

RAPID: Reliable Probabilistic Dissemination in Wireless Ad-Hoc Networks

Vadim Drabkin Roy Friedman Gabriel Kliot Marc Segal
Computer Science Department
Technion - Israel Institute of Technology
Haifa, 32000 Israel

Email:{dvadim,roy,gabik,marcs}@cs.technion.ac.il

Abstract

Reliable broadcast is a basic service for many collaborative applications as it provides reliable dissemination of the same information to many recipients. In this paper we propose a novel ReliAble Probabilistic Dissemination protocol, called RAPID, for mobile wireless ad-hoc networks that tolerates message omissions, node crashes, and selfish behavior.

The protocol employs a combination of probabilistic forwarding with deterministic corrective measures. The forwarding probability is set based on the observed number of nodes in each one-hop neighborhood, while the deterministic corrective measures include deterministic gossiping as well as timer based corrections of the probabilistic process. These aspects of the protocol are motivated by a theoretical analysis that is also presented in the paper, which explains why this unique protocol design is inherent to ad-hoc networks environments. Since the protocol only relies on local computations and probability, it is highly resilient to mobility and failures. By adding authentication, it can even be made malicious tolerant.

Additionally, the paper includes a detailed performance evaluation by simulation. We compare the performance and the overhead of RAPID with the performance of other probabilistic approaches. Our results show that RAPID achieves a significantly higher node coverage with a smaller overhead.

Keywords: Reliable Broadcast, Probabilistic Broadcast, Fault Tolerance, Ad-Hoc Networks.

1 Introduction

Wireless mobile ad-hoc networks (MANET) are formed when an ad-hoc collection of devices equipped with wireless communication capabilities happen to be in proximity to each other [29]. When some of these devices agree to forward messages for other devices, a multi-hop network is formed. One of the aspects of ad-hoc networks is that they are formed without any pre-existing infrastructure or management authority. Also, due to mobility, the physical structure of the network is continuously evolving.

MANETs offer a potential for a variety of new applications and improved services for mobile users, especially as the computing power of mobile devices becomes stronger. Example applications include interactive distributed games, ad-hoc transactions and e-commerce, collaborative (shared white-board and video conferencing) applications, and enhancing the bandwidth and reach of cellular communication (e.g., for Wi-Fi enabled cell-phones) [13, 14].

Broadcast is a basic service for many collaborative applications, as it enables any device to disseminate information to all other participants in the network. In particular, a useful broadcast service should provide a good level of reliability, meaning that most nodes in the system will receive almost every broadcasted message.

The simplest way to obtain broadcast in a multiple hop network is by employing flooding [28]. That is, the sender sends the message to everyone in its transmission range. Each device that receives a message for the first time delivers it to the application and also forwards it to all other devices in its range. While this form of dissemination is very robust, it is also very wasteful and may cause contention and a large number of collisions [30].

Common alternatives to flooding are either to perform a constrained flooding on top of a deterministic overlay, e.g., [35, 36], or to perform probabilistic flooding, e.g., [11, 18]. The problem with deterministic overlays is that due to the combination of mobility and the decentralized nature of MANETs, maintaining overlays in MANETs is a complex task. Finally, it is hard to make overlays resilient to malicious or even selfish behavior [6].

In the probabilistic approach, whenever a node receives a message, it applies some locally computable probabilistic mechanism to randomly determine whether it should broadcast the message or not [11, 18]. Probabilistic protocols are appealing since they are very simple, and are inherently robust to failures and mobility. Yet, as was discovered in [11, 18, 24], in order to obtain very high reliability levels with pure probabilistic broadcasting, one has to set the retransmission probability to relatively high values. Consequently, such schemes still generate a large number of redundant messages.

Other approaches [11, 30, 31] combine probabilistic forwarding with some additional locally computable mechanism, such as *counter-based*, *distance-based*, *location-based*, or any combination of those, to determine whether it should rebroadcast the message or not. That way, the number of messages is further reduced. Yet, those protocols suffer from increased latency. In addition, the results in those works are mainly based on simulations and very little theoretical analysis has been done in order to understand them. Finally, as we discuss in Section 4, those schemes cannot ensure high reliability for arbitrary topologies, and cannot cope with selfish (and malicious) behavior.

Contributions of this Work In this work, we first present several general theoretical results about probabilistic dissemination of messages in ad-hoc networks. These results are then used to motivate the development of a novel efficient reliable probabilistic broadcast protocol for wireless ad-hoc networks. Finally, we measure the performance of the protocol using simulations, including comparing our work to other probabilistic approaches. These simulations validate our approach and protocol design choices.

The formal part of this work includes the following results: We first analyze the relationship between the number of nodes that rebroadcast messages in each one-hop neighborhood in a probabilistic dissemination protocol and the expected reliability of this protocol (or in other words, the percentage of nodes that will receive the message). In particular, our probabilistic analysis shows that there is an optimal number, in the sense that this number of retransmitting nodes, which is relatively small, is enough to ensure good reliability. Moreover, this number does not depend on the network's density. Yet, in order to obtain even higher reliability using pure probabilistic forwarding, a much larger number of retransmitting nodes must be chosen. The conclusions from this formal analysis for the design of probabilistic broadcasting protocols in ad-hoc networks are twofold: First, the forwarding probability should be inversely proportional to the number of nodes in each one-hop neighborhood. Second, the forwarding probability should be kept to a relatively low value, which matches the optimal number mentioned above; in order to boost the protocol's reliability even further, it is better to use deterministic corrective measures, rather than increasing the forwarding probability.

Armed with this formal insight, we have deduced the following principles for obtaining an efficient and reliable probabilistic broadcast protocol: The retransmission probability of each node should be inversely proportional to the number of neighbors it observes at a given moment. Consequently, the number of retransmitting nodes is independent of the network's density, as discussed above. Also, this probability is chosen so that the expected number of retransmitting nodes will be optimal according to the formal analysis.

To further boost the reliability level of our protocol, we add the following two deterministic mechanisms. In parallel to the probabilistic dissemination process, every node gossips with its neighbors about the headers of the messages it obtains. This technique enables a node who misses some message and later learns about the missing messages through gossiping, to request those messages from another node that has them. Additionally, we employ timer based corrections that may cause a node to change its decision on whether to broadcast a message or not. Note that the benefit of gossiping comes from the fact that message headers are typically much smaller than the messages themselves. Moreover, as gossips are sent periodically, multiple gossip messages are aggregated into one packet, thereby greatly reducing the number of messages generated by the protocol. The timers are used to fix discrepancies between the probabilistic assumptions and the actual network, as well as to recover from "bad luck" scenarios, which might occur when using randomization.

The result is a protocol that sends a small number of messages compared to other known alternatives and guarantees high reliability with any topology. The protocol is also computationally very efficient, and it is highly resilient to mobility and even some forms of malicious behavior, due to its probabilistic nature and its reliance only on local information. The paper includes a detailed performance evaluation carried by simulation, validating our claims.

Paper's road-map The model and basic definitions and assumptions are described in Section 2. The theoretical results of this work are presented in Section 3. Section 4 describes the protocol. The results of the performance evaluation are given in Section 5. Section 6 compares our work with related work, and we conclude with a discussion in Section 7.

2 System Model and Definitions

We assume a collection of *nodes* placed in a given finite size area. A node in the system is a device owning an omni-directional antenna that enables wireless communication. A transmission by a node p can be received by all nodes within a disk centered at p whose radius depends on the transmission power, referred to in the

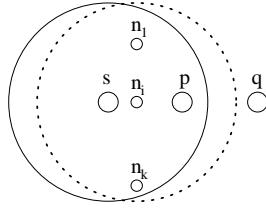


Figure 1: A transmission by a node s can be received by all nodes within its transmission range: p, n_1, \dots, n_k

following as the *transmission disk*; the radius of the transmission disk is called the *transmission range*.¹ The combination of the nodes and the transitive closure of their transmission disks forms a wireless ad-hoc network. Nodes can physically move across the network; new nodes may join and existing nodes may leave the network at any time, either gracefully or by suffering a crash failure. Nodes that crash or leave the network may rejoin it later.

We denote the transmission range of device p by r_p . This means that a node q can only receive messages sent by p if the distance between p and q is smaller than r_p . A node q is a *direct neighbor* of another node p if q is located within the transmission disk of p as illustrated in Figure 1. In the following, $N(p)$ refers to the set of direct neighbors of a node p . Additionally, messages can be lost. For example, if two nodes p and q transmit a message at the same time, and there exists a node r that is a direct neighbor of both, then r will receive neither message, in which case we say that there was a *collision*. Yet, we assume that a message is delivered with a positive probability.

Finally, in Sections 4.3 and 5.2.4, we assume that a few nodes might not execute their protocol correctly. Such nodes are called *malicious*. A specific form of maliciousness, which is also known as *selfishness*, is refusal to forward messages. The non-malicious nodes are called *correct*. Still, even in the parts of this work that tolerate malicious behavior, we assume that the correct nodes in the system continuously form a connected sub-network.

