

PAN: Providing Reliable Storage in Mobile Ad Hoc Networks with Probabilistic Quorum Systems *

Jun Luo Jean-Pierre Hubaux Patrick Th. Eugster
School of Computer and Communication Sciences
Swiss Federal Institute of Technology (EPFL), CH-1015 Lausanne, Switzerland
{jun.luo, jean-pierre.hubaux, patrick.eugster}@epfl.ch

ABSTRACT

Reliable storage of data with concurrent read/write accesses (or *query/update*) is an ever recurring issue in distributed settings. In mobile ad hoc networks, the problem becomes even more challenging due to highly dynamic and unpredictable topology changes. It is precisely this unpredictability that makes probabilistic protocols very appealing for such environments. Inspired by the principles of probabilistic quorum systems, we present a Probabilistic quorum system for Ad hoc networks (PAN), a collection of protocols for the reliable storage of data in mobile ad hoc networks. Our system behaves in a predictable way due to the gossip-based diffusion mechanism applied for quorum accesses, and the protocol overhead is reduced by adopting an asymmetric quorum construction. We present an analysis of our PAN system, in terms of both reliability and overhead, which can be used to fine tune protocol parameters to obtain the desired tradeoff between efficiency and fault tolerance. We confirm the predictability and tunability of PAN through simulations with *ns-2*.

Categories and Subject Descriptors

C.2.1 [Computer-Communication Networks]: Network Architecture and Design; C.2.2 [Computer-Communication Networks]: Network Protocols; C.2.4 [Computer-Communication Networks]: Distributed Systems; C.4 [Performance of Systems]: Fault Tolerance, Reliability; I.6 [Simulation and Modeling]: Modeling Methodologies

General Terms

Algorithms, Design, Experimentation, Performance, Reliability

*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>)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MobiHoc'03, June 1–3, 2003, Annapolis, Maryland, USA.
Copyright 2003 ACM 1-58113-684-6/03/0006 ...\$5.00.

Keywords

Mobile Ad Hoc Networks, Quorum Systems, Reliable Data Storage, Replication, Gossiping, Multicast

1. INTRODUCTION

As in any traditional distributed setting, reliable storage of data with concurrent read/write (*query/update*) accesses still appears to be a challenging problem in mobile ad hoc networks. In this paper, we aim at making, at a reasonable cost, small data objects highly available to nodes in an ad hoc network. This need arises in various data sharing services. Mobility management [1, 2, 3], for instance, continuously tracks locations of mobile nodes and serves requests to these data. The distributed management of cryptographic keys or certificates [4, 5, 6] represents another class of applications. In addition, a distributed naming or addressing service [7, 8, 9] is essential in order to provide each mobile node with an unambiguous name or IP address, or to convert the former to the latter. Due to the peculiarities of these applications, we consider a relaxed consistency model.

The “classic” academic approach to providing highly available data objects, the *state-machine* approach [10], consists in synchronizing a set of server replicas, each hosting a copy of the data, to handle all updates and queries in the same way (“write all – read one”). This approach provides perfect guarantees in theory, yet is expensive to implement even in wired networks. In particular, the *atomic broadcast* [11] protocol ensures that each server replica obtains all updates and in the same order, but it comes with a prohibitively large overhead. Clearly, with an underlying highly dynamic ad hoc network, the synchronism required for state machine replication becomes unpracticable. Even with weaker consistency guarantees (e.g., without ordering guarantees), state machine replication leads to poor performance in ad hoc networks. Further sacrificing safety guarantees such as atomicity through the use of a probabilistic dissemination of updates (e.g., *probabilistic broadcast* [12]) is indeed an option, yet the incurred overhead is not low enough to address a broad application context. A more drastic shift seems to be necessary.

Such a step has been initiated (again first in wired networks) through the introduction of *quorum systems* [13]. This approach improves the efficiency of the replication of the stored data by better balancing the load between updates and queries. A quorum system is, roughly, a set of quorums, each consisting of a subset of server replicas. By ensuring that the intersections of quorums meet certain

```

procedure LEAVE
  leaveFlag  $\leftarrow$  true

task UPDATE /* Executed every  $T$  ms */
  if leaveFlag = true then
    m.rmb  $\leftarrow$  id
    DS(1)  $\leftarrow$  set  $\subset_{\mathbb{R}}$  AView such that  $|set| = F$ (2)
    for all id  $\in$  DS do
      SEND(m, id)
  else
    while Buffer  $\neq$   $\emptyset$  do
      m  $\leftarrow$  entry  $\in$  Buffer
      Buffer  $\leftarrow$  Buffer  $\setminus$  {entry}
      m.rmb  $\leftarrow$  entry  $\in_{\mathbb{R}}$  RView
      m.mb  $\leftarrow$  entry  $\in_{\mathbb{R}}$  AView  $\cup$  PView
      DS  $\leftarrow$  set  $\subset_{\mathbb{R}}$  AView such that  $|set| = F$ 
      for all id  $\in$  DS do
        SEND(m, id)

```

(1) DS stands for *destination set*.

(2) A subscript \mathbb{R} stands for random selection.

