Probabilistic Quorum Systems in Wireless Ad Hoc Networks*

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

Chen Avin
Communication Systems Engineering Dep.
Ben Gurion University of The Negev
Beer Sheva, Israel
avin@cse.bgu.ac.il

February 22, 2009

Abstract

Quorums are a basic construct in solving many fundamental distributed computing problems. One of the known ways of making quorums scalable and efficient is by weakening their intersection guarantee to being probabilistic. This paper explores several access strategies for implementing probabilistic quorums in ad hoc networks. In particular, we present the first detailed study of asymmetric probabilistic bi-quorum systems and show its advantages in ad hoc networks. Such an asymmetric construction of probabilistic bi-quorum systems is also useful for other types of networks with non uniform access costs (e.g, peer-to-peer networks). The paper includes both a formal analysis of these approaches backed up by an extensive simulation based study. In particular, we show that one of the strategies that uses Random Walks, exhibits the smallest communication overhead, thus being very attractive for ad hoc networks.

Keywords: Wireless Ad Hoc Networks, Quorums Systems, Random Walks, Location Service, Distributed Middleware

1 Introduction

Quorums are a basic construction in many distributed systems. They can be used as building blocks (or at least as a design pattern) in solving various fundamental problems such as Consensus [25], distributed dictionaries and location services [20], distributed storage [4, 48], etc. While most implementations of quorum systems are deterministic, some works have suggested the use of probabilistic quorums as a way of improving their resilience, efficiency, and scalability [50].

The idea behind quorums is that they ensure intersection between subsets of nodes. The intersection property enables maintaining consistency of actions taken by nodes of a distributed system. This is achieved by contacting a quorum of nodes before taking an action that could change the state of the system, or before completing operations that might conflict other operations. This ensures that any two such actions are seen by at least one common node, which enables the detection and elimination of conflicts.

In ad-hoc networks [66], location services are one of the most important services for many of the envisioned applications, as they enable users to find information and services stored by others. As discussed in [20], a very large percentage of location services are based on bi-quorums, whether they are built explicitly or implicitly as part of the overall solution. Recently, there have been several attempts

*A shorter preliminary version of this paper appeared in the 38th IEEE International Conference on Dependable Systems and Networks, June 2008.

to solve Consensus and distributed storage in ad-hoc networks, e.g., [13, 16, 45]. As mentioned before, these also require quorums.

In this paper we investigate the implementation of scalable probabilistic quorums in ad-hoc networks. Ad-hoc networks are unique, compared to wired networks, in several aspects. In wireless ad-hoc networks, routing and flooding are extremely expensive. Hence, applications and services developed for these networks should avoid multiple hop routing and flooding as much as possible, and instead aspire to rely solely on local one-hop message exchanges. The dynamic nature of ad hoc network (caused by churn and nodes mobility) makes the usage of strict deterministic quorums highly costly. Therefore, for the sake of scale and efficiency, we relax the requirements of the quorum system to probabilistic ones, similar to [50].

One could potentially use geographical knowledge for construction of quorum systems in ad hoc network (e.g., [42, 64]). However, as GPS and other accurate positioning techniques may not always be available, and since the network's boundaries are not always known, in this paper we look for quorum systems that do not rely on geographical knowledge.

Contributions of this work. A central contribution of this work is the introduction of the first *asymmetric* probabilistic bi-quorum system. Specifically, in previous works, e.g., [44, 45, 50], accesses to all quorums of a probabilistic bi-quorum system are performed using the same access strategy.[1] In that respect, all previously known probabilistic bi-quorum systems are *symmetric*. In particular, we prove a "mix-and-match" lemma, showing that it is possible to combine different access strategies (and different quorum sizes) and still obtain intersection with a desired probability. This is an important enabling result for implementing probabilistic bi-quorum systems in any network in which the access cost is non-uniform, such as ad-hoc networks and peer-to-peer networks [23, 46]. This is because in a network with non-uniform communication cost, accessing a random set of nodes means having to contact at least some far away (or "expensive") nodes on each quorum access. Our result shows that in fact only one type of quorum access (e.g., advertise or update) needs to be "expensive", while the other (e.g., lookup or read) can be optimized to only access the closest (or "cheapest") nodes. Specifically, we demonstrate that such asymmetric bi-quorum systems indeed offer superior performance compared to symmetric ones in ad hoc networks.

Another contribution of our work is the orderly introduction and study of several schemes for accessing probabilistic quorums in ad hoc networks. We study the performance of the proposed schemes both analytically and by simulations. In particular, one of the schemes we investigate is based on random walks (RW). The use of RWs in wireless ad hoc networks has been previously proposed to solve various problems such as membership construction [17, 9], reliable multicast [17], routing [62] and querying [6]. RWs are attractive for ad hoc networks since they require neither multi-hop routing nor broadcasting, which are expensive in ad hoc networks [9]. Also, they offer fine grain control over the communication overhead as well as early halting capabilities, as elaborated later in this paper.

Additionally, we provide a number of useful formal results about random walks in random geometric graphs. First, we provide a novel result about a *partial cover time (PCT)* of RWs in random geometric graphs. We were able to show that covering a sub-linear fraction of nodes i takes a number of RW steps that is linear in i. In addition, we provide a definition of the *crossing time* of RWs and a lower bound on the crossing time in random geometric graphs. To capture the reliability of the quorum systems in dynamic environments we present a new metric, called *degradation rate*. These results help us reason about the costs and benefits of the various access strategies, as well as complete bi-quorum systems.

All our results are validated through an extensive simulation study, carried on the JIST/SWANS simulator from Cornell[2], which models mobility, collisions, and signal propagation, and implements the entire 802.11 MAC.

[1] In [44, 45], the authors use the term "asymmetric probabilistic quorums", however the asymmetry in [44, 45] refers to different quorum sizes and not different access strategies.

[2] `http://jist.ece.cornell.edu/`

Paper's road map. For methodological reasons, we first present the basic concepts of probabilistic bi-quorum systems and the various access strategies, and only then turn to the mix-and-match lemma and the simulations based evaluation. Specifically, we start by introducing some preliminaries and system model in Section 2. Section 3 describes the metrics used to evaluate Quorum Systems. In Section 4 we present a number of general strategies for accessing a single quorum. We then show in Section 5 how to mix those strategies and implement several different probabilistic ϵ-intersecting quorum systems and discuss their properties analytically. Quorum maintenance in the face of churn and mobility is discussed in Section 6 and certain optimizations of the basic access strategies are discussed in Section 7. Section 8 presents the simulation study. We discuss related work in Section 9 and conclude with a discussion in Section 10.

2 Preliminaries

2.1 Quorums and Bi-Quorums

Intuitively, a quorum system is a set of subsets such that every two subsets intersect. Moreover, a bi-quorum system consists of two sets of subsets such that each subset in one set intersects with each subset in the other set. Below, we provide a formal definition of these notions, following the works of [21, 24, 30, 54].

Definition 2.1 (Set System)**.** A set system $\mathcal{S}$ over a universe U is a set of subsets of U.

Definition 2.2 (Quorum System)**.** A quorum system $\mathcal{Q}$ over a universe U is a set system over U such that for any $Q_1, Q_2 \in \mathcal{Q}$, $Q_1 \cap Q_2 \neq \emptyset$.

Definition 2.3 (Bi-quorum System)**.** A bi-quorum system $\mathcal{Q}$ over a universe U is a couple of set systems $(\mathcal{Q}_1, \mathcal{Q}_2)$ such that for any $Q_1 \in \mathcal{Q}_1$ and $Q_2 \in \mathcal{Q}_2$, $Q_1 \cap Q_2 \neq \emptyset$.

In this work we focus on bi-quorums. We will also refer to them here as `lookup` and `advertise` quorums given that bi-quorums are often used in conjunction with lookup and advertise operations.[3] However, the discussion applies the same for any bi-quorum system.

A data discovery service as well as any distributed dictionary can be implemented using an advertise/lookup quorum system as follows: Publishing a data item is implemented by contacting all members of a single `advertise` quorum and having them store the information. Looking up the data is performed by contacting a `lookup` quorum. The intersection between any `advertise` quorum and any `lookup` quorum ensures that if a data item has been published, it will be found by the lookup operation.

2.2 Probabilistic Quorums

In *probabilistic quorums* [50], a quorum system is not fixed a-priori, but is rather picked in a probabilistic manner for each interaction with the quorum system. For example, in the case of bi-quorums, such as lookup/advertise quorums, it is ensured that each (randomly selected) `lookup` quorum intersects with every (randomly selected) `advertise` quorum with a given probability.

2.3 Ad Hoc Network System Model

Consider a wireless network comprising a finite set of nodes V, located arbitrary in the plane. A node in the system is a device owning an omni-directional antenna that enables wireless communication. Let X_i for $i = 1, 2, \ldots, |V|$ denote the location of the nodes. For simplicity, we also use X_i to refer to the i^{th} node itself. We denote by $|X_i - X_j|$ the Euclidian distance between node X_i and node X_j.

[3] We discuss implementing read/write registers via quorums in Section 10.

Communication model. We consider two possible models for a successful reception of a transmission over one hop, which are commonly used in the literature [27, 39].

The Protocol model. In this model, each node X_i is associated with a specific transmission range r_i. A transmission from X_i is received successfully by X_j if and only if $|X_i - X_j| \leq r_i$ and for every other simultaneously transmitting node X_k, $|X_k - X_j| \geq (1 + \triangle)r_k$. $\triangle > 0$ is the *interference parameter*. To simplify the analysis, we assume that all transmission ranges are the same: for each $X_i \in V, r_i = r$.

The Physical model. Let $\mathcal{T} \subseteq V$ be the set of nodes simultaneously transmitting at some time instant, and denote by P_k the transmit power of node $X_k \in \mathcal{T}$. The transmission from node X_i is received successfully by X_j if and only if

$$\frac{\frac{P_i}{|X_i - X_j|^\alpha}}{N_0 + \sum_{X_k \in \mathcal{T}, X_k \neq X_i} \frac{P_k}{|X_k - X_j|^\alpha}} \geq \beta$$

In this model, a minimum signal-to-interference-plus-noise ratio (SINR) of β is necessary for successful reception. The ambient noise power level is denoted N_0 and the signal decays deterministically with distance r as $1/r^\alpha$. As typical to this model, we assume that $\alpha = 2$. In our simulations, we use homogeneous transmit powers: for each $X_i \in V, P_i = P$.

Theoretical Graph model. We carry all our analysis in the protocol communication model, which greatly simplifies the formal analysis, without affecting the access strategies. In addition, all our results are validated by a simulation study, performed in the more realistic physical model, (which includes a realistic signal propagation model, signal interference, distortions, background noise, unidirectional links, etc.), as elaborated in Section 2.4.

In the protocol communication model, all nodes that are located within the transmission range r of a node X_i form the set of X_i's potential neighbors. The combination of the nodes and the neighborhood relations forms a wireless ad hoc network. The resulting network connectivity graph $G = (V, E)$ is a 2-dimensional *Unit Disk* graph, in which n nodes are embedded in the surface of a unit torus[4], and any two nodes within Euclidean distance r of each other are connected. When the nodes are placed uniformly at random on the surface the resulting graph is known as a *Random Geometric Graph* (RGG) [58] and is denoted by $\mathcal{G}^2(n, r)$. Specifically, the $\mathcal{G}^2(n, r)$ graph is often used to model the network connectivity graph of 2-dimensional wireless ad hoc networks and sensor networks [26, 27].

We assume that nodes do not know their position and we do not use any geographic knowledge. Yet, each node knows all of its temporal direct potential neighbors (nodes with which it can currently communicate directly). This can be implemented, e.g., by a simple heartbeat mechanism that is present in any case in most routing protocols in ad hoc networks [33]. In addition, a node can communicate with its non-direct neighbors (if exist), if other nodes agree to relay its messages.

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. The rate at which nodes join and leave the system in known as the *churn rate* of the system. We assume that the network remains continuously connected.

2.4 Simulations Setup

Our simulations were performed using the JiST/SWANS simulator [67] from Cornell university, which is a discrete event based network simulator that includes an accurate model of a full networking stack. All simulation parameters are summarized in Figure 1.

The signal interference model was RadioNoiseAdditive, which is based on a cumulative noise computation with SINR and capture effect. This is equivalent to the Physical model of successful reception

[4] In practice, the network is flat rather than a torus. However, this does not affect the access strategies, but greatly simplifies the formal model. At any event, our simulations are carried on a flat topology with a physical reception model.

PHY	
Signal Propagation model (PathLoss)	Two-Ray ground reflection
Signal Interference model	Cumulative noise with SNR and capture effect
Transmit power	15 dBm (31.62 mW)
Receive Thresh (RXThresh in ns2)	-71 dBm (7.9432e-8 mW)
Sensitivity Thresh (CSThresh in ns2)	-77 dBm (1.9952e-8 mW)
Background noise (thermal noise)	-101 dBm (8.0080e-11 mW)
SNR (β) (CPThresh in ns2)	10
TX/RX Antenna gain	0 dBm (1 mW)
Fading	None
Ideal Reception range	200 meter
Carrier Sensing Range	299 meter
MAC	
Modulation	DSSS with long preamble and PLCP header
Slot Time	20 μs
DIFS	50 μs
Bandwidth	11 Mbps for unicast, 2 Mbps for broadcast
Simulation Scenarios	
Message size	512 bytes + IP + MAC + PHY headers
Node count	50, 100, 200, 400, 800
Density (# of one hop neighbors)	**10 default.** Varying: 7, 10, 15, 20, 25.
Mobility	Random WayPoint, **default speed 0.5-2 m/s** Varying max speed: 2,5,10,20 m/s, pause 30s
Routing protocol (when applicable)	AODV
Heartbeat cycle	10 sec
# of advertisements	100
# of lookups	1000 (by 25 random nodes)
Java pseudo random number generator, initialized with the current time in millis as seed	

Figure 1: Simulation parameters

and is also identical to the interference model used in Glomosim and NS2 version 2.33 simulators (refer to [40] for more details). The ideal reception range, without any collisions, was set to 200m and the average number of neighbors, d_{avg}, ranged from 7 to 25 in various simulations (default $d_{\text{avg}} = 10$). This was achieved by scaling the area size according to $a^2 = \frac{\pi r^2 n}{d_{\text{avg}}}$ and resulted in all networks being connected (according to the connectivity result of [26], d_{avg} should be $\pi r^2 n = C \ln n$, for $C > 1$ and in our case $d_{\text{avg}} = 7$ bounded $C \ln n$ for all n's we used). The nodes were placed at uniformly random locations in a square plane. All simulations were performed on networks of 50, 100, 200, 400 and 800 nodes.

The multi-hop routing protocol used for accessing *RANDOM* quorum is AODV. The mobility pattern was the Random Waypoint model with different speeds. The default speed of movement ranged from 0.5-2 m/s, which corresponds to slow and fast walking speeds, and an average pause time of 30s.

3 Quorum Systems Metrics

Any implementation of a probabilistic quorum system can be analyzed according to the following quality measures [50]:

Intersection probability: Probabilistic quorum system $\mathcal{Q}$ is an ε-intersecting if the probability of any pair of quorums to intersect is at least $1 - \varepsilon$.

Formally: Let $\mathcal{Q}$ be a set system, let w be an access strategy for $\mathcal{Q}$, and let $0 < \varepsilon < 1$ be given. The tuple $\langle \mathcal{Q}, \varepsilon \rangle$ is an ε-intersecting quorum system if $Pr[Q \cap Q' \neq \varnothing] \geq 1 - \varepsilon$, where the probability is taken with respect to the strategy w.

Access cost: The cost (in messages) of accessing a quorum.

Load: The request load on a single node. The target is to balance the request load equally between the nodes.

Failure resilience: The resilience of the quorum system to failures. It is measured by two parameters:

1) Fault tolerance of a quorum system $\mathcal{Q}$ is the size of the smallest set of nodes that intersects all quorums in $\mathcal{Q}$ (i.e., the minimal number of nodes whose crash will leave the system without any quorum). As shown by [50], the fault tolerance of a probabilistic quorum system of size $k\sqrt{n}$ is $n - k\sqrt{n} + 1 = \Omega(n)$. In an ad hoc network it is also required that the quorum nodes form a connected graph (see Section 6 for an elaborative discussion on connectivity of ad hoc networks).

