PILOT: Probabilistic Lightweight group communication system for Ad Hoc Networks[†]

Jun Luo Patrick Th. Eugster Jean-Pierre Hubaux

School of Computer and Communication Sciences

Swiss Federal Institute of Technology (EPFL), CH-1015 Lausanne, Switzerland

Email: {jun.luo, patrick.eugster, jean-pierre.hubaux}@epfl.ch

Abstract

Providing reliable group communication is an ever recurring topic in distributed settings. In mobile ad hoc networks, this problem is even more significant since all nodes act as peers, while it becomes more challenging due to highly dynamic and unpredictable topology changes. In order to overcome these difficulties, we deviate from the conventional point of view, i.e., we "fight fire with fire"[1], by exploiting the nondeterministic nature of ad hoc networks. Inspired by the principles of gossip mechanisms and probabilistic quorum systems, we present in this paper PILOT (Probabilistic Lightweight group communication system) for ad hoc networks, a two layer system consisting of a set of protocols for reliable multicasting and data sharing in mobile ad hoc networks. The performance of PILOT is predictable and controllable in terms of both reliability (fault tolerance) and efficiency (overhead). We

[†]The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322. (http://www.terminodes.org)

[1]This expression was recently used for another gossip-based protocol [1], albeit one with a different goal from this paper.

present an analysis of PILOT's performance, which is used to fine tune protocol parameters to obtain the desired tradeoff between reliability and efficiency. We confirm the predictability and tunability of PILOT through simulations with *ns-2*.

Index Terms—Mobile ad hoc networks, group communication systems, quorum systems, reliable multicast, gossiping, data sharing, replication

NOMENCLATURE

$\mathcal{R}_{ds}$	Reliability degree of single packet dissemination (Sections 3.2, 5 and 6).
$\mathcal{R}_{dc}$	Reliability degree of continuous packet dissemination (Sections 3.2, 5 and 6).
$\mathcal{R}_{da}$	Reliability degree of access (Sections 3.2, 5 and 6).
$\mathcal{N}_l$	Network load (Sections 3.2, 5 and 6).
$\mathcal{F}$	Cumulative distribution function of the reliability degree (Sections 3.2 and 5).
λ_o	Overall access rate to PILOT (Sections 3.2, 5 and 6).
λ_u	Update rate to PILOT (Sections 3.2, 5 and 6).
λ_q	Query rate to PILOT (Sections 3.2, 5 and 6).
F	Gossip *fanout* (Sections 4.2.1, 5, and 6).
τ_q	Gossip *quiescence threshold* (Sections 4.2.1, 5, and 6).
τ_a	Gossip *age threshold* (Sections 4.2.1, 5, and 6).
ξ_W, $\hat{\xi}_W$	Write quorum *nominal* size and *real* size, respectively (Sections 4.4, 5 and 6).
ξ_R, $\hat{\xi}_R$	Read quorum *nominal* size and *real* size, respectively (Sections 4.4, 5 and 6).
n	Group (or, in particular, STS) size (Sections 5 and 6).
H	Random variable representing the length of an arbitrary routing path (Section 5).
p, q	Probability of infection and non-infection, respectively (Section 5).
p_f	Failure probability for each hop (Section 5).
p_r	Probability of a query occurring within rounds $r+1$ after an update (Section 5).
p_e	Server unavailability in the case of query (Sections 5 and 6).
r, $\tilde{r}$	A certain gossip round and the final round, respectively (Section 5).
S_r	Number of infected nodes after round r (Section 5).
ν_r, ν	Distribution of S_r (also $\hat{\xi}_W^r$) and its eventual value, respectively (Section 5).
μ	Distribution of $\hat{\xi}_R$ (Section 5).

1 INTRODUCTION

A *Group Communication System* (GCS) [2] is a useful infrastructure on which various reliable distributed computing functions can be built. The need for such a system arises not only in wired networks but also in mobile ad hoc networks. Even some mechanisms traditionally relying on a centralized service have to be implemented in a distributed way in ad hoc networks, since the service provided by a single node is not dependable enough. Mobility management [3, 4], for instance, relies on a special group of nodes to continuously track locations of mobile nodes and to serve requests to these location data. The distributed management of cryptographic keys or certificates [5, 6] and group security functions like access control or key agreement [7, 8] represent another class of applications. Last but not least, distributed dynamic host configuration protocols such as naming or addressing services [9], which are essential to build a functional network, need to make agreements within the whole network.

Unfortunately, the complexity of building *reliable* GCSs, which is prohibitively high already in wired networks, is further amplified in ad hoc networks due to highly dynamic and unpredictable topology changes. In fact, even guaranteeing reliability of multicast, a key building block of GCSs, becomes extremely hard. As a result, many distributed computing functions that would depend on reliable GCSs have to either rely on the fragile "reliability" provided by flooding [9] or make assumptions about such a service while waiting for it to appear [5].

In this paper, we identify two fundamental problems in the context of group communication, namely (i) multicast and (ii) data sharing, and we define notions of probabilistic reliability for these problems, aimed at ad hoc networks. We then present our protocol suite, called ProbabilistIc LightweIght grOup communication sysTem (PILOT) for ad hoc networks, as a solution. Innovating on the principles of gossip mechanisms and probabilistic quorum systems, PILOT provides probabilistic reliability for multicasting and data sharing, based only on a unicast primitive (rather than a multicast primitive like MAODV [10]) in order to improve the adaptability to

future technology developments. We present analytical results predicting the performance of PILOT in terms of message overhead and reliability degree. We then compare these results with simulation results obtained with the *ns-2* simulator to show that we can have useful predictions on the performance of PILOT. To the best of our knowledge, the work presented in this paper, as part of the *MICS/Terminodes* project [11], is the first to provide a complete solution to the problems of reliable multicast and data sharing in ad hoc networks, along with both analytical and simulation results. It smoothly integrates, expands and completes our previous individual results [12, 13] into a compound group communication system.

The remainder of this paper is structured as follows. Section 2 overviews related work. Section 3 details the network model and the problem to be solved. Section 4 presents our PILOT system. Section 5 analyzes PILOT in terms of reliability and efficiency. Section 6 compares those values with simulation results, and also investigates other aspects of PILOT, such as its sensitivity to node failures. Finally, Section 7 concludes the paper.

2 RELATED WORK

The prosperous research on group communication toolkits has led to a multitude of results in wired networks, such as Ensemble [14] and Spread [15]. However, similar systems have not yet appeared in ad hoc networks, although certain supporting mechanisms like token circulation [16], random walk agent [17], reliable broadcast [18], and membership management [19] have been proposed. Our PILOT system is a first step towards building a prototype for a group communication toolkit. Rather than emphasizing the discussion in the framework of GCSs, we will hence focus on the relevant underlying building blocks.

2.1 Gossip-based Probabilistic Reliable Multicast

As opposed to the "perfect" reliability guarantee for multicast, (cf. *reliable broadcast* [20]), approaches to a form of probabilistic reliable multicast (e.g., *probabilistic broadcast* (*pbcast*) [21] and *lightweight probabilistic broadcast* (*lpbcast*) [22]) reduce the protocol overhead by sacrificing

safety guarantees such as atomicity through the use of a gossip-based dissemination scheme. These protocols also equally distribute the load over nodes and thus outperform the so-called "best effort [21]" reliable multicast (e.g. [23, 24]) by improving the resilience to arbitrary node failures and providing prediction on protocol reliability.

The *Anonymous Gossip* (AG) protocol [25], a descendant of the *pbcast* protocol, pioneered the recent research efforts on gossip-based multicast for ad hoc networks. Through the concept of anonymous gossip, any agreement on membership is avoided during the gossip-based repair phase. This however shifts the responsibility for the membership management to the MAODV layer [10], which the AG protocol also relies upon for a preliminary, rough packet dissemination. These prerequisites make the AG protocol more difficult to apply in a broader context than the one offered by MAODV. Furthermore, the property of predictable behavior, an important merit of gossip-based protocols, is lost due to the dependence on MAODV to guide the gossips.

2.2 Probabilistic Quorum Systems

Quorum systems [26] have been proposed as an alternative to the *state-machine* approach [27] for reliable data sharing. They improve the efficiency of the replication of the stored data by better balancing the overhead between updates and queries. Unfortunately, "original" quorum systems, also termed *strict quorum systems*, do not apply well to highly dynamic environments. This is because the very construction of these quorums is not a trivial task, the outcome of this task being strongly subject to membership changes. By introducing *probabilities* for the intersection of individual quorums, *probabilistic quorum systems* [28] relax the construction rules for quorums and leave more freedom for trading protocol overhead for reliability. While this smoother tradeoff has constituted the driving force behind probabilistic quorum systems, it turns out that the resulting reduced determinism makes such an approach also more viable for ad hoc networks than a strict approach. The overhead considered in [28] is the charge of computation for individual servers. Our definition of overhead, however, focuses on the consumption of network

resources, because computation is much cheaper than communication in wireless networks.

Haas and Liang [29] first introduced probabilistic quorum systems into ad hoc networks for mobility management, under the name of *randomized database groups*. They propose a very interesting way to express both fault tolerance and load as costs of their system, and optimize those costs numerically. Considering the similarity between their system and PILOT, we provide some comparisons between the two solutions in Section 4.6.

2.3 Data Management in Ad Hoc Networks

The 7DS system presented in [30] shares certain features of our PILOT system, with respect to the diffusion scheme used for data dissemination. However, since the two systems are designed for different network environments (7DS assumes a rarely connected network, whereas PILOT considers networks of relatively high density), the underlying diffusion mechanisms are quite different. Whereas 7DS passively exploits node mobility to relay data from one node to the other, which can result in a considerable delay for data spreading but has the potential to improve power and bandwidth usages, PILOT more actively "pushes" data to other nodes with a gossip-based protocol. As a result, the analytical models for the two diffusion processes are also different (diffusion controlled process for 7DS and epidemic model for PILOT).

Both [31] and [32] try to guarantee data accessibility upon network partitioning in a replication system by investigating the problem of dynamic replica allocation. While [31] makes assumptions (e.g., data items are not updated) that seem to be too strong to capture the reality of mobile networks and hence has limited application scope, the approach in [32] is more practical in the sense that it takes into consideration topology information (e.g., connection stability) when replicating data; and data replication only happens when necessary, according to certain partition detection schemes. As far as system models are concerned, the problem we solve is somewhat orthogonal to the one of [32]. The mobility model they propose assumes strong correlations between different nodes (e.g., nodes are organized into mobility groups), which might lead to

frequent network partitions. We, however, consider a purely random mobility pattern, in which network partitions seldom happen and mobility prediction does not make much sense.