(a) UPDATE/LEAVE emission

```

upon RECEIVEUPDATE(m) do
  /* Step 1: Update Buffer with new messages */
  if  $\nexists$  entry  $\in$  midList such that entry  $\geq$  m.mid then
    Buffer  $\leftarrow$  Buffer  $\cup$  {m}
    for all entry  $\in$  midList such that entry  $<$  m.mid do
      midList  $\leftarrow$  midList  $\setminus$  {entry}
      midList  $\leftarrow$  midList  $\cup$  {m.mid}
      COMMIT(m)
  /* Step 2: Remove obsolete server from view */
  AView  $\leftarrow$  AView  $\setminus$  {m.rmb}
  PView  $\leftarrow$  PView  $\setminus$  {m.rmb}
  RView  $\leftarrow$  RView  $\cup$  {m.rmb}
  while  $|RView| > |RView|_M$  do
    RView  $\leftarrow$  RView  $\setminus$  {entry  $\in_{\mathbb{R}}$  RView}
  /* Step 3: Add new server to view */
  if m.mb  $\notin$  (AView  $\cup$  PView) then
    if  $\exists$  routing/location info. for m.mb then
      AView  $\leftarrow$  AView  $\cup$  {m.mb}
    else
      PView  $\leftarrow$  PView  $\cup$  {m.mb}

```

(b) UPDATE reception

Figure 3: Data update and membership management at node i

```

upon QUERY(mq) /* mq infused by client query */ do
  if  $\exists$  entry  $\in$  midList such that entry  $>$  mq.mid then
    mq.ver  $\leftarrow$  entry.ver
    Countermq  $\leftarrow$  0
    DS  $\leftarrow$  set  $\subset_{\mathbb{R}}$  AView such that  $|set| = \xi_R - 1$ 
    for all id  $\in$  DS do
      SEND(mq, id)
      timerid  $\leftarrow$  0 /* set a timer for the id */

upon RECEIVEQUERY(mq) do
  if  $\exists$  entry  $\in$  midList such that entry  $>$  mq.mid then
    mq.mid  $\leftarrow$  entry
    mq.data  $\leftarrow$  queried data object(3)
    SEND(mq, idagent)

```

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

(a) QUERY emission and reception

```

upon RECEIVEQUERYREPLY(mq)  $\vee$  timer = timeout do
  Countermq  $\leftarrow$  Countermq + 1
  if  $\nexists$  entry  $\in$  midList such that entry  $\geq$  mq.mid then
    for all entry  $\in$  midList such that entry  $<$  mq.mid do
      midList  $\leftarrow$  midList  $\setminus$  {entry}
      midList  $\leftarrow$  midList  $\cup$  {mq.mid}
      COMMIT(mq)
  if Countermq =  $\xi_R - 1$  then
    Invoke the client query protocol

```

(b) REPLY reception at an agent

Figure 4: QUERY operation and the responses to it

ing *mid* into the *midList*. The COMMIT primitive delivers the message to the upper layer. Periodically (every T ms), a task named UPDATE is executed to disseminate messages stored in the *Buffer*. The parameter *fanout* (F), key to the performance of PAN, denotes the number of receivers chosen by each sender, or forwarder, within the UPDATE task. The server update protocol is gossip-based [12, 26] in nature since these receivers are randomly chosen. At last, all servers that effectively receive a message make a write quorum. The size of the quorum, ξ_W , is predictable thanks to the epidemic nature of the gossip-based protocol. The SEND primitive is a direct call to the underlying unicast protocol.

While performing the message dissemination, the server update protocol also takes the responsibility of tracking the membership. Due to the node mobility and frequent membership changes, it is not practical to have a full membership view at each server. In fact, even if it is possible to have the *ids* of all servers, there is no guarantee that the corresponding routing or location information is available. Our

routing/location oriented membership management scheme tries to provide each server a partial view, approximately random in nature, by exchanging membership information between servers. The underlying principle is that the scheme has a similar effect as the reshuffling of the partial view, together with sporadic losses and discoveries of the routing or location information⁵.

Considering the fact that the locality of the traffic can reduce the network load, we apply a general optimization by raising the awareness of topology. This optimization is based on the assumption that the underlying routing protocol can provide partial topological information, e.g., we can have the path length information from the routing table of DSR, or the mobility management service itself can provide distance information. Our heuristics, in the case of DSR, works like

⁵The 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.

for all servers). The influence of particular network conditions is represented by p_f , the probability of packet loss for a certain link. Since the diffusion process is relatively short in time, we approximate p_f by considering only the effects of node mobility, buffer overflow, and packet collision, but omitting packet loss due to node crashes.

We consider only the second query to a data object modified by the most recent update, while considering the first query as happening before the update.⁷ For example, as shown in Fig. 6, only the pairs of (update β , query β_2) and

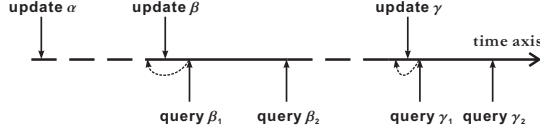


Figure 6: The occurrence of events in terms of absolute time.

(update γ , query γ_2) are considered, whereas queries β_1 or γ_1 are supposed to request previous updates (i.e., updates α 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. 7. Since there is always some delay for the message dissemination, the probability that the actual events occurrence will follow the order of our assumption at that server is very high, according to different distributions of time interval between two events within a Poisson process. This makes the present analysis a “viable” lower bound. We

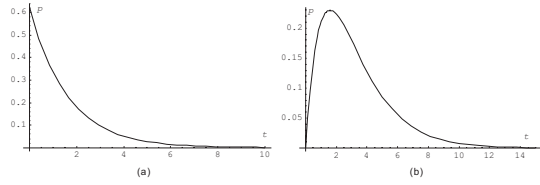


Figure 7: The distributions of time interval between two events. (a) exponential distribution for consecutive events; (b) Erlang distribution for non-consecutive events.

continue using p_f to represent the network condition, but an empirical value p_e is used to represent the server *unavailability* due to failure, at any time instant. One might argue that the server failure should be treated as a random process with the time as its parameter [3], this is however not justifiable with a failure recovery model, which is a general case in ad hoc networks (e.g., nodes switching off for the purpose of battery replacement or operating system rebooting).

4.2 Stochastic Behavior of PAN System

We convey the predictability of the metrics for PAN, \mathcal{R}_d and \mathcal{N}_l , in two subsections. Certain preliminary results presented in [27] are omitted for conciseness.