2) Failure probability of a quorum system is the probability that the system becomes disabled when individual nodes crash independently with a fixed probability p. As shown by [50], the failure probability of a probabilistic quorum system of size $k\sqrt{n}$ is $e^{-\Omega(n)}$ for all $p \leq 1 - \frac{k}{\sqrt{n}}$. Again, in ad hoc network, quorum nodes must also form a connected graph.

Failure resilience refers to the resilience of the whole quorum system to failures, rather than the resilience of a single quorum. As long as the entire quorums system has not failed, a new live quorum can be found. But it does not say anything about the liveness or a chance of survival of a previously accessed quorum. Therefore, failure resilience is not enough to measure the resilience of a dynamic quorum system. For this reason we introduce the following novel measure:

Degradation rate: The rate of change in the intersection probability, as a function of network churn. For probabilistic quorums, this metric translates to the probability that two quorums accessed at different times will intersect despite the fact that between these two accesses some nodes have crashed or new nodes have joined. Hence, degradation rate captures the resilience of a single quorum in the face of dynamic changes. The degradation rate helps determining when should the quorum system be reconfigured, or refreshed, in order to recover from failures, node departures and joins.

4 Quorum Access Strategies

An *access strategy* defines the way in which a client trying to access a probabilistic quorum propagates its requests. The access strategy may impact all the measures of a quorum system we presented above. In a bi-quorum system, it is possible to mix and match between the access strategies used for lookup quorums and those used for advertise quorums based, e.g., on the relative frequency of requests of each type. We now present several such access strategies.

In this work we focus on three main strategies: *RANDOM*, *PATH* and *FLOODING*. *RANDOM* simply accesses a set of random, uniformly chosen nodes. *PATH* is a Random Walk, which traverses the network graph until it covers a sufficient set of different nodes. *FLOODING* performs a limited scope flooding of the network, which covers a set of different nodes.

In addition, we consider a number of significant optimizations, which can turn some of the strategies even more appealing. The main novelty of our approach is an ability to mix those strategies in different ways, achieving various tradeoffs discussed below. More specifically, we show that in order to construct probabilistic quorum systems, some quorums can be accessed by optimized, non-random strategies. This is in contrast with the previous methods [45, 50], which used a *RANDOM* strategy to access every quorum.

Figure 2 provides a summary of the asymptotic costs and qualitative properties of various access strategies, for general networks and for Random Geometric Graphs. We elaborate on each of the strategies and the corresponding entries in the table below. The first line refers to the type of accessed nodes: random uniform or arbitrary. The second and third lines capture the cost (the number of messages) to access $|Q|$ nodes. Number of lookup replies refers to whether multiple redundant replies will be sent in the response for a lookup access and early halting refers to the ability to stop the lookup operation the moment the looked-up object is found, without the need to access the full quorum. We now turn to the details.

	RANDOM		PATH	FLOODING
	Membership based	Sampling based		
Accessed Nodes	Random Uniform	Random Uniform	Arbitrary	Arbitrary
General Network	$\lvert Q\rvert Diameter$	$\lvert Q\rvert T_{mix}$	$PCT(\lvert Q\rvert)$	$\lvert Q\rvert$
Random Geometric Graph	$\lvert Q\rvert\sqrt{\frac{n}{\ln(n)}}$	$\lvert Q\rvert n$	$\lvert Q\rvert$, for $\lvert Q\rvert = o(n)$	$\lvert Q\rvert$
Need Routing	Yes	No	No	No
Need Membership Service	Yes	No	No	No
# Lookup Replies	Multiple*	Multiple*	One	Multiple
Early Lookup Halting	No*	No*	Yes	No

* unless accessed serially

Figure 2: Asymptotic and qualitative comparison of different access strategies, for general networks and for Random Geometric Graphs.

4.1 *RANDOM*

In this method, a quorum (be it `lookup` or `advertise`) is simply any random, uniformly chosen, set of nodes Q. We consider two implementations to access such a random set.

Membership Service based Implementation. If a full membership is available (it can be obtained through a standard membership service [15], implemented, e.g., by every node occasionally flooding the network with its id), a node can simply randomly select node ids from its membership list. Alternatively, nodes can utilize a random membership service for ad hoc networks, such as RaWMS [9], which provides every node with a set of uniform-randomly chosen node ids.

Once nodes' ids for a given quorum have been fixed, accessing this quorum can be done by sending a message to each of these nodes through unicast routing. For a quorum of size $|Q|$ we need to send messages to $|Q|$ nodes. Thus, at the application level, the cost is $|Q|$. However, since we are operating over ad hoc networks, the true number has to take into account the cost of multi-hop routing, which includes both the cost of using the routes and establishing the routes. In the general case, the cost of using the route is $\Theta(|Q| \cdot Diameter)$, while $Diameter$ denotes the diameter of the network.

Cost in Random Geometric Graphs. It is well known ([26]) that the diameter of a random geometric graph with transmission range r is $\Theta(1/r)$ and the minimal r to guarantee network connectivity is $\Omega(\sqrt{\frac{\ln n}{n}})$. Thus, assuming the nodes are uniformly distributed, the price of accessing a quorum of this type is

$$\Theta(|Q| \cdot Diameter) = \Theta(|Q| \cdot 1/r) = O\left(|Q|\sqrt{\frac{n}{\ln n}}\right)$$

When $|Q| = \Theta(\sqrt{n})$, the cost is $O(\frac{n}{\sqrt{\ln n}})$.

As for the cost of establishing the routes, it is hard to predict analytically. This cost also depends on routes reuse. In slow moving ad hoc networks with low churn rate, it is best to reuse the same quorum between consecutive invocations as long as all its members are reachable. This amortizes the initial route discovery cost over several requests.

Direct Sampling based Implementation. If no membership service exists, a quorum can be picked

directly by using a sampling service. One possible implementation of a random uniform sampling service for ad hoc networks based on random walks (RWs) is described in [9]. Generally, RWs sample nodes in a non-uniform manner, proportionally to nodes' degrees. To provide uniform samples, the sampling algorithm in [9] utilizes a special form of RW, called a Maximum Degree RW (MD RW). Every uniform sample is obtained by a single MD RW, whose length equals the network mixing time.

Using this method, the *RANDOM* quorum can be accessed directly by starting an appropriate number of Maximum Degree RWs every time a quorum should be accessed. The data item is then published (or looked for) at the end node. Since two or more RWs may end in the same node, then for a quorum of size $|Q|$, we may need more than $|Q|$ RWs. However, for similar arguments as in the birthday paradox, as long as $|Q| = O(\sqrt{n})$, the chance of such collisions are very small. Hence, no more than $|Q|$ RWs are needed (for a more precise analysis refer to [9]). As a result, the cost of accessing a *RANDOM* quorum by this method is $\Theta(|Q| \cdot T_{mix})$, while T_{mix} denotes the network graph mixing time.

Cost in Random Geometric Graphs. The mixing time of a MD RW in Random Geometric Graphs was studied in [9] and was found to be approximately $n/2$. Thus, the total communication complexity of accessing a quorum of size $\Theta(\sqrt{n})$ in this way is $\Theta(n\sqrt{n})$. Yet, here we never invoke routing.

4.2 *PATH*

Another way to pick a quorum is by performing a single *Simple Random Walk (RW)*, which traverses the underlying network graph until it visits $|Q|$ different nodes. Naturally, the size of the quorum, $|Q|$, should be set such as to guarantee the intersection property of the quorum system. At every step, the RW picks one of the neighbors of a current node at uniformly at random and moves to this neighbor. In addition, the RW counts the number of distinct nodes it has visited. This can be implemented, e.g., by storing the list of all visited nodes in the RW header. Another way to implement it is for every node to store the last hop of every RW that passes through it. Since we target at using short RWs (in the order of $\sqrt{n}$), the list of visited nodes in the header does not introduce a significant additional communication overhead.

The biggest advantage of a RW is that it does not require multiple-hop routing. In addition, the RW implementation does not assume anything about the properties of the underlaying network graph (aside from being connected). RW simply proceeds until a required number of distinct nodes has been encountered. However, RW can generally revisit the same nodes more than once and may even include loops. In order to estimate the efficiency of the RW based method, we must estimate the average number of steps that takes a RW to visit i different nodes.

The number of steps required for a simple RW to visit i different nodes is called the *Partial Cover Time* and is denoted by $PCT_G(i)$ (for a given graph G). Formally, for $v \in V$, let $PCT_v(i)$ be the expected number of steps needed for a simple random walk starting at v to visit i distinct nodes in G, and the partial cover time of G is $PCT_G(i) = \max_v PCT_v(i)$. The *cover time*, C_G, of G is the expected time to visit all nodes in G, i.e., $C_G = \max_v PCT_G(n)$.

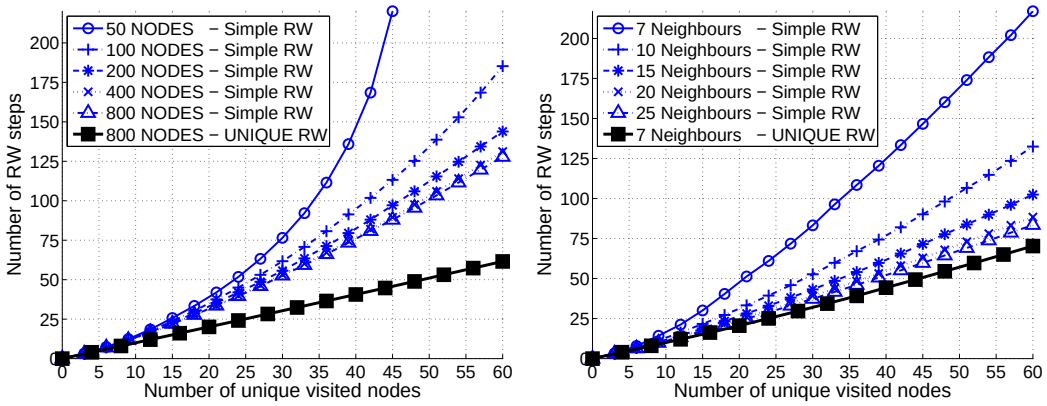
PCT of Random Geometric Graphs. It is already known that for Random Geometric Graphs $\mathcal{G}^2(n, r)$ and $0 \leq c < 1$, $PCT_{\mathcal{G}}(cn) \leq O(n)$ [6] and the full cover time ($c = 1$) is $PCT_{\mathcal{G}}(n) = O(n \log n)$ [7]. We now extend the result of [6] to achieve a tighter bound on covering a sub-linear number of nodes.

The below theorem establishes a novel result about partial cover time of random geometric graphs: covering $t = o(n)$ different nodes in $\mathcal{G}^2(n, r)$ is linear in t.

Theorem 4.1. *Let $t = o(n)$. For $c > 1$, if $r^2 \geq \frac{c8 \log n}{n}$, then* w.h.p. *for $\mathcal{G}^2(n, r)$*

$$PCT_{\mathcal{G}}(t) \leq 2\alpha t$$

where, for large enough n, α is a constant not depending on n.



(a) Varying network size, average 10 neighbors

(b) Varying network density, $N = 400$

Figure 3: RW Partial Cover Time - Number of RW steps required to visit unique nodes. UNIQUE-RW almost never revisits nodes (for a small number of visited nodes).

Proof idea. The detailed proof is deferred to the Appendix. It is based on bounding the expected number of visits to each node during the walk of length $t = o(n)$ by a constant α not dependant on n and then showing that the number of distinct nodes visited by the RW of this length is at least $t/(2\alpha)$.

Theorem 4.1 establishes an asymptotic result, without specifying the exact constants. Therefore, we have performed an empirical study of the partial cover time. Figure 3 depicts simulation results for the number of different nodes visited by a single RW (and *UNIQUE-PATH* RW explained below), for different network sizes, densities and quorum sizes. In these runs, performed on the same setting as reported in Section 2.4, we run a RW until it visits a fixed number of different nodes and count the number of required steps (averaging across multiple RWs and runs). According to Figure 3(a), for example, with $n = 800$, in order to visit $\sqrt{800} = 28$ nodes, a simple RW has a length of 45, thus $PCT_{\mathcal{G}}(\sqrt{n}) = 1.6\sqrt{n}$. For smaller networks, the same constant of 1.6 is enough: $PCT_{\mathcal{G}}(\sqrt{n}) \leq 1.6\sqrt{n}$ for any $n \leq 800$. Actually, $PCT_{\mathcal{G}}(i)$ for even larger values of i is still linear in i. For example, for $n = 100$, $PCT_{\mathcal{G}}(50) = 127$, thus $PCT_{\mathcal{G}}(n/2) \approx 1.3n$.

Figure 3(b) depicts the influence of a network density on $PCT_{\mathcal{G}}$. Intuitively, in sparse networks RW is expected to revisit the same nodes more often, since it has less choices at every step. On the other hand, a dense network, in which every node has a very large amount of neighbors, may resemble a complete graph. A complete graph is known to have a very small PCT (for example, $PCT_{complete}(n/2) = \ln(2)n \approx 0.69n$ on a complete graph, which can be shown by a simple balls and bins argument, [56]). Note that the sparsest network that is still connected has 7 neighbors on average. Less than 7 neighbors resulted in disconnected networks with multiple partitions in almost all our simulations, as also predicted by [26]. Even in this network, $PCT_{\mathcal{G}}(\sqrt{n}) = 2.5\sqrt{n}$, for $n = 400$.

When implementing lookup operations, *PATH* quorums have the following advantage: whenever the searched data has indeed been published, the RW is likely to find the advertisement before visiting all $|Q|$ nodes. In this case, called *early halting*, a reply can be returned immediately and the RW is stopped. This reduces the communication overhead of lookups. As we show in Section 8, early halting usually halves the length of the RW.

4.3 *UNIQUE-PATH*

An optimization of the *PATH* strategy is to perform the RW in a way that avoids visiting the same nodes more than once, also known as a *self-avoiding* RW [49]. That is, at every step, a RW picks at random one of the neighbors of a current node that has not been visited yet and moves to this neighbor. In a rare event that all the neighbors of a current node have been visited by the RW, an arbitrary random neighbor

is chosen (as in a simple RW).

A self-avoiding RW is expected to have a smaller partial cover time than the simple RW, since more different nodes are visited by the same number of steps. We have empirically explored the covering properties of a self-avoiding RW. As can be seen from Figure 3, empirically, indeed *UNIQUE-PATH* almost never revisits a node (at least for $|Q| = O(\sqrt{n})$). As opposite to the simple RW, the self-avoiding RW is almost not affected by the network density. Even at the sparsest network of 7 neighbors, $PCT_{\mathcal{G}}(60) = 70$, for $n = 400$. This comprises the biggest advantage of the *UNIQUE-PATH* strategy.

In addition, note that the *UNIQUE-PATH* strategy incurs the same communication bit-complexity as the simple *PATH*. This is since we need to remember the ids of the visited nodes in the RW message header anyway, to count the number of different visited nodes in order to make sure the correct number of quorum nodes have been accessed.

If the quorum system is to be used for a location service (or any other service that requires quorum nodes to reply to the originating node) the node that stores the location information has to send back a reply, specifying the actual location. If *PATH* or *UNIQUE-PATH* are used, it would be more beneficial to send the reply back on the reverse path of the RW instead of invoking a costly routing to send the reply. Additional RW related optimizations and implementation details are described in Sections 6 and 7.