3 GOALS AND ASSUMPTIONS

This section models the considered environment and states the problem to be solved.

3.1 Model

We consider an ad hoc network consisting of a set $\mathbb{N}$ of nodes and assume that every node $i \in \mathbb{N}$ has a unique id. Nodes may fail only by crashing, i.e., stopping to function. Failures are not permanent and can be recovered from.[2] All communications between different nodes are assumed to rely on the underlying unicast protocol. We use DSR [33] as an example in this paper but, in practice, our solution can be made to work with any on-demand routing protocol.

3.2 Problem Statement

We consider an ad hoc network where reliable group communication primitives are required by mobile nodes. Within the broad scope of group communication, we address two fundamental problems, namely multicast and data sharing, and associate each of them with a notion of *probabilistic reliability*.

1) Reliable Multicast Protocol: The multicast protocol disseminates packets within a multicast group $\mathbb{G} \subset \mathbb{N}$, which, for brevity, will be referred to as *group* hereafter. We define the following two metrics to measure the probabilistic reliability achieved by this protocol:

- *Reliability Degree of Single Packet Dissemination $\mathcal{R}_{ds}$:* The fraction of group members that receive the packet sent by a certain member.
- *Reliability Degree of Continuous Packet Dissemination $\mathcal{R}_{dc}$:* The fraction of all packets that are received by a certain member, assuming that packets are continuously sent from the same member with rate λ_o.

[2]This failure model also captures the case where nodes are deliberately switched off (e.g., for the purpose of battery replacement or operating system rebooting, or because the users do not intend to make use of their devices for a while).

Both metrics are described by respective *cumulative distribution function* (cdf) $\mathcal{F}(x) : [0,1] \rightarrow [0,1]$; it means that $\mathcal{F}(x)$ is the probability that $\mathcal{R}_{ds}$ (or $\mathcal{R}_{dc}$) is at most x.

2) Reliable Data Sharing Service: Let STS[3] $\subset \mathbb{N}$ be a storage entity and ρ be a set of access protocols for STS. The STS holds shared data in a replicated fashion, and the consistency model for data replication is considered to be *shared-private* [36], i.e., the service does not commit itself to any access ordering except FIFO order.[4] Given access rates λ_u and λ_q for updates and queries, respectively, the data sharing service is probabilistically reliable in nature if a query access $\rho_q(\text{STS}, \lambda_q)$ obtains, with a certain probability, the latest version of a data object resulting from an update access $\rho_u(\text{STS}, \lambda_u)$. The metric for the service is:

- *Reliability Degree of Access $\mathcal{R}_{da}$:* The probability that a query operation acquires the most recent update of the corresponding data object, considering both node and channel failures.

The overhead is measured by the *Network Load* $\mathcal{N}_l$, which is the average number of *unicast packet*$\times$*hop* per multicast packet to achieve a certain $\mathcal{R}_{ds}$ or per unit time to achieve a certain $\mathcal{R}_{dc}$ or $\mathcal{R}_{da}$. This definition is adapted to ad hoc networks by taking into account the number of hops to route a particular packet. $\mathcal{N}_l$ considers only the load generated by our protocols, which is independent of the various possible implementations of the underlying networking functions.

Our goal is to design a set of protocols that achieve a high reliability degree $\mathcal{R}_d$ (representing $\mathcal{R}_{ds}$, $\mathcal{R}_{dc}$, and $\mathcal{R}_{da}$ hereafter) even under large arrival rates λ_o (the sum of λ_u and λ_q for data sharing), while incurring reasonable overhead $\mathcal{N}_l$. We target relatively large scale networks, i.e., networks with tens or even hundreds of nodes and a random mobility pattern. Under a certain λ_o, the optimal performance with respect to both $\mathcal{R}_d$ and $\mathcal{N}_l$ does not exist, since one can always be sacrificed to improve the other. Hence, we will study the trade-off between the two metrics and show how to fine tune parameters to trade either for the other.

[3]STS is an abbreviation for *Storage Set*, a special group in the network. The algorithm used to initialize the STS will not be discussed here since it is out of the scope of this paper. Refer to [34, 35] for examples of initialization algorithms.

[4]All the applications we have mentioned in the introduction comply with this model.

4 PILOT: PROBABILISTIC GROUP COMMUNICATION SYSTEM

In this section, we first present the structure of our PILOT system, then we detail each component of PILOT separately.

4.1 Overview: Layered Architecture of PILOT

PILOT is a two layer system, illustrated by the dark grey part in Fig. 1(a). It has a probabilistic multicast protocol, Route Driven Gossip (RDG), as its basis. The protocol is gossip-based [21]

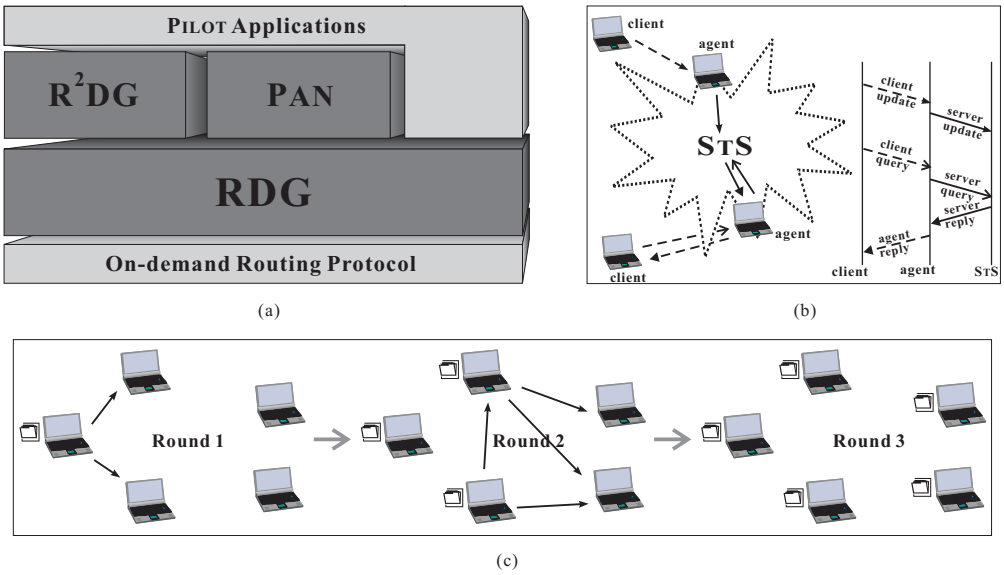


Fig. 1. Principles of PILOT. (a) Architecture of PILOT: the basic probabilistic multicast protocol (RDG) is at the bottom; R^{2}DG and PAN are built upon the basic protocol. (b) Message exchanges for updating and querying the STS in PAN. (c) Gossip-based multicasting in RDG.

in nature: it proceeds round by round while the receivers in each round are randomly chosen and they *relay* packets to the receivers of the later round(s), as shown in Fig. 1(c). Upon this layer, two dedicated services are built. R^{2}DG (Reliable RDG) is devised for continuous packet dissemination. It exploits the fact that packet losses can be detected by observing gaps in the pid (see Section 4.2.1) sequence, and thus piggybacks a negative acknowledgement with each packet sent (or relayed) to *pull* the lost packet back. The other service, Probabilistic quorum system for Ad hoc Networks (PAN), provides reliable data sharing by assuming the existence of an STS

to store the shared data in a replicated manner. Any node $i \in$ STS is termed *server*, whereas the rest of the nodes are termed *clients* of the STS. Data queries and updates are directed to an arbitrary server in the STS while the message dissemination within the STS is performed by RDG, as shown in Fig. 1(b). According to their requirements, applications can either use the upper layer services or directly call RDG if only single packet dissemination service is required.

4.2 RDG: Basic PILOT Multicast Protocol

Our RDG protocol uses a *pure* gossip scheme, as it is not built upon any underlying multicast protocol, in contrast to [25] (the only related approach we are aware of). As opposed to "traditional" gossip protocols that only consider the membership information of a group, RDG adapts to the peculiarity of ad hoc networks by also taking the availability of routing information into account. Although the resulting membership view for each member is just a random subview due to the randomness of routing information that nodes can have, the protocol still works very well in the sense that the reliability is in practice very high and also predictable.

4.2.1 Protocol Overview

Each packet multicast by RDG is uniquely identified by its identifier pid, defined as a tuple [group ID (gid), source ID (sid), pkt seq. no. (seq)]. The protocol has four data structures. In the data management part, $pidList$ stores the pids of the received packets, and $Buffer$ temporarily stores these packets. The other two are for the membership management of the protocol. $gidList$ stores the identifiers of all groups that a node belongs to. $View$ is composed of three fields: (i) $AView$ stores the ids of known members, whose corresponding routing or location information is known; (ii) $PView$ stores the ids of known members, whose corresponding routing or location information is currently unavailable; and (iii) $RView$ stores the ids of members having indicated their willingness to leave.[5] All these records are divided into several subsets with each subset

[5] $AView$, $PView$, and $RView$ stand for *active* view, *passive* view, and *remove* view, repectively.

being dedicated to a certain group, i.e., each $node_i$ has four subrecords ($pidList_i^{gid}$, $Buffer_i^{gid}$, $gidList_i^{gid}$, and $View_i^{gid}$) for a certain group $\mathbb{G}$ (with identifier gid) that it belongs to. In addition, each record is of limited size, noted $|R|_{max}$, for a given record R.

RDG offers seven operations, which are grouped into three sessions corresponding to their functionality. The *join* session defines the behavior of the node interested in joining a group and the reactions of other group members. The *leave* session defines the behavior of the node intending to leave the group and the reactions. In the *gossip* session, newly received packets are periodically propagated by a node. Furthermore, nodes respond to the gossip messages received. In relation to the GOSSIP task, three protocol parameters are defined here: (i) the *fanout* (F) is the number of gossip destinations randomly selected from the $AView$ for each gossip emission, (ii) the *quiescence threshold* (τ_q) is related to each data packet: a packet will be removed from $Buffer$ after having been gossiped for τ_q rounds by individual nodes, and (iii) the *age threshold* (τ_a) limits the propagation range of each packet. These parameters are set by the upper layer to control the behavior of the protocol (see Section 4.2.3).