⁷The time of an event is when it happens at an agent.

4.2.1 Reliability Degree \mathcal{R}_d

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.

We use $\hat{\xi}_W^r$ and $\Delta\hat{\xi}_W^r = \mathbf{E}[\hat{\xi}_W^r - \hat{\xi}_W^{r-1}]$ for the values of real quorum size **after** r gossip rounds accomplished by the server update protocol and the average increment of this size **within** round r , respectively. According to the analysis in [27], $\hat{\xi}_W^r$ is estimated with the following recurrence relation:

$$\nu_{r+1}^T = \nu_r^T \mathcal{P}_\delta \quad (1)$$

where $\nu_r(i) = P(\hat{\xi}_W^r = i)$ is the i th element of the column vector ν_r , with the initial value $\nu_0(1) = 1$. There exists an \bar{r} for which the diffusion process is finished, i.e., no new server is infected afterwards, when $r \geq \bar{r}$. We denote $\hat{\xi}_W^r|_{r \geq \bar{r}}$ by $\hat{\xi}_W$ with distribution ν . Here \mathcal{P}_δ is the transition matrix with its element $p_{(i,j)\delta}$ expressed as follows:

$$\begin{aligned} p_{(i,j)\delta} &= P(\hat{\xi}_W^{r+1} = j | \hat{\xi}_W^r = i, \Delta\hat{\xi}_W^r = \delta) \\ &= \begin{cases} \binom{n-i}{j-i} (1-q^\delta)^{j-i} q^{\delta(n-j)} & j \geq i \\ 0 & j < i \end{cases} \quad (2) \end{aligned}$$

where $q = f(F, p_f)$, a function of fanout F and p_f , is the probability that a certain server is not infected in a given gossip round. The detailed computation of q is omitted here.

Based on the assumption of synchronization, we divide the time axis after a given update event β into $\bar{r} + 1$ intervals, as shown in Fig. 8. A read quorum, resulting from a query

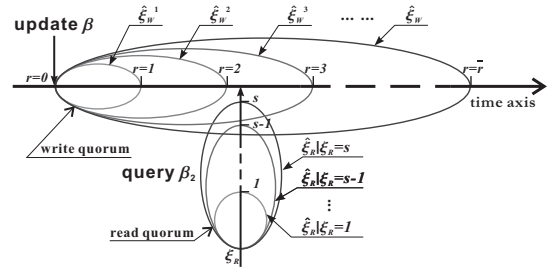


Figure 8: Incremental processes of read and write quorum size: $\hat{\xi}_W$ increases round by round, while $\hat{\xi}_R$ increases with the amount of queries sent by an agent.

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 the 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 according to the following recursive procedure, with an initial value $P(\hat{\xi}_R = 1 | \xi_R = 1) = 1$ and the convention

that $P(\hat{\xi}_R = k | \xi_R = s) = 0$ if $s < 0, k \leq 0$ or $k > s$:

$$\begin{aligned} \mu(k) &= P(\hat{\xi}_R = k | \xi_R = s) \\ &= P(\hat{\xi}_R = k - 1 | \xi_R = s - 1) \cdot p \\ &+ P(\hat{\xi}_R = k | \xi_R = s - 1) \cdot (1 - p) \end{aligned} \quad (3)$$

$k = 1, \dots, s$

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. Here H is defined as a random variable representing the length of an arbitrarily chosen routing path. The distribution of this value is discussed in [27]. The estimation of μ is somewhat conservative because servers with relatively old version data 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 that the event arrival is described by a Poisson process. Therefore, we have

$$p_r = \begin{cases} \int_{t_r}^{t_r+1} \lambda_q^2 t e^{-\lambda_q t} dt & r < \bar{r} \\ \int_{t_r}^{\infty} \lambda_q^2 t e^{-\lambda_q t} dt & r = \bar{r} \end{cases} \quad (4)$$

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

$$\mathcal{R}_d = \sum_{r=0}^{\bar{r}} \sum_{i=1}^n \sum_{j=1}^{\xi_R} \left(1 - \frac{\xi_R - \xi_W}{\xi_R}\right) \mu(j) \nu_r(i) p_r \quad (5)$$

4.2.2 Network Load \mathcal{N}_l

For a certain \mathcal{R}_d 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.

The loads generated by a single update and query are calculated separately, and then \mathcal{N}_l is obtained by summing products of the loads of the individual operations and their corresponding arrival rates:

$$\begin{aligned} \mathcal{L}_W &= \sum_{i=1}^n \sum_h (\hat{\xi}_W \cdot F \cdot h) P(H = h) \nu(i) \\ &= \mathbf{E}[\hat{\xi}_W] \cdot F \cdot \mathbf{E}[H] \end{aligned} \quad (6)$$

$$\begin{aligned} \mathcal{L}_R &= \sum_h (2 \cdot \xi_R \cdot h) P(H = h) \\ &= 2 \cdot \xi_R \cdot \mathbf{E}[H] \end{aligned} \quad (7)$$

$$\mathcal{N}_l = \lambda_u \mathcal{L}_W + \lambda_q \mathcal{L}_R \quad (8)$$

This estimation is conservative in the same sense as we mentioned before. Again, it is relatively rough compared with the one for \mathcal{R}_d , because we do not take into account the following two facts: (1) many packets get dropped before reaching their destinations, and (2) packets, especially those eventually dropped, may travel quite a long way due to stale routing information. We will show with simulation that the former factor has a dominating effect in most cases, but these factors tend to offset each other in some cases.

5. SIMULATION

This section presents the simulation results of our PAN system. Some results are compared with the corresponding analytical ones, to confirm our claim that both the reliability

degree \mathcal{R}_d and the network load \mathcal{N}_l of PAN are predictable. The flexibility of PAN, as well as its sensitivity to server failures, are also investigated by simulations in different settings.

