawback of lpbcast is that according to [26], lpbcast fails to provide uniform views. Moreover, the knowledge graph induced by views can sometimes be a star or a small-world graph, rather than a random graph. In addition, the views at the same node but in different rounds are not truly independent, since nodes gossip at round \( t + 1 \) only with nodes they had in their view in round \( t \). As a result, it was shown in [26] that lpbcast has a non-negligible chance of partitioning. Moreover, when partitions do occur in lpbcast, or any other similar gossip algorithm, they cannot self-heal.

5.2.2 Newscast

Newscast [25] corresponds to (rand, pushpull, full, head). Newscast possesses the same properties and drawbacks of lpbcast. Namely, it uses routing, it is not fully uniform, it suffers from partitioning, and there is no self-healing of partitions.

\(^4\) More precisely, \( v \) chooses \( F \) random nodes from its view. However, the view gradually converges to a random sample.
5.2.3 Shuffling

Shuffling [17], corresponds to (neighbor, pushpull, rand, shuffle). Shuffling was first introduced in the context of sensor networks and originally used for information dissemination. However, shuffling can also be used for construction of random views, by disseminating nodes identifiers. In shuffling, each node communicates only with its direct neighbors. The main idea of shuffling is that unlike other gossiping algorithms, the shuffle protocol avoids any 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, not only do they preserve the data that is collectively stored in the network, but also “move” this data around in a seemingly random fashion.

**Shuffling’s performance** We analyze the performance of shuffling by adapting some RW techniques to it. In the following analysis, let us assume that each node already possess a random view. We are interested in determining the number of rounds and the number of messages required for a new node joining the system to incorporate its identifier uniformly into the views of other nodes in the system.

Every round each node randomly picks $X$ identifiers out of its view and shuffles them with its randomly chosen neighbor. Since the views are random, when two nodes shuffle, they pass to each other almost completely different set of identifiers. Therefore, almost all ids that node $v$ passes to node $u$ will migrate to $u$ and will be removed from $v$’s view. In this process, ids already present in the network are not discarded, and almost never duplicated. This view exchange process has some resemblance with RWs; each identifier traverses the network from one node to its randomly chosen neighbor. However, there are a number of differences: 1) in shuffling, the “walks” of different identifiers are not independent since an exchange is performed on a batch of $X$ identifiers, 2) an identifier may not be passed to a neighbor node every round, since only $X$ identifiers out of the entire view are exchanged every round.

The first difference can be controlled by the size of the exchanged batch, $X$. Large values of $X$ indeed increase the dependence between disseminations of different identifiers. However, for small values of $X$, the effect of dependence is not significant, especially since in every round each node picks a different set of $X$ ids from its view for an exchange. Indeed, shuffling is usually run with small, constant $X$.

As for the second difference, we can measure the pause time, i.e., the average amount of time that each identifier spends in the view before being shuffled. If the whole view is shuffled, the pause time is zero. If only $X$ identifiers out of the entire $s$ (the view size) are shuffled every time, the pause time is a geometric random variable with a mean of $\frac{s}{X}$. Therefore, the number of rounds until an arbitrary identifier reaches a random place (assuming no duplications and discarding and fanout 1) is $\frac{s}{X} \cdot T_{mix}$, where $T_{mix}$ is the mixing time of a RW of the underlying graph. Since we are interested in a situation when $s$ random nodes have the identifier of the new node in their view, a new node must publish itself $s$ times, once in each successive round. This yields a total of $s + \frac{s}{X} \cdot T_{mix}$ rounds and messages until convergence.

5.2.4 Flooding

An efficient implementation of flooding requires memory which is linear in the number of nodes in the system. That is, in order to prevent nodes from delivering (and retransmitting) the same message more than once, each node must remember the identifiers of messages it has already seen. One efficient way to do this is for every node to remember the identifiers of the last few broadcast messages initiated by every other node. This requires an amount of memory that is at least linear in the overall number of nodes.

Since the implementation of flooding itself requires memory space that is linear with the overall number of nodes, there is no point in limiting the view to include fewer than $n$ identifiers. Therefore, in
a flooding-based dissemination algorithm, each node periodically floods the network with its identifier and all nodes store the identifiers of every node in the network. Each identifier is attached with a timeout. An identifier \( u \) is removed from a view of a given node \( v \) if \( v \) does not receive any message containing \( u \) for the duration of the timeout.

**Flooding in rounds:** It is possible to improve the message complexity of the basic flooding algorithm by using synchronous rounds. That is, in each round every node informs all of its neighbors about all the identifiers it knows about. This way, the size of the messages is increased until they contain all nodes identifiers. In this case, the number of rounds required for each identifier to be known everywhere equals the diameter \( D \) of the network. The total number of messages is also fairly small, i.e., \( D \cdot n \). However the communication complexity (overall number of bits sent) is the same as with asynchronous flooding (see 5.4 for a full comparison).

### 5.3 Probabilistic starvation

One of the main usages of partial membership services is 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. 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 every message, there is a small number of nodes that 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, then those nodes will be constantly denied messages and therefore suffer from probabilistic starvation. On the other hand, views constructed by RaWMS are proven to be uniform and therefore any probabilistic multicast algorithm build a top of it will not suffer from such a phenomenon.

### 5.4 Comparison

An asymptotic comparison of all the methods we mentioned above appears in Table 1. It compares the time and the communication complexity of the convergence period. The maintenance cost for each method is the communication cost during the convergence period divided by the convergence time.

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

This section presents our simulation results. We start by investigating RaWMS’ properties, and then compare the performance of RaWMS with lpbcast.

**Model** The simulations were performed on the JiST/SWANS simulator [39] from Cornell university. Nodes use an IEEE 802.11 radio and MAC model with 1Mbps transmission rate. 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 a walking speed, and an average pause time of 30s. All simulations were run on networks of 10, 50, 100, 200, 400 and 800 nodes.
The nodes were placed at uniformly random locations in a square universe.\textsuperscript{5} 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 regarding the critical transmission range. For a square area $a^2$ the radius of the critical transmission range is $r = a \sqrt{C \ln n / n}$, $r \in [0, a]$. We have set the average number of nodes in the transmission range of any node to $d_{avg} = 3 \ln n$. This means that the simulation area $a^2$ for $n$ nodes was picked such that $d_{avg} = \pi r^2 n / a^2 = \pi a^2 \ln n / a^2 = \pi C \ln n = 3 \ln n$, resulting in $a^2 = 300 \sqrt{\ln n}$ and $C \approx 1$. According to Proposition 3.5, for such a radius $d_{max} \approx 2d_{avg}$.

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 4.1.1. These advertisements were spread over the whole simulation period. We have used the default Java pseudo random number generator, initialized with the current system time in milliseconds as a seed.

**Uniformness of RaWMS** We have performed a number of tests to compare the views constructed by RaWMS with the ideal uniformly sampled views. Those tests were picked to reflect the most important structural properties of the system: distribution of the path lengths from every node to all nodes in its view, view size distribution, dependence between views of neighboring nodes, and connection between a node’s degree and its view size.

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 $\chi^2$ statistical test to compare the distribution of nodes in the view of

\textsuperscript{5}We run our simulation on a flat topology rather than a torus. This places our scheme in a slight disadvantage, since the communication graph tends to be less uniform in a flat topology.