3 Formal Motivation

In this section, we establish some formal results, that serve as motivation and explanation for our protocol's design in Section 4. Subsections 3.1 and 3.2 below present the theoretical results, and are therefore fairly technical. We then discuss these results and relate them to our protocol design choices in Subsection 3.3.

3.1 Setting the Retransmission Probability - Probabilistic motivation

Our theoretical analysis in this section relies on the famous *Random Geometric Graph* (RGG) model, which is often used to model the network connectivity graph of 2-dimensional wireless ad hoc networks and sensor networks [10]. A 2-dimensional RGG, also known as the *Unit Disk* graph and denoted $G^2(n, r)$, is obtained by placing n nodes uniformly at random on the surface of a 2-dimensional unit torus, and connecting nodes

¹ In practice, the transmission range does not behave exactly as a disk due to various physical phenomena. However, for the description of the protocol it does not matter, and on the other hand, a disk assumption greatly simplifies the formal model. In any case, our simulation results are carried on a simulator that simulates a real transmission range behavior including distortions, background noise, unidirectional links, etc.

within Euclidean distance r of each other [21]. In our case we assume n nodes are placed uniformly at random in the rectangular area $[a, b]$ and the transmission radius r is set such as $G^2(n, r)$ is connected with high probability. It has been shown (by Gupta and Kumar [10] and [20]) that for r satisfying $\pi r^2 \geq ab \frac{\log n + c(n)}{n}$, $G^2(n, r)$ is asymptotically connected with probability one, if and only if $c(n) \rightarrow \infty$ as $n \rightarrow \infty$. We will therefore assume that r satisfies the above condition and the network is connected.

We stress here that the uniform distribution of nodes in space is only used in the theoretical analysis of this section, in order to set the retransmission probability in the most efficient way. The correctness of the actual algorithm does not depend on this assumption. If the uniformity assumption does not hold, our protocol in Section 4 will ensure reliable delivery in any case, alas possibly with higher communication cost.

Denote by d_{avg} the average number of neighbors of any node in $G^2(n, r)$. It is well known that $d_{\text{avg}} \leq \frac{\pi r^2(n-1)}{ab}$ and for large networks, when the edges effect is negligible, $d_{\text{avg}} \sim \frac{\pi r^2(n-1)}{ab}$. We have previously shown in [2] that the maximal and minimal degrees, denoted by d_{max} and d_{min} , are of the order of d_{avg} with high probability. That is, the actual degree of any node in $G^2(n, r)$ is close to d_{avg} with high probability.

Assume some broadcasting algorithm A , which picks for every message m , a set of nodes S that transmit m . Every node in S is picked with probability $\mathcal{P} = \frac{\beta}{d_{\text{avg}}}$ from all network nodes, independently from all other nodes, where β is a parameter called the *reliability factor* of algorithm A . Informally, β is the average number of nodes in each one-hop neighborhood that retransmit m . Also assume that a message that was sent has a probability \mathcal{Q} to be successfully received by a neighboring node. We can now calculate the probability that an arbitrary node will not receive m .

Claim 3.1 *For any node p , the probability that p does not receive a message m is bounded from above by $e^{-\beta\mathcal{Q}}$.*

Proof: S is the set of all nodes that transmit message m . Notice that the size of S is a binomial random variable with mean $n\mathcal{P}$.

For each $q \in S$, let X_q be a 0-1 random variable indicating whether the node p receives a message m that was sent by the node q or not. Node p can receive a message m sent by q if and only if q is a neighbor of p in $G^2(n, r)$ and m has not collided with other messages. Since two nodes are neighbors if and only if they are at distance at most r from each other, then $\Pr(X_q = 1) = \mathcal{Q} \frac{\pi r^2}{ab}$.

Let Y_p be the random variable indicating the number of times node p has received m . Naturally, if $p \in S$, $\Pr(Y_p = 0) = 0$. Otherwise,

$$\begin{aligned} \Pr(Y_p = 0) &= \prod_{q \in S} \Pr(X_q = 0) = \left(1 - \mathcal{Q} \frac{\pi r^2}{ab}\right)^{n\mathcal{P}} = \\ &\left(1 - \mathcal{Q} \frac{\pi r^2}{ab}\right)^{n \frac{\beta ab}{\pi r^2(n-1)}} \leq \left(1 - \mathcal{Q} \frac{\pi r^2}{ab}\right)^{\frac{\beta ab}{\pi r^2}} \leq \left(e^{-\mathcal{Q} \frac{\pi r^2}{ab}}\right)^{\frac{\beta ab}{\pi r^2}} = e^{-\beta\mathcal{Q}}. \end{aligned}$$

We have used the inequality $1 - x < e^{-x}$, which holds for all $x > 0$. ■

Figure 2 depicts the values of $\Pr(Y_p = 0)$ for an arbitrary node p as a function of β and \mathcal{Q} .

It can be seen from the figure that the probability that a given node does not receive a message m is small for quite small values of β . For example, for $\mathcal{Q} = 0.9$, $\Pr(Y_p = 0)$ is approximately 0.1 for $\beta = 2.5$. That is, if there are only $\beta = 2.5$ nodes in every one-hop neighborhood that transmit m and $\mathcal{Q} = 0.9$, approximately 90% of all nodes will receive m .

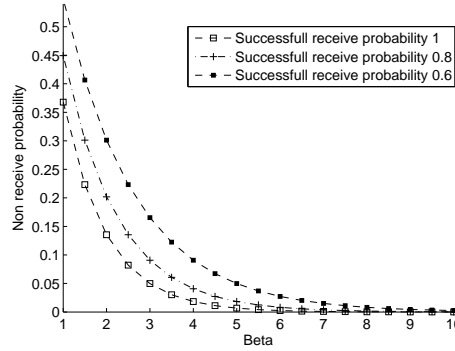


Figure 2: The probability that an arbitrary node does not receive a message m

3.2 Geometric motivation

Ni et al. [30] showed that a broadcast of some node p covers at least 39% of the one-hop neighborhood of any of its neighboring nodes. Therefore, any message rebroadcast can provide a maximum of 61% additional coverage over the area already covered by the previous transmissions. Moreover, the *average* additional coverage (under the assumption of uniformly distributed locations) is around 41%. Further analysis in [30] established the benefit of a host rebroadcasting a message after it has heard a message k times, by applying the same geometric argument of additional coverage. Not surprisingly, the resulting graph of the number of heard transmission vs. additional coverage is quite similar to our graph in Figure 2. The more times a message is being retransmitted in the same area, the smaller is the additional coverage. For example, when $k \geq 4$, the expected additional coverage is below 5%.

This result provides an alternative explanation to the same phenomena we have shown in Section 3.1. That is, there is a certain minimal number of rebroadcasting in every neighborhood (denoted by β) which is sufficient to cover large percentage of the network. In order to achieve even better coverage, a much larger number of rebroadcasting, which is disproportional to the expected additional coverage (or reliability), should be used.

3.3 Discussion

A broadcasting algorithm that sets the retransmission probability \mathcal{P} inversely proportional to the average degree has a number of advantages. First, the number of transmissions, which is equal to the average size of set S , is constant with respect to the number of nodes n and to the nodes' density.

$$E(|S|) = n\mathcal{P} = \frac{n\beta}{d_{\text{avg}}} = \frac{n\beta}{\frac{\pi r^2(n-1)}{ab}} \leq \frac{ab\beta}{\pi r^2}.$$

That is, the number of transmissions does not depend on the overall number of nodes, but rather only on the physical size of the network, the transmission radius and the required reliability level. Hence, for a given physical network, there is a minimal number of retransmissions that is required to guarantee high broadcasting reliability, and this number is constant with respect to the number of nodes and to the nodes' density. In particular, such a broadcast protocol is highly efficient in dense networks.

Second, a probabilistic broadcasting algorithm that picks nodes uniformly at random with probability inversely proportional to the average degree, can achieve high coverage of the network with relatively few

redundant messages. Most (but not all) of the network nodes will receive almost every message while using a relatively small group S .

On the other hand, in order to guarantee very high reliability, a much larger number of rebroadcasts should be used. This motivates the usage of probabilistic broadcasting to achieve high nodes coverage with relatively low price and then to rely on other deterministic strategies to boost the reliability even further.

4 The RAPID Protocol

For didactic purposes, we develop our protocol in three steps. The basic version of our protocol appears in Figure 3; an enhanced version of the protocol that sends even fewer messages and provides higher delivery ratio is depicted in Figure 4, and the malicious resilient version of our protocol appears in Figure 5. In all figures we make use of two primitives. The primitive `prob_bcast` denotes an immediate broadcast to all the direct neighbors of the sender with a given probability. The primitive `lazycast` initiates periodic broadcasting of the given message to the direct neighbors of the sender.