4.2.2 Join Session

A node intending to join a group floods the network with a GROUPREQUEST message to search for other group members while announcing its existence. Upon receiving such a message from a certain member, all members update their $AView$ with the new id. They also return a GROUPREPLY to the request initiator with probability P_{reply}. The probability is set by each node, according to its own estimation of the group size, in order to avoid GROUPREPLY storms. The initiator of the GROUPREQUEST also updates its $AView$ after receiving the GROUPREPLY. The detailed description can be found in [12].

By recording the route of each incoming packet, DSR ensures that a new element in $AView$ has a corresponding route entry in the DSR routing table. The validity of this relationship is periodically checked and the $AView$ and $PView$ are updated accordingly. When the size of

$AView$ drops below some threshold, the node has to reinitiate a *join* session.

4.2.3 Gossip/Leave Session

When a node wants to multicast a packet p, it inserts the packet in its $Buffer$ as shown in Fig. 2 (a). A node intending to leave a group sets a $leaveFlag$ for that group as shown in Fig. 2 (b). Each member of the group periodically (every T ms)[6] gossips packets stored in $Buffer$ to F other nodes randomly chosen from $AView$ (see Fig. 2 (c) lines 8∼16). It also piggybacks part of its view of the membership. A data packet is removed from $Buffer$ after having been gossiped for τ_q times. The SEND primitive is a direct call to the underlying unicast protocol, which will also be used by other parts of PILOT for the same purpose. If the node intends to leave, only the field of rmb is used (see Fig. 2 (c) lines 3∼6). As illustrated in Fig. 2 (d), a group member receiving a gossip packet will (i) update the $Buffer$ with new packets (lines 3∼8), (ii) remove the obsolete member from its $View$ (lines 10∼14), and (iii) add the new member to the $View$ (lines 16∼20). Note that a packet relayed τ_a times will not be gossiped again.

RDG performs message dissemination and membership tracking at the same time. Due to the node mobility and frequent membership changes, it is not practical to have a full membership view for each member. In fact, even if it is possible to have the ids of all members, there is no guarantee that the corresponding routing or location information is available. Our routing/location oriented membership management scheme tries to provide each member with a partial view, approximately random in nature, by exchanging membership information between members. The underlying scheme, together with sporadic losses and discoveries of the routing or location information[7], has a similar effect as the reshuffling of the partial view.

[6]In order to save bandwidth, we apply the *binary exponential backoff* algorithm to adjust the period when there is no new packet to be sent.

[7]The information could be lost due to the node mobility or the timeout of route cache timer. On the other hand, a node can also obtain new information by requesting it or tapping it from packets under transmission.

```
1: procedure RDG-CAST(gid, p)
2:     p.gid ← gid
3:     Buffer_i^gid ← Buffer_i^gid ∪ {p}
```

(a) Multicast

```
1: procedure LEAVE(gid)
2:     leaveFlag_i^gid ← true
```

(b) Node leave

```
1: task GOSSIP(gid)                    /* Executed every T ms */
2:   if leaveFlag_i^gid = true then
3:     p.rmb ← id_i
4:     DS (1) ← set ⊂_ℜ AView_i^gid s.t. |set| = F (2)
5:     for all id ∈ DS do
6:       SEND(id_i, id, p)
7:   else
8:     while Buffer ≠ ∅ do
9:       p ← pkt ∈ Buffer_i^gid
10:      if pkt has been gossiped τ_q times then
11:        Buffer_i^gid ← Buffer_i^gid \ {pkt}
12:      p.rmb ← rmb ∈_ℜ RView_i^gid
13:      p.mb(3) ← mb ∈_ℜ AView_i^gid ∪ PView_i^gid
14:      DS ← set ⊂_ℜ AView_i^gid s.t. |set| = F
15:      for all id ∈ DS do
16:        SEND(id_i, id, p)
```

(1) DS stands for *destination set*.

(2) A subscript $\Re$ stands for random selection.

(3) mb and rmb stands for member and removed member.

(c) Packet emission

```
1: upon RECEIVEGOSSIP(id_s, id_i, p) do
2:   /* Step 1: Update Buffer with new packets */
3:   if p.pid ∉ pidList_i^p.gid then
4:     pidList_i^p.gid ← pidList_i^p.gid ∪ {p.pid}
5:     p.age ← p.age + 1
6:     if p.age < τ_a then
7:       Buffer_i^p.gid ← Buffer_i^p.gid ∪ {p}
8:     DELIVER(p)                  /* to the upper layer */
9:   /* Step 2: Remove obsolete member from View */
10:  AView_i^p.gid ← AView_i^p.gid \ {p.rmb}
11:  PView_i^p.gid ← PView_i^p.gid \ {p.rmb}
12:  RView_i^p.gid ← RView_i^p.gid ∪ {p.rmb}
13:  while |RView_i^p.gid| > |RView_i^p.gid|_max do
14:    RView_i^p.gid ← RView_i^p.gid \ {rmb ∈_ℜ RView_i^p.gid}
15:  /* Step 3: Add new member to View */
16:  if p.mb ∉ (AView_i^p.gid ∪ PView_i^p.gid) then
17:    if there exists a route to that node then
18:      AView_i^p.gid ← AView_i^p.gid ∪ {p.mb}
19:    else
20:      PView_i^p.gid ← PView_i^p.gid ∪ {p.mb}
```

(d) Packet reception

Fig. 2. *Gossip/leave* session at node i

Considering that the locality of network traffic can reduce the network load, we apply a general optimization by raising the awareness of the topology. This optimization is based on the assumption that the underlying routing protocol can provide partial topological information. Our heuristics in the case of DSR work like this: for a given group member, different weights are assigned to the members in $AView$ according to the lengths of the routing paths to them, i.e., the longer a path the lower its weight, such that a "near" member is chosen with higher probability to relay a packet. A more detailed protocol description can be found in [12].

4.3 R^{2}DG: Continuous Packet Multicasting Service

If a stream of packets is multicast from a source, the pid sequence of received packets, at a certain group member, provides important information about packet loss. Based on RDG, our R^{2}DG protocol exploits this feature to enhance the reliability of multicasting.

R^{2}DG has its own data structures that are the same as for the data management part of RDG, except that the $Buffer$ is much larger than that of RDG in order to have enough packets to respond to a negative acknowledgement (or *pull*). Before invoking the RDG primitive, R^{2}DG

```
1: procedure P-RDG-CAST(gid, p)
2:     p.pull ← pid of the most recent missing packet
3:     RDG-CAST(gid, p)
4: task PULL(gid)              /* Executed periodically */
5:     p.pull ← pid of the most recent missing packet
6:     RDG-CAST(gid, p)
```

(a) Multicast and *pull* task

```
1: upon RECV(p) do
2:     pidList_i^{p.gid} ← pidList_i^{p.gid} ∪ {p.pid}
3:     Buffer_i^{p.gid} ← Buffer_i^{p.gid} ∪ {p}
4:     if p.pull ∈ pidList then
5:         SEND(id_i, p.sid, pkt_{p.pull})
6:     DELIVER(p)              /* to the upper layer */
```

(b) Packet reception and the response to *pull*

Fig. 3. Multicast and *pull* session at node i

(as shown in Fig. 3 (a) lines 1∼3) inserts the information about a missing packet into the packet header. In addition (task PULL in Fig. 3 (a) lines 4∼6), a packet with an empty payload (*pull*-packet) is periodically sent to the lower layer with similar information attached to it. The period is dynamically adjusted according to the number of missing packets. A group member receiving such a packet will try to respond to the pull with the packets it has, see Fig. 3 (b).

Considering that R^{2}DG passes pull-packets to RDG irregularly, RDG behaves intelligently when gossiping, in the sense that it tries to piggyback the pull information along with a data packet instead of sending the pull-packet directly.

4.4 PAN: Reliable Data Sharing Service

Our PAN system relies on the underlying RDG to provide reliable data sharing services. It includes two protocols: a client protocol and a server protocol, as shown in Fig. 1(b). In both

cases of update and query, a client sends a request to an arbitrary server in the STS.[8] This server, termed *agent* for that client, then performs a corresponding operation of the server protocol. We assume that all messages (updates and queries) for our protocols have relatively small sizes such that they can be fit into single network packets. This requirement is justified by considering the applications we aim at. For example, a public key is only hundreds of bits long and location information might be just a coordinate in a three-dimensional space. We further require that each message be uniquely identified by its identifier mid, which is a tuple [source ID (sid), object ID (oid), version no. (ver)][9], and that there is a way to establish a FIFO order among mids.[10] Since the client protocol, a one-to-one connection, can always implement certain mechanisms (e.g., ARQ [37]) to ensure reliability, we will not consider this protocol in our analysis and simulations. In the rest of this section we focus on the server protocol.

The server protocol maintains a quorum system building upon the STS with the support from the underlying RDG protocol. We distinguish two types of quorums within the quorum system. A quorum can be a *write quorum*, accessed by an update, or a *read quorum* in the case of an access by query. Throughout the presentation, as well as in the analysis and simulations of the server protocol, we will use two symbols $\xi_?$ and $\hat{\xi}_?$ to represent the *nominal quorum size* and the *real quorum size*, where "?" can be "W" for a write quorum or "R" for a read quorum. The nominal size is the number of servers that a certain update or query attempts to access, while the real size is the number of servers effectively accessed.

4.4.1 Server Update Protocol

The agent diffuses an update message m_u within the STS by invoking the RDG protocol, as shown in Fig. 4 (a). Two parameters F and τ_a (see Section 4.2.1 for the definition of these

[8]A client may have several ways to acquire information (e.g., identity and routing) about servers, depending on certain implementations of the STS initialization algorithm.

[9]The elements oid and sid stand for the ID of the data object to be queried or updated and of the object owner, respectively.

[10]$mid_1 > mid_2$ implies that $mid_1.sid = mid_2.sid \wedge mid_1.oid = mid_2.oid \wedge mid_1.ver > mid_2.ver$.

parameters) are set in order to control the size of the resulting quorum. In this paper, the value of τ_a is always set to ∞ for the server update protocol to simplify the analysis and simulations.

In order to keep track of the data access, each server keeps a record $midList$. It stores the mids of the most recent updates. Each server receiving a new update, including the agent, substitutes the old mid with the mid of the new update message in its $midList$ before delivering the message to the upper layer, as shown in Fig. 4 (b). At last, all servers that effectively receive the update form a write quorum. The size of the quorum, $\hat{\xi}_W$, is predictable thanks to the epidemic nature of the underlying gossip-based protocol, as we will see in Section 5.3.