<table>
<thead>
<tr>
<th></th>
<th>#rounds</th>
<th>time of a round</th>
<th>total time</th>
<th>msgs per round</th>
<th>total msgs sent</th>
<th>msg size</th>
<th>com. overhead</th>
<th>mem. overhead</th>
<th>other overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>RaWMS</td>
<td>$r(n)$</td>
<td>$T_{actual, mix} = \frac{n + r(n)}{n}$</td>
<td>$n^2$</td>
<td>$n^2 \cdot r(n)$</td>
<td>1</td>
<td>$n^2 \cdot r(n)$</td>
<td>view size $s$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1pbcast</td>
<td>$\log n$</td>
<td>$\sqrt{n \log n}$</td>
<td>$n \sqrt{n \log n}$</td>
<td>$\frac{n \sqrt{n \log n}}{F}$</td>
<td>$\frac{n \sqrt{n \log n}}{F \cdot \log n}$</td>
<td>view size $s$</td>
<td>$\log n \cdot F \cdot s$</td>
<td>linear memory for routing</td>
<td></td>
</tr>
<tr>
<td>Shuffle</td>
<td>$\frac{T_{mix}}{n}$</td>
<td>$\sqrt{n \log n}$</td>
<td>$\frac{n^2 \sqrt{n \log n}}{X}$</td>
<td>$X$</td>
<td>$n^2 s$</td>
<td>view size $s$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flooding</td>
<td>1</td>
<td>$\sqrt{n \log n}$</td>
<td>$\sqrt{n \log n}$</td>
<td>$n^2$</td>
<td>$n$</td>
<td>$n$ for flood impl.</td>
<td>Memory for flooding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flooding in rounds</td>
<td>Max. path length</td>
<td>$\sqrt{n \log n}$</td>
<td>$n$</td>
<td>$n$</td>
<td>$n$</td>
<td>$n$ for flood impl.</td>
<td>Memory for flooding</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Comparing gossip-based membership with RaWMS
every node at the end of the convergence period with the desired uniform distribution. Namely, the whole simulation area was partitioned into a number of bins. For every node \( v \) we have calculated the following score:

\[
\text{Score}_v = \frac{\sum_{j=1}^{\text{bins}} (\text{Actual}_{v,j} - \text{Expected}_{v,j})^2}{\text{Expected}_{v,j}},
\]

\( \text{Actual}_{v,j} \) being the actual number of nodes from bin \( j \) found in the view of node \( v \) and \( \text{Expected}_{v,j} \) is the number of nodes from bin \( j \) that are expected to be found in the view of node \( v \). The network score corresponds to the average of all \( \text{Score}_v \)'s. We used this score as the statistical test for the difference between the distributions obtained by simulations and the assumed uniform distribution. Clearly, the actual division into bins has a major influence on the test and should be picked such as to represent important features of the tested distribution. For our evaluation, we have divided all the nodes into bins according to their distance from the tested node. In a perfect uniform sample, the view of each node should be constructed with nodes from all network distances and the number of nodes at each distance in the view should be proportional to the actual number of nodes at this distance in the network. Therefore, we have tested the distribution of the paths lengths from every node to all nodes in its view.

The results of the path length distribution test for static networks are depicted in Figure 4(a). The simulations were run with 5 different lengths of the random walk, corresponding to 5 different candidates for the mixing time, \( T_{\text{mix}} \). Clearly, the longer the walk is, the closer is the distribution reached by the RW to the uniform stationary distribution. Intuitively, this means that a long walk has a better chance to reach a uniformly random node. We can see that for lengths of \( n \) and \( n/2 \) the test score 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. For shorter walks we can see a dramatic degradation in the test’s score. Those walks are shorter than \( T_{\text{mix}} \) and do not have enough steps to reach the uniform stationary distribution. For shorter walks we can see a dramatic degradation in the test’s score. Those walks are shorter than \( T_{\text{mix}} \) and do not have enough steps to reach the uniform stationary distribution. The larger the network is, the worse 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 that are geographically closer to it in its view and with fewer nodes that are geographically far from it. This confirms our formal analysis.

Figure 4(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 random walk. Still, as can be seen from the graph, very short walks of length \( n/16 \) obtain worse results even with mobility.
View size distribution  We have tested the distribution of the view sizes (also denoted as the degrees of the nodes in the knowledge graph). Recall that according to Section 4.2.3, 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 5 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 \( \frac{\sum_{i=1}^{n}(E(d_i) - d_i)^2}{n} \). For all network sizes and for all walk lengths, the average size of the view is almost equal (typically up to 90%) to the ideal expected mean size. The variance in the view sizes is also very close to the expected one, presenting another evidence to the fact that the view size is a binomial distributed random variable. The only exception to the above is in 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 therefore the resulting view includes only the node itself. The variance is zero in such a case. For small networks the mean view size is a bit larger than expected. This is due to the fact that \( \frac{2n}{m^2} \) is not a tight bound of \( r(n) \) for \( s \) of the order of \( n \) (see Lemma 4.1). In these cases, nodes simply start too many RWs.

In mobile networks, the mean values of view sizes are very close to the expected ones, since ac-
according to our implementation, if a node \( v \) does not succeed to forward a RW message to the neighbor chosen in a given step, then \( v \) makes a new attempt to send this message to another neighbor within the same step. This way, a RW never gets lost.

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. This is because if \( v \) passes next to a node \( u \) that has the RW message, \( v \) disappears from the transmission range of \( u \) before 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, than fast moving nodes. This phenomena becomes even worse in long RWs, since the longer the RW becomes, the greater is the chance that it will be “stuck” at a static or slow moving node. The situation could be improved by a more aggressive and frequent neighborhood discovery protocol. These results also suggest that the length of RWs should be adjusted in reverse proportion to the observed mobility in the network.

**Intersection between views of neighboring nodes** In this test we have checked the amount of correlation between the views of neighboring nodes. For ideal uniformly chosen views there should not be any special correlation between the views of neighbor nodes. We have measured the average size of the intersection between the views of all pairs of neighboring nodes. Since the average view size is \( \sqrt{n} \), the expected intersection is \( \sqrt{n} \sqrt{\frac{\sqrt{n}}{n}} = 1 \), for all network sizes. It can be seen from Figure 6(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 the expected one. On the other hand, walks shorter than the mixing time do not have enough steps to get far away from the originating node. Short walks tend to stop at a proximity of an originating node instead of at a random node, and as a result the neighbors of an originating node have a greater chance to have the initiating node in their views.

In mobile networks intersection between views of neighboring nodes is greatly reduced. Short RWs can now get a chance to escape the proximity of its originating node, due to mobile nodes carrying the RW message. In long walks the intersection is even smaller than expected. This can be explained by the same argument as with view size variance in mobile networks. Long RWs tend to stop at a static and slow moving nodes. Those nodes are surrounded by fast moving mobile nodes and are usually not neighbors of each other. Therefore, the intersection between views of static nodes with their fast moving neighbors is very small.
Correlation between node degree and view size

Next, we test whether there is a correlation between node degrees and view sizes. Figure 7 shows the distribution of view sizes accumulated into bins according to node degrees. The nodes were sorted by degree (number of their neighbors in the ad hoc network) and then separated into 10 deciles, each containing 10% of the nodes. For each decile, the bar chart shows the ratio between the average view size of this decile and the average view size of the whole network. The results in Figures 7 and 8 were generated for walk length $n/2$. The same results were observed for other walk lengths both in static and mobile networks.

The stationary distribution of a RW without self loops is degree-dependent, resulting in higher degree nodes having more RWs stopping at them. Indeed, it can be seen from Figure 7(b) that there is a significant bias towards high degree nodes - much more RWs stop at these nodes than at lower degree nodes, resulting in unbalanced view sizes. On the other hand, our Maximum Degree RW balances the node degree with self loops, generating a regular graph on which the RW has a uniform stationary distribution. As a result, we can see from Figure 7(a) that there is no bias towards high degree nodes and that the views have almost the same average size for all deciles (recall that self-loops do not cost anything, since no messages are sent when they are chosen).