5.1 Model and Parameters

The simulator we use is *ns-2* [30] 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 [31] as the radio propagation model.

We simulate ad hoc networks with 50 and 100 nodes in a square area of 1km². The movement pattern is defined by the ‘‘random waypoint’’ model [32] where the following process is repeated: nodes randomly choose a destination, move towards it at a speed uniformly distributed between zero and a maximum speed, and upon reaching the destination stay there for some pause time. In our simulations, 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. We pair the two simulation parameters in order to make the scenario more realistic, since it is a common knowledge that low speed nodes, like pedestrians, tend to be more restless than high speed nodes, like cars.

The STS contains half of the network nodes. We do not justify this number, but only use it as an example. The servers in the STS are assumed to be predefined in order to simplify the simulation.⁸ The client protocol is omitted to reduce side effects. There is a Poisson traffic source, generating packets of 128 bytes, attached to each server to emulate the arrivals of queries or updates. If there is an update message, the server diffuses it with the update protocol (the gossip period is set to $T = 200$ ms), otherwise the message is disseminated by the query protocol. The impact of the overall access rate λ_o on the performance of PAN is first investigated, then an appropriate value is taken for all simulations. We further assume that $\lambda_o = a\lambda_u$. Due to space limitations, we use $a = 8$ for most simulations, and change to $a = 4$ in Section 5.5 in order to show the fact that PAN is tunable. As the last simulation parameter, p_e is first set to 1%, and then varied to show the sensitivity of PAN to server failures in Section 5.6. For some cases, i.e., certain values for the network size, maximum speed, pause time, and other simulation parameters, we vary the design parameters F and ξ_R in order to show the trade-off between the two metrics \mathcal{R}_d and \mathcal{N}_l .

PAN is 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 was carried out 10 times with different scenario files created by *ns-2*.

5.2 Impact of λ_o on PAN Performance

Fig. 9 shows the performance of PAN with respect to λ_o , the overall access rate. We observe that PAN performs in a

⁸Although 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.

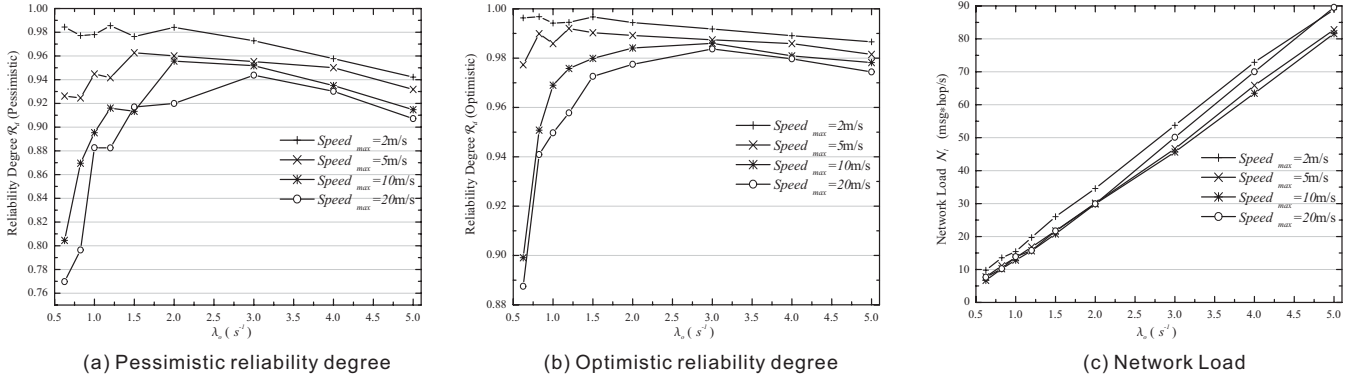


Figure 9: Reliability degree \mathcal{R}_d vs. overall access rate λ_o for 50 nodes networks in a 1km^2 square, with $F = 2$ and $\xi_R = 4$.

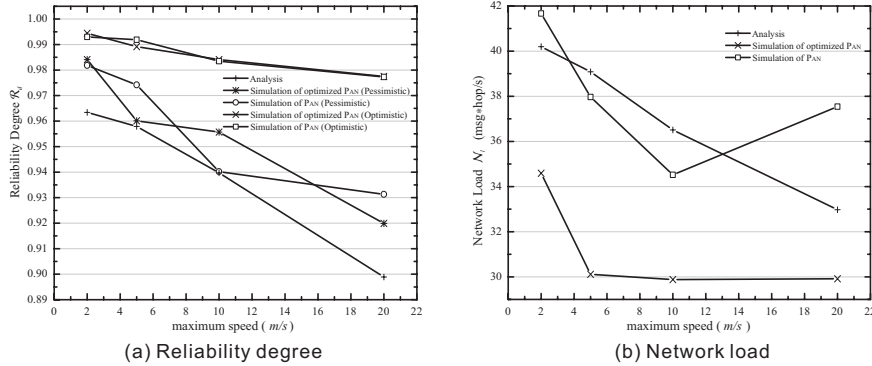


Figure 10: Comparisons between optimized and non-optimized PAN for 50 nodes networks in a 1km^2 square, with $F = 2$ and $\xi_R = 4$.

relatively stable way for $1.5\text{s}^{-1} \leq \lambda_o < 3\text{s}^{-1}$, and \mathcal{R}_d 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 increases linearly with λ_o by Equation (8). However, it may seem somewhat odd to observe that \mathcal{R}_d 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}_d are presented in two ways. The ‘‘pessimistic’’ \mathcal{R}_d refers to the probability that a query reaches the most recent update (with the same assumption as in Section 4.1 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 two notations for all graph illustrations in this section.