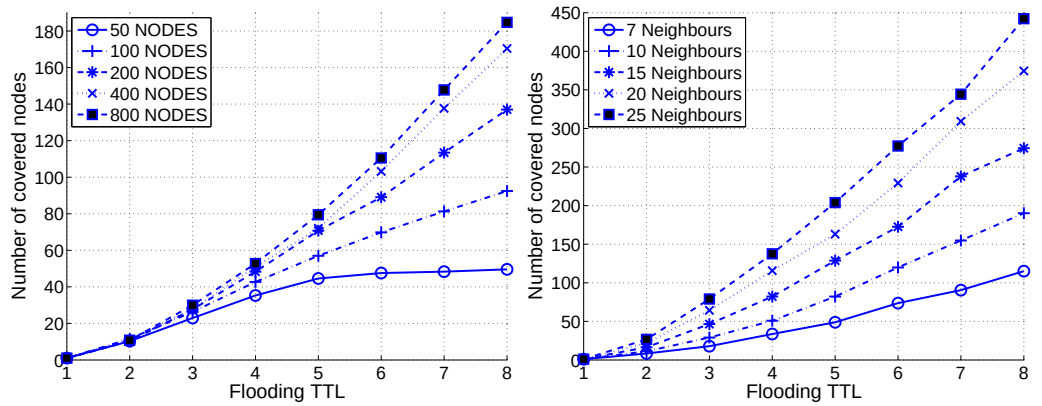
4.4 *FLOODING*

Another way to access a quorum is by flooding. Since in the general case flooding will reach all network nodes, which might not be necessary, a *limited scope flooding* can be used instead. That is, the request is broadcasted from a given node to all its neighbors with a given Time-To-Live (TTL). Each neighbor that receives the request for the first time decreases the TTL by one, and if the result is larger than 0, rebroadcasts the message to its neighbors, etc. All nodes that receive the message are members of the quorum. *FLOODING* can also be used to implement `advertise` quorums, by flooding the whole network and every node picking to take part in the `advertise` quorum with probability $|Q|/n$. To prevent multiple simultaneous broadcast transmissions from colliding, we use a random jitter ([36]) of 10 millisecond, as suggested in [11].

The biggest challenge of using *FLOODING* is how to set the TTL in order to ensure that the message is received by k nodes, for a given k. The amount of nodes covered by flooding directly depends on the network topology and network density. Here we suggest 2 possible implementations. First implementation is based on the explicit assumption that the nodes are uniformly distributed over the network and the average density is known (this holds, e.g., in mobile networks with random movement pattern). In such a case, the value of the TTL can be theoretically approximated.

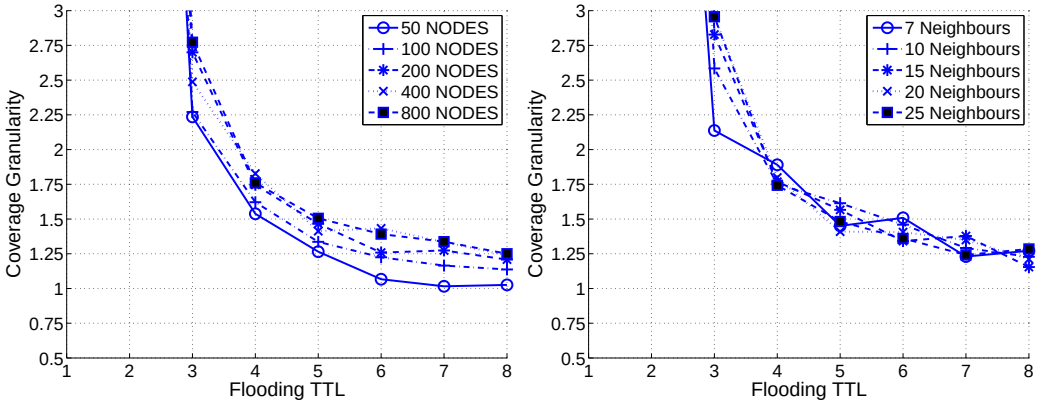
However, if the topology is different or the density is unknown or may change unpredictably, a different implementation strategy, termed *expending ring*, can be used (widely used in the reactive routing protocols in MANET [11]). According to this strategy, the originating node starts a series of floodings with increased TTLs. In every round, the originating node should estimate the amount of accessed nodes and continue until roughly $|Q|$ different nodes have been accessed. The counting can be implemented by nodes sending back acknowledgements. In order to reduce the excessive amount of acks a number of techniques can be used. One such technique combines the acks from different nodes along the ack reverse path. Another technique is for nodes to ack probabilistically and for the originating node to estimate the amount of accessed nodes based on the number of received acks and the reply probability. The expending ring is a robust implementation that ensures that the required number of nodes are accessed on any topology. This robustness however comes at the increased communication cost.

The main drawback of flooding is the inability to have a fine grain control over the exact number of nodes that receive the message (this is a shortcoming of both techniques described above). This is captured by the notion of *coverage granularity* (CG). Coverage granularity measures the difference in the flooding coverage when increasing the TTL by one. That is, $CG(i) = \frac{N_i}{N_{i-1}}$, when N_i is the expected



(a) Flooding Coverage, average 10 neighbors

(b) Flooding Coverage, $N = 800$



(c) Coverage Granularity, average 10 neighbors

(d) Coverage Granularity, $N = 800$

Figure 4: Flooding Coverage for varying network sizes and densities. (a) and (b) Number of nodes covered by flooding as a function of TTL. (c) and (d) Coverage Granularity: $CG(i) = \frac{N_i}{N_{i-1}}$. Flooding has high CG for small (effective) TTLs.

number of nodes that are covered by flooding with TTL i [38]. $N_1 = 1$ (only the originating node). A small CG allows a more adaptive flooding, that adjusts to varying densities and failures, for example by increasing the TTL by one in the expended ring implementation. Figure 4 depicts simulation results of TTL influence on the flooding coverage. We can see the substantial growth in the number of covered nodes, when increasing TTL. $CG(3)$ is always above 2, $CG(4)$ and $CG(5)$ is between 1.75 and 1.25 for different network sizes. When varying network density, CG is approximately the same for all densities and is around 1.75 for TTL 4. Such a big flooding coverage granularity impedes its efficient use.

An additional disadvantage of *FLOODING* is that it does not posses the *early halting* property - there is no way to stop the flooding from expanding, before the TTL expires. Another drawback for location services is the numerous number of replies that will be sent back to the starting node, which also increases the communication cost. Last but not least, broadcasting in wireless networks has a number of network level disadvantages. For example, in 802.11 WiFi networks [35] broadcasting is less reliable since nodes do not acknowledge broadcast messages; broadcasts are usually sent at the lowest possible rate of 1 or 2 Mbps since low rates enable a better signal capture, which is especially important due to the lack of acks; broadcast is less energy efficient than sending point-to-point messages: the 802.11 MAC PSM protocol can put nodes to sleep, which can significantly save energy. However, PSM is disabled when nodes communicate by broadcasts.

Advertise	RANDOM				FLOODING		PATH
Lookup	RANDOM	RANDOM-OPT	PATH	FLOODING	PATH	FLOODING	PATH
Advertise Cost	$\frac{n}{\sqrt{\ln(n)}}$				Combined Cost	Combined Cost	Combined Cost Lower bound*: $\frac{n}{\ln(n)}$
Lookup Cost	$\frac{n}{\sqrt{\ln(n)}}$	$\sqrt{n\ln(n)}$	$\sqrt{n}$	$\sqrt{n}$	n	n	Simulations: n

* lower bound is based on the crossing time

Figure 5: Asymptotic comparison of combinations of different access strategies for a target quorums size $|Q| = \Theta(\sqrt{n})$, for Random Geometric Graphs.

4.5 *RANDOM-OPT* Access Strategy

Yet another way to access a quorum is by optimizing the *RANDOM* strategy. As discussed above, *RANDOM* either utilizes routing or sampling. In both cases, *RANDOM* strategy sends messages that pass through intermediate nodes, without making an efficient use of these nodes. On the other hand, *RANDOM-OPT* strategy adds these intermediate nodes to the quorum.

Whenever a quorum access message passes through an intermediate node p, the networking layer of this node can pass this message to the location service. The location service will either perform a local lookup in case it is a `lookup` access or store the advertisement in case of an `advertise` access. In the former case, if the data is found at node p, p can respond immediately to the originator and instruct its own networking layer not to forward the lookup request any further.

The benefit of this approach is that on average, the same quorum size can be accessed by explicitly contacting much fewer nodes. Since the average length of a random route in the network is $\sqrt{\frac{n}{\ln n}}$, we can issue much fewer routing requests. However, those routes may not necessarily pass in different nodes. As we show by simulation in Section 8, when using the *RANDOM-OPT* access strategy for `lookup`, we only need to invoke the lookup request to $O(\ln n)$ random nodes instead of $O(\sqrt{n})$ nodes.

Note however, that in contrast with the *RANDOM* strategy, *RANDOM-OPT* is not guaranteed to access randomly chosen nodes and thus it cannot simply substitute *RANDOM.*

5 Implementing Probabilistic Quorum Systems

As mentioned before, we can use any of the access strategies described in Section 4 to implement any of the `advertise` and `lookup` quorums and we can mix and match them.

Below, we present a formal evaluation of quorum systems obtained by several of these combinations. Figure 5 summarizes the various combinations of different access strategies for $|Q| = \Theta(\sqrt{n})$.

5.1 Symmetric Combination with Two *RANDOM* Quorums

`advertise` *RANDOM,* `lookup` *RANDOM.* This is the method of Malkhi et al. [50]. Lemma 3.4 from [50] states:

Lemma 5.1. *Let Q_a and Q_ℓ be quorums of size $k\sqrt{n}$ each chosen uniformly at random. Then the non-intersection probability is $Pr(Q_a \cap Q_\ell = \varnothing) < e^{-k^2}$.*

Loosely speaking, quorums of size $\Theta(\sqrt{n})$ ensure intersection with good probability. It can also be easily shown (for example, by using the same argument as we do below in Lemma 5.2) that if Q_a and Q_ℓ are quorums of sizes $|Q_a|$ and $|Q_\ell|$ accordingly, each chosen uniformly at random, then $Pr(Q_a \cap Q_\ell = \varnothing) < e^{-\frac{|Q_a||Q_\ell|}{n}}$.

As we have shown above in Section 4, for $|Q| = \Theta(\sqrt{n})$ the cost of invoking a *RANDOM* access in ad hoc network is $\frac{n}{\sqrt{\ln n}}$ plus the cost of establishing multi-hop routes in case a membership service is used or $\Theta(n\sqrt{n})$ when sampling directly with RWs.

5.2 Asymmetric Combinations with One *RANDOM* Quorum

Below we provide a proof of the main result of our paper: to build probabilistic bi-quorums it is sufficient that only one of the quorums is *RANDOM*, while the other quorum can be picked in an arbitrary (non adversarial) way.

Lemma 5.2. ***(Mix and Match Lemma)*** *Denote Q_a an* `advertise` *quorum and Q_ℓ a* `lookup` *quorum. The probability that $Q_a \cap Q_\ell$ is empty is:* $\Pr(Q_a \cap Q_\ell = \varnothing) \leq e^{-\frac{|Q_a||Q_\ell|}{n}}$.

Proof. Since the selection of a lookup quorum is performed independently of the selection of an advertise quorum, we claim that the order of the selection does not influence the intersection probability between the two quorums. We can therefore look at the quorum selection process as if the lookup quorum Q_ℓ was selected first.

Thus, suppose that a subgroup Q_ℓ of different nodes is picked out of a network of n nodes (note that the selection may be arbitrary, not necessarily uniform). Next, another subgroup Q_a of nodes is picked, while each node in Q_a is picked uniformly at random out of the network, but without repetitions. For the intersection of Q_ℓ and Q_a to be empty, we can look at this process as if the first node of Q_a is picked out of $n - |Q_\ell|$ nodes uniformly at random, the next node is picked out of $n - |Q_\ell| - 1$ nodes, etc. Hence, we have:

$$\Pr(Q_a \cap Q_\ell = \varnothing) = \prod_{i=0}^{|Q_a|} \frac{n - |Q_\ell| - i}{n - i} \leq \prod^{|Q_a|} \frac{n - |Q_\ell|}{n} = \left(1 - \frac{|Q_\ell|}{n}\right)^{|Q_a|} \leq e^{-\frac{|Q_a||Q_\ell|}{n}}$$

□

The above result can be used to set the size of the quorums Q_a and Q_ℓ to guarantee a non-empty intersection of two quorums with a given probability. This is established below:

Corollary 5.3. *In order for two sets, Q_a and Q_ℓ of sizes $|Q_a|$ and $|Q_\ell|$ respectively chosen in the manner described above to have a non empty intersection with probability at least $1-\varepsilon$, the following must hold:* $|Q_a| \cdot |Q_\ell| \geq n \ln(1/\varepsilon)$.

Proof.

$$\Pr(Q_a \cap Q_\ell \neq \varnothing) \geq 1 - e^{-\frac{|Q_a||Q_\ell|}{n}} \geq 1 - \varepsilon \Rightarrow -\frac{|Q_a||Q_\ell|}{n} \leq \ln(\varepsilon) \Rightarrow \frac{|Q_a||Q_\ell|}{n} \geq \ln(1/\varepsilon)$$

□

To get a feel for the result, consider the example in which $1 - \varepsilon = 0.9$. In this case, $|Q_a| \cdot |Q_\ell|$ must be at least $2.3n$. Thus, we can pick both $|Q_a|$ and $|Q_\ell|$ to be $\Theta(\sqrt{n})$.

Summary. Such an asymmetric quorum system, in which only one of the quorums has to be picked randomly while the other one can be picked in an arbitrary (non adversarial) way, has a significant advantage over the pure *RANDOM*x*RANDOM* strategy mix. While the cost of a *RANDOM* access is almost linear in n (plus the cost of establishing multi-hop routes), the second quorum can be accessed by a considerably cheaper access strategy. In addition, a combination in which at least one quorum is *RANDOM* has an important practical advantage. The intersection probability does not depend on the

network topology or density (Lemma 5.2 is correct on any network graph). It means that combinations with *RANDOM* are robust and yet efficient, since the second (non-random) quorum can be picked in a very efficient way.

`advertise` *RANDOM*, `lookup` *PATH*. In this case, the selection of the `advertise` quorum is performed uniformly at random, while the selection of the `lookup` quorum is performed by a RW. As we have shown in Section 4.2, the cost of the *PATH* access strategy is linear in the quorum size for $|Q| = \Theta(\sqrt{n})$. This can be further optimized by using *UNIQUE-PATH*, which does not effect the intersection probability, but can lower the communication cost. We can also switch the `advertise` and `lookup` access strategies: use *RANDOM* for `lookup` and *PATH* for `advertise`. We explore this direction in more details in Section 5.4.

`advertise` *RANDOM*, `lookup` *FLOODING*. Instead of using the *PATH* strategy used for `lookup`, we can use the *FLOODING*. The intersection probability in these cases is similar to the ones obtained with *PATH* `lookup`. This is because a *FLOODING* that covers (at least) k nodes, visits k different nodes, just like a RW. However, in most cases more than k nodes will actually receive the message. Hence, while asymptotically the cost will be the same as *PATH*, the constants for *FLOODING* are higher. In addition, *FLOODING* does not provide early halting and sends multiple replies, which also increases the communication cost and is not energy efficient.

`advertise` *RANDOM*, `lookup` *RANDOM-OPT*. We can also use the *RANDOM-OPT* strategy to access the `lookup` quorum. This is expected to produce the same intersection as by the *RANDOM*x*RANDOM* mix, while sending fewer message.

5.3 Symmetric Combinations without *RANDOM* Quorum

We have further explored combinations that do not use the *RANDOM* access strategy at all.

`advertise` *PATH*, `lookup` *PATH*. Both `advertise` and `lookup` are performed using RWs. This neither requires any sampling or membership services nor routing and thus appears very appealing. However, a deeper look reveals that this combination is less attractive than it seems, since it has a high communication and memory costs. One must ensure these two RWs intersect in at least one node. For that matter we define the *crossing time* of two RWs.

Crossing Time of Two Simple RWs in Random Geometric Graphs. We say that two random walks cross, if there exists at least one node in which these two RWs visit during their run (not necessarily simultaneously). Therefore, crossing time captures the smallest number of steps required for two simple RWs to intersect in at least one node.