The results for mobile networks are depicted in Figure 8. The results for RWs with self loops are essentially the same as in static networks: there is no bias towards high degree nodes. On the other hand, RWs without self loops in mobile network have no bias as well. This is since in a mobile network the neighborhood of a node changes frequently. During the run every node has different degrees, and all nodes have approximately the same degree averaged over the whole simulation time. Therefore, we can note here that again mobility assists in introducing uniformity into the RW.

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 throughout all simulations. The view size limit was set to $\sqrt{n}$, to establish the same conditions as with RaWMS. As can be seen in Figure 9(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 9(b), the uniformity of the views is
Figure 8: Correlation between node’s degree and its view size. Mobile network, walk length $n/2$

dramatically improved when nodes are mobile.

RaWMS versus Lpbcast - communication overhead Figure 10 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 5.4. In RaWMS, each node starts roughly $\sqrt{n} + 2$ RWs, each walk sending $T_{\text{actual,mix}} \leq T_{\text{mix}} \frac{d_{\text{max}}}{D}$ messages. $T_{\text{mix}}$ was approximated by $n/2$ according to the results in Figure 4 and $D$ was set large enough to bound $d_{\text{max}}$. The measured $T_{\text{actual,mix}}$ was about $T_{\text{mix}}/2$. Therefore, each node sends a total number of $\frac{n\sqrt{n}}{4}$ messages.
In lpbc, every node starts $4 \log n$ rounds with fanout 3 and each message traverses the network over an average path of $\sqrt{\frac{n}{\log n}}$. Each node sends $12\sqrt{n \log n}$ messages in total.

As is evident from Figure 10(a), lpbc generates fewer application messages than RaWMS, as expected by our previous analysis. Yet, recall that in lpbc each message contains the whole view, while in RaWMS messages carry only a single node identifier. Therefore, the total bit communication overhead of lpbc is $12n \sqrt{\log n}$. In addition, lpbc has a significant message overhead due to routing. When adding the cost of routing, RaWMS becomes considerably more efficient than lpbc.

Figure 10(b) illustrates the communication costs of RaWMS with a walk length of $n/8$ and lpbc with $2 \log n$ rounds. Again, those parameters guarantee a uniform distribution of views at the lowest possible cost. Here, the cost of RaWMS is significantly lower than lpbc. This is due to a decreased walk length, yet without compromising the uniformness of the views. In this scenario, each node sends about $n$ messages. On the other hand, lpbc 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 lpbc in terms of network messages.

7 Related work

Random walks Comprehensive surveys of random walk techniques and their analysis appear in [28] and [21]. The idea of using a “maximum-degree” random walk in order to reach a uniform limit distribution on the state space has been used before in a number of contexts [4, 7].

Lv et al. [30] propose to use simulated random walks for searching in unstructured peer-to-peer networks. They report that such a search is preferable to the standard practice of searching by flooding. The suitability of random walks is attributed to their adaptiveness to termination conditions. RWs allow fine-grain control of the search space, whereas with flooding, small increases of the TTL may increase the search space exponentially. The work in [30] reported attractive empirical results, but does not provide any analytical evaluation of the RW properties.

In [18], Gkantsidis et al. explore the performance of random walks for searching and uniform sampling in peer-to-peer networks. An important result of this paper is that it is possible to simulate the selection of a uniform sample of elements from the entire network by performing a random walk 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 [34], which are not expanders. Awan et al. [3] use random walks to uniformly sample nodes from a large network in a distributed fashion. Their algorithm is not tailored to any specific network model, and in particular they do not address the random geometric graphs modeling ad hoc networks.

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

Dolev, Schiller and Welch [12] propose a randomized self-stabilizing group membership service for ad hoc networks. The group membership list is carried through the network by a random walk of a single mobile agent. The agent traverses the network and accumulates the membership data from every node. The length of the RW is set to guarantee, w.h.p, that an agent will visit every node. There are two major differences between [12] and our work. First, [12] only constructs a full membership while RaWMS can be used to construct partial membership views. Second, they apply a single RW that covers the whole network and runs for a period of time that is equal to the cover time of the induced communication graph. We use multiple RWs simultaneously each running for a period that is equal to the mixing time. This results in a major difference in the time and communication complexity. Specifically, the time and communication complexities of the algorithm in [12] are $O(n^3)$, while in RaWMS each RW runs for only $O(n)$ steps. The communication complexity of RaWMS depends on the desired view size. For example, to construct a view of size $O(\sqrt{n})$ 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. Thus, the total communication complexity in this case is $O(n^2\sqrt{n})$.

In [36] random walks 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. In particular, a general gossip-based peer sampling service was introduced in [26]. Examples of gossip-based lightweight membership services complying to this general framework are reported in [1, 14, 16]. We discuss gossip-based algorithms for membership construction in more detail in Section 5.

SCAMP [16] 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. Note that [16] 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 actual 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.

A gossip-based membership service for sensor networks and for mobile ad hoc networks is described in [17]. We discuss it in detail in Section 5.

RDG [29] is an adaptation of [14] to ad hoc networks. It reduces the cost of routing compared to [14] 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 removal of nodes that left the network from the views. As we show in section 5.2.4, 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 func-
tions in ad hoc networks [22]. 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 in order 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. We 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 random walks present an attractive paradigm for implementing partial view based membership services in ad hoc networks. This is due to the fact that random walks do not require multi-hop routing and avoid flooding altogether. Moreover, when the network is mobile, random walks reach their target uniformity even faster than on static networks. In these cases, the mobility helps to disseminate messages to random places in the network.

A surprising result of our study is that in mobile networks short random walks obtain better results than long ones. The conclusion is that nodes should consider the degree of mobility in the network when determining the length of the RWs that they start. The faster nodes move, the shorter the random walks need to be. Recognizing the level of mobility can be implemented, e.g., by analyzing the frequency of neighbor changes in nodes proximity.

Our work leaves several open problems. These include, e.g., a more detailed investigation of the relation between random walks and gossip. In particular, combining random walks with occasional gossiping to far away nodes.

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.

References


A Random geometric graphs

We provide below a formal definition of the Random Geometric Graph $G^2(n, r)$. To this end, we need to introduce some basic facts about the geometry on the surface of a torus.

Geometry on the surface of a torus A 2-dimensional unit torus is the set of points in the unit square $[0, 1] \times [0, 1]$ endowed with a special measure of distance, called the geodesic distance. It is convenient to visualize a torus as taking the flat unit square, and then “gluing” together the two vertical edges and the two horizontal edges. What we get is a surface of 3-dimensional object, whose shape resembles a holed donut. The important point to notice is that because of the gluing points near the left vertical edge are close to points near the right vertical edge, and similarly points near the top horizontal edge are close to points near the bottom horizontal edge.

Every point $u$ on the surface of a torus has two coordinates: $u_x \in [0, 1]$ and $u_y \in [0, 1]$. Every two points $u, v$ on a torus have two straight lines connecting them (going in opposite directions). The geodesic distance between $u$ and $v$ is the length of the shorter of these two lines. To formally define the geodesic distance we introduce the following notion of “circle distance” between real numbers.

Every real number $a \in \mathbb{R}$ can be embedded into a circle whose circumference is 1 as follows:

$$a \mod 1 = \begin{cases} a - \lfloor a \rfloor & \text{if } a \geq 0 \\ 1 - (\lfloor a \rfloor \mod 1) & \text{if } a < 0 \end{cases}$$

For two numbers $a, b \in [0, 1]$, we define the circle distance between $a$ and $b$ as:

$$\text{cd}(a,b) = \min\{(a - b) \mod 1, (b - a) \mod 1\}.$$ 

For example, if $a = 7/8$ and $b = 1/8$, then $(a - b) \mod 1 = 6/8$, while $(b - a) \mod 1 = 2/8$, and hence $\text{cd}(a,b) = 2/8$. Note that the circle distance between any two numbers is always at most $1/2$.