5.3 Optimization vs. Non-optimization

As described in Section 3.1 and 3.2, raising the topology

awareness improves the performance of PAN. Fig. 10 highlights this point. It shows that the optimized version has nearly the same \mathcal{R}_d as the non-optimized one (the difference is around 1%), but with a remarkable reduction (up to 20%) on \mathcal{N}_l . Therefore, we will provide simulation results of the optimized version throughout the rest of this section.

We also observe that \mathcal{N}_l diminishes when the node speed increases. This is not a surprise because the definition of \mathcal{N}_l considers not only the number of messages sent but also the number of hops that each message travels. As a result, \mathcal{N}_l decreases as more and more packets get lost due to increased channel failure probability in high mobility scenario. The abnormal increase of \mathcal{N}_l in very high speed scenario is a direct result of the second factor explained in Section 4.2.2.

5.4 Comparisons between Analytical and Simulation Results

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 . The maximum speed and pause time are varied to test the impact of mobility on the performance of PAN. The design 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 the update, takes $F = x$ with probability $1 - y/10$ and $F = x + 1$ with probability $y/10$.

The following conclusions can be drawn: (1) the simu-

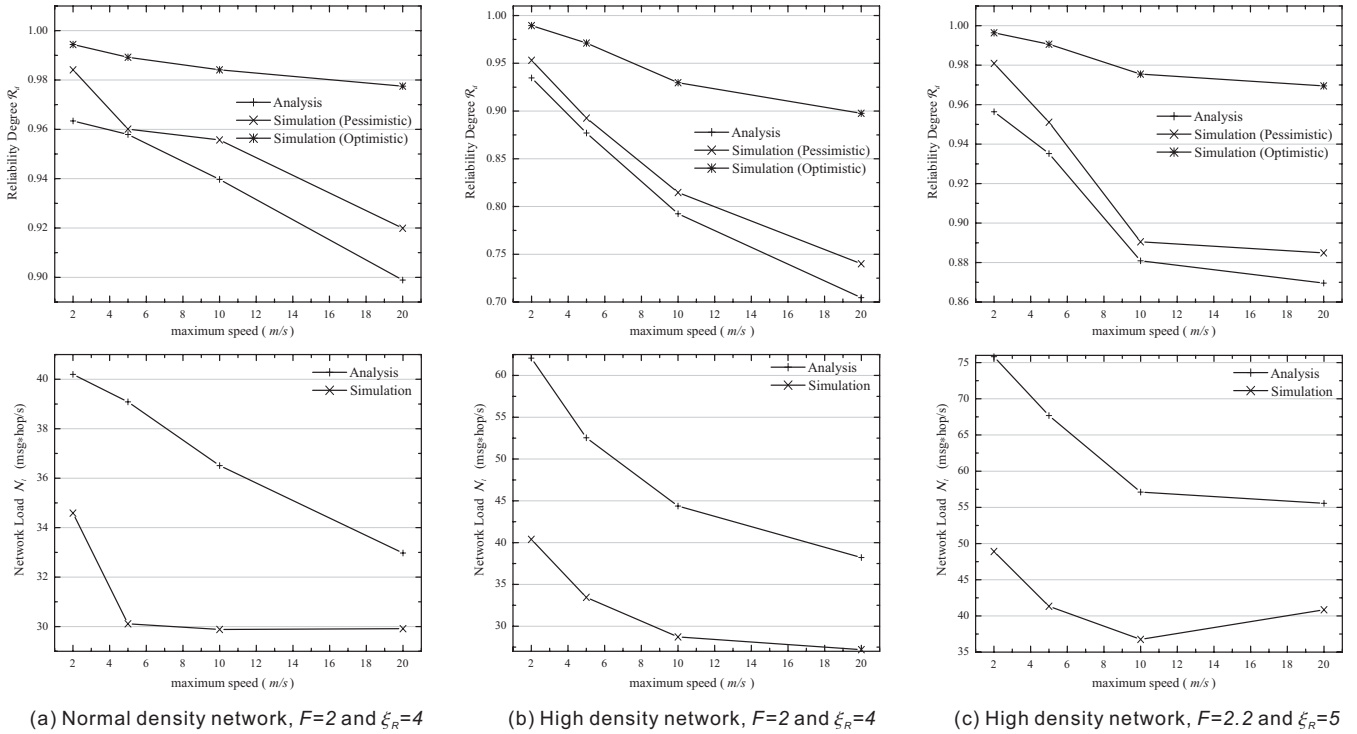


Figure 11: Analytical and simulation results for reliability degree \mathcal{R}_d and network load \mathcal{N}_l vs. mobility

lation and analytical results of \mathcal{R}_d match very well. This confirms the predictability on \mathcal{R}_d ; (2) the analytical results of \mathcal{N}_l are quite rough due to the reason we stated in Section 4.2.2, as well as the optimization we apply, but they still provide certain information about the system overhead such as the trend of its changes in different situations; (3) the optimistic \mathcal{R}_d 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; (4) as the network size and the maximum node speed grow, design parameters have to be adjusted to maintain a good performance of \mathcal{R}_d , at a cost of increased system overhead.

5.5 Adapting PAN to a New Environment

We simulate a new application environment by setting $a = 4$, i.e., increasing the update rate and decreasing the query rate. Intuitively, one would expect that shrinking the write quorum (decreasing F) and enlarging the read quorum (increasing ξ_R) would reduce \mathcal{N}_l , while still maintaining \mathcal{R}_d . We investigate this hypothesis with simulations shown in Fig. 12.