1: **upon** UPDATE(m_u) /* m_u infused by client update */ **do**
2: $p.data \leftarrow m_u$
3: RDG-CAST(gid_{sts}, p, F, $\tau_a = \infty$)

(a) UPDATE emission

1: **upon** RECEIVEUPDATE(m_u) **do**
2: **if** $\nexists\ mid \in midList$ s.t. $mid \geq m_u.mid$ **then**
3: **for all** $mid \in midList$ s.t. $mid < m_u.mid$ **do**
4: $midList \leftarrow midList \setminus \{mid\}$
5: $midList \leftarrow midList \cup \{m_u.mid\}$
6: DELIVER(m_u)

(b) UPDATE reception

1: **upon** QUERY(m_q) /* m_q infused by client query */ **do**
2: **if** $\exists\ mid \in midList$ s.t. $mid > m_q.mid$ **then**
3: $m_q.ver \leftarrow mid.ver$
4: $Counter_{m_q} \leftarrow 0$
5: $p.data \leftarrow m_q$
6: RDG-CAST(gid_{sts}, p, $F = \xi_R - 1$, $\tau_a = 1$)
7: $timer_{m_q} \leftarrow 0$ /* set a timer for the query */
8: **upon** RECEIVEQUERY(m_q) **do**
9: **if** $\exists\ mid \in midList$ s.t. $mid > m_q.mid$ **then**
10: $m_q.mid \leftarrow mid$
11: $m_q.data \leftarrow$ queried data object(1)
12: SEND(id_i, id_{agent}, m_q)

(1) The data object is retrieved from the upper layer with certain callback procedures.

(c) QUERY emission and reception

1: **upon** RECEIVEQUERYREPLY(m_q) **do**
2: $Counter_{m_q} \leftarrow Counter_{m_q} + 1$
3: **if** $\nexists\ mid \in midList$ s.t. $mid \geq m_q.mid$ **then**
4: **for all** $mid \in midList$ s.t. $mid < m_q.mid$ **do**
5: $midList \leftarrow midList \setminus \{mid\}$
6: $midList \leftarrow midList \cup \{m_q.mid\}$
7: DELIVER(m_q)
8: **if** $Counter_{m_q} = \xi_R - 1$ **then**
9: Invoke the client query protocol
10: **upon** $timer_{m_q} = timeout$ **do**
11: Invoke the client query protocol

(d) REPLY reception at an agent

Fig. 4. UPDATE/QUERY operation at node i

4.4.2 Server Query Protocol

In the case of a query, the agent again uses RDG to disseminate the query message to other servers. The value of τ_a is set to 1 to simplify the protocol evaluation later on. Also, since we consider that the arrival rate of queries is higher than that of updates in most cases, it is justifiable to have a relatively small read quorum.[11]

After receiving a query message from a client, the agent sends it to other servers immediately, along with the version number of the corresponding local data object. The agent also sets a counter and a timer in order to guarantee proper termination of the query session (Fig. 4 (c) lines 1∼7). Each server belonging to the read quorum, upon receiving the message, responds with its own copy of the data object, if its version is more recent than the one of the agent (Fig. 4 (c) lines 8∼12). The agent always delivers a new update returned from other servers. It invokes the corresponding client protocol, after every request either yields a reply or times out, as illustrated in Fig. 4 (d).

4.5 Examples of Protocol Operations

Fig. 5(a) gives a visual illustration of the behaviour of our RDG protocol with respect to the dissemination of one packet, assuming a single group $\mathbb{G}$ of size $|\mathbb{G}| = 10$ within a 20 nodes network. Another example in Fig. 5(b) illustrates a simple execution of our PAN system in a network of 50 nodes, assuming an STS consisting of 25 nodes.

4.6 Comparing PILOT with Randomized Database Group

In this section, we compare the work of Haas and Liang [29] with PAN. The comparisons are qualitative rather than quantitative, because [29] does not provide simulation results to evaluate

[11]By setting $\tau_a = 1$, the nominal read quorum size ξ_R is directly determined by F (Fig. 4 (c) line 6), since a server receiving the query will not relay it further.

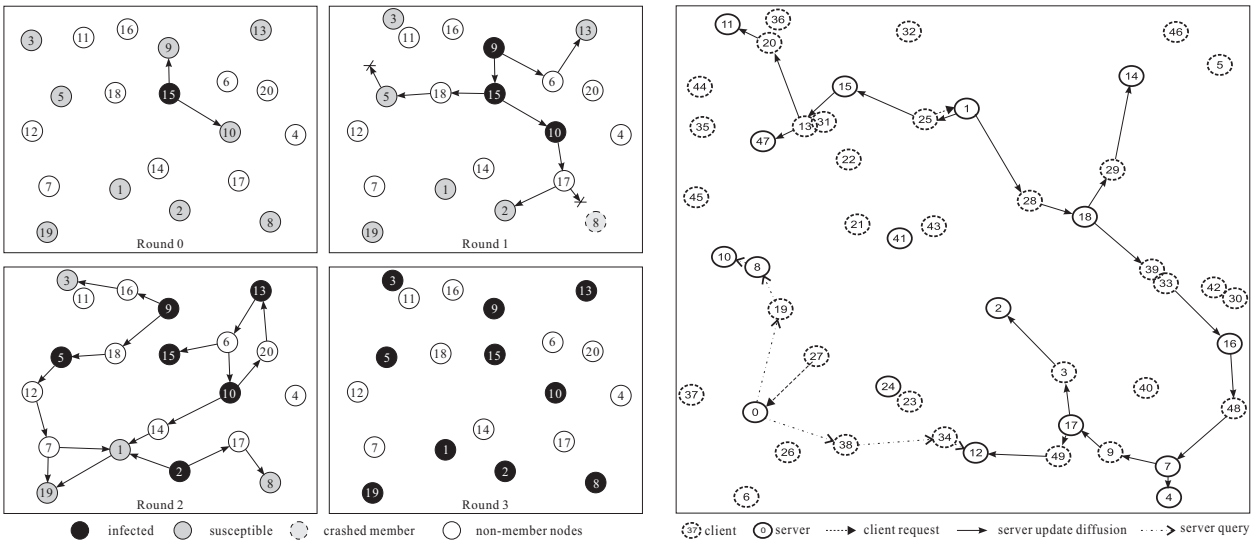


Fig. 5. Illustrations of PILOT. (a) An example of one "run" of RDG with $F = 2$ and $\tau_q = 2$ within a group of size 10. The packet initiated by member 15 infects the whole group in only 3 rounds in spite of the fact that nodes move and even fail. A member may receive duplicates of the same packet (e.g., member 1 at round 2). On the other hand, the packet can get lost at a certain round due to nodes crashing or moving (e.g., members 8 and 3 in round 1), but these losses will be compensated with high probability at a later round. (b) An illustration of an update/query pair in PAN within a network of 50 nodes located in a square area of 1km^2. When node 25 wants to perform an update, it sends a request to its agent, node 1. The request of this update is diffused to other servers by node 1, using the gossip-based scheme (Only the valid transmissions are shown here. Duplicated transmissions are omitted to simplify the visualization.). If node 27 wants to access the data, it also requests its agent, node 0. Node 0 in turn requests other servers, node 8, 10, and 12. In this case, node 12 is the intersection of the read and write quorum. It is able to reply the requested data of node 25 to node 27. The query reply is omitted here for simplicity.

the system performance and to confirm the precision of their numerical analysis. In a nutshell, PAN outperforms randomized database group in two aspects. On one hand, the protocol used to access the database group in [29] consists in multiple unicasts, based on the assumption of perfect routing information. Obviously such an approach fails under the more realistic assumption of incomplete routing information, while PAN can cope with such incompleteness. On the other hand, the symmetric construction of quorum systems in [29], i.e., the same size for all quorums, is not suitable for all replication systems (for instance when the arrival rates of queries and updates, respectively, diverge strongly). PAN, on the contrary, can adapt to a given situation

by appropriately adjusting parameters. As far as the analytical methodology is concerned, the model in [29] is more application oriented than the one of PAN. It provides an insight into the probabilistic quorum systems from a different perspective.

5 ANALYSIS

In this section, we show that the two metrics, $\mathcal{R}_d$ and $\mathcal{N}_l$ (defined in Section 3.2), are predictable given certain protocol parameters and information about the network. These analytical results are confirmed by simulations in the next section. Since the behavior of R^{2}DG pull depends on far more factors than that of RDG gossip, we will not consider this part of the protocol in the analysis. However, we will show the enhanced reliability by simulations.

5.1 Model

For the multicast protocol, we consider a single group $\mathbb{G}$ composed of $|\mathbb{G}| = n$ members and observe its behavior in terms of the dissemination of a *single* packet ("one run"), but also a *continuous* stream of packets (which is more realistic than related research proposals considering only the "one run" part). Each gossiping operation is modeled as a uniform random selection of F members out of n, i.e., without considering the topology-awareness, in order to simplify the tractability. According to the terminology of epidemiology [38], a member that has received a certain packet is termed *infected*, otherwise *susceptible*. An infected member attempting to share the packet with others (i.e., a member who keeps gossiping the packet) is called *infectious*. We analyze our protocol in a network composed of a static set of nodes running closely "synchronized". More precisely, nodes gossip in synchronous rounds (T ms, identical for all nodes), and there is an upper bound on the network latency which is smaller than T.

The probability of packet loss is closely related to the movement and traffic pattern, as well as to the length of the considered routing path. By assuming an identical and independent probability of failure p_f for each hop along a routing path in a certain network environment, the probability

of losing a certain gossip message can be expressed as a function of the number of hops, H, of that routing path. We further assume that the lengths H of all routing paths between any two members follow the same distribution $f(h)$. On the other hand, p_f can be split into two parts: (i) p_{f_c} represents the probability of packet loss due to node crash; (ii) $p_{f_{mo}}$ reflects the effects of node mobility and buffer overflow. Since $p_{f_c} \ll p_{f_{mo}}$ in general for mobile wireless networks, we directly use $p_{f_{mo}}$ to approximate p_f.

As for the data sharing service, we consider only the server protocol (including both update and query protocols) for analysis. The STS is assumed to consist of n servers. We also assume that query and update accesses arrive randomly at an arbitrary server, following Poisson processes with intensities of λ_q and λ_u, respectively. By further assuming that these two processes are independent, the overall access rate is given by $\lambda_o = \lambda_q + \lambda_u$.