Given two points $u, v$ on the surface of a unit torus, we define the geodesic distance between $u$ and $v$ as follows:

$$\text{gd}(u, v) = \sqrt{\text{cd}(u_x, v_x)^2 + \text{cd}(u_y, v_y)^2}.$$
Random geometric graphs  Let \( n \) be a positive integer and let \( r \geq 0 \) be a real number. The random geometric graph \( G^2(n, r) \) is generated as follows. The graph has \( n \) vertices associated with \( n \) uniformly chosen points on the surface of a 2-dimensional torus. Two vertices \( u, v \) are connected by an edge if and only if \( gd(u, v) \leq r \).

\[ \tag{72} \]

B  Chernoff bounds

We state below the exact version of the Chernoff bounds [9] we use in this paper. A proof can be found, e.g., in [32].

Theorem B.1 (Chernoff bounds). Let \( X_1, X_2, \ldots, X_n \) be independent and identically distributed Bernoulli random variables with probability of success \( p \). (That is, for all \( 1 \leq i \leq n \), \( X_i \) is a 0-1 random variable and \( \Pr[X_i = 1] = p \).) Let \( X = \sum_{i=1}^{n} X_i \) and let \( \mu = E(X) = np \). Then, for any \( 0 < \delta < 1 \),

\[ \Pr(X > (1 + \delta) \mu) < e^{-\mu \delta^2 / 3} \quad \text{[Upper tail]}, \]

\[ \text{and} \]

\[ \Pr(X < (1 - \delta) \mu) < e^{-\mu \delta^2 / 2} \quad \text{[Lower tail]} \]

Note that by combining the upper and lower tail bounds we obtain:

\[ \Pr(|X - \mu| > \delta \mu) < 2e^{-\mu \delta^2 / 3}. \]

C  Mixing time bound for the MD random walk

In this section we prove the upper bound on the actual mixing time of the Maximum Degree random walk on a random geometric graph \( G^2(n, r) \) (Theorem 3.6). Our proof is based on Sinclair’s bound [38] on the spectral gap of a random walk.

C.1  Sinclair’s bound

In this section we overview Sinclair’s bound [38] on the spectral gap of a Markov chain. Sinclair’s result holds for general reversible Markov chains. Yet, in order to avoid cumbersome notation, we restrict to random walks on regular graphs, which is the case of interest to us.

Let \( G = (V, E) \) be a connected non-bipartite \( D \)-regular graph on \( n \) nodes. \( G \) possibly has weighted self loops but does not have parallel edges. The probability transition matrix \( P \) corresponding to a random walk on \( G \) is an \( n \times n \) stochastic matrix defined as follows. For every \( u \neq v \), \( P_{uv} = \frac{1}{D} \) if \( u \) and \( v \) are connected by an edge and \( P_{uu} = 0 \), if they are not. The diagonal entries are \( P_{uu} = 1 - \sum_{v' \neq u} P_{uv'} \).

Since \( P \) is a symmetric matrix, the stationary distribution of this random walk is the uniform distribution. The principal eigenvalue of \( P \) is 1. Let \( \lambda_{\max} \) denote its second largest eigenvalue in absolute value. Sinclair showed a bound on the spectral gap \( 1 - \lambda_{\max} \) using the notion of canonical paths.

A family of canonical paths is a collection of paths \( \gamma = \{ \gamma_{uv} \}_{u \neq v \in V} \), one for each pair of distinct node \( u, v \) in \( G \). Sinclair defines two parameters of such a family: the maximum path length and the maximum edge load. The maximum path length of a family of canonical paths \( \gamma \) is defined as:

\[ \ell(\gamma) = \max_{\gamma_{uv} \in \gamma} |\gamma_{uv}|. \]

For an edge \( e \in E \), we denote by \( \rho(\gamma, e) \) the number of paths in \( \gamma \) that pass through \( e \):

\[ \rho(\gamma, e) = |\{ \gamma_{uv} \in \gamma \mid \gamma_{uv} \ni e \}|. \]

32
The maximum edge load of $\gamma$, denoted $\rho(\gamma)$, is defined as:

$$\rho(\gamma) = \max_{e \in E} \rho(\gamma, e).$$

Sinclair’s bound is then the following:

**Theorem C.1 (Sinclair).** For any $D$-regular graph $G$ and for any family of canonical paths $\gamma$ on $G$,

$$1 - \lambda_{\max} \geq \frac{n}{D \cdot \ell(\gamma) \cdot \rho(\gamma)}.$$

To derive bounds on the mixing time of the Maximum Degree random walk on $G^2(n, r)$, we need to resort to a stronger version of Sinclair’s bound. (Our extension of Sinclair’s bound is identical to the extension done by Boyd et al. [7] to the bound of Diaconis and Stroock [11].) Let $\Gamma$ be the collection of all possible families of canonical paths. Let $p$ be a probability distribution over $\Gamma$. Let $\text{SUPP}(p)$ be the set of canonical path families that have non-zero probability under $p$. The maximum path length of $p$ is defined as:

$$\ell(p) = \max_{\gamma \in \text{SUPP}(p)} \ell(\gamma).$$

The maximum expected load of $p$ is defined as:

$$\rho(p) = \max_{\gamma \in \Gamma} \sum_{e \in E} p(\gamma) \cdot \rho(\gamma, e).$$

We prove the following:

**Theorem C.2.** For any $D$-regular graph $G$ and for any distribution $p$ over $\Gamma$,

$$1 - \lambda_{\max} \geq \frac{n}{D \cdot \ell(p) \cdot \rho(p)}.$$

**Proof.** We use the variational characterization of the second eigenvalue (cf. [24]):

$$1 - \lambda_{\max} = \frac{n}{D} \cdot \inf_{\psi} \frac{\sum_{e \in E} \psi(e^+) - \psi(e^-))^2}{\sum_{u, v \in V} (\psi(u) - \psi(v))^2},$$

(2)

where the infimum is over all non-constant functions $\psi : V \rightarrow \mathbb{R}$, and $e^+$ and $e^-$ denote the two vertices comprising an edge $e$. Consider any term $(u, v)$ in the denominator. Using any $\gamma \in \Gamma$, we can rewrite this term as the following telescopic sum:

$$(\psi(u) - \psi(v))^2 = \left( \sum_{\gamma \in \Gamma_{uv}} (\psi(e^+) - \psi(e^-)) \right)^2. $$

Since this holds for all $\gamma$, then we can also write:

$$(\psi(u) - \psi(v))^2 = \left( \sum_{\gamma \in \Gamma} p(\gamma) \cdot \sum_{e \in \Gamma_{uv}} (\psi(e^+) - \psi(e^-)) \right)^2. $$

Applying the Cauchy-Schwarz inequality, we have:

$$(\psi(u) - \psi(v))^2 \leq \left( \sum_{\gamma \in \Gamma} p(\gamma) \right) \cdot \left( \sum_{\gamma \in \Gamma} \sum_{e \in \Gamma_{uv}} p(\gamma) \cdot (\psi(e^+) - \psi(e^-))^2 \right).$$
We first bound the left factor:
\[ \sum_{\gamma \in \Gamma} \sum_{e \in \Gamma_{uv}} p(\gamma) = \sum_{\gamma \in \text{supp}(p)} p(\gamma) \cdot |\gamma_{uv}| \leq \ell(p). \]

Substituting the above back in the denominator of the expression appearing in Equation 2, we have:
\[ \sum_{u,v \in V} (\psi(u) - \psi(v))^2 \leq \ell(p) \cdot \sum_{u,v \in V} \sum_{\gamma \in \Gamma} \sum_{e \in \Gamma_{uv}} p(\gamma) \cdot (\psi(e^+) - \psi(e^-))^2 \]
\[ = \ell(p) \cdot \sum_{e \in E} (\psi(e^+) - \psi(e^-))^2 \cdot \sum_{\gamma \in \Gamma} \sum_{\gamma_{uv} \in e} 1 \]
\[ = \ell(p) \cdot \rho(p) \cdot \sum_{e \in E} (\psi(e^+) - \psi(e^-))^2. \]

Substitution in Equation 2 completes the proof.