Definition 5.4. Given two RWs starting at any two nodes in the network, the *crossing time* (CRT) is the expected first time at which there is a node that was visited by the two RWs.

Formally, given two random walks X_u, Y_v, starting at u and v respectively, let $\gamma_{uv} = \min\{t : \{X_u(0), \ldots, X_u(t)\} \cap \{Y_v(0), \ldots, Y_v(t)\} \neq \emptyset\}$, be the first time that the two walks cross. The crossing time of a graph is defined as $\max_{uv} E[\gamma_{uv}]$. We prove the following:

Theorem 5.5. *The crossing time of two RWs in $\mathcal{G}^2(n, r)$ is $\Omega(r^{-2})$.*

Proof. Divide the unit square of $\mathcal{G}^2(n, r)$ into bins of size $\frac{1}{r} \times \frac{1}{r}$. Each bin is indexed by a column i and a row j where $0 \leq i, j \leq \lfloor \frac{1}{r} \rfloor$. Now look at the projection of the walk on the columns of the bins, when the walk is at column i it can only move to a bin at columns $i-1, i+1$ or to stay at column i. This resembles a simple random walk on a line with some additional probability to stay at the same node (i.e, self-loop).

Take two nodes, u at column 0 and v at column $\lfloor \frac{1}{r} \rfloor$. For the two walks to cross, at least one walk has to reach the column in the middle. This will amount to the expected time it takes to a simple random walk on the line to move from 0 to $\lfloor \frac{1}{2r} \rfloor$, which is known to be $\Omega(r^{-2})$ [43]. □

Note that when $r = \Theta(\sqrt{\frac{\log n}{n}})$ (the minimal radius that ensures connectivity), the crossing time of $\mathcal{G}^2(n, r)$ is at least $\Omega(\frac{n}{\log n})$. Thus, if both `advertise` and `lookup` quorums are accessed by the *PATH* strategy, at least one of them will incur a communication cost which is almost linear in n. Recall that this is a lower bound, and, as a matter of fact, our simulation results in Section 8 indicate that **both** `advertise` and `lookup` need to be of that order, even if we use a unique RW. In addition, there is an increased storage cost. The `advertise` RW has to publish the data in every node it visits and if the length of this RW is $\Omega(n/\log n)$, accessing the `advertise` quorum will incur almost linear storage.

The shortcoming of the *PATH*x*PATH* combination is that the exact value of the crossing time depends on the network density and is thus hard to set without overestimating it or running in danger of not ensuring the desired intersection probability.

`advertise` *UNIQUE-PATH*, `lookup` *UNIQUE-PATH*. Instead of using a Simple RW we can use the Unique RW. As before, the sizes of 2 quorums should be set to guarantee that the two quorums intersect. For example, setting $|Q_a| + |Q_\ell > n$ provides such a guarantee. In such a case our result about partial cover time from Section 4.2 can be used as an estimate of the message complexity of this combination.

`advertise` *FLOODING*, `lookup` *FLOODING*. Here, in the same manner as with RWs, one must ensure that the two sets of nodes accessed by two flooding intersect. It is thus clear that the combined cost of both floodings is linear with n.

5.4 Deriving the Optimal Cost for Asymmetric Combinations

An ability to build quorum systems while using different access strategies and different quorum sizes raises the following interesting question. What is the best strategy mix and what is the best relative size of the `lookup` and `advertise` quorums, that guarantees a given intersection probability with an optimal minimal cost. The answer to this question depends on the usage pattern of `lookup` vs. `advertise`. Intuitively, if `lookup` is used more frequently than `advertise`, one would suggest to optimize the `lookup` access strategy, by using a smaller `lookup` quorum size. The question we raise is how much exactly to optimize?

To this end, we consider the total cost of accessing all quorums. This total cost depends on the size of each quorum, the cost of accessing a single node in this quorum and the relative ratio of requests to access `lookup` vs. `advertise`. More formally, we define the function $Cost$ to be the cost of accessing a node or a set of nodes. The cost function could reflect the number of messages or any other measure. $Cost(Q_a)$ is the cost of accessing an `advertise` quorum and $Cost_a = \frac{Cost(Q_a)}{|Q_a|}$ is the average cost of accessing a single node in the `advertise` quorum. $Cost(Q_\ell)$ and $Cost_\ell$ are defined similarly for the `lookup` quorum.

We assume a parameter τ, which is the network-wide ratio between the number of times `lookup` is being accessed vs. `advertise`.

Lemma 5.6. *The optimal ratio between the sizes of* `lookup` *and* `advertise` *quorums, which minimizes the total cost of accessing all* `lookup` *and* `advertise` *quorums, is:*

$$\frac{|Q_\ell|}{|Q_a|} = \frac{1}{\tau}\frac{Cost_a}{Cost_\ell}$$

Proof. Define $\#\texttt{advertise}$ to be the total number of advertisements issued by all nodes during the considered period of time and by $\#\texttt{lookup}$ the total number of lookups. The total cost is:

$$TotalCost = \#\texttt{advertise} \cdot |Q_a| \cdot Cost_a + \#\texttt{lookup} \cdot |Q_\ell| \cdot Cost_\ell$$

We are subject to the constrain that $|Q_a| \cdot |Q_\ell| = n \ln(1/\varepsilon)$ and are given $\tau = \frac{\#\texttt{lookup}}{\#\texttt{advertise}}$. Therefore,

$$TotalCost = \frac{\#\texttt{lookup}}{\tau} \cdot \frac{n \ln(1/\varepsilon)}{|Q_\ell|} \cdot Cost_a + \#\texttt{lookup} \cdot |Q_\ell| \cdot Cost_\ell$$

The minimal $TotalCost$ is achieved when $|Q_\ell| = \sqrt{\frac{n \ln(1/\varepsilon) Cost_a}{\tau Cost_\ell}}$ (obtained by derivation of $Total-Cost$). Thus, $\frac{|Q_\ell|}{|Q_a|} = \frac{|Q_\ell|^2}{n \ln(1/\varepsilon)} = \frac{1}{\tau}\frac{Cost_a}{Cost_\ell}$.

□

Consider an example in which $\tau = 10$ (there are 10 times more lookups than advertisements). Let `advertise` quorum be accessed by a *RANDOM* strategy and `lookup` by a *UNIQUE-PATH* strategy. $Cost_a = D$ (the diameter of the network) and $Cost_\ell \approx 1$ (as we have shown in 4.2). For a network with $D = 5$, we must pick $\frac{|Q_\ell|}{|Q_a|} = \frac{5}{10} = 1/2$. Thus, to yield the minimal total access cost, the size of the `advertise` quorum should be twice the size of the `lookup` quorum.

The parameter τ is a global network parameter. It might be known a-priory to all nodes or configured based on common usage patterns (such as the distribution of advertisement vs. lookups in file sharing P2P applications). In case τ is not known and cannot be assumed, it can be dynamically estimated based on the usage statistics.

6 Maintenance - Handling Failures, Dynamism, and Mobility

Probabilistic constructions are inherently very suitable to handle dynamic environments, such as networks with frequent nodes joins and failures or mobility. This is since they do not rely on strict determinist sets of nodes, which are costly to update and reconfigure. In this section we describe some of the formal properties of probabilistic quorums w.r.t the dynamism, as well as provide additional implementation details to improve dynamism handling even further.

6.1 Handling Churn

Churn is caused by frequent joins and leaves/failures of nodes. As mentioned in Section 3, the resilience of the quorum system to failures is measured by fault tolerance and failure probability. The fault tolerance of a probabilistic quorum system with quorum sizes of $\sqrt{n}$ is $\Omega(n)$. Similarly, the failure probability of a probabilistic quorum system of size $k\sqrt{n}$ is $e^{-\Omega(n)}$ for all $p \leq 1 - \frac{k}{\sqrt{n}}$. However, in ad hoc networks, we must also require that the network remains connected. We discuss the necessary condition for connectivity below.

Connectivity in Face of Failures. The connectivity of $\mathcal{G}^2(n, r)$ 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 [26] 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 [57], the authors obtain a similar result when nodes are distributed in the unit square $[0, 1]^2$. The above result implies that if the transmission range r is set such that $r = \sqrt{\frac{C \ln n}{\pi n}}$ for $C > 1$, the network is connected *w.h.p.*. Such value of r implies an average number of neighbors, d_{avg}, which is $d_{\text{avg}} = \pi r^2 n = C \ln n$.

With failures, one would like to know how many failures leave the network connected. We look at a network with fixed r and assume a failure model in which individual nodes crash independently with fixed probability. In such a model, after i nodes fail, the remaining network forms a Random Geometric Graph, $\mathcal{G}^2(n-i,r)$. This network remains connected if $n-i$ satisfies the necessary connectivity condition, namely, $r \geq \sqrt{\frac{\ln(n-i)}{\pi(n-i)}}$. For example, in the network of 1000 nodes the minimal d_{avg} that guarantees connectivity is 7. Thus, if the initial density is $d_{\text{avg}} = 14$, this network can withstand a failure of up to half of the nodes in the network.

Degradation Rate. Degradation rate captures the probability that two quorums accessed at different times will intersect despite the fact that between these two accesses f nodes have crashed or joined. We analyze the degradation rate as a function of the percentage of network change (percentage of crashed or joined nodes). We calculate the probability of an intersection of a `lookup` quorum with a previously established `advertise` quorum. Denote by $Q_a(t)$ the live nodes of a given `advertise` quorum at time t. $Q_a(0)$ is the initial `advertise` quorum at the time it was established (before the churn started) that guarantees intersection with at least $1-\varepsilon$ probability. $Q_\ell(t)$ is a `lookup` quorum at the moment the access is being issued (`lookup` quorum accesses only live nodes at time t, so they are not affected by failures). $n(t)$ is the network size at time t (at the time of a `lookup` access). We denote the non-intersection probability at time t by $\Pr(miss(t))$. By definition, $\Pr(miss(0)) \leq \varepsilon$.

$$\Pr(miss(t)) = \Pr(Q_a(t) \cap Q_\ell(t) = \varnothing) \leq e^{-\frac{|Q_a(t)||Q_\ell(t)|}{n(t)}}$$

We separate our discussion into cases of failures only, joins only, and both. Note that when nodes fail or join, $n(t)$ changes as well. We can furthermore separate the discussion into 2 additional categories: whether the size of the `lookup` quorum, $|Q_\ell(t)|$, is adjusted to $n(t)$ or not. Practically, this means that the data location service that uses quorums can periodically estimate the network size $n(t)$ and adjust $|Q_\ell(t)|$ dynamically to its size (e.g., $|Q_\ell(t)| = C\sqrt{n(t)}$), or keep $|Q_\ell(t)|$ constant ($|Q_\ell(t)| = |Q_\ell(0)|$) until the next time the whole quorum system is refreshed (as discussed below).

1. *Failures only.* We assume nodes crash independently, with some fixed probability. In such a case $n(t) = (1-f)n(0)$ and $|Q_a(t)| = (1-f)|Q_a(0)|$, while f is the fraction of failed nodes. Thus,

$$\Pr(miss(t)) \leq e^{-\frac{|Q_a(t)||Q_\ell(t)|}{n(t)}} = e^{-\frac{|Q_a(0)|(1-f)|Q_\ell(t)|}{(1-f)n(0)}} = e^{-\frac{|Q_a(0)||Q_\ell(t)|}{n(0)}}$$

   (a) If $|Q_\ell(t)| = |Q_\ell(0)|$ ($|Q_\ell(t)|$ is kept constant and is not adjusted to $n(t)$), then $\Pr(miss(t)) = \Pr(miss(0))$ (it does not change)! This is despite the fact that a fraction f of the network, including the advertisement nodes, has failed.

   (b) If $|Q_\ell(t)|$ is adjusted to $n(t)$, e.g., $|Q_\ell(t)| = C\sqrt{n(t)}$ then:

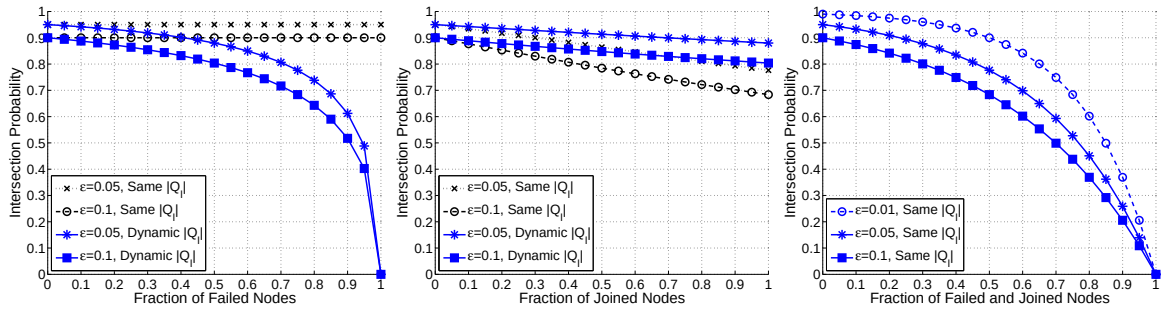
$$\Pr(miss(t)) \leq e^{-\frac{|Q_a(0)||Q_\ell(t)|}{n(0)}} = e^{-\frac{|Q_a(0)|C\sqrt{n(t)}}{n(0)}} = e^{-\frac{|Q_a(0)|C\sqrt{(1-f)n(0)}}{n(0)}} = e^{-\frac{|Q_a(0)|\sqrt{(1-f)}|Q_\ell(0)|}{n(0)}} \leq \varepsilon^{\sqrt{1-f}}$$

2. *Joins only.* In such a case $n(t) = (1+f)n(0)$ and $|Q_a(t)| = |Q_a(0)|$, while f is the fraction of joined nodes.

$$\Pr(miss(t)) \leq e^{-\frac{|Q_a(t)||Q_\ell(t)|}{n(t)}} = e^{-\frac{|Q_a(0)||Q_\ell(t)|}{(1+f)n(0)}}$$

   (a) If $|Q_\ell(t)| = |Q_\ell(0)|$, then: $\Pr(miss(t)) \leq e^{-\frac{|Q_a(0)||Q_\ell(t)|}{(1+f)n(0)}} = e^{-\frac{|Q_a(0)||Q_\ell(0)|}{(1+f)n(0)}} \leq \varepsilon^{\frac{1}{1+f}}$

   (b) If $|Q_\ell(t)| = C\sqrt{n(t)}$, then: $\Pr(miss(t)) \leq e^{-\frac{|Q_a(0)|C\sqrt{n(t)}}{(1+f)n(0)}} = e^{-\frac{|Q_a(0)|C\sqrt{(1+f)n(0)}}{(1+f)n(0)}} \leq \varepsilon^{\frac{1}{\sqrt{1+f}}}$



(a) Failures only (b) Joins only (c) Failures and Joins

Figure 6: The degradation of the intersection probability as a function of the f - the fraction of crashed and joined nodes. ε is the initial non-intersection probability.

3. *Both Joins and Failures.* If the same amount of nodes that have failed have also joined, then $n(t) = n(0)$, $|Q_a(t)| = (1-f)|Q_a(0)|$, and $|Q_\ell(t)| = |Q_\ell(0)|$.

$$\Pr(miss(t)) \leq e^{-\frac{|Q_a(t)||Q_\ell(t)|}{n(t)}} = e^{-\frac{(1-f)|Q_a(0)||Q_\ell(0)|}{n(0)}} \leq \varepsilon^{1-f}$$

Figure 6 illustrates the evolution of the intersection probability as a function of the churn rate for all cases. We can see that if we keep using the same `lookup` quorum size while nodes fail, the intersection probability remains the same. This result indicates a remarkable resilience of probabilistic bi-quorum systems to failures. If there are no new joins, then the intersection probability stays the same despite failures! In the case of joins, a bi-quorum system can withstand a churn of a linear fraction of nodes without a significant intersection deterioration. For example, when starting with an initial intersection probability of 0.95, after 30% of the network changed (30% of the nodes have failed and new ones have joined), the intersection probability deteriorates to only slightly below 0.9.