Our protocol is based on the following principles: Each node calculates its broadcast probability according to the number of observed neighbors at a given moment. Since in our protocol each node needs to know the number of its one-hop neighbors, every node periodically sends a heartbeat message (unless it has already sent another message during a predefined time interval).

As already mentioned in Section 3.2, every broadcast of a message m by any node p covers at least 39% of the area of its neighbor q . Thus, at least $0.39 \times |N(q)|$ neighbors of q will receive the transmission of p with high probability, assuming that m did not collide with any other message. Therefore, w.h.p., m will be rebroadcasted by q or by at least one of q 's neighbors that received m from p , if each node k that received m from p sets the rebroadcast probability to $\min(1, \frac{1}{0.39 \times |N(k)|}) \approx \min(1, \frac{\beta}{|N(k)|})$ with $\beta = 2.5$, which corresponds to the optimal value discussed in Section 3.1.

In parallel, every node p periodically broadcasts to its neighbors the headers of messages p received from other nodes, which is called *gossiping*. This technique enables nodes who miss some messages that exist in the system to request these messages from their neighbors. Notice that nodes only send headers of messages they possess. Hence, the header of a message that does not exist will not be disseminated in the network. Also, whenever possible, gossip messages are piggybacked on other messages, in order to further reduce the generated traffic. Unlike many other gossiping mechanisms from distributed computing [3], in our case, gossiping is deterministic, in the sense that a gossip message from p is broadcasted to all of p 's neighbors at once.

Note that without the gossip/recovery protocol, one cannot guarantee reliable delivery of all messages even if all nodes broadcast all the messages with very high probability. First of all, this is due to the possibility of collisions in wireless networks. Moreover, consider the following scenario w.r.t Figure 1. Node s broadcasts a message m such that p and all n_i s receive m . If we only rely on probabilities, it is possible that p will decide not to rebroadcast m . Consequently, q will never receive it. However, if we use the gossip/request mechanism, eventually q will find out that it is missing m and will ask its neighbors that have m to rebroadcast it.

4.1 Basic RAPID

4.1.1 The Dissemination Task in Details

Message dissemination in our protocol consists of the following steps: (1) The originator p of a message m sends $m||header(m)$ to all nodes in $N(p)$ (Lines A01–A04 in Figure 3). The header part of m includes a

```

Upon send(msg) by application do
(A01) header := msg_id||node_id;
(A02) data_msg := header||msg;
(A03) gos_msg := header;
(A04) prob_bcast(prob = 1, data_msg, DATA);
(A05) lazycast(gos_msg, GOSSIP);

Upon receive(msg, DATA) sent by  $p_j$  do
(A06) if (have not received this msg before) then
(A07)   Accept( $p_i, p_j, msg$ ); /*forward to the application*/
(A08)   prob_bcast(prob =  $\min(1, \frac{\beta}{|N(p)|})$ , msg, DATA);
(A09)   lazycast(gos_msg, GOSSIP);
(A10) endif;

Upon receive(gos_msg, GOSSIP) sent by  $p_j$ : do
(A11) if (there is no msg that fits the gos_msg) then
(A12)   /*Ask the neighbors to send the real message*/
(A13)   prob_bcast(prob =  $\min(1, \frac{\beta}{|N(p)|})$ , gos_msg, REQUEST);
(A14) endif;

Upon receive(gos_msg, REQUEST) sent by  $p_j$  do
(A15) if (I have the msg that matches gos_msg) then
(A16)   prob_bcast(prob =  $\min(1, \frac{\beta}{|N(p)|})$ , msg, DATA);
(A17) endif;

```

Figure 3: Basic RAPID (executed by node p)

sequence number and the identifier of the originator. (2) The originator p of m then starts a periodic gossip of $header(m)$ to all nodes in $N(p)$ (Line A05). (3) When a node p receives a message m for the first time, p accepts m (Lines A06–A07). (4) p broadcasts m with probability $\min(1, \frac{\beta}{|N(p)|})$ (Line A08 – our protocol was simulated with β equals to 2.5). (5) p starts a periodic gossip of $header(m)$ to all nodes in $N(p)$ (Line A09). (6) If a node p receives a message m it has already received beforehand, then m is ignored.

4.1.2 Gossiping and Message Recovery in Detail

The gossiping and message recovery part of the protocol is composed of the following subtasks:

1. When p receives a message m , p gossips $header(m)$ to other nodes in $N(p)$ (Lines A09). Note that p does not forward gossips about messages it has not received yet. This is done in order to make the recovery process more efficient.
2. When p receives a gossip $header(m)$ for a message m it has not received yet, p asks its neighbors to forward m to itself using a REQUEST message (Lines A11–A14). Intuitively, since p received a GOSSIP message about m , one of p 's neighbors should have m and supply it when needed.
3. When p receives a REQUEST for a message m , yet p has not received m , p ignores this request. Otherwise, p broadcasts the missing message (Lines A15–A17).

One issue that needs to be taken care of is purging received messages, in order to avoid unbounded memory requirements. This can be done either using timeouts, or by employing a stability detection mechanism [9,

27]. In this work, we have chosen to use timeout based purging due to its simplicity. Clearly, in this case there is a tradeoff in setting the timeout value: a long timeout increases the reliability, but also increases the memory consumption. From our experiments, it turns out that that even with short timeouts we can reach reliability above 99.9% in most cases.

4.2 Enhanced RAPID

The basic RAPID protocol has two important drawbacks. First, two nodes can choose to broadcast the same message even if they are very close to each other. Obviously, retransmissions by nodes that are located close to each other cover very little additional area. The second drawback is even more severe: if all nodes in a given neighborhood decide not to broadcast a message, the dissemination of this message would be severely delayed, as it will only be propagated through the gossip/request protocol, which is slow.

In order to deal with these drawbacks and improve the reliability and the latency of RAPID, we slightly change the protocol by adding two complementing corrective measures that are based on having each node monitor its neighbors. That is, instead of immediately rebroadcasting a message m with a given probability, a node p adds m to its casting queue for a random amount of time and in parallel monitors its neighbors. If p overhears a transmission of m by one of its neighbors, p removes m from its casting queue without broadcasting m . Otherwise, after the random timeout elapses, p broadcasts m with the probability as discussed above.

The second corrective measure is that whenever p initially probabilistically decides not to rebroadcast m , but later on p does not hear any other rebroadcasting of m , then p adds m to its casting queue. Thus, either p will hear a retransmission of m by one of its neighbors, or p will retransmit m .

We now explain how these two timer based corrective measures complement each other. The optimization of deciding to rebroadcast m even if initially a node p probabilistically chose not to, but later did not hear any of its neighbors rebroadcast m helps boosting the reliability of the protocol, by ensuring that a message will be propagated to almost every neighborhood of the network.

Yet, the other optimization, of cancelling a retransmission might seem, at first, to hurt the probabilistic behavior of the protocol. The reason for this optimization is that, as been discussed in Section 3.2, according to [30], on average, beyond the first two transmissions of a message m in a given one-hop neighborhood, any additional transmission in the same neighborhood will only deliver the message to very few additional nodes. Hence, if a node p has heard two transmissions of the same message m (the first transmission and the later rebroadcast), there is little point for it to retransmit it as well. Taken from another point of view, the formal analysis of Section 3.1 is based on certain assumptions (e.g., that transmitting nodes are spread uniformly at random in the area). The two timer based corrective measures help approximating these conditions.

Notice that even with the timer based corrections, gossiping is still needed. For example, consider another scenario w.r.t Figure 1. Node s broadcasts a message m and p and n_1, \dots, n_k receive it. If p cancels the transmission of m , for example, due to a retransmission by n_i , then q will never receive m .

4.2.1 The Dissemination task in details

The pseudo-code for the enhanced version of RAPID is listed in Figure 4. In this code, we use a queue called `cast_queue`. The `add` method of this queue accepts the following parameters. The sending probability, a time parameter, the message itself and the type of the message. The time is used in order to set a timer to expire after the corresponding amount of time elapses. The probability and type are stored alongside the message inside the queue.