C.2 Mixing time bound for \( G^2(n, r) \)

In this section we show that the Maximum Degree random walk on the random geometric graph \( G^2(n, r) \) has a mixing time of about \( n \) with high probability. The proof basically repeats the argument made by Boyd et al. [7]. We need to repeat the analysis, in order to figure out the best constants. Moreover, we provide details that are missing in the current version of [7].

**Theorem 3.6 (restated)** 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 \( T_{\text{mix}}(\epsilon) \) be the mixing time of the MD random walk on this graph, when applied with the value \( D \). Let \( T_{\text{actual mix}}(\epsilon) \) 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),
\[ T_{\text{mix}}(\epsilon) \leq \frac{30}{(1 - \frac{7}{\sqrt{C}})^2} \cdot \frac{D}{n} \cdot \frac{1}{r^4} \cdot (\ln n + \ln \epsilon^{-1}). \]
\[ T_{\text{actual mix}}(\epsilon) \leq \frac{120}{(1 - \frac{7}{\sqrt{C}})^2} \cdot \frac{1}{r^2} \cdot (\ln n + \ln \epsilon^{-1}). \]

We prove the theorem by bounding the spectral gap of the Maximum Degree random walk using Theorem C.2. In order to define a distribution over canonical paths on \( G^2(n, r) \), we introduce the notion of a square grid on the unit torus.

**Square grid** Let \( t = \lceil \frac{\sqrt{5}}{2} \rceil \). We divide the unit square \([0, 1]^2\) into \( t^2 \) squares, each one of side length \( 1/t \). Each square is surrounded by eight neighboring squares. Consider any two nodes \( u, v \in G^2(n, r) \) belonging to neighboring squares. Since the square side length is at most \( r/\sqrt{5} \), then \( cd(u_x, v_x) \leq 2r/\sqrt{5} = r/\sqrt{2} \) and similarly \( cd(u_y, v_y) \leq 2r/\sqrt{5} = r/\sqrt{2} \). It follows that \( gd(u, v) \leq r \), implying \( u \) and \( v \) are neighbors in the graph \( G^2(n, r) \). We next prove that with high probability each square in the grid contains about \( n/t^2 \) nodes:

**Proposition C.3.** Fix any \( 0 < \alpha_s < 1 \). Let
\[ \delta_s = \sqrt{\frac{3t^2}{n} \cdot \ln \frac{2t^2}{\alpha_s}}. \]
With probability at least $1 - \alpha_s$ (over the choice of the random graph), every square in the square grid contains between $(n/t^2) \cdot (1 - \delta_s)$ and $(n/t^2) \cdot (1 + \delta_s)$ nodes of $G^2(n, r)$.

**Proof.** Fix any square $C$ in the square grid. For each $i = 1, \ldots, n$, let $X_i$ be the 0-1 random variable indicating whether the $i$-th node of $G^2(n, r)$ lands in $C$ or not. Clearly, $E(X_i) = \Pr(X_i = 1) = 1/t^2$.

Let $X = \sum_{i=1}^n X_i$ be the total number of nodes of $G^2(n, r)$ that fall into $C$. By linearity of expectation, $E(X) = n/t^2$. By Chernoff bounds,

$$\Pr(|X - E(X)| > \delta_s E(X)) \leq 2 \cdot \exp\left(-\frac{\delta_s^2 E(X)}{3}\right).$$

Thus,

$$\Pr(|X - \frac{n}{t^2}| > \delta_s \cdot \frac{n}{t^2}) \leq 2 \cdot \exp\left(-\frac{\delta_s^2 n}{3t^2}\right) = \alpha_s.$$

The total number of squares is $t^2$. Hence, by the union bound, the probability there is a square that contains less than $\frac{n}{t^2} \cdot (1 - \delta_s)$ nodes or more than $\frac{n}{t^2} \cdot (1 + \delta_s)$ nodes is at most $\alpha_s$. \hfill \Box

**Square paths** Fix any realization $G$ of the random graph $G^2(n, r)$ that has at least one node in each of the squares of the square grid (by Proposition C.3 the vast majority of the realizations of $G^2(n, r)$ have this property). Let $u \neq v$ be any two distinct nodes in this graph. We next define a family of paths between $u$ and $v$, which we call square paths. Let $C_u$ be the square to which $u$ belongs and let $C_v$ be the square to which $v$ belongs (possibly, $C_u = C_v$). Let $L_{uv}$ be the shortest straight line connecting $u$ and $v$. Let $C_1, \ldots, C_k$ be the sequence of squares through which $L_{uv}$ passes. Clearly, $C_1 = C_u$ and $C_k = C_v$. A square path is a sequence $u_1, \ldots, u_k$ of $k$ nodes that satisfies the following:

1. $u_1 = u$.
2. $u_k = v$.
3. For every $i = 1, \ldots, k$, $u_i$ belongs to $C_i$.

Note that there can be many square paths connecting $u$ and $v$. We next show an upper bound on the length of square paths:

**Proposition C.4.** Let $G$ be any realization of the random graph $G^2(n, r)$ that has at least one node in each square. Let $u \neq v \in G$ be any two distinct nodes. Then, every square path between $u$ and $v$ is of length at most $t + 2$.

**Proof.** Let $L_{uv}$ be the shortest straight line connecting $u = (u_x, u_y)$ and $v = (v_x, v_y)$. Let $C_1, \ldots, C_k$ be the squares through which $L_{uv}$ passes and let $(x_1, y_1), \ldots, (x_k, y_k)$ be the bottom-left corners of these squares, respectively. For every $i = 1, \ldots, k - 1$, the squares $C_i$ and $C_{i+1}$ are neighboring squares, meaning that either $\text{cd}(x_i, x_{i+1}) = 1/t$ and/or $\text{cd}(y_i, y_{i+1}) = 1/t$. Since $\text{cd}(u_x, v_x) \leq 1/2$, then the number of $i \in \{2, \ldots, k - 1\}$ for which $\text{cd}(x_i, x_{i+1}) = 1/t$ is at most $\left\lfloor \frac{1}{2} \cdot \frac{1}{t} \right\rfloor \leq \frac{1}{2}$. Similarly, the number of $i \in \{2, \ldots, k - 1\}$ for which $\text{cd}(y_i, y_{i+1}) = 1/t$ is at most $\frac{1}{2}$. We conclude that $k$ can be at most $t + 2$. \hfill \Box

**Canonical path distribution** Fix any realization $G$ of the random graph $G^2(n, r)$ that has at least one node in each square. Let $\Gamma_G$ be the set of all families of canonical paths $\gamma = \{\gamma_{uv}\}_{u,v \in G}$ on $G$. We now define a probability distribution $p_G$ on $\Gamma_G$. The support of $p_G$ will consist only of families of square paths. We pick such a family $\gamma$ as follows. The $\binom{n}{2}$ paths in $\gamma$ are selected independently. For each $u \neq v \in G$, a canonical path between $u$ and $v$ is chosen uniformly at random among all the square paths between $u$ and $v$. By Proposition C.4, we have an immediate bound on the maximum path length of $p_G$:

$$\ell(p_G) \leq t + 2.$$
Before we prove the upper bound on the maximum expected edge load of \( p_G \), we show the next upper bound on the number of paths that pass through each square:

**Lemma C.5.** Let

\[
\delta_{\ell} = \frac{t}{\sqrt{\binom{n}{2}} \cdot \alpha_{\ell}}.
\]

With probability at least \( 1 - \alpha_{\ell} \) (over the choice of the random graph), the number of paths passing through each square of the square grid is at most \( \binom{n}{2} \cdot \left( \frac{1}{7} + \frac{2}{k^2} \right) \cdot (1 + \delta_{\ell}) \).