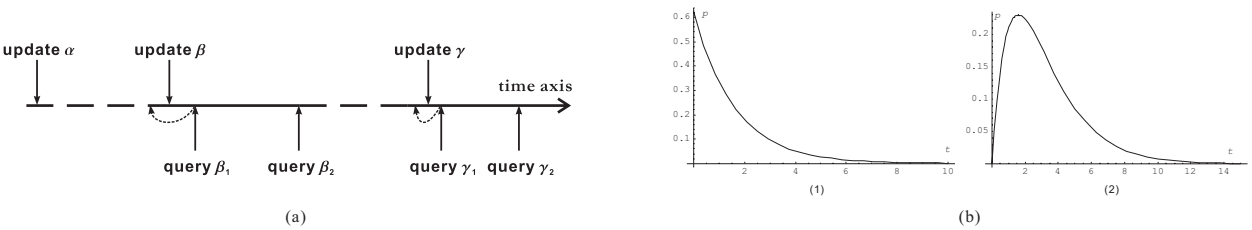


Fig. 6. Time intervals between events and their distributions. (a) The occurrence of events in terms of absolute time. (b) The distributions of time interval between two events: (1) exponential distribution for consecutive events, (2) *Erlang* distribution for non-consecutive events.

The dissemination process of the server update is performed by RDG. As this process finishes, all infected servers form a write quorum with real size $\hat{\xi}_W$ following a certain probability distribution. We consider only the second query to a data object that was modified by the most recent update, while considering the first query as happening before the update.[12] For example, as shown in Fig. 6(a), only the pairs of (update β, query β_2) and (update γ, query γ_2) are considered, whereas queries β_1 or γ_1 are supposed to request previous updates (i.e., updates α

[12]The time of an event is when it happens at an agent.

and β, respectively). This assumption makes sense when we consider the time with respect to a server where updates and queries arrive, and also the property of a Poisson process shown in Fig. 6(b). Since there is always some delay for the message dissemination, the probability that the actual occurrence of events will follow the order of our assumption at that server is very high, according to different distributions of the time interval between two events within a Poisson process (see Fig. 6 (b)). This makes the present analysis a "viable" lower bound.

We continue using p_f to represent the network condition, but an empirical value p_e is also used in the case of queries to represent the server *unavailability* due to failure, at any time instant. One might argue that the server failure should be treated as a Poisson process [29], but this is not justifiable with a failure recovery model, which is usually the case in ad hoc networks (e.g., nodes switching off for the purpose of battery replacement or operating system rebooting).

5.2 Stochastic Behavior of RDG

Considering a packet multicast by a member, we use $S_r \in \{0, \cdots, n\}$ to denote the number of members infected with the packet **after** round r. With the convention that $\Pr\{S_r = 0\} = 1$ for $r < 0$, it is easy to show that the sequence of random vectors $\mathbf{S}_r = [S_r, S_{r-1}, \cdots, S_{r-\tau_q}]^T_{r \geq 0}$ forms a Markov chain with values taken from the *state space* $\mathcal{E} = \overbrace{\{0, \cdots, n\} \times \cdots \times \{0, \cdots, n\}}^{\tau_q + 1}$.

1) Recurrence Relation: Given the probability p that a certain member is infected by a specific gossip message, $q = 1 - p$ represents the probability of non-infection. Let $S_r = i$ (the number of infected members) and $S_r - S_{r-\tau_q} = k$ (the number of infectious members) in the current round; we introduce a binary random variable, X_l, for each of the remaining $n - i$ susceptible members, where $\Pr\{X_l = 0\} = q^k$, i.e., the probability that a certain susceptible member is not infected in the next round is the probability that it is not infected by any of the k infectious members. It is clear that $S_{r+1} - S_r = \sum_{l=1}^{n-i} X_l$ follows a binomial distribution. Let j be the

number of infected members in the next round; the transition probability is expressed as:

$$\begin{aligned} & \Pr\{S_{r+1} = j \mid S_r = i, S_r - S_{r-\tau_q} = k\} \\ = \; & \Pr\{\sum_{l=1}^{n-i} X_l = j - i \mid S_r - S_{r-\tau_q} = k\} \\ = \; & \begin{cases} \binom{n-i}{j-i}(1-q^k)^{j-i} q^{k(n-j)} & j \geq i \\ 0 & j < i \end{cases} \end{aligned} \tag{1}$$

which leads to the following global balance equation of the chain:

$$\Pr\{\mathbf{S}_{r+1} = \mathbf{s}_{r+1}\} = \sum_{i_{\tau_q}=0}^{i_{\tau_q-1}} \binom{n-i}{j-i} (1-q^{i-i_{\tau_q}})^{j-i} q^{(i-i_{\tau_q})(n-j)} \Pr\{\mathbf{S}_r = \mathbf{s}_r\} \tag{2}$$

where $\mathbf{s}_r = [i, i_1, \cdots, i_{\tau_q}]^T$, $\mathbf{s}_{r+1} = [j, i, i_1, \cdots, i_{\tau_q-1}]^T$, and $i = i_0$. Let the column vector ν_r, with $\nu_r(i) = \Pr\{S_r = i\}$ as its ith element, be the marginal distribution of S_r. Given the initial distribution $\nu_0 = [0, 1, 0, \cdots, 0]^T$ and (2), ν_r is then computed as:

$$\nu_r(i) = \sum_{i_1=0}^{i} \sum_{i_2=0}^{i_1} \cdots \sum_{i_{\tau_q}=0}^{i_{\tau_q-1}} \Pr\{\mathbf{S}_r = \mathbf{s}_r\} \tag{3}$$

2) Computation of p: According to our assumptions, the probability of infection p can be estimated by taking two conditions into account: (i) the considered node is chosen as the gossip destination and (ii) the gossip message is successfully received. This results in the following expression (remember that F is the protocol parameter fanout):

$$p = \overbrace{P_{gossip}}^{\text{(i)}} \overbrace{P_{succ}}^{\text{(ii)}} = \left(\frac{F}{n-1}\right) P_{succ} \tag{4}$$

Given a certain length (in hops) h of a routing path, the probability of a successful delivery is expressed as $P_{succ} = (1-p_f)^h$, i.e., there is no failure in each of the h hops. So we have:

$$P_{succ} = \sum_h (1-p_f)^h \Pr\{H = h\} = \mathbf{E}_H[(1-p_f)^H] \tag{5}$$

Therefore, p is expressed as:

$$p = \left(\frac{F}{n-1}\right) \mathbf{E}_H[(1-p_f)^H] \tag{6}$$

The distribution of H and the value of p_f are the network information we need. We refer to [12] for discussions about their estimations.

3) Reliability Degree $\mathcal{R}_{ds}$ and $\mathcal{R}_{dc}$: With the recurrence relation (3) of the single packet dissemination, the reliability degree can be expressed[13] in terms of $\nu(i)$ as follows. Note that the distribution of $\mathcal{R}_{ds}$ is always related to the group size n, while the distribution of $\mathcal{R}_{dc}$ is related to the number of packets in a stream, denoted by M in the formula.

$$\text{cdf of } \mathcal{R}_{ds}: \qquad \mathcal{F}_n(x) = \sum_{i=1}^{\lfloor nx \rfloor} \nu(i) \tag{7}$$

$$\text{cdf of } \mathcal{R}_{dc}: \qquad \mathcal{F}_M(x) = \sum_{i=0}^{\lfloor Mx \rfloor} \binom{M}{i} p_1^i (1-p_1)^{M-i} \tag{8}$$

where $p_1 = \sum i \cdot \nu(i)/n$ is the probability that a certain group member receives a single packet in a stream. Here we assume that the receptions of two distinct packets are independent events.

4) Network Load $\mathcal{N}_l$: The $\mathcal{N}_l$ for single packet dissemination is estimated straightforwardly by counting the number of unicast packets sent and the number of hops traveled by each of them:

$$\mathcal{N}_l = \mathbf{E}[S_{\tau_a}] \cdot F \cdot \tau_q \cdot \mathbf{E}[H] \tag{9}$$

Recall that τ_a limits the number of gossip rounds and τ_q defines how many times a packet is repeatedly relayed by a certain group member. The expression for $\mathcal{N}_l$ in the case of continuous packet dissemination is omitted as it becomes trivial with (9) and a given λ_o. This prediction is relatively rough because it is hard to find a way to precisely estimate the distribution of H as the distribution depends on several factors.

5.3 Stochastic Behavior of PAN

Since PAN directly uses RDG to diffuse an update, the distribution of $\hat{\xi}_W^r$ can be estimated with (3). The distribution of $\hat{\xi}_R$ can be expressed in a similar but more precise way, since τ_a is set to 1 (see Section 4.4.2). $\mathcal{N}_l$ is computable given the two distributions, but information about the time interval between a query and an update is necessary to compute $\mathcal{R}_{da}$.

[13]The subscript r is omitted hereafter, because we always consider the final distribution (i.e., after the last round).

5) Reliability Degree $\mathcal{R}_{da}$: According to the definition and the protocol description, this value is in fact the probability that a read quorum intersects the most recent corresponding write quorum. More precisely, we are looking for the probability that two subsets with sizes $\hat{\xi}_W$ and $\hat{\xi}_R$, taken from a set of n servers, intersect. Note that $\hat{\xi}_R$ is defined as the number of servers that effectively reply to the query back to its forwarding agent.

There exists an $\tilde{r}$ for which the dissemination process is finished, i.e., no new server is infected when $r \geq \tilde{r}$. Based on the assumption of synchronization, we divide the time axis after a given update event β into $\tilde{r}+1$ intervals, as shown in Fig. 7. A read quorum, resulting from a query

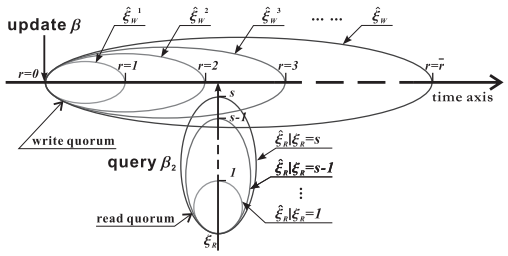


Fig. 7. Incremental processes of read and write quorum size: $\hat{\xi}_W$ increases round by round, while $\hat{\xi}_R$ increases with the amount of queries sent by an agent.

happening in-between two consecutive gossip rounds r and $r+1$, would have to intersect a write quorum of size $\hat{\xi}_W^r$ with a distribution ν_r. In order to find the probability of intersection, we need to calculate the read quorum size $\hat{\xi}_R$ (with a distribution μ) and p_r, the probability that the query event occurs in-between rounds r and $r+1$.