Upon send(msg) by application **do**
 (B01) *header* := *msg_id*||*node_id*;
 (B02) *data_msg* := *header*||*msg*;
 (B03) *gos_msg* := *header*;
 (B04) `prob_bcast(prob = 1, data_msg, DATA)`;
 (B05) `lazycast(gos_msg, GOSSIP)`;

Upon receive(msg, DATA) sent by p_j **do**
 (B06) **if** (have not received this msg before) **then**
 (B07) `Accept(p_i, p_j, msg); /*forward it to the application*/`
 (B08) `cast_queue.add(prob = $\min(1, \frac{\beta}{|N(p)|})$, time=random(0, short_jitter), msg, DATA)`;
 (B09) `lazycast(gos_msg, GOSSIP)`;
 (B10) **endif**;

Upon receive(gos_msg, GOSSIP) sent by p_j : **do**
 (B11) **if** (there is no message that fits the gos_msg) **then**
 (B12) */*Node asks from its neighbors to send the real message*/*
 (B13) `cast_queue.add(prob = $\min(1, \frac{\beta}{|N(p)|})$, time=random(0, short_jitter), gos_msg, REQUEST)`;
 (B14) **endif**;

Upon receive(gos_msg, REQUEST) sent by p_j **do**
 (B15) **if** (I have the msg that matches gos_msg) **then**
 (B16) `cast_queue.add(prob = $\min(1, \frac{\beta}{|N(p)|})$, time=random(0, short_jitter), msg, DATA)`;
 (B17) **endif**;

Interceptor
 (B18) **if** (*msg that appears in cast_queue* was received) **then**
 (B19) `cast_queue.remove(msg)`;
 (B20) **endif**;

Upon Expiration of timer of msg in cast_queue **do**
 (B21) `cast_queue.remove(msg)`;
 (B22) *pr* = the probability attached to *msg*;
 (B23) *type* = the message type associated with *msg*;
 (B24) `prob_bcast(prob = pr, msg, type)`;
 (B25) **if** (*msg* was not broadcasted) **then**
 (B26) `cast_queue.add(prob = 1, time=long_jitter, msg, type)`;
 (B27) **endif**;

Figure 4: Enhanced RAPID (lines that were modified w.r.t Figure 3 are boxed while lines B18–B27 were added)

Dissemination in Enhanced RAPID works the same as in Basic RAPID (Section 4.1.1) except for Step 4 of Section 4.1.1. In Enhanced RAPID, whenever a node p receives a message m for the first time, it schedules a rebroadcast of m with probability $\min(1, \frac{\beta}{|N(p)|})$ to occur after some random jitter (Line B08 in Figure 4). If during this time p overhears a retransmission of m by another node, then p cancels its own retransmission of m (Lines B18–B20). However, if a received message has never been rebroadcasted, neither by p nor by any of its neighbors, then p decides to rebroadcast m after all, by invoking `prob_broadcast` with probability 1 (Lines B25–B27).

4.2.2 Gossiping and Message Recovery in Detail

As in the dissemination task, the only difference between Basic RAPID and Enhanced RAPID is in the fact that in the enhanced protocol every node p monitors its neighbors and if p planned to broadcast a message m , but p heard a transmission of m by its neighbor node, then p cancels the transmission of m . In addition, if p decided not to broadcast m , but it does not hear the transmission of m by any of its neighbors, p broadcasts m . These issues are handled in Lines B13, B15–B17, B18–B20, and B21–B27.

4.2.3 Latency of RAPID

In both RAPID and *counter-based* protocols nodes wait for a certain amount of time before they rebroadcast a message. Yet, the average waiting time is much shorter in RAPID than in counter based protocols. Notice that in Figure 4 we employ two jitter lengths, *short_jitter* and *long_jitter*. The first is used to prevent collisions, while the second is used as a corrective measure, as discussed above, and is similar to the counter based approach. The expected number of concurrent transmitters competing for transmission due to the probabilistic mechanism is quite small, and hence *short_jitter* can be much shorter than *long_jitter*. Moreover, most times, the timer-based corrective measure will not be used, and hence the average latency is mostly dominated by *short_jitter*.

4.3 Maliciousness Resilient RAPID

Due to its probabilistic nature, RAPID can be resilient to many forms of malicious behavior. Since the decisions that every node takes are based only on the number of its neighbors and the transmissions it hears, the attacks that a malicious node can perform are quite limited. We describe below how the protocol was modified in order to overcome these attacks.

4.3.1 Malicious Tolerant RAPID in Details

We use digital signatures in order to prevent a malicious node from forging others' messages or trying to impersonate other nodes. Each device p holds a private key k_p , known only to itself, with which p can digitally sign every message it sends [25]. We assume a malicious node cannot forge signatures and that each device can obtain the public key of every other device, and can thus authenticate the sender of any signed message.

The originator p of a message m adds two signatures to m before it broadcasts m . The first signature is calculated on the concatenation of m , p 's node id, and m 's message id, in order to bind between the context of the message, the node id of its originator and the message id. The second signature is performed on the p 's node id and the message id. The objective of the second signature being attached to the message is to speed up the dissemination of gossip messages in the system. That is, in our protocol, every time a node q receives a data message m , then q sends a gossip message about m to its neighbors. However, the first signature binds both the message header (sender id and message id) with the message data. Thus, a node that receives a message m cannot generate a valid gossip message for m only based on the first signature. The second signature is the one that should be sent with the gossip message. This enables any node that receives m to immediately start gossiping about m , and be able to attach a valid signature that was generated by the originator of m , to the gossip message. Otherwise, without the second signature, a receiver q of m would have had to wait for a separate gossip message about m before q could have started gossiping about m .

The pseudo-code for the maliciousness resilient protocol appears in Figure 5. Here we introduce four new primitives: `send`, `verify_signature`, `suspect` and `expect`, and the retransmission probability

Upon send(msg) by application **do**

```
(C01) gos_msg := msg_id||node_id||sig(msg_id||node_id);
(C02) data_msg := msg_id||node_id||msg||sig(msg_id||node_id||msg)||sig(msg_id||node_id);
(C03) prob_bcast(prob = 1, data_msg, DATA);
(C04) lazycast(gos_msg, GOSSIP);
```

Upon receive(msg, DATA) sent by p_j **do**

```
(C05) if (verify_signature(msg) = TRUE) then
(C06)   if (have not received this msg before) then
(C07)     Accept( $p_i, p_j, msg$ ); /*forward it to the application*/
(C08)     cast_queue.add(prob = min(1,  $\frac{\beta}{|trusted\_neighbors|}$ ), time=random(0, short_jitter), msg, DATA);
(C09)     lazycast(gos_msg, GOSSIP);
(C10)   endif;
(C11) else /* the message is not correct */
(C12)   suspect( $p_j$ );
(C13) endif;
```

Upon receive(gos_msg, GOSSIP) sent by p_j : **do**

```
(C14) if (verify_signature(gos_msg) = TRUE) then
(C15)   if (there is no message that fits the gos_msg) then
(C16)     expect(gos_msg, $p_j$ );
(C17)     /* The node asks from the node that sent the gossip to send the real message */
(C18)     send(gos_msg, REQUEST,  $p_j$ );
(C19)   endif;
(C20) else /* the message is not correct */
(C21)   suspect( $p_j$ );
(C22) endif;
```

Upon receive(gos_msg, REQUEST, p_k) sent by p_j **do**

```
(C23) if (verify_signature(gos_msg) = TRUE) then
(C24)   if (I am  $p_k$  and I have the msg that matches gos_msg) then
(C25)     prob_bcast(prob = 1, msg,DATA);
(C26)   endif;
(C27) else /* the message is not correct */
(C28)   suspect( $p_j$ );
(C29) endif;
```

Figure 5: Maliciousness Resilient RAPID (lines that were modified w.r.t Figure 4 are boxed)

is being computed based on the number of trusted neighbors (*trusted_neighbors*). The one-hop neighbors of a node p that it has not suspected yet of being malicious form its set of trusted neighbors. The primitive send is a point to point send. The primitive verify_signature verifies that $sig(m)$ matches m . If it does not then m is ignored and the process that sent it is suspected by the receiver of the message. The primitive suspect permanently removes a node p_j that was caught forging a message from the list of trusted neighbors (i.e., p_j sent a message with a signature that fails to authenticate). On the other hand, expect accepts two parameters: a gossip message and a node id p_j . In response, the node p that executed expect, sets a timer such that the given message must be received from p_j before the timer expires. If

such a message is not received in time, then p_j is temporarily removed from the list of trusted neighbors of p . We use it to temporarily suspect a node, which sent us a gossip but refused to deliver the corresponding message, from the list of trusted nodes.

As mentioned before, in the malicious resilient version of RAPID, each node only counts its one-hop neighbors that it has not suspected yet of being malicious. This is because if a node is malicious, it might not execute the protocol correctly, and in particular refuse to forward some messages even when it should do so probabilistically. Hence, if a correct node p is located in an area with many malicious nodes, then p 's broadcast probability will become higher due to the fact that it will ignore those malicious nodes in counting its neighbors. Even if malicious nodes manage to mislead a correct node p by pretending to be correct nodes, the worst thing that can happen is that p 's broadcast probability will be lower. In this case, any message m that is not sent by the probabilistic rebroadcasting mechanism will still be forwarded to p 's neighbors either if p does not hear a retransmission by any of its neighbors or via the gossip/request protocol. Either way, the reliability of the protocol will not be degraded. The only thing that can suffer is the latency of delivering the message to all the nodes.