**Proof.** Fix any square \( C \). Let \( U_1, \ldots, U_n \) be the \( n \) random points chosen on the surface of the unit torus. For each \( i \neq j \), let \( L_{U_i,U_j} \) be the shortest straight line connecting \( U_i \) and \( U_j \). We define \( X^C_{ij} \) to be the 0-1 random variable indicating whether \( C \) intersects \( L_{U_i,U_j} \) or not. Let \( X^C = \sum_{1 \leq i < j \leq n} X^C_{ij} \) be the number of paths passing through \( C \). Our goal is to show that \( X^C \) is small with high probability. To this end, we first bound the expectation of \( X^C \) and then use Chebyshev’s inequality to prove that with high probability \( X^C \) does not exceed its expectation by much.

By linearity of expectation, \( E(X^C) = \sum_{1 \leq i < j \leq n} E(X^C_{ij}) \). It thus suffices to bound the expectation of \( X^C_{ij} \):

**Claim C.6.** For all \( C, i, j, \frac{1}{7} \leq E(X^C_{ij}) \leq \frac{1}{7} + \frac{2}{k^2} \).

**Proof.** For every square \( C \) and every two points \( u, v \) on the torus, define:

\[
T(C, u, v) = \begin{cases} 
1 & \text{if } C \text{ intersects the line } L_{uv} \\
0 & \text{otherwise}
\end{cases}
\]

Let \( U \) and \( V \) be uniformly chosen points on the torus surface. Clearly, \( E(X^C_{ij}) = E(T(C, U, V)) \). \( E(T(C, U, V)) \) is the probability that the line \( L_{UV} \) intersects \( C \). We next show that this probability is the same for all \( C \):

**Claim C.7.** Let \( U \) and \( V \) be uniformly chosen points on the surface of the torus and let \( L_{UV} \) be the shortest straight line connecting \( U \) and \( V \). Then all squares in the square grid are equally likely to intersect \( L_{UV} \).

**Proof.** The proof is based on the symmetry of the torus. For each square \( C \), let \( S_C \) be the set of pairs of points whose shortest connecting line passes through \( C \):

\[
S_C = \{(u, v) \mid L_{uv} \cap C \neq \emptyset\}.
\]

Fix any two squares \( C, C' \). We would like to show that \( |S_C| = |S_{C'}| \). That would imply that all squares are equally likely to intersect the line \( L_{UV} \) connecting two random points \( U, V \) on the torus surface. To this end, we define a 1-1 function \( f \) from the torus surface to itself and prove that \( f \) induces a 1-1 mapping from \( S_C \) onto \( S_{C'} \).

Let \((x, y)\) and \((x', y')\) be the leftmost bottom corners of \( C \) and \( C' \), respectively. We define the function \( f \) as follows. For every point \( w = (w_x, w_y) \):

\[
f(w_x, w_y) = ((w_x + \text{cd}(x, x')) \mod 1, (w_y + \text{cd}(y, y')) \mod 1).
\]

To show \( f \) is 1-1 we present an inverse mapping:

\[
g(z_x, z_y) = ((z_x - \text{cd}(x, x')) \mod 1, (z_y - \text{cd}(y, y')) \mod 1).
\]

36
Indeed, let \( w = (w_x, w_y) \) be any point on the torus surface. The \( x \)-coordinate of \( g(f(w)) \) is:

\[
\begin{align*}
((w_x + cd(x, x')) \mod 1 - cd(x, x')) \mod 1 \\
= (w_x \mod 1 + cd(x, x') \mod 1 - cd(x, x') \mod 1) \mod 1 \\
= (w_x \mod 1) \mod 1 = w_x.
\end{align*}
\]

Similarly, the \( y \)-coordinate of \( g(f(w)) \) is \( w_y \) and hence \( g(f(w)) = w \).

We observe that \( f \) maps lines to lines and squares to squares. Furthermore, \( f(C) = C' \). Let \( F \) be the following mapping from \( S_C \): \( F(u, v) = (f(u), f(v)) \). Next, we prove that \( F \) is a 1-1 mapping from \( S_C \) onto \( S_{C'} \). Let \( (u, v) \) be any pair in \( S_C \). This means that \( C \) intersects the line \( L_{uv} \). Let \( w \) be a point in \( C \cap L_{uv} \). Since \( f \) maps lines to lines, then \( f(w) \) must lie also on the line \( L_{f(u)f(v)} \) connecting \( f(u) \) and \( f(v) \). On the other hand, since \( w \in C \) and \( f(C) = f(C') \), then \( f(w) \in C' \). We conclude that \( L_{f(u)f(v)} \) intersects \( C' \), and thus \( (f(u), f(v)) \in S_{C'} \). \( F \) is then a mapping from \( S_C \) to \( S_{C'} \). It is 1-1 due to the fact \( f \) is 1-1. A similar argument can show that \( G(u', v') = (g(u), g(v)) \) (where \( g = f^{-1} \)) is the inverse mapping of \( F \). Hence, \( F \) is a 1-1 mapping from \( S_{C'} \) onto \( S_C \) implying \( S_C \) and \( S_{C'} \) are of equal size.

Going back to the proof of Claim C.6, since \( E(T(C, U, V)) \) is independent of \( C \), we can write it as:

\[
E(T(C, U, V)) = \frac{1}{t^2} \sum_{C'} E(T(C', U, V)),
\]

where the summation is over all squares \( C' \) in the grid and \( t^2 \) is the number of such squares. By linearity of expectation,

\[
\frac{1}{t^2} \sum_{C'} E(T(C', U, V)) = \frac{1}{t^2} E(\sum_{C'} T(C', U, V)).
\]

Since for every \( u, v \), the number of squares that intersect \( L_{uv} \) is at least 1 and at most \( t + 2 \) (Proposition C.4), then

\[
1 \leq \sum_{C'} T(C', U, V) \leq t + 2.
\]

We conclude that:

\[
E(X_{ij}^C) = E(T(C, U, V)) \leq \frac{1}{t^2} \cdot (t + 2) = \frac{1}{t} + \frac{2}{t^2}
\]

and

\[
E(X_{ij}^C) = E(T(C, U, V)) \geq \frac{1}{t^2}.
\]

We conclude from the above claim that

\[
E(X^C) \leq \left( \frac{n}{2} \right) \cdot \left( \frac{1}{t} + \frac{2}{t^2} \right).
\]

Next, we prove that the sequence of random variables \( \{X_{ij}^C\}_{1 \leq i < j \leq n} \) is pairwise independent:

**Claim C.8.** For every \((i, j) \neq (i', j')\), \( X_{ij}^C \) and \( X_{ij'}^C \) are independent.

**Proof.** If \((i, j) \cap \{i', j'\} = \emptyset \), then the independence of \( X_{ij}^C \) and \( X_{ij'}^C \) follows from the independence of the two pairs \((U_i, U_j)\) and \((U_i', U_j')\). So suppose, e.g., that \( i = i' \) but \( j \neq j' \). The independence will follow from the following stronger claim: for every two points \( u, v \) on the torus surface,

\[
\Pr(X_{ij}^C = 1 \mid U_i = u, U_{j'} = v) = \Pr(X_{ij}^C = 1).
\]