The distribution of $\hat{\xi}_R$, conditioned on $\xi_R = s$, is calculated as follows, with an initial value[14] $\Pr\{\hat{\xi}_R = 1 | \xi_R = 1\} = 1$ and the convention $\Pr\{\hat{\xi}_R = k | \xi_R = s\} = 0$ if $s < 1, k < 1$ or $k > s$:

$$\begin{aligned} \mu_s(k) &= \Pr\{\hat{\xi}_R = k | \xi_R = s\} = \mu_{s-1}(k-1)p + \mu_{s-1}(k)(1-p) \\ &= \Pr\{\hat{\xi}_R = k-1 | \xi_R = s-1\} \cdot p \\ &+ \Pr\{\hat{\xi}_R = k | \xi_R = s-1\} \cdot (1-p) \qquad k = 1, \cdots, s \text{ and } s \geq 2 \end{aligned} \tag{10}$$

[14]Because the agent, one of the servers, has already received the query, it is sure that $\hat{\xi}_R = 1$, if ξ_R is set to 1.

where $p = \mathbf{E}_H[(1-p_f)^{2H}](1-p_e)$ is the probability that the agent forwarding a query receives the reply from a server belonging to the corresponding read quorum. The estimation of μ is somewhat conservative because servers with a relatively old data version do not reply to a query.

The time interval between an update and the second query to it is characterized by an Erlang distribution $\lambda_q^2 t e^{-\lambda_q t}$, with the assumption of a Poisson arrival process. Therefore, we have

$$p_r = \begin{cases} \int_{t_r}^{t_{r+1}} \lambda_q^2 t e^{-\lambda_q t} dt & r < \tilde{r} \\ \int_{t_r}^{\infty} \lambda_q^2 t e^{-\lambda_q t} dt & r = \tilde{r} \end{cases} \tag{11}$$

Now, the probability of intersection, i.e., $\mathcal{R}_{da}$, is expressed by taking an average over all possible cases:

$$\mathcal{R}_{da} = \sum_{r=0}^{\tilde{r}} \sum_{i=1}^{n} \sum_{j=1}^{s} \left(1 - \frac{\binom{n-\hat{\xi}_W^r}{\hat{\xi}_R}}{\binom{n}{\hat{\xi}_R}} \right) \mu_s(j) \nu_r(i) p_r \tag{12}$$

6) Network Load $\mathcal{N}_l$: For a certain $\mathcal{R}_{da}$ with its parameter pair F and ξ_R, we evaluate the corresponding $\mathcal{N}_l$ by averaging the load over a certain time unit (e.g., 1s), taking into account the arrival rate of updates and queries. Therefore, the loads generated by a single update and query are calculated separately, and then $\mathcal{N}_l$ is obtained by summing the products of the loads of the individual operations and their corresponding arrival rates.

$$\mathcal{L}_W = \mathbf{E}[\hat{\xi}_W] \cdot F \cdot \tau_q \cdot \mathbf{E}[H] \tag{13}$$

$$\mathcal{L}_R = 2 \cdot \xi_R \cdot \mathbf{E}[H] \tag{14}$$

$$\mathcal{N}_l = \lambda_u \mathcal{L}_W + \lambda_q \mathcal{L}_R \tag{15}$$

This estimation is conservative in the same sense as we mentioned before. Again, it is relatively rough compared with the one for $\mathcal{R}_{da}$, because we do not take into account the following two facts: (i) many packets get dropped before reaching their destinations, and (ii) packets, especially those eventually dropped, may travel quite a long way due to stale routing information. We will show with simulations that the former fact has a dominating effect in most cases, but these facts tend to offset each other in some cases.

6 SIMULATIONS

This section presents the simulation results of our PILOT system in five parts. Two subsections, 6.2 and 6.3, are dedicated to RDG/R^{2}DG and the other three, 6.4 to 6.6, are devoted to PAN. The main goal is to confirm our claim that both the reliability degree $\mathcal{R}_d$ and the network load $\mathcal{N}_l$ of PILOT are predictable. The impact of the message arrival rate λ_o and of the server failures p_e on PAN is also investigated by simulations in different settings. We refer to [12] for comparisons of simulation results between RDG/R^{2}DG and AG [25].

6.1 Model and Parameters

The simulator we use is *ns-2* [39] with the Monarch Project wireless and mobile extensions. It provides both implementations of DSR and wireless MAC, based on the Lucent WaveLAN IEEE 802.11 product, with a 2Mbps transmission rate and a nominal range of 250m. We adopt the two-ray ground reflection model [40] as the radio propagation model.

We simulate ad hoc networks in a square area of 1km^2. The movement pattern is defined by the "random waypoint" model [41]. The simulation parameters such as network size and maximum node speed are specified for each simulation. The STS for PAN always contains half of the network nodes. We do not justify this number[15], but only use it as an example. The servers in the STS are assumed to be predefined in order to simplify the simulation[16]. The client protocol is omitted to reduce side effects.

The gossip period is set to 200ms. For RDG/R^{2}DG, a CBR traffic generator produces 64 byte packets at regular intervals of 200ms, which gives a $\lambda_o = 5$pkt/s. The effect of the sending rate will be investigated in our future work. The arrival of queries or updates in PAN is emulated by

[15]It is not the goal of this paper to find the optimal size for an STS, but we note that generally, the larger the size, the heavier the network is loaded, whereas the load on individual servers becomes smaller.

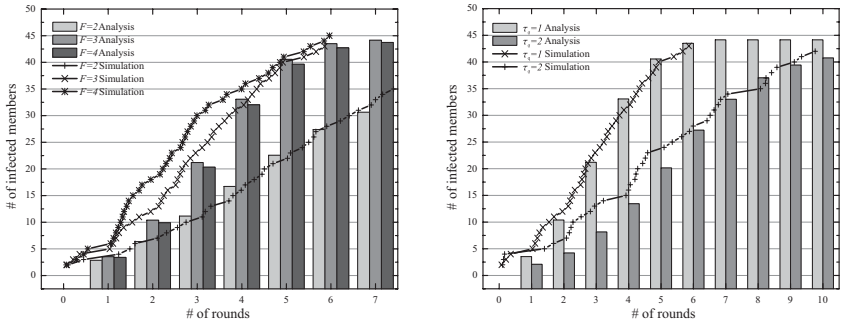
[16]Although the clustering algorithm is a popular way to elect some representatives of the network, introducing such an algorithm into our simulation may only bring more overhead to this task, without any help to show the essence of our system.

a Poisson traffic source attached to each server, generating packets of 128 bytes with rate λ_o. We first investigate the impact of the overall access rate λ_o on the performance of PAN, then we take an appropriate value for all simulations. Due to space limitations, we use $\lambda_o = 8\lambda_u$ for all simulations. With certain simulation parameters (network size, maximum speed, pause time, and arrival rate), we vary the protocol parameters F and ξ_R in order to show the trade-off between the two metrics $\mathcal{R}_{da}$ and $\mathcal{N}_l$. As the last simulation parameter, p_e is first set to 1%, and then varied to show the sensitivity of PAN to server failures.

Both RDG/R^{2}DG and PAN are operated over 400 seconds of simulated time. The first 50 seconds of the simulation are used for system initialization. Then each traffic source continues generating traffic according to the predefined intensity until the end. Each simulation is carried out 10 times with different scenario files created by *ns-2*.

6.2 Single Packet Dissemination Reliability $\mathcal{R}_{ds}$

Fig. 8 shows the analytical and simulation results (i.e., the evolution of the infection processes) of the basic RDG protocol. The figure contrasts one process with another instead of providing the value of $\mathcal{R}_{ds}$ explicitly. These comparisons basically confirm that the theoretical prediction of the relationship between the reliability and the latency is valid.



(a) τ_q=*1* with different values for *F* (b) *F=3* with different values for τ_q

Fig. 8. Average number of infected members (simulation) and expected number of infected members (analysis) in time (expressed in rounds) with $n = 50$ in a network of 100 nodes. Each node has a maximum speed of 2m/s and an average pause time of 40s.

It is easy to observe that the reliability of the protocol with $F = 3$ is better than the one with $F = 2$, because the fanout has a significant effect on the reliability. However, when we further increase the fanout, the reliability decreases instead of increasing (analysis) or only marginally increases (simulation). The reason is that increasing the fanout has the same effect as increasing the number of connections, and p_f increases dramatically because of the network congestion. A similar reason accounts for what happens when τ_q changes from 1 to 2.

In fact, there is always a trade-off between certain requirements on reliability and the introduced overhead, characterized by the values of F and τ_q. Considering the network capacity imposes a further limitation not considered in other research proposals (considerably large F [42] or unbounded τ_q [22]). Therefore, for all simulations later in this paper, we always take $F \leq 3$ and $\tau_q = 1$ for RDG.