Also, notice that the protocol in Figure 5 uses point-to-point requests (for missing messages) and unconditional replies (node that was requested a message will send it to the requesting node regardless of other nodes and other messages), rather than probabilistically broadcasting requests and replies as in the previous versions of the protocol. This is done in order to prevent attacks in which malicious nodes "convince" some nodes not to send their messages. For example, consider the following scenario, which is possible with the recovery scheme of Figure 4. A malicious node p can continuously broadcast REQUEST messages such that its close neighbors will hear the transmission of the messages, while the rest of its neighbors will not hear the transmissions of those REQUEST messages. Consequently, the nearby neighbors of p will not broadcast REQUEST messages even if they miss some messages, since they have heard the transmissions of the corresponding REQUEST messages by p . Hence, these neighbors of p will never obtain messages that they failed to receive using the probabilistic dissemination phase. A similar attack is for a malicious node p to always rebroadcast DATA messages in response for REQUEST messages, but to do so such that only the close neighbors of p will receive that DATA message, and will therefore never retransmit it themselves. In this case, the other neighbors of p might never receive such messages. Hence, by using point-to-point requests for missing messages, we slightly enlarge the overhead of the protocol on one hand, but on the other hand, we increase the reliability of the protocol.

It would have been possible to use a similar mechanism to the one used in Enhanced RAPID in lines B13 and B16, but that would have required an additional twist. In order to continue using the scheme of lines B13 and B16, each node would have had to store additional information about messages it has decided not to broadcast due to broadcasts by its neighbors. If some node p receives the same REQUEST (GOSSIP) message several times and p has cancelled the rebroadcast of the corresponding DATA (REQUEST) message, then p would have to rebroadcast the message (with probability 1) immediately. The code in Figure 5 does not include this optimization for simplicity.

4.3.2 Resilience Against Malicious Attacks

Below we specify a number of specific attacks, which are being overcome by Maliciousness Resilient RAPID. Those attacks include: (1) forwarding a message with the wrong data, (2) not forwarding some/all messages (this is known as *selfish* behavior²), (3) sending gossip messages without ever supplying the real

²Giving incentives for nodes to participate is beyond the scope of this work. Here we only focus on overcoming selfish behavior so that it does not prevent correct nodes from receiving messages, assuming that the correct nodes form a connected sub-network.

messages in order to confuse other nodes, (4) trying to collide others' messages, and (5) sending messages as point-to-point messages instead of broadcast messages, thus causing a correct node to decide not to rebroadcast a message, even if it is the only one among all its neighbors that has received the message.

As mentioned above, the first attack is solved by adding signatures. That is, the originator of a message m signs the message with its private key and attaches this signature to the message. Thus, every node p that receives m from q checks m 's signature and if the signature does not match the content of m , p will suspect q and will not accept the message. Moreover, p will no longer count q as one of its neighbors for the purpose of calculating the rebroadcasting probability.

The second attack is solved by the monitoring mechanism. If a malicious node does not rebroadcast a message m to all its neighbors, then our protocol guarantees that in any case one of its neighbors will do it. Hence, as long as the correct nodes form a connected sub-network, every message will be disseminated to all of them.

The third attack is solved using a simple timeout mechanism. When a node p receives a gossip from q about a message m that p is missing, then in addition to sending a request for m to q , p starts a timer. If p does not receive the requested message from q after the timeout, it starts suspecting q as being malicious. In this case, p stops counting q for calculating its rebroadcasting probability.

As for the fourth attack, in our model we assumed that all messages are delivered with a non-zero probability. Hence, by assumption, the fourth attack is not possible. The rationale behind this is twofold: first, if malicious nodes are allowed to collide all messages, then no protocol can ensure reliable delivery. Second, if all nodes are battery operated, jamming the channel will drain the battery very quickly, and hence such an attack cannot last for too long. In particular, whenever malicious nodes are only selfish, rather than mean, then the fourth attack does not make sense in any case, since it hurts everyone, including themselves.

Finally, if a malicious node sends a point-to-point message instead of rebroadcasting it, our gossip mechanism will ensure that the message will still be propagated, yet with an increased delay. In addition, some lower level mechanisms can be used, such as forcing nodes to send messages and listen to messages only on IP-multicast addresses. Moreover, it is possible to verify that a received IP-multicast message was also sent to a MAC destination broadcast address rather than to a point-to-point destination address.

5 Simulations

In this section, we evaluate the performance of RAPID and compare it with the performance of flooding and with the performance of the GOSSIP3 protocol [11]. In GOSSIP3, when a node q receives a message, it broadcasts the message to its neighbors with probability \mathcal{P} and with probability $1 - \mathcal{P}$ it discards the message. In addition, q broadcasts a message if initially q got a message and did not broadcast it, but later q did not get the message from at least M other nodes³. The reason for choosing GOSSIP3 is that it is one of the best studied probabilistic protocols in the literature and was found to be the best probabilistic broadcast mechanism among all the ones explored in [11]. In our simulations we have measured the percentage of messages delivered to all the nodes (*delivery ratio*), the latency to deliver a message to varying percentages of the nodes, the load imposed on the network (number of transmitted messages) and the influence of mute (selfish) nodes on the performance of our protocol.

³GOSSIP3 was simulated with probability 0.65 and $M=1$.

5.1 Setup

We have used the JiST/SWANS simulator [32] to evaluate the protocols. In JiST/SWANS, nodes use two-ray ground radio propagation model with IEEE 802.11 MAC protocol and 54Mb/sec throughput. Communication between nodes is by broadcast. Two concurrent broadcasts can collide, in which case, the messages will not be received by some of the nodes. The collision may occur without the broadcasting node detecting the problem, a phenomenon known as the hidden terminal problem [1]. In order to reduce the number of collisions, we have employed a staggering technique (Figure 4). That is, each time a node is supposed to send a message, it delays the sending by a random period of up to several milliseconds. The transmission range was set to roughly 200 meters⁴. The nodes were placed at uniformly random locations in a square area of 3500x3500 m^2 , and unless mentioned otherwise, the results are reported for networks of 1,000 nodes, which corresponds to roughly 10 neighbors per node. We have also checked other network sizes (2500x2500 m^2 and 4500x4500 m^2) with similar density, but the results were qualitatively the same, regardless of the specific network size and exact number of nodes. An additional analysis of varying network density is presented in Section 5.2.3.

Mobility was modelled by the Random-Waypoint model [16]. In this model, each node picks a random target location and moves there at a randomly chosen speed. The node then waits for a random amount of time and then chooses a new location, etc. In our case, the speed of movement ranged from 1-10 m/s. Being aware of recent criticisms of the Random-Waypoint model [4], we set the pause time to be 0 seconds and discarded the first 1000 seconds of simulation time.

In our simulations the number of broadcasting nodes varied from 1 to 200 and the size of data messages was set to 512 bytes (less than one UDP/IP packet). In every simulation, every broadcasting node sends 10 messages and then after a cool down period the simulation is being terminated. Each data point was generated as an average of 10 runs. Unless otherwise mentioned, we use the default values defined in JiST/SWANS. We have used the default Java pseudo random number generator, initialized with the current system time in milliseconds as a seed.

In the graphs, we have used the following notation: the flooding protocol is denoted FLOODING; the enhanced version of our probabilistic dissemination protocol from Figure 4 is denoted RAPID; a restricted version of the enhanced RAPID in which the gossip and the recovery mechanism were disabled is denoted RAPID-No-Gossip (in the simulation that involved selfish nodes, we run the malicious resilient version of RAPID from Figure 5); GOSSIP3 is the probabilistic protocol by Haas et al. [11]. We limited the number of times each message is gossiped by nodes in RAPID to 1. Additional gossip attempts slightly improve the delivery ratios, at the cost of additional messages.

5.2 Results

5.2.1 Changing the Number of Broadcasting Nodes

Figures 6 and 7 present results for mobile networks. Figure 6 shows the percentage of nodes that received all messages vs. the number of nodes that initiate one new broadcast per second. As expected, FLOODING delivers all messages to all the nodes. This is due to the extremely high redundancy in FLOODING (with FLOODING, every node rebroadcasts every message), which overcomes even very high number of collisions that occur. RAPID also delivers a very high percentage of messages (99.9%). GOSSIP3 delivers about 90% of the messages when the number of broadcasting nodes is relatively small (about 50 nodes). Yet, when

⁴ In SWANS one can choose the transmission power which translates into a transmission range based on power degradation and background noise.