37
Since \(X_{ij}^C\) and \(U_i\) are independent of \(U_{ij}\), it suffices to show that for all \(u\),

\[
Pr(X_{ij}^C = 1 \mid U_i = u) = Pr(X_{ij}^C = 1).
\]

That is, even if we know that \(U_i\) was chosen to be \(u\), this does not change the probability of the line \(L_{U_iU_j}\) to pass through \(C\). This statement follows by a symmetry argument, similar to the one done in the proof of Claim C.7: for every fixed point \(u\), all squares are equally likely to intersect the line \(L_{uV}\) (where \(V\) is chosen at random).

\[\square\]

We now finally return to the proof of Lemma C.5. Since \(E(X^C) \leq \binom{n}{2} \cdot \left( \frac{1}{t} + \frac{2}{t^2} \right) (1 + \delta_t)\), then

\[
Pr(X^C > \binom{n}{2} \cdot \left( \frac{1}{t} + \frac{2}{t^2} \right) (1 + \delta_t)) \leq Pr(X^C > E(X^C) \cdot (1 + \delta_t)) \leq Pr(|X^C - E(X^C)| > \delta_t \cdot E(X^C)).
\]

By Chebyshev’s inequality,

\[
Pr(|X^C - E(X^C)| > \delta_t \cdot E(X^C)) \leq \frac{\text{var}(X^C)}{\delta_t^2 \cdot E^2(X^C)}.
\]

Recall that \(X^C = \sum_{i,j} X_{ij}^C\). The random variables \(\{X_{ij}^C\}_{i,j}\) are identically distributed and pairwise independent. Let \(p = Pr(X_{ij}^C = 1)\). Then, \(\text{var}(X^C) = \binom{n}{2} \cdot p(1 - p) \leq \binom{n}{2} \cdot p\). On the other hand, \(E(X^C) = \binom{n}{2} \cdot p\). Therefore,

\[
\frac{\text{var}(X^C)}{\delta_t^2 \cdot E^2(X^C)} \leq \frac{1}{\delta_t^2 \binom{n}{2} p}.
\]

By Claim C.6, \(p \geq 1/t^2\). Also recall that \(\delta_t = t/\sqrt{\binom{n}{2} \cdot \alpha}\). We conclude that

\[
Pr(X^C > \binom{n}{2} \cdot \left( \frac{1}{t} + \frac{2}{t^2} \right) (1 + \delta_t)) \leq \frac{\alpha}{t^2}.
\]

Using the union bound and based on the fact there are \(t^2\) squares, the probability there is at least one square \(C\) for which \(X^C > \binom{n}{2} \cdot \left( \frac{1}{t} + \frac{2}{t^2} \right) (1 + \delta_t)\) is at most \(\alpha\).

\[\square\]

We are now ready to prove the upper bound on the maximum expected edge load of \(p_G\):

**Lemma C.9.** Fix any \(0 < \alpha, \alpha_s < 1\). Let \(G\) be a random realization of the random graph \(G^2(n, r)\) and let \(p_G\) be the canonical path distribution defined above. Then, with probability at least \(1 - \alpha - \alpha_s\) (over the choice of \(G\)),

\[
\rho(p_G) \leq \frac{t^3}{16} \cdot \left( \frac{1}{t} + \frac{2}{t^2} \right) \cdot \frac{1 + \delta_t}{(1 - \delta_s)^2}.
\]

**Proof.** Fix any square \(C\). By Lemma C.5, with probability at least \(1 - \alpha\), the number of paths that pass through \(C\) is at most

\[
\binom{n}{2} \cdot \left( \frac{1}{t} + \frac{2}{t^2} \right) (1 + \delta_t).
\]

The canonical path distribution disseminates the paths that pass through \(C\) evenly among the nodes in \(C\). By Proposition C.3, with probability at least \(1 - \alpha_s\), the number of nodes in \(C\) is at least

\[
\frac{n}{t^2} \cdot (1 - \delta_s).
\]
Fix any node \( u \in C \). We conclude that with probability at least \( 1 - \alpha_\ell - \alpha_s \), the expected number of paths that pass through \( u \) is at most

\[
\left( \frac{n}{2} \right) \cdot \left( \frac{t}{1 + \delta_\ell} \right) \cdot \frac{1 + \delta_\ell}{1 - \delta_s} \leq \frac{n}{2} \cdot t \cdot \frac{1 + \delta_\ell}{1 - \delta_s}.
\]

A symmetry argument similar to the one shown in the proof of Claim C.6 can show that in expectation exactly 1/8 of the paths that pass through square \( C \) go to each one of its neighboring squares. Fix a neighboring square \( C' \). Recall that any node in \( C \) is connected to any node in these neighboring squares. Hence, 1/8 of the paths that pass through \( u \) are expected to use the edges that connect \( u \) with nodes in \( C' \). Since the canonical path distribution picks a random node from each square independently, then all the edges that connect \( u \) and \( C' \) are expected to carry the same load. This load then equals the number of paths that pass through \( u \) divided by 8 and divided again by the number of nodes in \( C' \). We already know (Proposition C.3) that the number of nodes in \( C' \) is at least \( \frac{n}{2} \cdot (1 - \delta_s) \), hence the expected load on edges connecting \( u \) with \( C' \) is at most:

\[
\frac{n}{2} \cdot t \cdot \left( \frac{1 + \frac{1}{2}}{1 - \delta_s} \right)^2 = \frac{t^3}{16} \cdot \frac{1 + \frac{1}{2}}{1 - \delta_s} \cdot \frac{1 + \delta_\ell}{1 - \delta_s}.
\]

Since the choice of \( C, C' \) and \( u \) was arbitrary this is also the maximum expected load on edges of the graph \( G \).

We are now ready to prove Theorem 3.6:

**Proof of Theorem 3.6.** Suppose \( D \) is the upper bound on \( d_{\text{max}} \) used in the random walk. We start by analyzing the standard mixing time of the MD random walk, including the self loops. Let \( P \) be the probability transition matrix of the random walk. By Theorem 3.3,

\[
T_{\text{mix}}(\epsilon) \leq \frac{\ln n + \ln(1/\epsilon)}{1 - \lambda_{\text{max}}(P)}.
\]

By the strong version of Sinclair’s bound (Theorem C.1),

\[
\frac{1}{1 - \lambda_{\text{max}}(P)} \leq \frac{D}{n} \cdot \ell(p_G) \cdot \rho(p_G).
\]

Hence,

\[
T_{\text{mix}}(\epsilon) \leq \frac{D}{n} \cdot \ell(p_G) \cdot \rho(p_G) \cdot (\ln n + \ln(1/\epsilon)).
\]

We set \( \alpha_d = \alpha_s = \alpha_\ell = 1/9 \). Then, with probability at least \( 2/3 \), the chosen random graph \( G \) satisfies the three following conditions:

1. By Proposition 3.5, its maximum degree, \( d_{\text{max}} \), is at most \( \pi t^2 (n - 1) \cdot (1 + \delta_d) \).
2. By Proposition C.4, \( \ell(p_G) \leq t + 2 \).
3. By Lemma C.9, \( \rho(p_G) \leq \frac{t^3}{16} \cdot \left( 1 + \frac{1}{2} \right) \cdot \frac{1 + \delta_\ell}{(1 - \delta_s)^2} \).