6.3 Continuous Packet Dissemination Reliability $\mathcal{R}_{dc}$ and Network Load $\mathcal{N}_l$

Fig. 9 shows $\mathcal{R}_{dc}$ and $\mathcal{N}_l$ of both RDG and R^{2}DG with different mobility patterns and group sizes. We provide here the mean value of $\mathcal{R}_{dc}$ and its standard deviation, which characterize

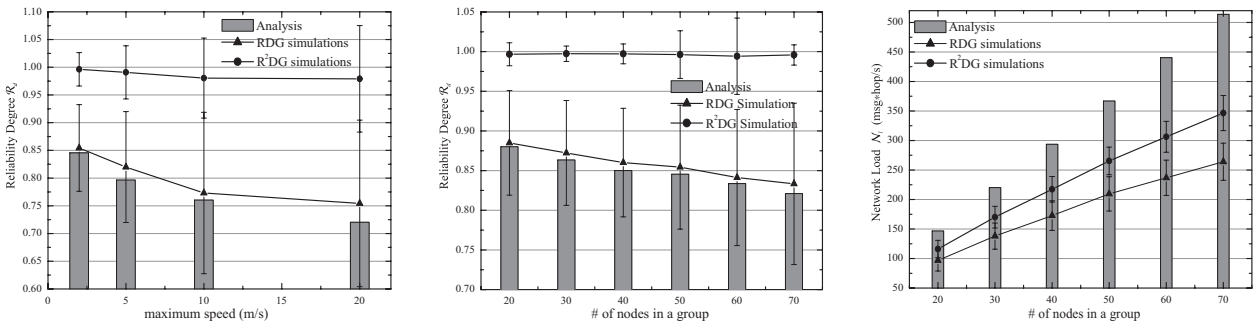


Fig. 9. Reliability degree $\mathcal{R}_{dc}$ and network load $\mathcal{N}_l$ vs. mobility and group size in 100 nodes networks

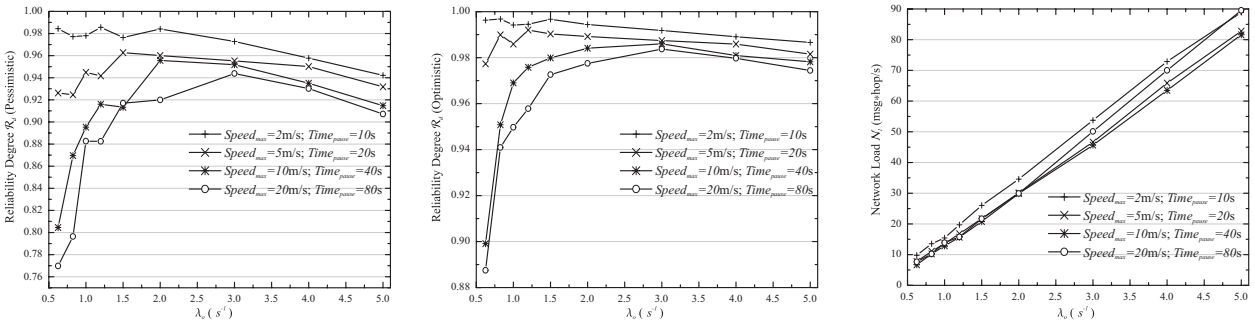
the distribution function $\mathcal{F}$. The figures again exhibit the similarity between the simulation and analytical results with respect to RDG (see Section 5.2 for the explanation of the rough prediction

on $\mathcal{N}_l$). As expected, R^{2}DG always performs better than RDG in terms of reliability, while the improvement is significant in high mobility and large group scenarios, thanks to the gossiper-pull mechanism. We also note that only a slight reliability degradation is observed (especially in the case of R^{2}DG) when the mobility or group size is increased (with a sub-linear increment of $\mathcal{N}_l$ in the case of increasing group size), illustrating the scalability of our protocols.

Note that two simulation parameters are paired to represent the mobility pattern such that each node has a maximum speed of 2m/s, 5m/s, 10m/s, and 20m/s, and a corresponding average pause time of 10s, 20s, 40s, and 80s, respectively (maximum speed is used as a symbol of the mobility pair in this case). This concept of mobility pattern will be used throughout the rest of this section.

6.4 Impact of λ_o on PAN Performance

Fig. 10 shows the performance of PAN (assuming $F = 2$ and $\xi_R = 4$) with respect to λ_o, the overall access rate. We observe that PAN performs in a relatively stable way for $1.5\text{s}^{-1} \leq \lambda_o < 3\text{s}^{-1}$, and $\mathcal{R}_{da}$ begins to degrade if we further increase λ_o, since the request arrival rate becomes larger than the service rate that PAN can provide. It is also natural to see that $\mathcal{N}_l$



(a) Pessimistic reliability degree (b) Optimistic reliability degree (c) Network Load

Fig. 10. Reliability degree $\mathcal{R}_{da}$ and network load $\mathcal{N}_l$ vs. overall access rate λ_o for 50 nodes networks.

increases linearly with λ_o by (15). However, it may seem somewhat odd to observe that $\mathcal{R}_{da}$ is very low in high mobility scenarios, when $\lambda_o < 1.5\text{s}^{-1}$. The main reason for this is the increased

amount of stale routing information. In practice, this effect does not appear in the presence of background traffic. This problem can also be solved actively by requiring each STS server to send control packets during idle time in order to keep routing information fresh. Based on these observations, we apply $\lambda_o = 2\text{s}^{-1}$ for all other simulations.

The evaluations of $\mathcal{R}_{da}$ are presented in two ways. The "pessimistic" $\mathcal{R}_{da}$ refers to the probability that a query reaches the most recent update (with the same assumption as in Section 5.1 about the event order), whereas for the "optimistic" one, we consider a query to be successful even if it only retrieves the result of an update that occurred right before the most recent update. This second evaluation makes sense because, in practice, there are different data objects stored in an STS, and the probability that a queried data object has been modified by the most recent update is quite low. We will use these notations for all graph illustrations in the rest of this section.

6.5 Access Reliability $\mathcal{R}_{da}$ and Network Load $\mathcal{N}_l$

Fig. 11 shows comparisons between simulation and analytical results for networks of "normal" density, i.e., 50 nodes in an area of 1km^2, and "high" density, i.e., 100 nodes in an area of 1km^2. We vary the maximum speed and pause time to test the impact of mobility on the performance of PAN. The protocol parameters F and ξ_R are adjusted to cope with the increased network size. We note that a real number $x.y$ for the value of F means that each server, when gossiping an update, takes $F = x$ with probability $1 - y/10$ and $F = x + 1$ with probability $y/10$.

We make the following observations: (i) The simulation and analytical results of $\mathcal{R}_{da}$ match very well; this confirms the predictability on $\mathcal{R}_{da}$. (ii) The analytical results of $\mathcal{N}_l$ provide certain information about the system overhead, such as the trend of its changes in different situations. (iii) The optimistic $\mathcal{R}_{da}$ is always much higher than the pessimistic one; this basically means that the potential of PAN is much higher than what could be expected from the analytical results. (iv) As the network size and the maximum node speed grow, protocol parameters have to be

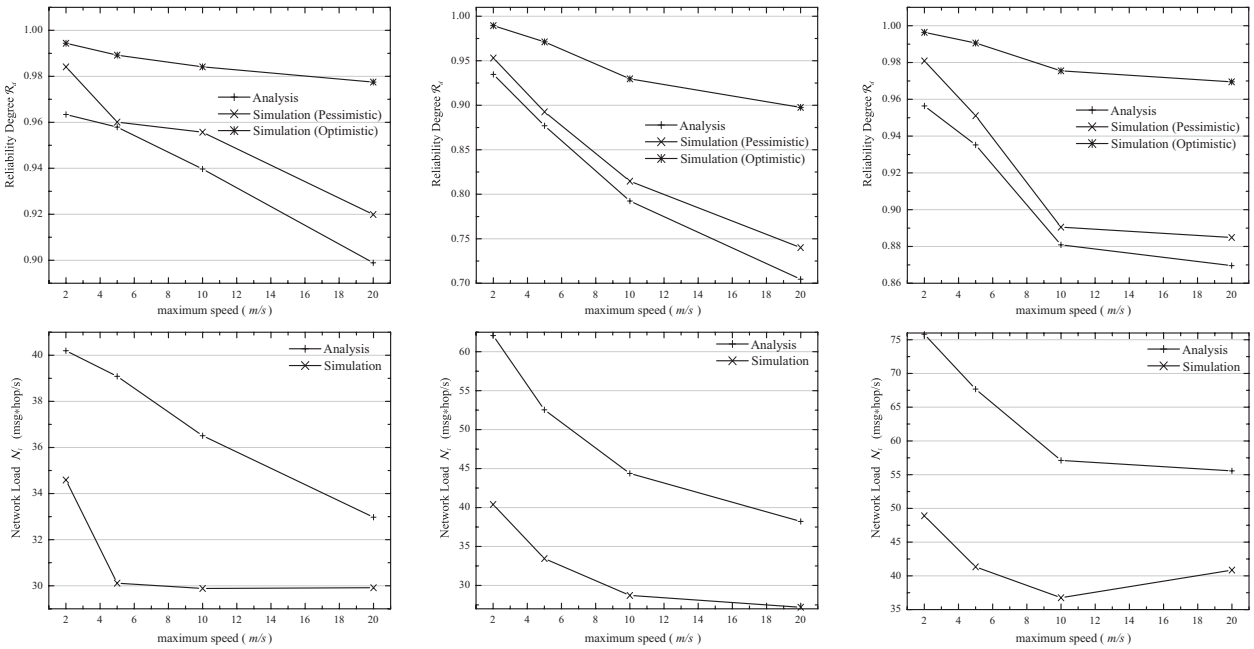


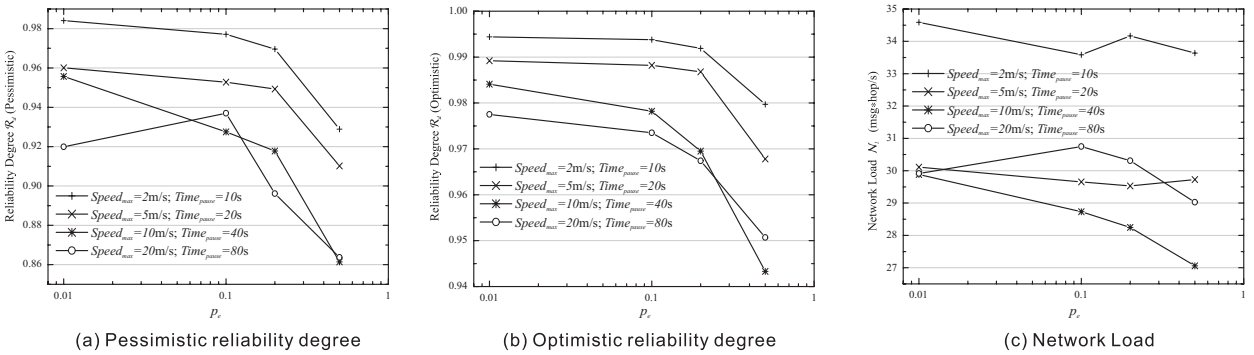
Fig. 11. Analytical and simulation results for reliability degree $\mathcal{R}_d$ and network load $\mathcal{N}_l$ vs. mobility pattern.

adjusted to maintain a good performance of $\mathcal{R}_{da}$, at the cost of an increased system overhead.

6.6 Sensitivity to Server Unavailability p_e

According to the simulation results shown in Fig. 12, the sensitivity of PAN (assuming $F = 2$ and $\xi_R = 4$) to p_e increases as the node mobility grows. In addition, the sensitivity of PAN considering optimistic $\mathcal{R}_{da}$ is lower than the sensitivity considering pessimistic $\mathcal{R}_{da}$.