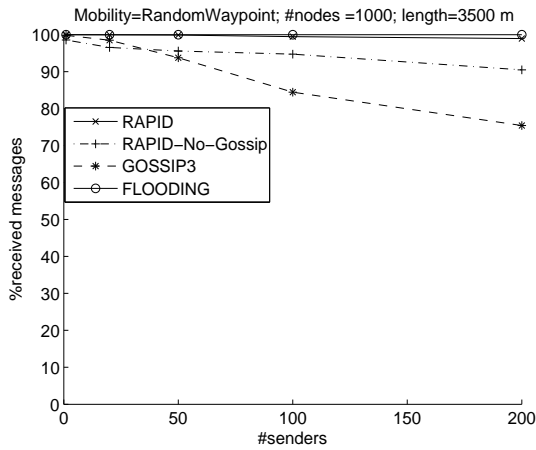


Figure 6: Message delivery ratio when all nodes are mobile

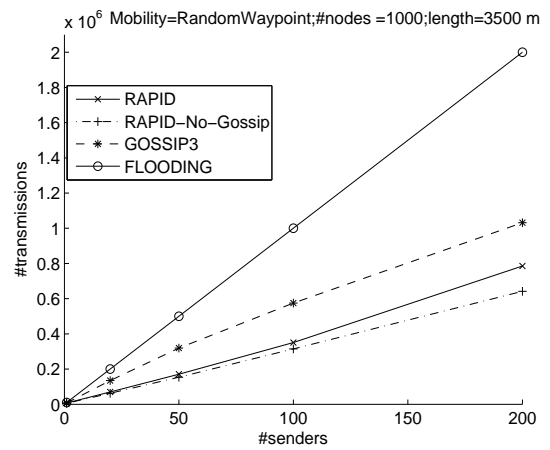


Figure 7: Network load in terms of total number of transmissions when all nodes are mobile.

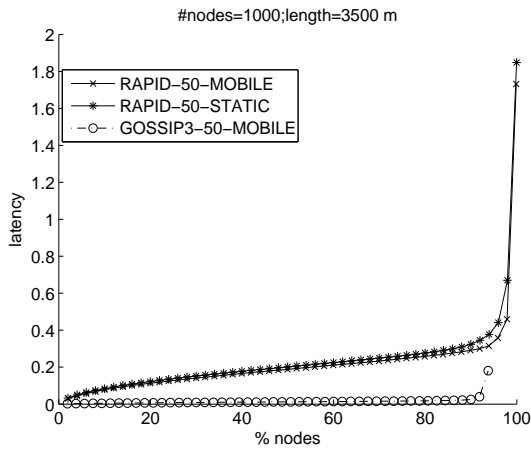


Figure 8: Latency to deliver a message to X% of the nodes (with 50 broadcasting nodes)

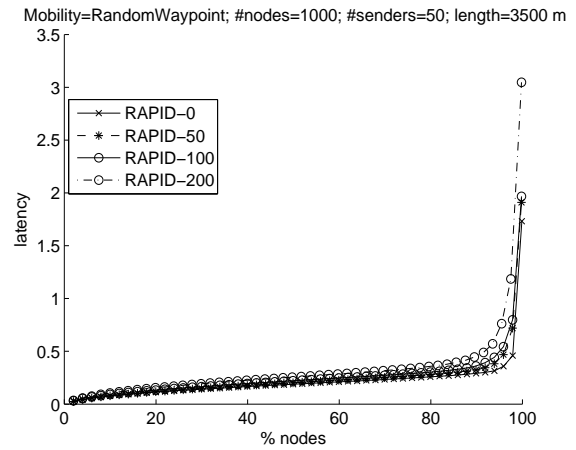


Figure 9: Latency to deliver a message to X% of the nodes when all nodes are mobile with varying number of malicious nodes (with 50 broadcasting nodes)

the number of broadcasting nodes increases and more messages are injected into the network, the percentage of messages that GOSSIP3 delivers to all the nodes decreases substantially. In particular, GOSSIP3 delivers only 75% of the messages to all nodes with 200 broadcasting nodes. The reason for this degradation is the fact that when the number of concurrent messages in the system is too high, many collisions occur causing messages to be lost. Given that GOSSIP3 only employs a probabilistic dissemination mechanism, it cannot recover these lost messages.

Interestingly, RAPID-No-Gossip manages to deliver about 90% of messages to all the nodes even with 200 concurrent senders. This is because RAPID-No-Gossip generates significantly fewer messages than GOSSIP3. Recall that the rebroadcasting probability of GOSSIP3 is fixed at 0.65. Conversely, in RAPID-No-Gossip (and RAPID) the rebroadcasting probability is set to the minimal number required to ensure

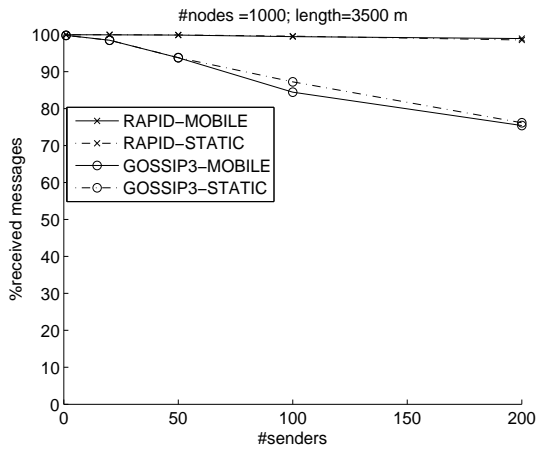


Figure 10: Message delivery ratio

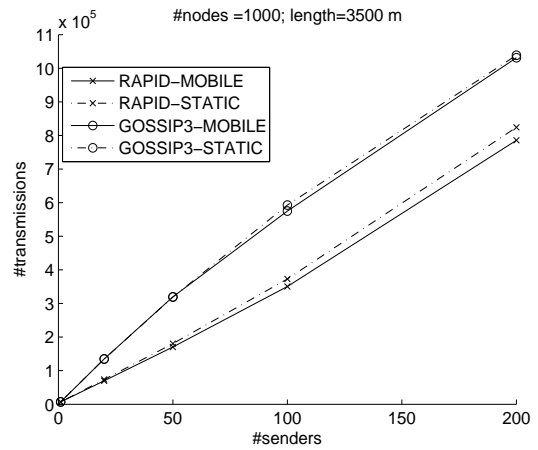


Figure 11: Network load in terms of total number of transmissions

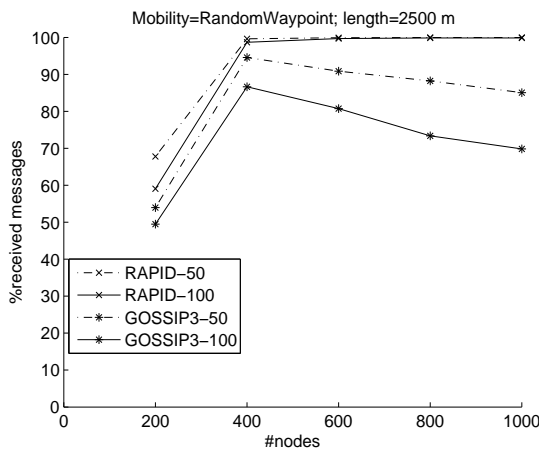


Figure 12: Message delivery ratio with varying density

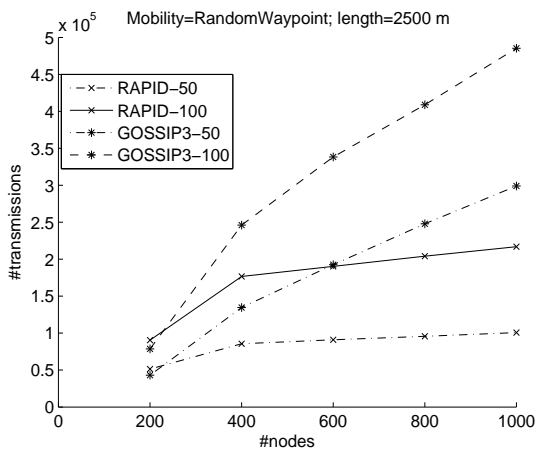


Figure 13: Message overhead with varying density

continued dissemination with high probability, depending on the number of observed neighbors of each node. Practically, with this specific network density, in our protocol the rebroadcasting probability is close to 0.35. This can also be observed when looking at the total number of transmissions, which is reported in Figure 7.

We can also observe in Figure 7 that when the number of broadcasting nodes is smaller than 100, the overhead of RAPID is only slightly larger than the overhead of RAPID-No-Gossip. Only when the number of broadcasting nodes is as high as 200 (20% of all the nodes in the system), RAPID sends more messages than RAPID-No-Gossip, in order to overcome the collisions and message loss. Hence, the decision of whether to use RAPID or RAPID-No-Gossip can be made based on the tradeoff between reliability and load for a given application.

Figure 8 explores the latency to deliver messages to a varying percentage of the nodes when the number of broadcasting nodes is 50. As can be seen by the graphs, GOSSIP3 is significantly faster than RAPID.

Yet, GOSSIP3 delivers messages only to 93% of the nodes, while RAPID delivers the messages to 98% of the nodes within 0.46 seconds and to 99.9% of the nodes within 1.732 seconds, which is good enough for most envisioned applications of MANET. In the famous “no free lunch” analogy, RAPID trades off latency (but still keeps it reasonable) for increased reliability and reduced message overhead.