The first set of design parameter adjustments (changing F from 2 to 1.7 and ξ_R from 4 to 5) results in a reduction of \mathcal{N}_l (more than 5%) with virtually no expense of \mathcal{R}_d . Further parameter adjustments (reducing ξ_R to 4) yield significant reduction on \mathcal{N}_l (up to 25%), but with a modest degradation of \mathcal{R}_d (less than 5%). These results confirm what our intuition suggested.

5.6 Sensitivity to Server Unavailability p_e

According to the simulation results shown in Fig. 13, the sensitivity of PAN to p_e increases as the maximum node

speed grows. In addition, the sensitivity of PAN considering optimistic \mathcal{R}_d is lower than the sensitivity considering pessimistic \mathcal{R}_d .

We also observe that the increase of p_e leads to an improvement of \mathcal{R}_d in some cases. This paradox indeed suggests a way to optimize our PAN system, i.e., a server belonging to a certain read quorum does 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 can avoid that 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}_d . However, we do not put it as a formal optimization of PAN, because it is not as stable as the topology-awareness optimization in dynamic environments.

6. DISCUSSION AND RELATED WORK

In this section, we summarize the novelties of the design of our PAN system and overview relevant work on quorum system designs and data management schemes.

6.1 Characteristics of PAN

We explain the three main characteristics of PAN as follows:

We have proposed a gossip-based protocol for quorum access to cope with the “hostile” environment and to avoid the need for a global membership tracking for quorum systems. The design of access protocols and membership management schemes for quorum systems was not well covered in previous literature. This problem, however, has to be explicitly tackled in ad hoc networks for the following reasons: (1) Unicast protocols, as one choice to access

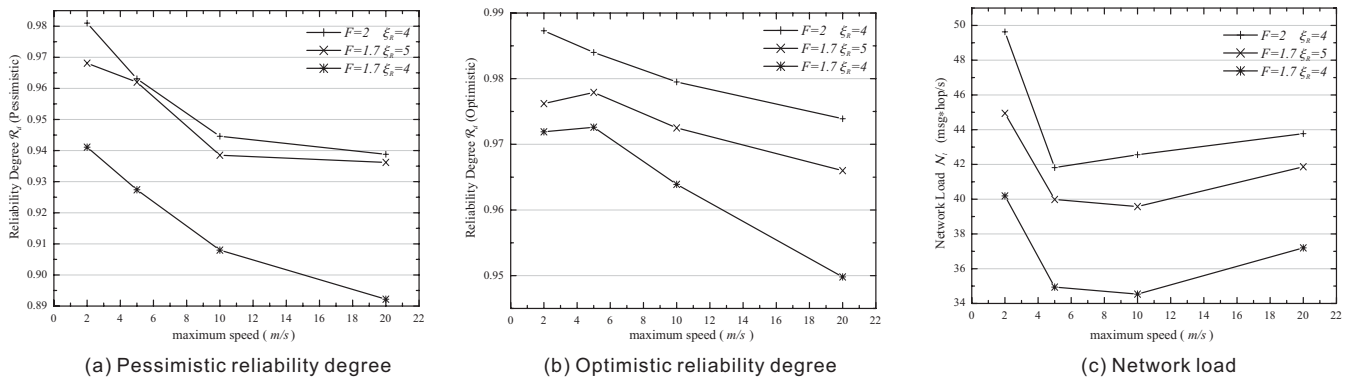


Figure 12: Reliability degree \mathcal{R}_d and network load \mathcal{N}_l vs. maximum speed with $a = 4$, for 50 nodes networks in a 1km^2 square.

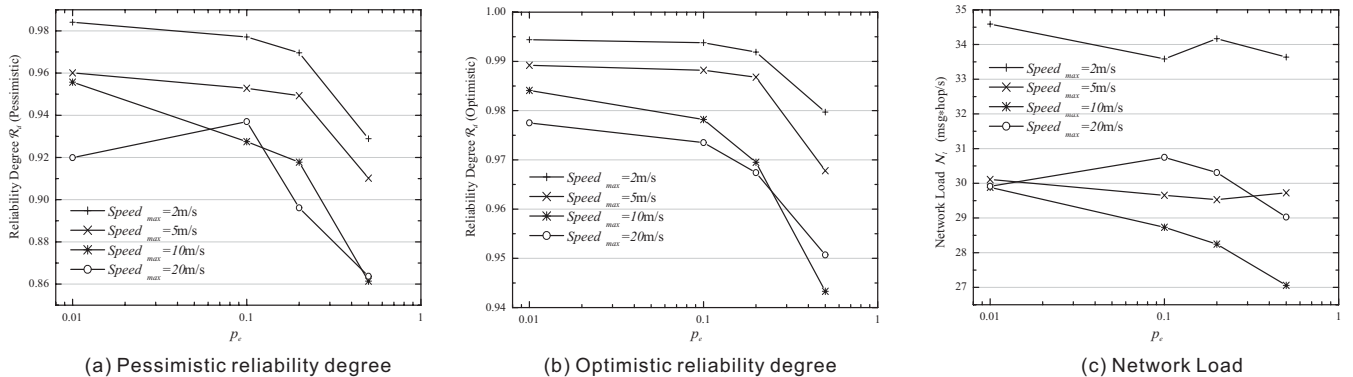


Figure 13: Reliability degree \mathcal{R}_d and network load \mathcal{N}_l vs. server unavailability p_e for 50 nodes networks in a 1km^2 square, with $F = 2$ and $\xi_R = 4$.

quorum systems, are not as stable as those for wired networks; Even worse, the performance of the unicasting can hardly be predicted due to node mobility and packet collision, which invalidates the commitment made by quorum systems. (2) Multicast protocols for ad hoc networks (e.g., MAODV), as another choice, always have a relatively large overhead for maintaining the multicast group, and are therefore not suitable for delivering packets in several groups with memberships that change frequently, such as probabilistic quorum systems. (3) It is a main purpose of ad hoc networks to refrain from any centralized membership tracking that is an often made assumption for quorum systems. Our gossip-based protocol, as demonstrated before, behaves in a predictable way, while requiring no separate membership tracking.