Handling Quorums Degradation. In the case of deterministic quorums, recovering from quorums degradation requires both reconfiguring the quorum system (finding a new quorum set that will replace the previous one) and refreshing its contents, in order to ensure data continuity, as been done in [48] and [2]. In the case of probabilistic quorums, there is no need to reconfigure the system after failures in order to ensure quorum liveness. All that is needed is to refresh the quorum system, e.g., by re-advertising every data item to ensure data continuity. The frequency of this re-advertising is determined by the degradation rate. Consider the example depicted in Figure 6(c). Suppose the minimum accepted intersection probability of a given system is 0.9, the intersection probability before the churn started was 0.95, and the time it takes 30% of the nodes to change is one day. Then in this example, every data item should be refreshed once a day.

6.2 Handling Mobility - Network and Application Adaptation

In ad hoc networks, our construction should also handle mobility. If mobility maintains the uniform distribution of nodes, then it does not impact the intersection probability. Thus, in these cases, nothing is needed to be done. However, if mobility substantially skews the structure of the network, then refreshing by re-advertising is required. The rate of refreshing in such cases depends on the exact mobility model.

Another important aspect of mobility is network level adaptation. Consider for example a RW implementation. The next hop along the RW is picked randomly out of node's direct neighbors, while the neighbors list is constructed by a heartbeat like mechanism [33]. However, due to mobility, the neighbors set can change rapidly, which might result in attempting to forward the RW to a non-neighbor

or even a failed node. Another example is *RANDOM* quorum with membership implementation. The accessed node can leave the network or simply fail. However, this might not be immediately indicated by the membership service. Those examples demonstrate a need for: (1) reliable indication of failures to access nodes; (2) an ability to dynamically adapt for this failures and fix them.

Network Level Notifications. Indicating failures in accessing quorums can be achieved for example by applying end-to-end reliability [60] design: the accessed node will send an ack to the originator. This however has a significant cost of doubling the traffic and may also introduce some unnecessary latency. Instead, we suggest to use low level network notifications, in a cross layer design. For example, if the MAC protocol fails to receive an ack for a unicast transmission (after multiple attempts, default 7 in 802.11 MAC [35]), instead of simply dropping the packet, it can signal a higher layer about this failure. This notification should be propagated in the networking stack all the way to the application, allowing it to act accordingly. Another example is a failure of the routing protocol to establish a routing path. This will happen if the routed-to node has left the network. We have also observed such behaviors with very fast mobility, so this problem is real. In such a case, instead of silently dropping the message, routing will notify a higher layer, which will propagate this notification to the application.

Application Adaptation to Network Failures. When the application is notified about a failure of a specific message, it should employ some adaptation mechanism. Simply re-sending the same message again to the same destination is a bad choice, since it will fail to arrive again with high probability, consuming valuable network resources. Instead, in the case of *RANDOM* quorum we could pick another random node to access. In the case of RWs, we can use the *RW salvation* technique of [9] to prevent dropping of RW messages. If node v does not succeed to forward a RW message to the neighbor chosen in a given step (did not receive a MAC level ack), v makes a new attempt to send this message to another random neighbor within the same step. This is especially useful in mobile networks, which experience frequent breakages of neighborhood connections. We demonstrate the usefulness of this technique in Section 8.

Another example is the reply message of the RW; according to the *PATH* strategy, the RW stops at the first intersecting node and then sends the reply back to the looking node, following the reverse RW path recorded in the RW message. When following the reverse path in mobile network, there is a non-negligible chance that the reply messages will be dropped by at least one hop. Such dropping becomes more intense with growing RW length and with increased mobility. We thus suggest to use the following *reply path local repair* technique to combat fast mobility. When node v wishes to pass the reply message to node u, which is the next hop along the reverse path, it first sends a direct unicast MAC message to u without relying on routing. If v's MAC indicates a failure to send this message (v did not receive a MAC level ack from u, probably since u moved out of v's range), v gives up on sending the message to u and tries to pass it to the next node along the path (node w). Since w is not necessarily v's neighbor, v sends this message with the help of routing. However, we do not want to use routing too aggressively. A naïve use of routing might cause a network wide flooding, which will happen if the routing algorithm will search the path to w and w has also moved far from v. To prevent this costly effect, we limit the routing scope with TTL 3. Thus, routing will not search the path more than 3 hops away from v. The value of 3 was chosen as a good tradeoff between the amount of exceeding traffic generated by routing search packets and the probability of receiving the reply. In the case routing fails to find the path to w (which is indicated at v by a routing level notification), v attempts the next hop along the reverse path in the same manner, also using TTL 3. If w is the last hop along the path, and routing with TTL 3 failed to find it, then v has no choice but to invoke routing to w with a large TTL. Another option in this case is for v to drop this reply message. As we explore in Section 8, in relatively low mobility scenarios, the local repairs are sufficient to fix temporal breakages in the reverse path. However, in a very fast mobility scenario of 20m/s (a VANET scenario), local repairs alone are not enough and occasional global routing is inevitable.

6.3 Network Size Estimation

In order to calculate the quorum size in all our access strategies, the number of nodes in the network n must be known. There are several methods for obtaining a loose upper bound on the network size, e.g., [19]. Notice that overestimating the network size will not hurt the intersection probability and will only incur additional communication cost. Once we have such a loose upper bound, a technique in which the nodes count the number of collisions between RWs and estimate n in the manner similar to the birthday paradox principle, can be applied. This technique was previously suggested in [9].

7 Additional Optimizations

7.1 Service Dependent Optimizations

Typically in data location services, the mapping of an object to its home node remains valid for a long time. Thus, caching of advertisement requests or lookup replies that pass through nodes can significantly reduce the lookup overhead. Here we distinguish between the node who is part of the `advertise` quorum, which we call an *owner* of the mapping, and the other nodes that happen to cache it, which we call *bystanders*. Once a node runs low on memory, it can forget all entries for which it is a bystander, but it is supposed to maintain the entries for which it serves as an owner.

With this optimization, and especially when using the *PATH* strategy, lookup requests for popular data items can terminate much faster and also have a higher chance of success.

7.2 Random Walks Optimizations

We propose two additional optimizations for RW based techniques. The first one is called *path reduction*. It is applicable for *PATH* or *UNIQUE-PATH* `lookup` quorums and is used for reply messages. Whenever a lookup RW hits an `advertise` quorum, a reply is sent over the reverse path of the RW. Whenever a reply message arrives at some node v and its next hop in the reverse path is u, v checks if any of its neighbors w appears on the reverse path further after u. In the affirmative, v sends the reply message directly to w skipping u. This optimization reduces the reply length, as demonstrated by simulations.

The second optimization utilizes the broadcasting nature of ad hoc networks. Nodes can overhear messages, e.g., by switching their MAC to a promiscuous mode. If a node u that hears a RW `lookup` request passing through one of its neighbors v is part of the matching `advertise` quorum, u can send a reply immediately to v, which will stop the RW and send the reply back to the `lookup` originator. Thus, the number of nodes covered by a RW is significantly increased. Exploring the benefits of this technique is left for future work.

8 Simulations

We have compared our different implementation strategies by simulations under a wide range of varying conditions. In particular, we have considered the impact of the number of nodes and nodes density, mobility, churn, quorum strategy mixes and sizes. Simulation setup is described in Section 2.4.

Simulation scenarios. Each simulation comprised of two parts. In the first part, a total of 100 advertisements were performed by random nodes, each by *RANDOM* access to a quorum of size $2\sqrt{n}$ (except for *UNIQUE-PATH* `advertise` in Section 8.5). *RANDOM* access was based on a membership service. In the second part 1000 lookups were performed (by 25 random nodes, each making 40 lookups). `lookup` quorum was accessed by 4 different methods: *RANDOM*, *RANDOM-OPT*, *UNIQUE-PATH* and *FLOODING*. On a hit, a node sends a reply to the node that originated the lookup request. In

case of *RANDOM* and *RANDOM-OPT* the reply was sent using routing, while in *UNIQUE-PATH* and *FLOODING* it was sent over the reverse path of the lookup message, thus no routing was used at all. In all simulations, the number of messages denotes network layer messages (e.g., one application message sent to a random node that traverses a route of 4 hops is counted as 4 network layer messages). Additional routing overhead means routing layer specific messages, including path establishment and maintenance messages (RREQ, RREP and RERR in AODV). Hit ratio corresponds to the number of successful `lookup` quorum accesses, that intersected with the corresponding `advertise` quorum. Thus, hit ratio corresponds to the intersection probability.

Each simulation lasted for 1,000 seconds (of simulation time) and each data point was generated as an average of 10 runs. Simulations started after a 200 seconds initialization period, which was enough to construct the membership information (in case of *RANDOM* quorums). Every node maintained a membership list of random, uniformly chosen, $2\sqrt{n}$ nodes.

8.1 Cost of *RANDOM* `advertise`

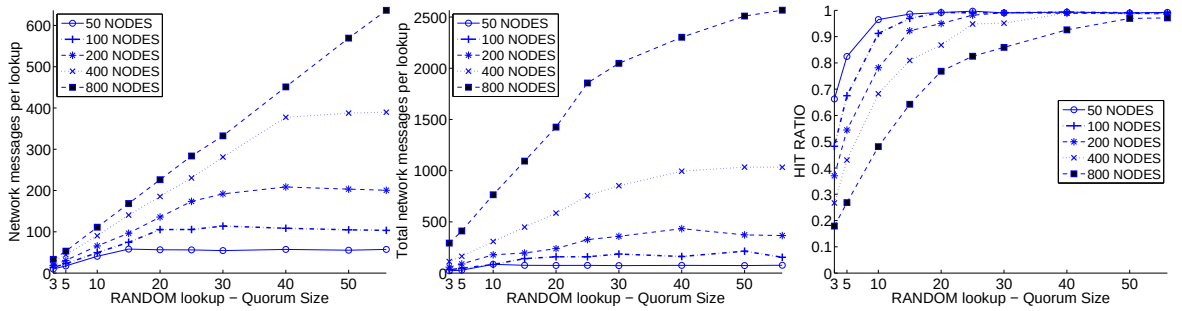
Figure 7 depicts the cost (number of messages) of `advertise` by the *RANDOM* access strategy. This cost does not include the cost of the membership service: we assume this cost is amortized over all `advertise` accesses and is also potentially shared by other applications using the membership service ([9] includes a detailed study of random membership costs). The number of messages per `advertise` request behaves as $\frac{|Q_\ell|\sqrt{n}}{\ln n}$. The number of messages stays constant for $|Q_\ell| \geq 2\sqrt{n}$ for all networks since we use random membership of size $2\sqrt{n}$ and only those nodes were accessed for `advertise`. One can see the dramatic communication overhead increase due to routing. This is primarily due to new routes establishment and route maintenance of the AODV protocol. One has to note that the price of establishing the routes is amortized over different quorum accesses due to routes reuse and its relative part will drop in a longer run. However, in moving networks, in which routes break and need to be reestablished, the price of routing remains a dominant performance factor.

8.2 *RANDOM* `advertise` with *RANDOM* `lookup` and *RANDOM-OPT* `lookup`

Figure 7(c) depicts the hit ratio of *RANDOM* `lookup` access strategy. Hit ratio of 0.9 is achieved when the quorum size is approximately $1.15\sqrt{n}$, just as predicted by the formal analysis in Lemma 5.1. For example, for a network of 800 nodes, quorum size of 33 achieves 0.9 hit ratio. The cost to access 33 random nodes for `lookup` is the same as the cost to access them for `advertise`, and thus can be deduced from Figure 7(a). The `lookup` quorum was accessed in parallel, thus we don't see the potential advantage of the early halting. If we were to access it serially, we would see a two times reduction in the number of accessed lookup nodes, at the cost of increased latency.

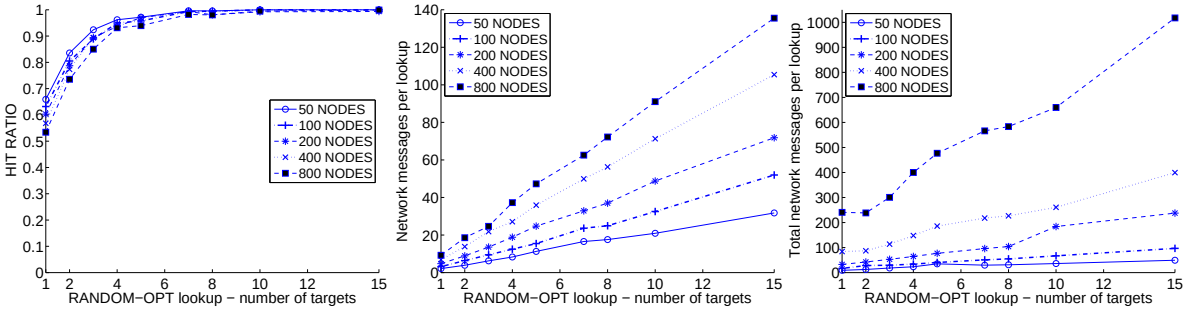
As for the *RANDOM-OPT* strategy in Figure 8, the hit ratio of 0.9 is achieved when starting somewhat between $X = \ln(n)$ and $X = \sqrt{\ln(n)}$ messages to random targets. Due to the cross layer optimization of *RANDOM-OPT*, in which a local lookup is performed in every node through which a message passes, the actual accessed quorum size is $X\sqrt{\frac{n}{\ln n}} \approx \sqrt{n \ln n}$. Thus, this optimization reduces the communication cost significantly compared to *RANDOM*. For example, in a static network of 800 nodes sending 4 `lookup` requests to random nodes achieves a hit ratio of above 0.9 at the cost of less than 40 network messages, which is around $1.4\sqrt{n}$. The routing price of *RANDOM-OPT* is also much less than with *RANDOM*, since it uses fewer multi-hop routes. Still, the additional cost of routing is high, which turns this method inefficient compared to the *UNIQUE-PATH* and *FLOODING* strategies.

In mobile networks, the hit ratio of *RANDOM-OPT* is only slightly smaller than the hit ratio achieved in static networks for the same quorum size (the results for *RANDOM* `lookup` in mobile networks had the same tendency and are thus not depicted due to the lack of space). This is since about 10% of the messages are lost due to mobility, mainly influencing the replies. The number of messages

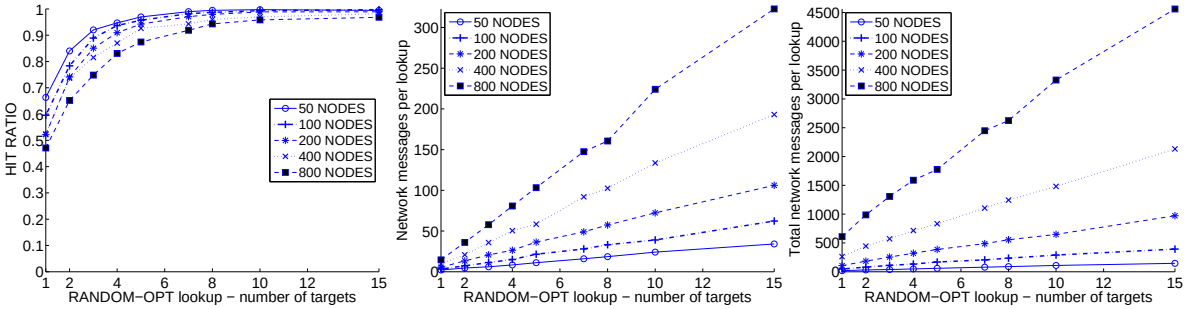


(a) Number of messages per adv. (b) Num. msgs per adv + routing. (c) *RANDOM* `lookup`, Hit ratio.