5.2.2 Impact of Mobility

Figures 8, 10, and 11 present the impacts of mobility. We have run simulations while varying the speed of nodes (from 1 to 10 meters/sec) and discovered that the results are qualitatively the same. Thus, we only present the results when the speed of nodes was between 1 and 5 meters/sec and when all the nodes are static. As can be seen in Figures 8, 10, and 11, when nodes are mobile, the performance of RAPID (in terms of latency, delivery ratio and number of transmitted messages) is slightly better than when all nodes are static. This is because with mobility, the information about messages propagates faster to all areas of the network. Additionally, when a node moves, its chances of overhearing a message in one of the visited locations are higher than when it stays in the same place. Finally, when nodes move, they appear to be in more neighborhoods, which slightly reduces the retransmission probability.

5.2.3 Network Density

Figures 12 and 13 explore the delivery ratio and the number of transmissions against the density of the nodes. We can see that when the number of nodes is 200 and the network size is $2500 \times 2500 \text{ m}^2$ (the average density is about 4 nodes per neighborhood), RAPID with 50 broadcasting nodes delivers all messages to 69% of the nodes and GOSSIP3 delivers all messages to 51% of the nodes. These results are explained by the very poor network connectivity. We can also see that when the number of nodes is 400 (the average density is about 8 nodes), GOSSIP3 with 50 broadcasting nodes delivers all messages to 93% of the nodes. This simulation shows that the delivery ratio of GOSSIP3 for both 50 and 100 broadcasting nodes is the best when the density is about 8 nodes.

Interestingly, this echoes the results of [23]. Moreover, we know from Gupta and Kumar’s connectivity bound for ad hoc networks [10] that the networks’ connectivity is ensured with high probability when $r \geq a\sqrt{\frac{C \ln(n)}{n}}$, with r being the transmission range, a the length of the network area, C is a constant such that $C > \frac{1}{\pi}$, and n the number of nodes. Recall that in our case, $r = 200$ and $a = 2,500$. With these numbers, we get that for $n = 200$, the network is not likely to be connected, but for $n = 400$, the network is already connected. Hence, with $n = 200$, no protocol can achieve high delivery ratios, yet with $n = 400$, good reliability can already be obtained.

When the number of nodes grows beyond 400 and consequently the network’s density increases, the delivery ratio of GOSSIP3 significantly decreases. This phenomenon is even more acute with 100 broadcasting nodes. It can be explained by the fact that in such a scenario, many nodes in the same neighborhood rebroadcast messages, so a lot of collisions occur, resulting in a high percentage of message loss. As we mentioned before, GOSSIP3 has no recovery mechanism for such lost messages. In contrast, RAPID loses fewer messages due to collisions since its probabilistic transmission part self adjusts to the network’s density. Moreover, RAPID recovers lost messages using its gossiping mechanism.

When looking at the total number of transmissions in Figure 13, we can observe that RAPID scales much better than GOSSIP3 with the density of the network. The number of transmission is almost constant (slightly increasing mainly due to the heartbeat messages and increased collisions) due to the fact that RAPID tunes its rebroadcasting probability based on the number of observed neighbors. This validates our theoretical analysis in Section 3.3.

# Selfish	Delivery (Mobile)	Delivery (Static)	# Messages (Mobile)	# Messages (Static)
0	99.9%	99.9%	170600	180357
1	99.8%	99.87%	170178	179824
20	99.98%	99.9%	169754	178753
50	99.97%	99.88%	167840	177912
100	99.96%	99.7%	165590	174703
200	99.87%	99.2%	160910	168706

Table 1: Delivery ratio and message count vs. the number of selfish nodes (with 50 broadcasting nodes)

5.2.4 Selfish Nodes

Figure 9 explores the latency to deliver a message to $X\%$ of the nodes when the total number of nodes in the system is 1,000 and some nodes are selfish, i.e., refuse to rebroadcast messages. In this graph, we use the notation RAPID- Y to indicate that RAPID was run with Y selfish nodes. Clearly, the latency grows with the number of selfish nodes. Yet, we can see that even when the number of selfish nodes is 200 (20% of all nodes), RAPID delivers the messages to 98% of the nodes within 1.18 seconds. We would like to point out that by fine tuning the rate of gossips and the other timers in the system, it is possible to reduce the quantitative latency numbers. The numbers here do not include such tuning, yet we have started exploring this option.

Table 1 presents the delivery ratio and the message overhead in mobile and static networks for varying numbers of selfish nodes. We can see that neither mobility nor the number of selfish nodes significantly influences the delivery ratio of RAPID, which consistently delivers more than 99% of the messages to all nodes. Interestingly, we also notice that the message overhead becomes smaller as the number of selfish nodes increases. One could expect that the message overhead should increase with the number of selfish nodes. In particular, the protocol must send more REQUEST messages for recovering missing messages that were not rebroadcasted by selfish nodes. However, selfish nodes do not send any messages except for heartbeat messages. This reduces both the number of retransmissions and the number of message collisions. Hence, overall, this results in a reduced number of message transmissions.

6 Related Work

A comprehensive study of broadcasting and multicasting protocols for wireless ad hoc networks can be found in [29, 34]. Here, we only discuss the most relevant protocols to our work.

The simplest probabilistic broadcast protocol is probabilistic flooding [11, 30]. In this scheme, each node rebroadcasts a message with a predefined probability \mathcal{P} . Works by Haas et al. [11] and Shiffer et al. [24] study the rebroadcasting probability \mathcal{P} with regard to the so called *phase transition phenomena*. Both works establish that the delivery distribution has a bimodal behavior with regard to some threshold probability $\bar{\mathcal{P}}$, in a sense that for any $\mathcal{P} > \bar{\mathcal{P}}$ almost all nodes will receive the message and for $\mathcal{P} < \bar{\mathcal{P}}$ almost none. Both works show that the threshold probability $\bar{\mathcal{P}}$ is around 0.59 – 0.65; in [24] this is done analytically based on percolation theory while in [11] it is obtained by simulations. It is also noted in [11] that the threshold probability depends on nodes density, yet without providing any theoretical means to evaluate this dependance. We have studied the delivery distribution using probabilistic methods in Section 3. We have shown that by making several probabilistic assumptions (the deterministic corrective actions of our protocol make sure they hold), the delivery distribution function behaves in a concave manner rather than

being bimodal. That is, nodes coverage initially grows fast with \mathcal{P} . Then, at some critical point, the added coverage becomes negligible with further increase of \mathcal{P} .

Other probabilistic approaches [11, 30, 31] include *counter-based*, *distance-based*, and *location-based* mechanisms. The main idea in these schemes is that the additional space coverage obtained by each additional broadcast decreases with the number of broadcasts. For example, [11] presents a variant of the probabilistic protocol in which every node monitors the transmissions of its neighbors and rebroadcasts a message if it has not heard M transmissions of the same message. Yet, those protocols suffer from increased latency due to the packet delay introduced at each hop (as explained in Section 4.2.3) and none of them guarantee a reliable dissemination of messages to all nodes (as explained in Section 4.2). On the other hand, RAPID guarantees reliable dissemination in any topology.

The works in [26, 37] utilize an adapted probabilistic flooding that makes use of local density. The approaches of those works are based on the observation that the retransmission probability \mathcal{P} should be adjusted relatively to the local nodes density. In [37] this is done through counters, while in [26] the uniform density is assumed. However, those works contain little theoretical analysis of the proposed schemes and like other counter-based schemes can also fail to provide reliability on certain topologies. To the best of our knowledge, our work is the first to provide a theoretical analysis of the optimal usage of nodes density in order to set \mathcal{P} .

The *color-based* scheme has been recently proposed in [17]. In this scheme, each node forwards a message if it can assign it a color from a given pool, which it has not already overheard after a random time. Using geometric analysis, they have shown that the size of the rebroadcasting group is within a small constant factor of the optimum. The *color-based* scheme is actually an advanced type of a counter-based scheme, and thus incurs similar latencies and does not guarantee high reliability on arbitrary topologies. The bounds on the size of the rebroadcasting set in homogenous dense network in [17] are similar to our analysis in Section 3.1. Yet, our analysis is much simpler and holds for every probabilistic algorithm that picks nodes uniformly at random in homogenous network, while their analysis only holds for color-based schemes.

A number of works have been designed to provide a reliable dissemination of messages to all nodes. An approach called Mistral tries to compensate for missing messages in probabilistic dissemination by using forward error correction techniques [22]. In contrast, our approach to recovery of messages is based on gossip. Also, Mistral cannot cope with malicious behavior.

The idea that a process can detect that it is missing a message by exchanging messages with other processes previously appeared in the MNAK layer of the Ensemble system in 1996 [12]. Additionally, randomized gossip has been used as a method of ensuring reliable delivery of broadcast/multicast messages while maintaining high throughput in the PBCast/Bimodal work [3] as well as in several followup papers, e.g., [7]. In a way, the idea in our work is an inverse of the idea at PBCast/Bimodal work. In the PBCast/Bimodal, each node deterministically sends every message to all the nodes and later gossips about the existing messages with a random subset of nodes. Conversely, in RAPID each node disseminates the messages to a random set of nodes (chosen among its physical neighbors) and later deterministically gossips about the existing messages with all its neighbors.