As a consequence of gossip-based access protocol, the design of quorum systems becomes more straightforward. According to the theory of probabilistic quorum systems [14], the ξ is designed to meet the requirement of a certain intersection property ε in a replication system with n servers. However, as any unreliable access protocol is concerned, the difference between ξ and $\hat{\xi}$ does exist. Also, it is $\hat{\xi}$ that is essential to the estimation of reliability. Our design is more straightforward in the sense that we directly obtain $\hat{\xi}$, probabilistic in nature, from the access protocol. Our approach is friendly to quorum systems in ad hoc networks

since it does not require any global membership tracking in order to design ξ , a function of n and ε , beforehand.

An asymmetric quorum construction is applied, with a flexibility to fine tune design parameters, in order to have different sizes for individual quorums. An important factor that limits the application of traditional quorum systems is their symmetric construction (i.e., the same size for each quorum). By considering the shared-private data model, for instance, it becomes clear that the arrival rate of queries could be quite higher than that of updates, since the number of potential clients that may send a query is much larger than the number of the update generator, which is basically one for each data object. Therefore, in order to reduce the \mathcal{N}_l required for achieving a certain \mathcal{R}_d , it is reasonable to have a large write quorum, but a relatively small read quorum. The flexibility of tuning quorum sizes improves the adaptability of our PAN system to various application requirements.

6.2 Probabilistic Quorum Systems

Probabilistic quorum systems were first proposed in [14]. The authors demonstrate that, compared with strict quorum systems, probabilistic quorum systems yield a substantial improvement on the *load* while keeping high fault tolerance. A simple construction of a probabilistic quorum system, called ε -intersecting quorum system, is also proposed.

The construction shows that the probability of nonintersection between two arbitrary quorums is less than $\varepsilon = e^{-l^2}$ by taking $\xi = l\sqrt{n}$ servers as a quorum in a system of n servers. The load considered in [14] is the charge of computation for an individual server. We define the load in a different way by focusing on the charge of network resources, since computation is much cheaper than communication in wireless networks.

Haas and Liang [3] 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 PAN, it is unfortunate that there are no simulation results provided to evaluate the system performance and to confirm the precision of the numerical analysis in [3], otherwise making comparisons between the two systems would have been desirable.

6.3 Data Management in Ad Hoc Networks

The 7DS system presented in [33] shares certain features of our PAN system with respect to the diffusion scheme used for data spreading. However, since the two systems are designed for different network environments (7DS assumes a rarely connected network, whereas PAN considers networks of relatively high density), the underlying diffusion mechanisms are quite different. 7DS passively exploits node mobility to relay data from one node to the other, which results in a considerable delay for data spreading but has a potential to improve power and bandwidth usages (a recent research work [34] applies a similar scheme to exchange security data), whereas PAN 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 PAN).

The work of Hara [35] assumes that all mobile nodes can carry a set of data replicas, which would be a special case for PAN with its STS being the whole network. In order to guarantee data accessibility upon network partitioning, their work focuses on optimizing the location of data replicas within a network, based on the assumptions that (1) the memory space on nodes are limited, (2) data items are not updated, and (3) access frequencies to data items from each node are known and fixed. Although it is interesting to consider the impact of memory space on data allocation, the other two assumptions are too strong to capture the reality of mobile networks, hence limiting the application scope of their protocol.

Similarly to [35], Wang and Li [36] consider the problem of replica allocation. However, their approach is more practical in the sense that it takes into consideration topology information (e.g., connection stability) when replicating data, while 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 theirs. The mobility model they propose assumes strong correlations between different nodes such that nodes in a network are organized into mobility groups, which might lead to frequent network partitions. Whereas we consider a purely random mobility pattern, in which network

partitions seldom happen and mobility prediction can hardly make sense.

7. CONCLUSIONS

In this paper, we are concerned with the high availability of relatively small data objects in mobile ad hoc networks. We have defined the problem of probabilistic reliable data storage and two performance metrics that take the peculiarities of ad hoc networks into account. We have proposed our PAN system, based on the principle of probabilistic quorum systems, as a solution to this problem. The performance of PAN has been analyzed by making use of, notably, epidemic theory. The evaluation and investigation of PAN have also been carried out by simulations in *ns-2*.

The probabilistic reliability problem and corresponding metrics that we have defined can be considered as a general framework for evaluating protocols designed to address similar issues. As one solution to the probabilistic reliability problem, our PAN system diffuses updates to a random set (write quorum) in a storage entity STS. It also accesses a random set (read quorum) in the STS, upon queries, to get the most recent update. The main contributions we have made on the design of PAN can be summarized in three aspects, namely (1) gossip-based quorum access protocol, (2) straightforward quorum systems design, and (3) asymmetric quorum construction.

We have proposed an analytical model to predict the performance of our PAN system. The validity of predictions is evaluated by simulations. The results show that our analytical model provides quite accurate predictions on the reliability degree \mathcal{R}_d , while the prediction on the network load \mathcal{N}_l is relatively rough due to various factors. We are in the process of improving this model by further investigating the effects of those factors. In addition, it would also be interesting to have an analytical model based on random graph theory, as in [37].

In addition to confirming the predictability of our PAN system, we have also investigated other aspects of it with intensive simulations. Our simulation results show that, even under frequent topology changes, the reliability degree of our PAN system is high enough to guarantee its quality of service. These results also confirm the robustness of PAN, in the sense that it can sustain a large access rate λ_o , different network sizes, and a certain amount of server failures. Finally, the results illustrate the tunability property of PAN, by which the system can adapt to various application environments. It is a part of our future work to further study the performance of PAN under different access rates and to propose an upgraded PAN that adapts itself well to changes of access rate.