Figure 7: *RANDOM* `advertise`, Static networks



(a) Hit ratio. Static (b) Num. msgs per lookup. Static (c) Num. msgs per lookup + routing. Static



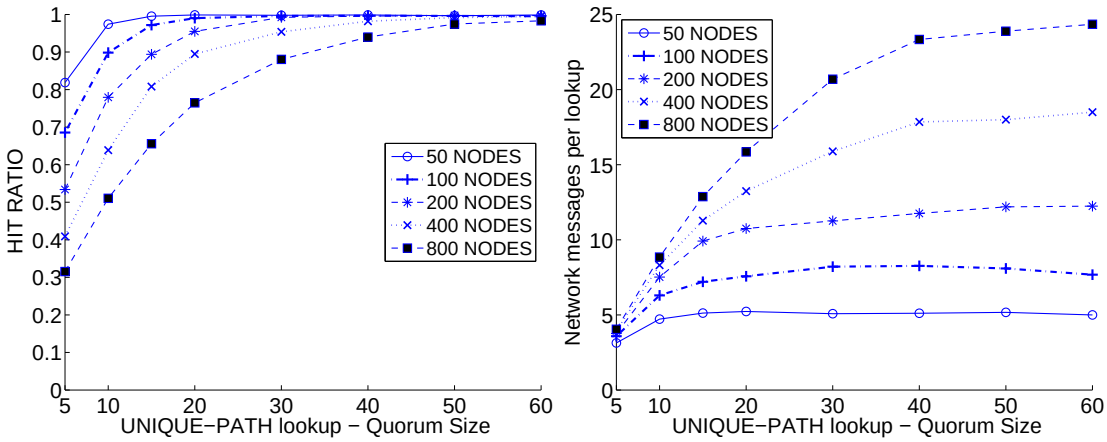
(d) Hit ratio. Mobile (e) Num. msgs per lookup. Mobile (f) Num. msgs per lookup + routing. Mobile

Figure 8: *RANDOM* `advertise`, *RANDOM-OPT* `lookup`, Static and Mobile networks

also increases. Generally speaking, the average path length in mobile networks tends to be longer than in static networks, mainly due to stale neighborhood information used by routing to find routes. When a message follows a wrong path, it takes it longer to finally get to its destination. The routing price in mobile networks also dramatically increases.

8.3 *RANDOM* `advertise` with *UNIQUE-PATH* `lookup`

Figure 9 depicts the performance of the *UNIQUE-PATH* strategy in mobile networks with speeds ranging between 0.5m/s and 2 m/s, which corresponds to walking speed. It performed identically in mobile and static networks and thus we depict only the mobile case here (further mobility impact is discussed in Section 8.6). A hit ratio of 0.9 is achieved when the target quorum size (the number of nodes that need to be covered by RW) is $\sim 1.15\sqrt{n}$, thus validating our analysis in Lemma 5.2 and testifying that a



(a) Hit ratio. Mobile.

(b) Number of messages per lookup. Mobile.

Figure 9: *RANDOM* `advertise`, *UNIQUE-PATH* `lookup`. Mobile Networks, $d_{\text{avg}} = 10$

non random choice of the `lookup` quorum results in the same intersection probability as a random one. The most interesting fact about the *UNIQUE-PATH* strategy is the extremely small number of messages it requires. One would expect that due to the reply message the total number of messages to access a quorum of size $|Q_\ell|$ will be $2|Q_\ell|$. Surprisingly, accessing a target quorum of size $|Q_\ell|$ requires fewer than $|Q_\ell|$ messages, including the reply message!

This happens due to the early halting. When a quorum of target size $|Q_\ell|$ is accessed, the first hit occurs at approximately half of the way. Thus, an average of $|Q_\ell|/2$ messages are sent until this hit. The reply follows the reverse path, but due to the *path reduction* optimization described in Section 7, the reply path length is usually shorter. In addition, the originator of the lookup includes itself in the `lookup` quorum, which further reduces the number of messages by one.

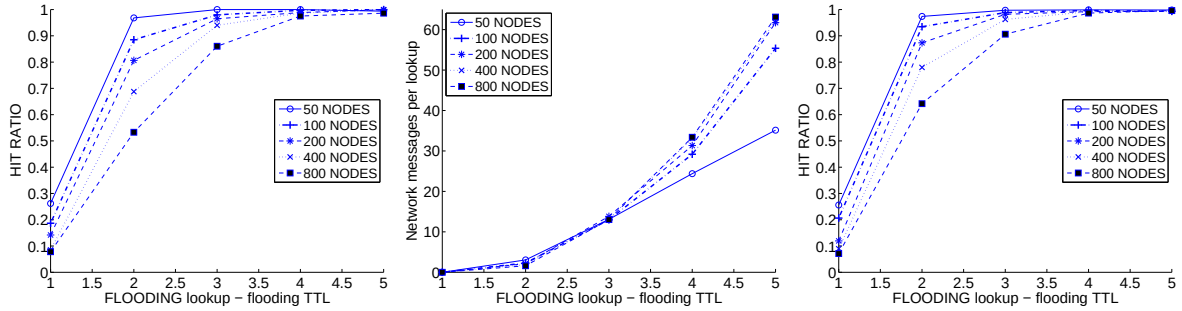
A big advantage of the RW based strategies is that they do not require multi-hop routing and are thus well suited for dynamic mobile networks. This is a significant advantage of the RW strategy, since its performance is not deteriorated by mobility. More details about the mobility impact and the usage of the RW salvation and reply path local repair techniques are discussed in Section 8.6.

Perhaps the biggest advantage of the *RANDOM* x *UNIQUE-PATH* mix is that the same intersection is guaranteed on any network topology and density. All that is needed is to access $|Q_\ell|$ different nodes by the RW and this number does not depend on network characteristics.

8.4 *RANDOM* `advertise` with *FLOODING* `lookup`

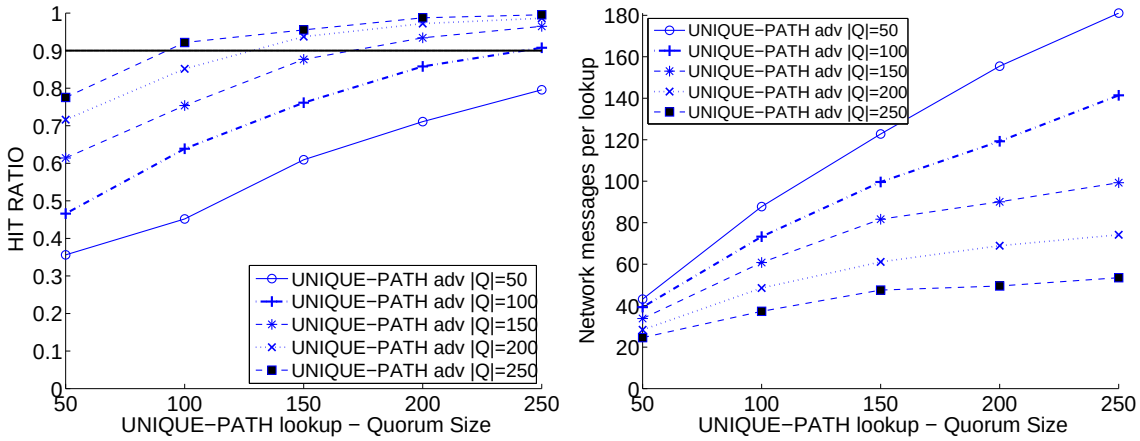
Figure 10 depicts the performance of the *FLOODING* access strategy. The hit ratio grows super linearly with TTL. For example, for 800 nodes, a hit ratio of 0.5 is achieved for TTL=2, whereas a hit ratio of 0.85 is achieved for TTL=3. The number of messages sent by *FLOODING* is quite small and is comparable with the *PATH* strategy. This is mainly due the broadcast nature of flooding: in the last hop of flooding, nodes that do not rebroadcast the flooding message further on can still reply to the `lookup` request if they posses the searched data.

However, notice that in order to increase the hit ratio to 0.9 in a network of 800 nodes one has to increase the TTL to 4, resulting in a significant communication cost increase. Instead of sending 14 messages with TTL 3, *FLOODING* with TTL 4 sends 35 messages (this is both due to the increased flooding scope and additional reply messages). This demonstrates the biggest disadvantage of the *FLOODING* strategy: coarse coverage granularity and the lack of a fine grain control over the intersection probability (as explained in Section 4.4). This is in contrast with the *PATH* strategy, in which one can control the



(a) Hit ratio. Static. (b) Number of messages per lookup. Static. (c) Hit ratio. Mobile.

Figure 10: *RANDOM* `advertise`, *FLOODING* `lookup`. $d_{\text{avg}} = 10$



(a) Hit ratio. Static networks. (b) Number of messages per lookup. Static networks.

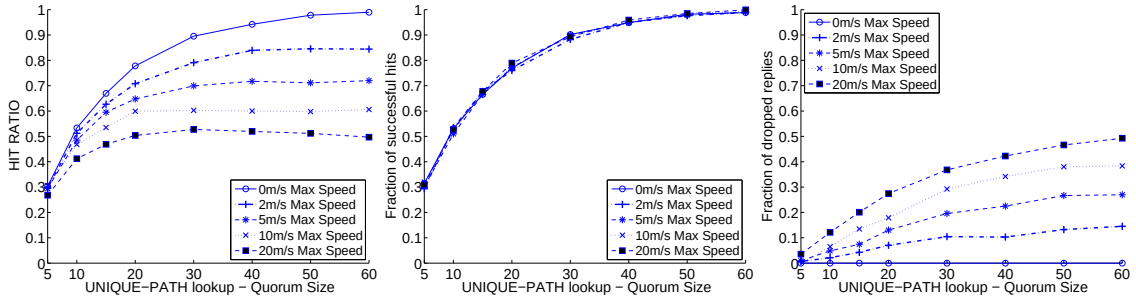
Figure 11: *UNIQUE-PATH* `advertise`, *UNIQUE-PATH* `lookup`, $n = 800$.

intersection probability in a very fine grain manner, by changing the number of nodes that need to be covered by the RW.

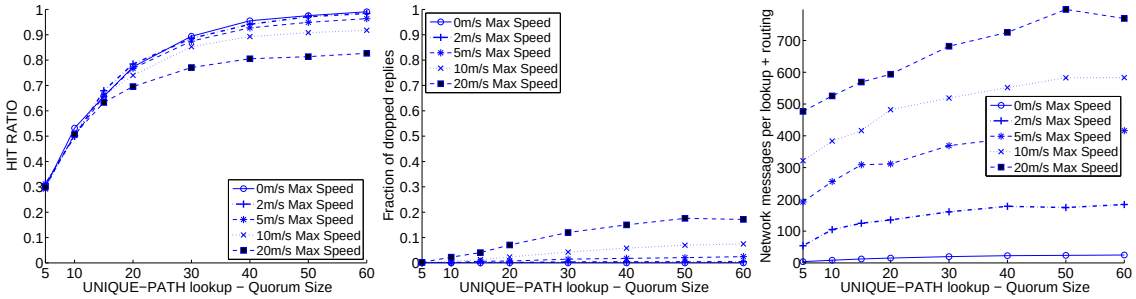
In mobile networks, *FLOODING* performs quite similar to static networks. Surprisingly, it achieves a slightly higher hit ratio for the same TTL as in static networks, while sending more messages. This is due to the mobility model artifact: it is well known that in the Random Waypoint model nodes tend to concentrate at the center of the network [12]. Thus, the average density increases and as a result given the same TTL more nodes are covered and more messages are being sent.

8.5 *UNIQUE-PATH* `advertise` with *UNIQUE-PATH* `lookup`

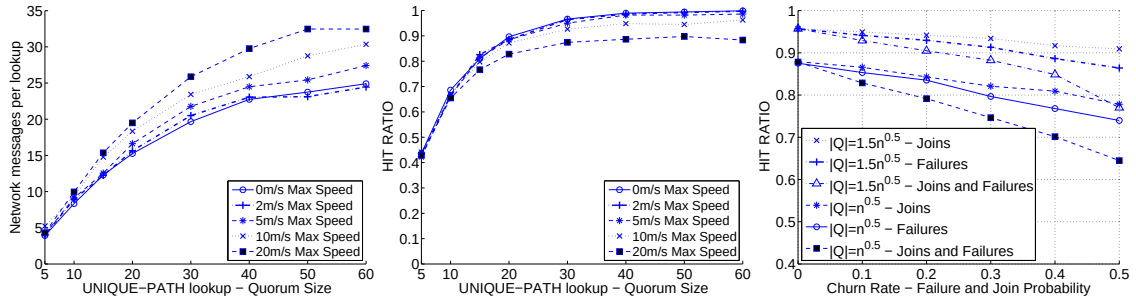
We have also explored the possibility to access the `advertise` quorum with the *UNIQUE-PATH* strategy. As proven in Theorem 5.5, at least one of the RWs has to be of length $\Omega(\frac{n}{\log n})$. Figure 11 depicts the hit ratio for various RW TTL values for a network of 800 nodes. Both `advertise` and `lookup` were accessed by *UNIQUE-PATH*, rather than by *PATH*, which considerably improved their performance. We can see that a 0.9 hit ratio is achieved when the length of `advertise` RW and `lookup` RW together is around 340, almost $n/2$. Thus, if both walks use the same target quorum size, it must equal approximately $|Q_a| = |Q_\ell| = 170 \approx 1.5\frac{n}{\log n} \approx n/4.7$. For such a choice of quorum sizes, the number of messages of the `lookup` access is about $|Q_\ell|/2$, since the first hit occurs at approximately half the way. Note however, that the exact constants of the crossing time depend on



(a) Hit ratio. Without reply path local repair.(b) Fraction of quorum intersections. Without reply path local repair.(c) Fraction of dropped reply msgs. Without reply path local repair.



(d) Hit ratio. `advertise` $|Q|=2\sqrt{n}$.(e) Fraction of dropped reply messages. With reply path local repair.(f) Num. msgs. per lookup + routing. With reply path local repair.



(g) Number of messages per lookup. With reply path local repair.(h) Hit ratio. `advertise` $|Q|=3\sqrt{n}$. With reply path local repair.(i) Degradation rate in face of churn. Static. $d_{\text{avg}} = 15$

Figure 12: *RANDOM* `advertise`, *UNIQUE-PATH* `lookup`, Varying Mobility, $n = 800$.

the network topology and density and are not the same for all networks. Thus, the target quorums sizes cannot be set in a generic way. This is in contract with the *RANDOM* x *UNIQUE-PATH* mix, in which the same quorum sizes guarantee the same intersection on any topology and density.

8.6 Fast Mobility Impact

In this section we have also explored the impact of varying speeds (in addition to the impact of a relatively slow mobility that was discussed above). According to Figure 12(a), the hit ratio deteriorates as the maximal speed increases. To shed a light on the reasons for this effect, we have looked at every stage of our protocol. Recall that to access a `lookup` quorum by the *UNIQUE-PATH* strategy, the RW first traverses the network until it encounterers $|Q_\ell|$ different nodes, stops upon the first hit and then sends the reply back to the looking node, following the reverse RW path. We can see in Figure 12(b) that the intersection probability itself (not taking into account the fail of the reply) is not impacted by mobility at

Advertise		RANDOM				UNIQUE-PATH
Lookup		RANDOM	RANDOM-OPT	UNIQUE-PATH	FLOODING	UNIQUE-PATH
Advertise Cost	Static	600				250
	Mobile	1600				
Cost of Lookup Miss	Static	350*	40*	33	50**	100
	Mobile	1000*	100*			
Cost of Lookup Hit	Static	350*	40*	23	35	35
	Mobile	1000*	100*			

* this does not include the routing cost

** 50 messages is the coverage range of flooding with TTL 3. Smaller TTL results in a non -sufficient intersection

Figure 13: Summary of simulation results for combinations of different access strategies, $n = 800$, $d_{\text{avg}} = 10$, intersection 0.9, $|Q_a| = 2\sqrt{n} = 56$, $|Q_\ell| = 1.15\sqrt{n} = 33$.

all. This is since we use the *RW salvation* technique to prevent dropping of RW messages, as discussed in Section 6.2. However, when following the reverse path, a growing number of reply messages is being dropped (Figure 12(c)). Basically, this happens since when a node is not able to reach a certain node that is the next hop along the reverse RW path, it drops the reply message. Naturally, the longer the RW is, the higher is the chance that at least one of the nodes that was previously traversed by the RW will move out of range and the reverse path will break.