Demers et al. were the first to use gossip in the context of replicated databases in [5]. A generic framework for presenting gossip protocols was proposed in [15], and in particular highlighted the advantages of designing gossiping protocols using a pull-push approach for higher reliability. This framework was later extended to ad-hoc networks in [2, 8]. An example of a protocol for ad-hoc networks that uses a pull-push approach and is easily expressed in the above framework is [19]. Our protocol can also be seen as a specific instantiation of pull-push dissemination.

An additional protocol for reliable broadcast and multicast in ad hoc networks called Scribble has been proposed in [33]. In Scribble, the responsibility for dissemination initially rests with the multicast originator, which periodically broadcasts the message, and is subsequently passed around to other nodes. The termination condition in Scribble is determined by piggybacking a bit vector for all known nodes that have received the broadcast message. Scribble does not employ probabilistic mechanisms and thus suffer from increased latency and is more message consuming.

7 Discussion and Conclusions

In this work, we have described a reliable broadcast protocol for mobile ad-hoc networks. The protocol takes advantage of the locally observed network's density in order to reduce the number of message transmissions while maintaining very high delivery ratios. The protocol also employs a deterministic gossip based mechanism to recover messages that were not delivered by the probabilistic dissemination process. We have also presented theoretical results, which motivate our protocol, and explain why our design choices are inherent to the environment. According to this analysis, if the retransmission probability is set inversely proportional to the node's density and if the rebroadcasting nodes are picked uniformly at random, then the actual number of retransmissions required to guarantee high coverage is constant with regards to the overall number of nodes.

Practically, since networks are seldomly perfectly uniform, pure probabilistic approaches fail to pick nodes with true uniform distribution. Consequently, pure probabilistic broadcasting protocols have the following limitation. In order to obtain very high delivery ratios, they must set the rebroadcasting probability to very high values. Even then, some messages might still be delivered only to a few nodes and these protocols do not handle well selfish and malicious behaviors. An important feature of our approach is that we employ both probabilistic dissemination with a set of corrective deterministic measures. Hence, we can ensure that the requirements of the formal analysis more or less hold in any environment. This results in very efficient delivery for homogenous topologies, without sacrificing reliability in arbitrary topologies.

Our measurements confirm that for non-sparse networks, our protocol behaves very well. That is, the protocol obtains very high delivery ratios while sending relatively few messages.

References

- [1] D. Allen. Hidden terminal problems in Wireless LAN's. In *IEEE 802.11 Working Group Papers*, 1993.
- [2] Z. Bar-Yossef, R. Friedman, and G. Kliot. RaWMS - Random Walk based Lightweight Membership Service for Wireless Ad Hoc Networks. In *Proc. of the 7th ACM Intr. Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 238–249, 2006.
- [3] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, , and Y. Minsky. Bimodal Multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, May 1999.
- [4] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing (WCMC)*., 2(5):483–502, 2002.
- [5] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proc. of the 6th annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–12, New York, NY, USA, 1987. ACM Press.
- [6] V. Drabkin, R. Friedman, and M. Segal. Efficient Byzantine Broadcast in Wireless Ad-Hoc Networks. In *Proc. of the 6th IEEE Conference on Dependable Systems and Networks (DSN)*, pages 160–169, June 2005.

- [7] P. Th. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight Probabilistic Broadcast. *ACM Transactions on Computing Systems*, 21(4):341–374, 2003.
- [8] D. Gavidia, S. Voulgaris, and M. van Steen. Epidemic-style Monitoring in Large-Scale Sensor Networks. Technical Report IR-CS-012, Vrije Universiteit, Netherlands, March 2005.
- [9] K. Guo and I. Rhee. Message Stability Detection for Reliable Multicast. In *Proc. of IEEE INFOCOM'2000*, March 2000.
- [10] P. Gupta and P. Kumar. Critical Power for Asymptotic Connectivity in Wireless Networks. In *Stochastic Analysis, Control, Optimization and Applications*, Birkhauser, Boston, pages 547–566, 1998.
- [11] Z. Haas, J. Halpern, and L. Li. Gossip-Based Ad Hoc Routing. In *Proc. of the 21st Conference of the IEEE Communication Society (INFOCOM)*, pages 1707–1716, June 2002.
- [12] M. Hayden. The Ensemble System. Technical Report TR98-1662, Department of Computer Science, Cornell University, January 1998.
- [13] F. Ingelrest, D. Simplot-Ryl, and I. Stojmenovic. Broadcasting in Hybrid Ad Hoc Networks. In *Proc. 2nd Annual Conference on Wireless On demand Network Systems and Services (WONS)*, 2005.
- [14] I. Ioannidis and B. Carbunar. Scalable Routing in Hybrid Cellular and Ad-Hoc Networks. In *1st IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, October 2004. Poster.
- [15] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In *Proc. of the 5th Middleware*, pages 79–98, 2004.
- [16] D.B. Johnson and D.A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, volume 353. 1996.
- [17] A. Keshavarz-Haddad, V. J. Ribeiro, and R. H. Riedi. Color-based broadcasting for ad hoc networks. In *Proc. of the 4th IEEE Int. Symposium on Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks (WiOpt)*, pages 49–58, April 2006.
- [18] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks, 2004.
- [19] J. Luo, P. Eugster, and J.-P. Hubaux. PILOT: Probabilistic lightweight group communication system for mobile ad hoc networks. *IEEE Trans. on Mobile Computing*, 3(2):164–179, April–June 2004.
- [20] P. Panchapakesan and D. Manjunath. On the Transmission Range in Dense Ad Hoc Radio Networks. In *Proc. of IEEE Signal Processing Communication (SPCOM)*, 2001.
- [21] M. D. Penrose. *Random Geometric Graphs*. Oxford Press, 2003.
- [22] S. Pleisch, M. Balakrishnan, K. Birman, and R. van Renesse. MISTRAL: Efficient Flooding in Mobile Ad-hoc networks. In *Proc. of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 1–12, 2006.
- [23] E. Royer, P. Melliar-Smith, and L. Moser. An Analysis of the Optimum Node Density for Ad hoc Mobile Networks. In *Proc. of the IEEE International Conference on Communications*, June 2001.
- [24] Y. Sasson, D. Cavin, and A. Schiper. Probabilistic Broadcast for Flooding in Wireless Mobile Ad hoc Networks. In *Proc. of the IEEE Wireless Comm. and Networking Conference (WCNC)*, March 2003.
- [25] B. Schneier. *Applied Cryptography*. Wiley, 1996.
- [26] D. Scott and A. Yasinsac. Dynamic probabilistic retransmission in ad hoc networks. In *Proc. of the Int. Conference on Wireless Networks (ICWN)*, pages 158–164, Las Vegas, Nevada, June 2004.
- [27] K. Singh, A. Nedos, G. Gaertner, and S. Clarke. Message Stability and Reliable Broadcasts in Mobile Ad-Hoc Networks. In *Proc. of the 4th ADHOC-NOW*, pages 297–310, October 2005.

- [28] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, 1996. 3rd Ed.
- [29] C.K. Toh. *Ad Hoc Mobile Wireless Networks*. Prentice Hall, 2002.
- [30] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless Networks*, 8(2/3):153–167, 2002.
- [31] Y.-C. Tseng, S.-Y. Ni, and E.-Y. Shih. Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc networks. In *Proc. of the 21st International Conference on Distributed Computing Systems (ICDCS)*, pages 481–488, 2001.
- [32] Cornell University. JiST/SWANS Java in Simulation Time / Scalable Wireless Ad Hoc Network Simulator. Available at <http://jist.ece.cornell.edu/>.
- [33] E. Vollset and P. Ezhilchelvan. Enabling reliable many-to-many communication in ad-hoc pervasive environments. In *Proc. of the 2nd Intr. Workshop on Mobile Peer-to-Peer Computing (MP2P)*, 2005.
- [34] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proc. of the 3rd Intr. Symp. on Mobile Ad Hoc Networking & Computing (MobiHoc)*, pages 194–205, 2002.
- [35] C.W. Wu and Y.C. Tay. AMRIS: A Multicast Protocol for Ad-Hoc Wireless Networks. In *Proc. of the IEEE MILCOMM*, Nov. 1999.
- [36] J. Wu and H. Li. On Calculating Connected Dominating Sets for Efficient Routing in Ad Hoc Wireless Networks. In *Proc. of the 3rd DialM*, pages 7–14, 1999.
- [37] Q. Zhang and D. P. Agrawal. Dynamic probabilistic broadcasting in MANETs. *Journal of Parallel Distributed Computing*, 65(2):220–233, 2005.