8. REFERENCES

- [1] 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.
- [2] 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, Apr 1999.
- [3] Z.J. Haas and B. Liang, “Ad hoc mobility management with randomized database groups,” in *Proc. of ICC’99*, 1999, vol. 3, pp. 1756–1762.

- [4] L. Zhou and Z.J. Haas, "Securing ad hoc networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, 1999.
- [5] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for mobile ad-hoc networks," in *Proc. of ICNP'01*, 2001, pp. 251–260.
- [6] J.-P. Hubaux, L. Buttyán, and S. Čapkun, "The quest for security in mobile ad hoc networks," in *Proc. of MobiHoc'01*, 2001, pp. 146–155.
- [7] C.E. Perkins, J.T. Malinen, R. Wakikawa, E.M. Royer, and Y. Sun, *IP address autoconfiguration for ad hoc networks*, July 2002, Internet-Draft, draft-ietf-manet-autoconf-01.txt. Work in progress.
- [8] N.H. Vaidya, "Weak duplicate address detection in mobile ad hoc networks," in *Proc. of MobiHoc'02*, 2002, pp. 206–216.
- [9] S. Nesargi and R. Prakash, "MANETconf: configuration of hosts in a mobile ad hoc network," in *Proc. of INFOCOM'02*, 2002, pp. 1059–1068.
- [10] F.B. Schneider, "Replication management using the state-machine approach," in *Distributed Systems*, chapter 6, pp. 169–197. Addison-Wesley, 2 edition, 1993.
- [11] V. Hadzilacos and S. Toueg, "Fault-tolerant broadcasts and related problems," in *Distributed Systems*, chapter 5, pp. 97–145. Addison-Wesley, 2 edition, 1993.
- [12] K.P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiú, and Y. Minsky, "Bimodal multicast," *ACM Trans. on Computer Systems*, vol. 17, no. 2, pp. 41–88, 1999.
- [13] D. Barbara and H. Garcia-Molina, "The reliability of vote mechanisms," *IEEE Trans. on Computers*, vol. 36, no. 10, pp. 1197–1208, 1987.
- [14] D. Malkhi, M.K. Reiter, and A. Wool, "Probabilistic quorum systems," *Information and Computation*, vol. 170, no. 2, pp. 184–206, 2001.
- [15] E.M. Royer and C.E. Perkins, "Multicast operation of the ad-hoc on-demand distance vector routing protocol," in *Proc. of MobiCom'99*, 1999, pp. 207–218.
- [16] S.J. Lee, M. Gerla, and C.C. Chiang, "On-demand multicast routing protocol," in *Proc. of WCNC'99*, 1999, vol. 3, pp. 1298–1302.
- [17] J.J. Garcia-Luna-Aceves and E.L. Madruga, "The core-assisted mesh protocol," *IEEE Journal on Selected Areas in Communications (Special Issue on Ad-hoc Routing)*, vol. 17, no. 8, pp. 1380–1394, 1999.
- [18] 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.
- [19] 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.
- [20] K. Xu, X. Hong, and M. Gerla, "An ad hoc network with mobile backbones," in *Proc. of ICC'02*, 2002, vol. 5, pp. 3138–3143.
- [21] P. Krishna, N.H. Vaidya, M. Chatterjee, and D.K. Pradhan, "A cluster-based approach for routing in dynamic networks," in *Proc. of ACM/SIGCOMM Computer Communication Review*, 1997, pp. 372–387.
- [22] C.R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7, pp. 1265–1275, 1997.
- [23] 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.
- [24] A.W. Fu and D.W. Cheung, "A transaction replication scheme for a replicated database with node autonomy," in *Proc. of VLDB'94*, 1994, pp. 214–225.
- [25] L.-G. Alberto and I. Widjaja, *Communications Networks*, McGraw Hill Higher Education, 2000.
- [26] Z.J. Haas, J.Y. Halpern, and L. Li, "Gossip-based ad hoc routing," in *Proc. of INFOCOM'02*, 2002, pp. 1707–1716.
- [27] J. Luo, P.Th. Eugster, and J.-P. Hubaux, "Route driven gossip: Probabilistic reliable multicast in ad hoc networks," in *Proc. of INFOCOM'03*, 2003.
- [28] K. Chen and K. Nahrstedt, "Effective location-guided tree construction algorithms for small group multicast in MANET," in *Proc. of INFOCOM'02*, 2002, pp. 1192–1201.
- [29] J.D. Murray, *Mathematical Biology*, Springer, Berlin, 2nd edition, 1993.
- [30] K. Fall and K. Varadhan, Eds., *The ns Manual*, The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, Apr. 2002, Available from <http://www.isi.edu/nsnam/ns/>.
- [31] T.S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall, Upper Saddle River, New Jersey, 2nd edition, 2002.
- [32] 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.
- [33] M. Papadopouli and H. Schulzrinne, "Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices," in *Proc. of MobiHoc'01*, 2001, pp. 117–127.
- [34] S. Čapkun, J.-P. Hubaux, and L. Buttyán, "Mobility helps security in ad hoc networks," in *Proc. of MobiHoc'03*, 2003.
- [35] T. Hara, "Effective replica allocation in ad hoc networks for improving data accessibility," in *Proc. of INFOCOM'01*, 2001, pp. 1568–1576.
- [36] K.H. Wang and B. Li, "Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks," in *Proc. of INFOCOM'02*, 2002, pp. 1089–1098.
- [37] A.-M. Kermarrec, L. Massoulie, and A. Ganesh, "Probabilistic reliable dissemination in large-scale systems," *IEEE Trans. on Parallel and Distributed Systems (to appear)*, 2003.