We have thus tackled the fast mobility with the *reply path local repair* technique, also discussed in Section 6.2 Figures 12(d) and 12(e) depicts the hit ratio with reply path repairs. We can see that the combination of local and global path repairs fixes the hit ratio and decreases the amount of reply message droppings significantly. As for the networking price, Figures 12(f) depicts the amount of network messages including routing control messages (AODV path establishment and maintainable). As speed increases, more and more reply path repairs are required and the price of routing increases. Note, however, that if there was no need to send the reply back, the RW salvation technique alone fully eliminates the mobility effects almost without any cost. The RW itself does not invoke routing at all, and the amount of generated network messages is almost the same for static and mobile networks (Figures 12(g)).

Another way to combat fast mobility in a proactive way is by increasing the `advertise` quorum. According to Figures 12(h), increasing $|Q_{\texttt{advertise}}|$ from $2\sqrt{n}$ to $3\sqrt{n}$ improves the hit ratio, since the `lookup` access needs shorter RWs, which also decreases the chance for a reply path breakage. Another possible way to deal with reply path breakages is by incorporating anycast logic into the routing layer, in a more tight cross layer design. Exploration of this option is left for a future work.

8.7 Churn

Figures 12(i) depicts the intersection probability in face of churn (is static 800 nodes network, with average density of 15 neighbors, which kept the network connected in all scenarios). After all advertisements finished, we fail every node with a given probability or/and add new nodes according to the average churn rate (the x-axis). The size of the `lookup` quorum was adjusted to the new network size. We can see an outstanding survivability of the probabilistic quorums: the intersection probability deteriorates very slowly with the increasing churn rate. For example, the initial intersection of 0.95 degrades to only 0.87 in face of as much as 50% failures (as long as the network remains connected). This also matches our analysis from Section 6.1.

8.8 Simulation Summary

Figure 8.8 presents a summary of a simulation study for some specific setup values. For `lookup` we have depicted both the cost of a hit and the cost of a miss. In case of a miss (the looked-up object is not present), the whole cost of a target quorum size is paid. In case of a hit, this cost is reduced due to the early halting of some strategies (but also includes the price of a reply). *RANDOM* and *RANDOM-OPT* lookup quorums were accessed serially and not in parallel, thus do not benefit from early halting.

Based on these values and our result from section 5.4, we can reason which strategy mix is better. For example, the relative cost of `advertise` vs. `lookup` for a *RANDOM*x*UNIQUE-PATH* combination in this setting is $\frac{|Q_a|Cost_a}{|Q_\ell|Cost_\ell} = \frac{600}{33} = 18$, while for a *UNIQUE-PATH*x*UNIQUE-PATH* combination it is $\frac{|Q_a|Cost_a}{|Q_\ell|Cost_\ell} = \frac{250}{100} = 2.5$. Therefore, in this setting, as long as $\tau > 2.5$ (τ is the frequency of `lookup`s vs. `advertise`s), it is more efficient to use *RANDOM*x*UNIQUE-PATH* rather than *UNIQUE-PATH*x*UNIQUE-PATH*.

9 Related Work

9.1 Quorum systems

Quorum systems were implicitly introduced by Gifford [22] and Thomas [65] as a weighted voting mechanisms for ensuring consistency. An explicit definition of quorums later appeared in [21], and was extended to bi-coteries, and therefore bi-quorums, in [54]. Herlihy used quorums and consensus to implement shared objects in [30]. Other quorum based implementations of distributed shared memory (or shared objects) include [4, 48]. Probabilistic quorum systems were initially introduced by Malkhi et al [50]. They were later explored, e.g., in [51, 55].

Reconfigurable quorum systems were first explored by Herlihy in [31]. Additional dynamical reconfiguration mechanisms of quorum systems appeared in [48] and [2], yet without analyzing the failure probability of a single quorum. Moreover, a method for dynamic update of quorums in the face of joins and leaves based on maintaining de Bruijn graph was presented in [2]. Those methods are unsuitable for ad hoc networks due to their large message complexity. Our calculations of the degradation rate can serve as a simple mechanism for determining how often a quorum system needs to be refreshed to ensure continued intersections (or lookup success).

Byzantine resilient quorum systems were investigated, e.g., in [3, 51, 52]. In order to deal with Byzantine systems, the size of the intersection of quorums must be large enough to mask possible false output by Byzantine nodes. In this paper, we do not address Byzantine failures.

An implementation of a probabilistic quorum system for sensor networks appears in [14]. The access to both write and read quorums is performed by flooding (gossiping) a corresponding request throughout the entire network. In writes, the data is saved by $\Theta(n)$ nodes, yet only $\Theta(\log n)$ nodes send an acknowledgement. For reads, the ids of $\Theta(\log n)$ random nodes are included in the header of the corresponding message; only these nodes are supposed to send a reply to the read request. This mix of access strategies is a special case of `advertise` *FLOODING*, `lookup` *RANDOM* strategies, while the choice of random `lookup` requires a random membership or sampling procedure, just like in our work.

9.2 Quorum Based Location Services

One of the most widely used application of quorums in ad hoc networks thus far has been in implementing location service. In quorum-based protocols (also known as *rendezvous-based*), all nodes in the network agree, implicitly or explicitly, upon a mapping that associates each node's unique identifier to one or more other nodes in the network. The mapped-to nodes are the location servers for that node.

They are the nodes where periodical location updates are being stored and location queries will be routed to and looked up at. The mapping of nodes to quorums should be such that the update quorum will intersect with the lookup quorum. Examples of quorum-based location services include, [28, 63, 34], Octopus [53], LLS [1], GLS [42]. Additional examples of data location services are GCLP [64], GHT [59] and Rendezvous Regions [61]. All those works differ from our in that most of them use geographic knowledge, do not use probabilistic quorums and do not utilize a-symmetric quorum systems. In that respect, our work is the first systematic study of probabilistic quorum systems in ad hoc network.

In the works of Haas and Liang [28, 29], a uniform random quorum system is used for mobility management. Nodes location information is maintained in location databases that form a virtual backbone. When a node moves, it updates its location with one quorum containing the nearest backbone node. Each source node then queries the quorum containing its nearest backbone for the location of the destination, and uses that location to route the message. In [29] the division of nodes into quorums is static and done a-priori, while ensuring uniformity in the sense that all nodes will be members of the same number of quorum sets. On the other hand, in [28] the selection of nodes into quorums is done randomly during runtime.

In PAN [44, 45] both read and write quorums are random subsets of nodes. The write quorum is accessed by random gossip, which also constructs random membership. The read quorum is accessed by contacting a set of random nodes picked out of the random membership directly (relying on routing), in the same way as in our *RANDOM* access. Both write and read quorums are directed only to a predefined subset of nodes, termed *Storage Set* (`StS`). StS can be any subset or overlay (such as connected dominating set), picked statically or dynamically and agreed upon by all nodes. However, PAN does not specify how to dynamically reconfigure the quorum system when StS changes. Our scheme could be extended in a straightforward way to use only a subset of nodes for quorums. Since the write updates in PAN are disseminated to the whole network (or the whole StS) and are eventually stored by all StS nodes, the size of the read quorum can be kept relatively small (it must still be more than one, to overcome node failures and message loss). Our work could be seen as a generalization of [45], since it provides a number of alternative quorum access strategies, while [45] only provides *RANDOM* `advertise` x *RANDOM* `lookup`. In addition, our scheme is more flexible since it does not disseminate the write updates to all (StS) nodes, but allows to adjust the sizes of the write/read quorums optimally w.r.t. the access pattern, as shown in Section 5.4.

The work in [37] focuses on the problem of efficiently utilizing quorum systems in a highly dynamic environment. Nodes are partitioned into fixed quorums, and every operation updates a randomly selected set, thus balancing the load. Their simulation study indicates that probabilistic quorums have a better recency rate than other strict quorum strategies.

In GeoQuorums [16], geometric coordinates determine the location of home servers. These focal point coordinates define geographic areas that must be inhibited by at least one server at any time. Sets of focal points are organized in intersecting quorums. The quorums are further used to implement an atomic memory abstraction in mobile ad hoc networks. The algorithm to dynamically reconfigure the set of available quorums is presented as well. At this stage, we focus on implementation of quorum systems and location services without utilizing geographic information.

In [10], a location service is implemented through randomly selected quorums. Yet, no means are provided in [10] to determine the required size of the random quorum and no theoretical evaluation of the quorum selection algorithms is supplied.

10 Discussion

In this paper we have explored various access strategies for implementing probabilistic bi-quorum systems in ad hoc networks. In particular, we have shown that asymmetric bi-quorums can offer better performance than symmetric ones. Moreover, we have shown that even without geographical knowl-

edge (or localization), it is possible to obtain efficient quorums. The bi-quorum system we have found most efficient is the one that uses *RANDOM* for `advertise` and *UNIQUE-PATH* for `lookup` (or vice versa). This is due to the use of random walks in *UNIQUE-PATH*, which eliminates the need for multiple hop routing.

We would like to stress that, as mentioned in the introduction, asymmetric constructions of probabilistic bi-quorum systems are useful not only for ad hoc networks, but also for any network with non uniform access costs (e.g, peer-to-peer networks). This is because in many workloads, one type of quorum access (e.g., lookup) is much more frequent than the other (e.g., advertise). Hence, the more common access type can only communicate with the closest nodes, and only the rare access type needs to communicate with random nodes, which are most certain to include some far away nodes. Of course, the result holds also when advertisements are more frequent than lookups, in which case advertisements are the ones that can be performed on nearby nodes.

The main driving application that we addressed in this work is data location services, which can also be trivially generalized to distributed dictionary services and bulletin boards. Yet, another appealing application of quorums is distributed shared objects. In particular, it was shown in [4] that atomic registers, also known as linearizable read/write objects [32], can be implemented using quorums. Yet, such an implementation requires both read and write operations to access one `advertise` and one `lookup` quorum [5, 41, 47].[5] When using probabilistic quorums, these protocols in fact implement what is known as probabilistic linearizability [24].

An interesting area for future research is the usage of probabilistic quorums for implementing decentralized publish/subsribe (pub/sub) systems [8]. For example, a natural way of implementing pub/sub using quorums is to disseminate a subscription to all members of an advertise quorum; a published event can be sent to all members of a lookup quorum. Each member of the lookup quorum checks if the event matches any of the subscriptions it is aware of, and if one exists, it sends a notification to the corresponding subscriber. In particular, as event publications typically occur much more frequently than subscriptions, the use of an asymmetric bi-quorum system like the ones we propose here is advantageous. Clearly, when using probabilistic quorums, the guarantees of the resulting pub/sub system become probabilistic as well. Notice, however, that an interesting challenge of such a system is how to execute unsubscriptions efficiently. That is, in probabilistic quorum systems each access to a quorum touches a possibly different set of nodes. Hence, sending an unsubscribe message to a single probabilistic quorum might not be enough.

References

[1] I. Abraham, D. Dolev, and D. Malkhi. LLS: a Locality Aware Location Service for Mobile Ad Hoc Networks. In *Proc. of the Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 75–84, 2004.

[2] I. Abraham and D. Malkhi. Probabilistic quorums for dynamic systems. In *Proc. of the 16th International Symposium on Distributed Computing (DISC)*, pages 60–74, 2003.

[3] L. Alvisi, E. T. Pierce, D. Malkhi, M. K. Reiter, and R. N. Wright. Dynamic Byzantine Quorum Systems. In *Proc. of the 2000 International Conference on Dependable Systems and Networks (DSN)*, page 283, 2000.

[4] H. Attiya, A. Bar-Noy, and D. Dolev. Sharing Memory Robustly in Message Passing Systems. *J. ACM*, 42(1):124–142, 1995.

[5] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. McGraw Hill, 1998.

[6] C. Avin and C. Brito. Efficient and Robust Query Processing in Dynamic Environments Using Random Walk Techniques. In *Proc. of the 3rd International symposium on Information Processing in Sensor Networks (IPSN)*, pages 277–286, 2004.

[7] C. Avin and G. Ercal. On the cover time and mixing time of random geometric graphs. *Theor. Comput. Sci.*, 380(1-2):2–22, 2007.

[5] There are certain optimizations by which when there is no contention, some of these accesses can be saved, but in the worst case, some read and some write operations have to access both quorums [18].

[8] R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg. Content-Based Publish-Subscribe over Structured Overlay Networks. In *Proc. of the 25th Intr. Conference on Distributed Computing Systems (ICDCS)*, pages 437–446, 2005.

[9] Z. Bar-Yossef, R. Friedman, and G. Kliot. RaWMS - Random Walk Based Lightweight Membership Service for Wireless Ad Hoc Networks. *ACM Transactions on Computer Systems (ACM TOCS)*, 26(2):1–66, June 2008.

[10] S. Bhattacharya. Randomized location service in mobile ad hoc networks. In *Proc. of the 6th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWIM)*, pages 66–73, 2003.

[11] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. of the 4th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 85–97, 1998.

[12] C. Bettstetter and G. Resta and P. Santi. The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 2(3):257–269, July-September 2003.

[13] G. Chockler, M. Demirbas, S. Gilbert, C. Newport, and T. Nolte. Consensus and Collision Detectors in Wireless Ad Hoc Networks. In *Proc. 24th Symposium on the Principles of Distributed Computing (PODC)*, 2005.

[14] G. Chockler, S. Gilbert, and B. Patt-Shamir. Communication-Efficient Probabilistic Quorum Systems for Sensor Networks. In *Proc. 4th Intr. Conf. on Pervasive Computing and Communication Workshops (PERCOMW)*, page 111, 2006.

[15] G. Chockler, I. Keidar, and R. Vitenberg. Group Communication Specifications: a Comprehensive Study. *ACM Computing Surveys*, 33(4):427–469, 2001.

[16] S. Dolev, S. Gilbert, N. Lynch, A. Shvartsman, and J. Welch. Geoquorums: Implementing atomic memory in mobile ad hoc networks. In *Proc. of the 17th International Symposium on Distributed Computing (DISC)*, 2003.

[17] S. Dolev, E. Schiller, and J. Welch. Random Walk for Self-Stabilizing Group Communication in Ad Hoc Networks. In *Proc. of the 21st ACM Symposium on Principles of Distributed Computing (PODC)*, pages 259–259, 2002.

[18] P. Dutta, R. Guerraoui, R. R. Levy, and A. Chakraborty. How fast can a distributed atomic read be? In *Proc. of the 23rd ACM symposium on Principles of Distributed Computing (PODC)*, pages 236–245, 2004.

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

[20] R. Friedman and G. Kliot. Location Services in Wireless Ad Hoc and Hybrid Networks: A Survey. Technical Report CS-2006-10, Technion, Haifa, Israel, Available at `http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-info.cgi?2006/CS/CS-2006-10`, January 2006.

[21] H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *Journal of ACM*, 32(4):841–860, 1985.

[22] D. K. Gifford. Weighted Voting for Replicated Data. In *Proc. of the 7th Symposium on Operating System Principles (SOSP)*, pages 150–162, December 1979.

[23] C. Gkantsidis, M. Mihail, and A. Saberi. Random Walks in Peer-to-Peer Networks. In *Proceedings of the 23rd Conference of the IEEE Communications Society (INFOCOM)*, pages 259–259, 2004.

[24] V. Gramoli. *Distributed Shared Memory for Large-Scale Dynamic Systems*. PhD thesis, University of Rennes 1, 2007.

[25] R. Guerraoui and M. Raynal. The Information Structure of Indulgent Consensus. *IEEE Transactions on Computers*, 53(4):453–466, 2004.

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

[27] P. Gupta and P. Kumar. The capacity of wireless networks. *IEEE Trans. Inform. Theory*, 46(2):388–404, March 2000.

[28] Z. Haas and B. Liang. Ad Hoc mobility management with randomized database groups. In *Proc. of IEEE ICC*, June 1999.

[29] Z. Haas and Ben Liang. Ad Hoc mobility management with uniform quorum systems. *IEEE/ACM Transactions on Networking*, 7(2):228–240, 1999.

[30] M. Herlihy. A quorum-consensus replication method for abstract data types. *ACM Trans. Comput. Syst.*, 4(1):32–53, 1986.