Therefore,

\[
T_{\text{mix}}(\epsilon) \leq \frac{D}{n} \cdot (t + 2) \cdot \frac{t^3}{16} \cdot \left( 1 + \frac{1}{2} \right) \cdot \frac{1 + \delta_\ell}{(1 - \delta_s)^2} \cdot (\ln n + \ln(1/\epsilon))
\]

\[
= \frac{D}{n} \cdot \frac{(t + 2)^2}{16} \cdot \frac{t^2}{(1 - \delta_s)^2} \cdot \frac{1 + \delta_\ell}{(1 - \delta_s)^2} \cdot (\ln n + \ln(1/\epsilon)).
\]
Now, recall that $t = \lceil \sqrt{8}/r \rceil \leq \sqrt{8}/r + 1$. Therefore, $t + 2 \leq \sqrt{8}/r + 3$. $r$ was chosen so that $r \leq 1/2$, hence $\sqrt{8}/r + 3 \leq (\sqrt{8} + 3/2)/r < 5/r$. Similarly, $t^2 \leq (\sqrt{8} + 1)^2 = \frac{8}{r^2} + 1 + \frac{2\sqrt{8}}{r} \leq \frac{12}{r^2}$. Therefore, $(t + 2) \cdot t^2 / 16 < 19/r^4$. We conclude that:

$$T_{\text{mix}}(\epsilon) \leq 19 \cdot \frac{D}{n} \cdot \frac{1}{r^4} \cdot \frac{1 + \delta_s}{(1 - \delta_s)^2} \cdot (\ln n + \ln(1/\epsilon)).$$

Recall that:

$$\delta_s = \frac{t}{\sqrt{\binom{r}{2}} \cdot \alpha_s}$$

and

$$\delta_s = \sqrt{\frac{3t^2}{n} \cdot \frac{2t^2}{r \ln(1/\epsilon)}}.$$

Since $r \leq 1/2$, then $t \leq \sqrt{8}/r + 1 \leq 4/r$. Also, $\alpha_s = 1/9$. Hence,

$$\delta_s \leq \frac{\sqrt{12 \sqrt{2}}}{r \sqrt{n(n - 1)}}.$$

Recall that $r = \sqrt{C \ln n}/n$ for $C > 49$ and that $n \geq 10$. Therefore, $\delta_s < 0.55$.

As for $\delta_s$, $t^2 \leq 12/r^2$ and $\alpha_s = 1/9$. Hence,

$$\delta_s \leq \frac{\sqrt{36 \ln 216}}{r^2 n}.$$

Rewriting $r$ as $\sqrt{C \ln n}/n$, we have:

$$\delta_s \leq \sqrt{\frac{36 \ln 216}{C \ln n \ln C \ln n}} = \left[ \frac{\sqrt{36 \ln 216}}{C \ln n} \right] \cdot \left[ \frac{\ln 216}{\ln C \ln n} \right].$$

Since $C > 49$ and $n \geq 10$, then $\delta_s < \sqrt{47/C} < 7/\sqrt{C}$. By incorporating the bounds on $\delta_s$ and $\alpha_s$, we obtain the desired bound on the mixing time:

$$T_{\text{mix}}(\epsilon) \leq \frac{30}{(1 - \frac{7}{\sqrt{C}})^2} \cdot \frac{D}{n} \cdot \frac{1}{r^4} \cdot (\ln n + \ln(1/\epsilon)).$$

We now turn to the calculation of the actual mixing time. Consider a run of the MD random walk, and let $U_1, U_2, \ldots$ be the distinct nodes visited during the random walk. (Note that $U_1, U_2, \ldots$ are random variables.) For each $i = 1, 2, \ldots$, let $X_i$ denote the number of steps the random walk spends at $U_i$. That is, $X_i$ is 1 plus the number of steps the random walk spends at the self loop of $U_i$ until moving to $U_{i+1}$. For every infinite sequence of nodes $v_1, v_2, \ldots$ the random variables $X_1, X_2, \ldots$ are independent given that $U_1 = v_1, U_2 = v_2, \ldots$ (that is, the number of self loop steps spent at $U_i$ depends only on $U_i$ and not on the other nodes visited during the random walk).

Consider any step $i$. Given that $U_i = v_1$, $X_i$ is a geometric random variable with probability of success $d_{v_1}/D$, where $d_{v_1}$ is the degree of $v_1$, excluding the weighted self loop. Hence, $E(X_i|U_1 = v_1, U_2 = v_2, \ldots, U_i = v_i, \ldots) = E(X_i|U_i = v_i) = D/d_{v_i}$. Let $d_{\text{max}} = \max_v d_v$. Then, $E(X_i|U_1 = v_1, U_2 = v_2, \ldots) \geq D/d_{\text{max}}$.

Let $m = T_{\text{mix}}(\epsilon)$ be the mixing time of the random walk. The random walk runs for $m$ steps, including self loop steps, until it is stopped. Let $T$ denote the number of non-self loop steps made by the random walk. Note that $T$ is a random variable and $E(T)$ is the actual mixing time $T_{\text{actual mix}}(\epsilon)$.
we wish to calculate. Furthermore, \( \sum_{i=1}^{T} X_i = m \). Since for every sequence of nodes \( v_1, v_2, \ldots \), the random variables \( X_1, X_2, \ldots \) are independent given that \( U_1 = v_1, U_2 = v_2, \ldots \), the conditions of Wald’s identity (cf. [37]) are met, implying that:

\[
E(\sum_{i=1}^{T} X_i \mid U_1 = v_1, U_2 = v_2, \ldots) \geq E(T \mid U_1 = v_1, U_2 = v_2, \ldots) \cdot \frac{D}{d_{\text{max}}}.
\]

Since \( \sum_{i=1}^{T} X_i = m \) always, we have:

\[
E(T \mid U_1 = v_1, U_2 = v_2, \ldots) \leq m \cdot \frac{d_{\text{max}}}{D}.
\]

This holds for every sequence \( v_1, v_2, \ldots \). Thus,

\[
E(T) \leq m \cdot \frac{d_{\text{max}}}{D}.
\]

Hence,

\[
T_{\text{actual mix}}(\epsilon) = E(T) \leq m \cdot \frac{d_{\text{max}}}{D} = T_{\text{mix}}(\epsilon) \cdot \frac{d_{\text{max}}}{D}
\]

\[
\leq \frac{30}{(1 - \frac{7}{\sqrt{C}})^2} \cdot \frac{D}{n} \cdot \frac{1}{r^4} \cdot (\ln n + \ln(1/\epsilon)) \cdot \frac{d_{\text{max}}}{D}
\]

\[
= \frac{30}{(1 - \frac{7}{\sqrt{C}})^2} \cdot \frac{d_{\text{max}}}{n} \cdot \frac{1}{r^4} \cdot (\ln n + \ln(1/\epsilon)).
\]

Recall that we assumed the chosen random graph satisfies

\[
d_{\text{max}} \leq \pi r^2(n - 1) \cdot (1 + \delta_d),
\]

where

\[
\delta_d = \sqrt{\frac{3 \cdot \frac{2n}{\ln \alpha_d}}{\pi r^2(n - 1)}}.
\]

Writing \( r = \sqrt{C \ln n / n} \) and recalling that \( \alpha_d = 1/9 \), we have:

\[
\delta_d = \sqrt{\frac{3n}{\pi C \ln n(n - 1) \cdot \ln(18n)}}.
\]

Since \( C > 49 \) and \( n \geq 10 \), we have: \( \delta_d < 0.25 \). Therefore, \( d_{\text{max}} \leq 1.25 \pi r^2(n - 1) < 4r^2n \).

Substituting in the bound for \( T_{\text{actual mix}}(\epsilon) \), we have:

\[
T_{\text{actual mix}}(\epsilon) \leq \frac{120}{(1 - \frac{7}{\sqrt{C}})^2} \cdot \frac{1}{r^2} \cdot (\ln n + \ln(1/\epsilon)).
\]

\[\square\]