We also observe that the increase of p_e leads to an improvement of $\mathcal{R}_{da}$ in some cases. This paradox indeed suggests a way to optimize our system, i.e., a server belonging to a certain read quorum would not always try to reply to a query back to its agent, even if the server is "alive" and has a new version of the queried data object. With such a behavior, PAN could avoid the case where more than one server replies to an agent with the same data object, thereby reducing the probability of packet collisions and, in turn, improving $\mathcal{R}_{da}$. However, we do not actually apply this optimization to PAN, because it is not as stable as the topology-awareness optimization in



(a) Pessimistic reliability degree (b) Optimistic reliability degree (c) Network Load

Fig. 12. Reliability degree $\mathcal{R}_{da}$ and network load $\mathcal{N}_l$ vs. server unavailability p_e for 50 nodes networks.

dynamic environments.

7 CONCLUSION

In this paper, we are concerned with probabilistic reliable group communication in mobile ad hoc networks. We have studied two fundamental aspects, i.e., multicast and data sharing, within this framework and specified performance metrics that take the peculiarities of mobile ad hoc networks into account. We have proposed our PILOT system as a solution, based on the principle of gossip mechanisms and probabilistic quorum systems. The performance of PILOT has been analyzed by making use of, notably, epidemic theory. The evaluation and investigation of PILOT have also been carried out by simulations in *ns-2*.

As the first step towards building a probabilistic group communication toolkit, our PILOT system consists of two layers: RDG, at the bottom layer, is a gossip-based probabilistic reliable multicast protocol. At the upper layer, two dedicated services, R^{2}DG and PAN, provide continuous reliable packet dissemination and reliable data sharing, respectively. Our main contributions are: (i) an ad hoc adapted gossip mechanism, (ii) a hybrid gossip including both push and pull, (iii) gossip-based quorum access protocols, and (iv) an asymmetric quorum construction.

We have proposed an analytical model to predict the performance of both RDG and PAN. The validity of these predictions has been evaluated by simulations. The results show that our

analytical model provides predictions that are adequate for tuning the tradeoff between reliability degree $\mathcal{R}_d$ and network load $\mathcal{N}_l$. Our simulation results also show that, even under frequent topology changes, the reliability degrees of RDG/R^{2}DG and PAN are fairly high in practice. Finally, we have investigated also other aspects of PAN with intensive simulations, which confirm its robustness, in the sense that it can sustain a large access rate λ_o, different network sizes, and up to 50% server failures.

We are in the process of determining a probabilistic notion of membership consistency and improving the analytical model by taking this notion into account. In addition, we are considering other models [42, 29] in order to further understand the benefits of gossip-based protocols and to provide numerical comparisons between PILOT and similar systems for ad hoc networks, which would better justify the deployment of PILOT. Finally, we intend to take into consideration, in our simulations, the recently recommended modifications to the "random waypoint" model [43].

8 ACKNOWLEDGEMENTS

The authors would like to thank Milan Vojnovic, Matthias Grossglauser, and Pierre Brémaud for several instructive discussions.

REFERENCES

[1] I. Gupta, K.P. Birman, and R. van Renesse, "Fighting fire with fire: Using randomized gossip to combat stochastic scalability limits," *Journal of Quality and Reliability Engineering International*, vol. 18, no. 3, pp. 165–184, 2002.

[2] D. Powell et al, "Group communication (special issue)," *Communications of the ACM*, vol. 39, no. 4, pp. 50–97, 1996.

[3] Z.J. Haas and B. Liang, "Ad hoc mobility management with uniform quorum systems," *IEEE/ACM Trans. on Networking*, vol. 7, no. 2, pp. 228–240, 1999.

[4] G. Pei and Mario Gerla, "Mobility management for hierarchical wireless networks," *ACM/Kluwer Mobile Networks and Applications (MONET)*, vol. 6, no. 4, pp. 331–337, 2001.

[5] L. Zhou and Z. Haas, "Securing ad hoc networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, 1999.

[6] S. Čapkun, L. Buttyán, and J.-P. Hubaux, "Self-organized public-key management for mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, pp. 52–64, 2003.

[7] L. Buttyán and J.-P. Hubaux, "Report on a working session on security in wireless ad hoc networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 4, 2002.

[8] N. Asokan and Philip Ginzboorg, "Key-agreement in ad-hoc networks," *Computer Communications*, vol. 23, no. 17, pp. 1627–1637, 2000.

[9] S. Nesargi and R. Prakash, "MANETconf: configuration of hosts in a mobile ad hoc network," in *Proc. of INFOCOM*, 2002, pp. 1059–1068.

[10] E.M. Royer and C.E. Perkins, "Multicast operation of the ad-hoc on-demand distance vector routing protocol," in *Proc. of MobiCom*, 1999, pp. 207–218.

[11] J.-P. Hubaux, T. Gross, J.-Y. Le Boudec, and M. Vetterli, "Toward self-organized mobile ad hoc networks: the terminodes project," *IEEE Communications Magazine*, vol. 39, no. 1, pp. 118–124, 2001.

[12] J. Luo, P.Th. Eugster, and J.-P. Hubaux, "Route driven gossip: Probabilistic reliable multicast in ad hoc networks," in *Proc. of INFOCOM*, 2003, pp. 2229–2239.

[13] 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 MobiHoc*, 2003, pp. 1–12.

[14] M.G. Hayden, *The Ensemble System*, Ph.D. thesis, Department of Computer Science, Cornell University, 1997.

[15] Y. Amir, C. Danilov, and J. Stanton, "A low latency, loss tolerant architecture and protocol for wide area group communication," in *Proc. of DSN*, 2000, pp. 327 –336.

[16] N. Malpani, N.H. Vaidya, and J.L. Welch, "Distributed token circulation in mobile ad hoc networks," in *Proc. of ICNP*, 2002, pp. 4–13.

[17] S. Dolev, E. Schiller, and J.L. Welch, "Random walk for self-stabilizing group communication in ad hoc networks," in *Proc. of SRDS*, 2002, pp. 70–79.

[18] E. Vollset, "The autograph protocol - a preliminary report on a reliable broadcast protocol for mobile ad-hoc networks," in *Proc. of DSN - Student Forum*, 2003.

[19] G.-C. Roman, Q. Huang, and A. Hazemi, "Consistent group membership in ad hoc networks," in *Proc. of ISCE*, 2001.

[20] V. Hadzilacos and S. Toueg, "Fault-tolerant broadcasts and related problems," in *Distributed Systems*, chapter 5, pp. 97–145. Addison-Wesley, 2 edition, 1993.

[21] K.P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, "Bimodal multicast," *ACM Trans. on Computer Systems*, vol. 17, no. 2, pp. 41–88, 1999.

[22] P. Eugster, R. Guerraoui, S. Handurukande, A.M. Kermarrec, and P. Kouznetsov, "Lightweight probabilistic broadcast," *ACM Transactions on Computer Systems (to appear)*, 2003.

[23] S. Floyd, V. Jacobson, C-G. Liu, S. McCanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Trans. on Networking*, vol. 5, no. 6, pp. 784–893, 1997.

[24] S. Paul, K.K. Sabnani, J.C. Lin, and S. Bhattacharyya, "Reliable multicast transport protocol," *IEEE J. Sel. Areas Commun.*, vol. 15, no. 3, pp. 784–893, 1997.

[25] R. Chandra, V. Ramasubramanian, and K. Birman, "Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks," in *Proc. of ICDCS*, 2001, pp. 275–283.

[26] D. Barbara and H. Garcia-Molina, "The reliability of vote mechanisms," *IEEE Trans. on Computers*, vol. 36, no. 10, pp. 1197–1208, 1987.

[27] F.B. Schneider, "Replication management using the state-machine approach," in *Distributed Systems*, chapter 6, pp. 169–197. Addison-Wesley, 2 edition, 1993.

[28] D. Malkhi, M.K. Reiter, and A. Wool, "Probabilistic quorum systems," *Information and Computation*, vol. 170, no. 2, pp. 184–206, 2001.

[29] Z.J. Haas and B. Liang, "Ad hoc mobility management with randomized database groups," in *Proc. of ICC*, 1999, vol. 3, pp. 1756–1762.

[30] M. Papadopouli and H. Schulzrinne, "Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices," in *Proc. of MobiHoc*, 2001, pp. 117–127.

[31] T. Hara, "Effective replica allocation in ad hoc networks for improving data accessibility," in *Proc. of INFOCOM*, 2001, pp. 1568–1576.

[32] K.H. Wang and B. Li, "Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks," in *Proc. of INFOCOM*, 2002, pp. 1089–1098.

[33] D.B. Johnson, D.A. Maltz, and Y-C. Hu, *The dynamic source routing protocol for mobile ad hoc networks (DSR)*, February 2003, Internet-Draft, draft-ietf-manet-dsr-08.txt. Work in progress.

[34] K. Xu, X. Hong, and M. Gerla, "An ad hoc network with mobile backbones," in *Proc. of ICC*, 2002, vol. 5, pp. 3138–3143.

[35] R. Sivakumar, P. Sinha, and V. Bharghavan, "CEDAR: a core-extraction distributed ad hoc routing algorithm," *IEEE Journal on Selected Areas in Communications (Special Issue on Ad-hoc Routing)*, vol. 17, no. 8, pp. 1454–1465, 1999.

[36] A.W. Fu and D.W. Cheung, "A transaction replication scheme for a replicated database with node autonomy," in *Proc. of VLDB*, 1994, pp. 214–225.

[37] L.-G. Alberto and I. Widjaja, *Communications Networks*, McGraw Hill Higher Education, 2000.

[38] J.D. Murray, *Mathematical Biology*, Springer, Berlin, 2nd edition, 1993.

[39] K. Fall and K. Varadhan, Eds., *The ns Manual*, The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, Apr. 2002, Availiable from http://www.isi.edu/nsnam/ns/.

[40] T.S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall, Upper Saddle River, New Jersey, 2nd edition, 2002.

[41] D.B. Johnson and D.A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Tomasz Imielinski and Hank korth, Eds., chapter 5, pp. 153–181. Kluwer Academic Publishers, 1996.

[42] A.-M. Kermarrec, L. Massoulie, and A. Ganesh, "Probabilistic reliable dissmination in large-scale systems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 14, no. 3, pp. 248–258, 2003.

[43] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," in *Proc. of INFOCOM*, 2003, pp. 1312–1321.