[31] M. Herlihy. Dynamic quorum adjustment for partitioned data. *ACM Trans. Database Syst.*, 12(2):170–194, 1987.

[32] M. Herlihy and J. Wing. Linearizability: A Correctness Condition for Concurrent Objects. *ACM Trans. on Programming Languages and Systems*, 12(3):463–492, 1990.

[33] IETF Mobile Ad hoc Networks Working Group. MANET Neighborhood Discovery Protocol (NHDP). Available at `http://tools.ietf.org/html/draft-ietf-manet-nhdp-07.txt`.

[34] J.-P. Hubaux, T. Gross, J.-Y. Le Boudec, and M. Vetterli. Towards Self-Organizing Mobile Ad Hoc Networks: the Terminodes Project. *IEEE Communications Magazine*, pages 118–124, 2001.

[35] IEEE Computer Society. 802.11: Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specifications.

[36] IETF Mobile Ad-hoc Networks Working Group. RFC 5148: Jitter considerations in Mobile Ad Hoc Networks (MANETs). Available at `www.ietf.org/rfc/rfc5148.txt`.

[37] G. Karumanchi, S. Muralidharan, and R. Prakash. Information Dissemination in Partitionable Mobile Ad Hoc Networks. In *Symposium on Reliable Distributed Systems*, pages 4–13, 1999.

[38] I. Keidar and R. Melamed. Evaluating Unstructured Peer-to-Peer Lookup Overlays. In *Proc. of the 2006 ACM Symposium on Applied Computing (SAC)*, pages 675–679, April 2006.

[39] A. Keshavarz-Haddad, V. Ribeiro, and R. Riedi. Broadcast capacity in multihop wireless networks. In *Proc. of the 12th Intr. Conference on Mobile Computing and Networking (MobiCom)*, pages 239–250, 2006.

[40] G. Kliot. Wireless Signal Interference models made simple. Available at `www.cs.technion.ac.il/~gabik/Jist-Swans/signal_interference`.

[41] L. Lamport. On Interprocess Communication. *Distributed Computing*, 1(2):77–101, 1986.

[42] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proc. of the 6th Intr. Conference on Mobile Computing and Networking (MobiCom)*, pages 120–130, August 2000.

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

[44] J. Luo, P. Eugster, and J. Hubaux. Pilot: Probabilistic lightweight group communication system for ad hoc networks. *IEEE Transactions on Mobile Computing*, 3(2):164–179, April 2004.

[45] J. Luo, J.-P. Hubaux, and P.Th. Eugster. PAN: Providing reliable storage in mobile ad hoc networks with probabilistic quorum systems. In *Proc. of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 1–12, 2003.

[46] C. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *Proceedings of the 16th International Conference on Supercomputing (ICS)*, pages 84–95, 2002.

[47] N. Lynch. *Distributed Algorithms*. Morgan Kaufman, 1996.

[48] Nancy A. Lynch and Alexander A. Shvartsman. RAMBO: A Reconfigurable Atomic Memory Service for Dynamic Networks. In *Proc. of the 16th International Conference on Distributed Computing (DISC)*, pages 173–190, 2002.

[49] N. Madras and G. Slade. *The self-avoiding walk*. Birkhauser, Boston, 1993.

[50] D. Malkhi, M. Reiter, A. Wool, and R. Wright. Probabilistic Quorum Systems. *The Information and Computation Journal*, 170(2):184–206, November 2001.

[51] D. Malkhi and M. K. Reiter. Secure and Scalable Replication in Phalanx. In *Proc. of the the 17th IEEE Symposium on Reliable Distributed Systems (SRDS)*, page 51, 1998.

[52] J.-P. Martin and L. Alvisi. A Framework for Dynamic Byzantine Storage. In *Proc. of the 34th International Conference on Dependable Systems and Networks (DSN)*, page 325, 2004.

[53] R. Melamed, I. Keidar, and Y. Barel. Octopus: A Fault-Tolerant and Efficient Ad-hoc Routing Protocol. In *Proc. of the 24th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 39–49, October 2005.

[54] N. Mitchell, M. Mizuno, and M. Raynal. A General Method to Define Quorums. In *Proc. of the 12th International Conference on Distributed Computing Systems (ICDCS)*, pages 657–664, 1992.

[55] K. Miura and T. Tagawa. A Quorum-Based Protocol for Searching Objects in Peer-to-Peer Networks. *IEEE Trans. Parallel Distributed Systems*, 17(1):25–37, 2006.

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

[57] P. Panchapakesan and D. Manjunath. On the Transmission Range in Dense Ad Hoc Radio Networks. In *Proc. of IEEE Signal Processing Communication (SPCOM)*, 2001.

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

[59] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A Geographic Hash Table for Data-Centric Storage in SensorNets. In *Proc. of the 1st ACM Intr. Workshop on Wireless Sensor Networks and Applications (WSNA)*, September 2002.

[60] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288, 1984.

[61] K. Seada and A. Helmy. Rendezvous Regions: A Scalable Architecture for Service Provisioning in Large-Scale Mobile Ad Hoc Networks. Proc of ACM SIGCOMM, Refereed poster, 2003.

[62] S. Servetto and G. Barrenechea. Constrained Random Walks on Random Graphs: Routing Algorithms for Large Scale Wireless Sensor Networks. In *WSNA*, 2002.

[63] I. Stojmenovic. A routing strategy and quorum based location update scheme for ad hoc wireless networks. Computer Science, SITE, University of Ottawa, TR-99-09, September 1999.

[64] J. Tchakarov and N.H. Vaidya. Efficient Content Location in Wireless Ad Hoc Networks. In *Proc. IEEE International Conference on Mobile Data Management (MDM)*, January 2004.

[65] R. H. Thomas. A Majority consensus approach to concurrency control for multiple copy databases. *ACM Trans. Database Syst.*, 4(2):180–209, 1979.

[66] C.K. Toh. *Ad Hoc Mobile Wireless Networks*. Prantice Hall, 2002.

[67] Cornell University. JiST/SWANS Java in Simulation Time / Scalable Wireless Ad Hoc Network Simulator. Available at `http://jist.ece.cornell.edu/`.

A Random Walk Preliminaries

Let $G(V, E)$ be an undirected graph, with V the set of nodes and E the set of edges. Let $n = |V|$ and $m = |E|$. For $v \in V$, let $N(v) = \{u \in V \mid (v, u) \in E\}$ be the set of neighbors of v, and let $\delta(v) = |N(v)|$ be the degree of v.

Let $X_v = \{X_v(\tau) : \tau \geq 0\}$ be a *simple random walk* starting from node v on the state space V with *transition matrix* Q. When the walk is at node v, the probability to move in the next step to u is $Q_{vu} = \Pr(v, u) = \frac{1}{\delta(v)}$ for $(v, u) \in E$ and 0 otherwise. Let $X_v(t) = \{X_v(\tau) : t \geq \tau \geq 0\}$ denote a walk of length t. Let $\mathcal{N}(t) = |X_v(t)|$ denote the number of distinct nodes visited by the random walk of length t.

The *hitting time*, $h(u, v)$, is the expected time for a random walk starting at u to arrive to v for the first time. Let $h_{\max}$ be the maximum $h(u, v)$ over all ordered pairs of nodes. The return time $h(v, v)$ is the expected time to return to v for the first time, starting at v. $h(v, v)$ is well known to be equal to $\frac{1}{\pi_v}$, from the theory of Markov chains [43] where $\pi = \{\pi_v : v \in V\}$ is the stationary distribution of the simple random walk. In [7] the authors prove that following:

Proposition A.1. *For $c > 1$, if $r^2 \geq \frac{c8 \log n}{n}$, then* w.h.p. *for $\mathcal{G}^2(n, r)$:*

(i) $h_{\max}$ is linear in n. That is, there exist a constant γ independent of n, s.t. w.h.p. *$h_{\max} \leq \gamma n$.*

(ii) $\forall v \ \pi_v = \Theta(\frac{1}{n}) \geq \frac{1}{\beta n}$ for a constant β independent of n.

The first statement is based on a resistance argument and the second is based on proving that *w.h.p.* the degree of every node is $\Theta(\log n)$.

B Partial Cover Time of Random Geometric Graphs

The *partial cover time*, $PCT_G(i)$, of a graph G is the expected time taken by a simple random walk on G to visit i distinct nodes in G. Formally, for $v \in V$, let $PCT_v(i)$ be the expected number of steps needed for the simple random walk starting at v to visit i distinct nodes in G, and the partial cover time of G is $PCT(i) = \max_v PCT_v(i)$. The *cover time*, C_G, of G is the expected time to visit all nodes in G, i.e., $C_G = \max_v PCT_G(n)$.

Theorem 4.1 (restated) *Let $t = o(n)$. For $c > 1$, if $r^2 \geq \frac{c8 \log n}{n}$, then* w.h.p. *for $\mathcal{G}^2(n, r)$*

$$PCT_{\mathcal{G}}(t) \leq 2\alpha t$$

where, for large enough n, α is a constant not depending on n.

Proof. The proof proceeds as follows. We will first bound the expected number of visits to each vertex during the walk and then show that the number of distinct nodes visited by the random walk of length $t = o(n)$ is at least $t/(2\alpha)$.

Given a graph G and a starting node v, we view the random walks as proceeding in phases. For $1 \leq i \leq n$ we identify phase i with a set of i vertices, $V_i \subseteq V$ and a starting vertex $s_i \in V$, which is the last vertex visited in phase $i-1$. Phase i starts with the random walk at s_i and ends with the random walk exiting V_i. Note that $V_1 = v$. If the walk is of length t and $k \leq t$ distinct nodes are visited during the walk, then for $k < j \leq n$, we define s_j and V_j to be empty sets.

For now consider only $X_v(t)$ where we may omit t if it is clear from the context. Let α_i denote the number of visits to s_i during the t steps. If $s_i = \emptyset$ then $\alpha_i = 0$. Clearly

$$t = \sum_{1}^{n} \alpha_i$$

Taking expectation over all possible walks of length t we have:

$$E[t] = t = E[\sum_{1}^{n} \alpha_i] = \sum_{1}^{n} E[\alpha_i]$$

Consider the graph in the theorem, we now show that the expected number of visits to each vertex during $X_v(t)$, is bounded by a constant.

Lemma B.1. $\max_i E[\alpha_i] \leq \alpha$

Proof. The proof is based on the Proposition A.1. Let $t = o(n)$. For a random walk X_v, let $\mathcal{E}_v^t$ denote the event that the random walk return to v within time t and $\rho_v = \Pr(\mathcal{E}_v^t)$. For the $\mathcal{G}^2(n,r)$ given in the theorem we can bound the return time in the following way:

$$\begin{aligned}
h(v,v) &= \frac{1}{\pi_v} && (1)\\
&\leq \beta n && (2)\\
&\leq \rho_v E[h(v,v) \mid \mathcal{E}_v^t] + (1-\rho_v) E[h(v,v) \mid \bar{\mathcal{E}}_v^t] && (3)\\
&\leq \rho_v t + (1-\rho_v)(t + h_{\max}) && (4)\\
&= t + (1-\rho_v)(h_{\max}) && (5)\\
&\leq t + (1-\rho_v)\gamma n && (6)
\end{aligned}$$

Where β, γ are the constants from Proposition A.1, which do not depend on n. Steps (2) and (6) are licensed by Proposition A.1. Step (4) is true since if the walk does not return to v within t steps we can bound the return time by taking t steps plus the time it takes to return to v from the node at which the walk is at time t. Now, for large enough n, we can lower bound $(1-\rho_v)$ as follows:

$$(1-\rho_v) \geq \frac{\beta n - t}{\gamma n} \geq c \tag{7}$$

where $0 < c < 1$ is a constant not depending on n (e.g., c could be $\frac{\beta}{2\gamma}$). Let R_v be the expected number of returns to v before step t. Using ρ_v we can bound R_v with the expected number of trials (every time the walk returns to v we start a new independent trial) until the walk does not return to v in t steps:

$$R_v \leq \frac{1}{1-\rho_v} \leq 1/c \tag{8}$$

By setting $\alpha = c + 1$, we can conclude that the expected number of visits of a walk of length t to every s_i at most α times (since we can see every phase i as a random walk of length less than t starting at s_i), so $\forall i \; E[\alpha_i] \leq \alpha$.

□

Lemma B.2. *For a random walk $X_v(t)$, if $\forall i \leq n \;\; E[\alpha_i] \leq \alpha$, then*

$$E[\mathcal{N}(t)] \geq \frac{t}{2\alpha} \tag{9}$$

Proof. We omit the t. We can express $E[\mathcal{N}]$ as follows:

$$E[\mathcal{N}] = \sum_{j=1}^{n} \Pr(\mathcal{N} \geq j) = \sum_{j=1}^{n} 1 - \Pr(\mathcal{N} < j) \tag{10}$$

Now will bound $\Pr(\mathcal{N} < j)$ by $\Pr(\mathcal{N} < j) = \Pr(\mathcal{N} \leq j - 1) \leq (j-1)\alpha/t$:

$$\Pr(\mathcal{N} \leq j) \leq \min\{\frac{j\alpha}{t}, 1\} \tag{11}$$

Clearly $\Pr(N \leq j) \leq 1$. We will prove that $\Pr(N \leq j) \leq j\alpha/t$. Let $\mathcal{E}_j$ be the event that $N \leq j$ and $\bar{\mathcal{E}_j}$ be its compliment. We can express $E[\alpha_i]$ via conditional expectation:

$$E[\alpha_i] = \Pr(\mathcal{E}_j)E[\alpha_i \mid \mathcal{E}_j] + \Pr(\bar{\mathcal{E}_j})E[\alpha_i \mid \bar{\mathcal{E}_j}] \tag{12}$$

Assume by contradiction that $\Pr(\mathcal{E}_j) > \frac{j\alpha}{t}$. Since all walks are of length t, for all walks that visit at most j distinct nodes we have $t = \sum_{i=1}^{j} \alpha_i$, and

$$\sum_{i=1}^{j} E[\alpha_i \mid \mathcal{E}_j] = t$$

Since the average value is t/j, there exist i^* for which $E[\alpha_{i^*} \mid \mathcal{E}_j] \geq t/j$. Otherwise, if $\forall i$: $E[\alpha_i \mid \mathcal{E}_j] < t/j$ we will get $\sum_{i=1}^{j} E[\alpha_i \mid \mathcal{E}_j] \leq j \cdot \max_j\{E[\alpha_i \mid \mathcal{E}_j]\} < j \cdot t/j = t$. Now, from Eq (12) we get:

$$E[\alpha_{i^*}] > \frac{j\alpha}{t} \cdot \frac{t}{j} + \Pr(\bar{\mathcal{E}_j})E[\alpha_{i^*} \mid \bar{\mathcal{E}_j}] \tag{13}$$

$$> \alpha \tag{14}$$

this is a contradiction to $\forall i \leq n \;\; E[\alpha_i] \leq \alpha$, so $\Pr(\mathcal{E}_j) \leq \frac{j\alpha}{t}$.

Let $k = \lfloor \frac{t}{\alpha} \rfloor$, now we can put everything together:

$$E[\mathcal{N}] = \sum_{j=1}^{n} 1 - \Pr(\mathcal{N} < j) \tag{15}$$

$$\geq \sum_{j=1}^{n} 1 - (j-1)\frac{\alpha}{t} \tag{16}$$

$$\geq \sum_{j=1}^{k} 1 - (j-1)\frac{1}{k} \tag{17}$$

$$\geq k - \frac{1}{k}\sum_{j=1}^{k-1} j \tag{18}$$

$$\geq k - \frac{1}{k}\frac{(k-1)k}{2} \tag{19}$$

$$= \frac{1}{2}(k+1) \tag{20}$$

$$> \frac{t}{2\alpha} \tag{21}$$

which proves the assertion of the Lemma. □

From Lemma B.2, the expected number of distinct nodes a walk visits in t steps is at least $t/2\alpha$. Thus, we conclude the statement of the Theorem that $PCT_{\mathcal{G}}(t) \leq 2\alpha t$. □