

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

In this paper, we propose a novel ReliAble Probabilistic Dissemination protocol, 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.

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.

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 [30]. 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 applica-

tions and improved services for mobile users, especially as the computing power of mobile devices becomes stronger. Example applications include interactive distributed games, collaborative 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 devices in the network. A useful broadcast service should be both efficient and provide a good level of reliability, meaning that most nodes in the system will receive almost every broadcasted message. A simple implementation of broadcast in a multiple hop network is by employing flooding [29]. 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 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 a large number of collisions [31].

Common alternatives to flooding are either to perform a constrained flooding on top of a deterministic overlay, e.g., [19, 28, 36, 37], or to perform a probabilistic flooding, e.g., [12, 20]. 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. It is also hard to make overlays resilient to malicious or even selfish behavior (i.e., nodes that only send their own messages [7]). 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 [5, 12, 20]. Probabilistic protocols are appealing since they are very simple, and are inherently robust to failures and mobility. Yet, as was discovered in [12, 20, 25], in order to obtain very high reliability levels with pure probabilistic broadcasting, one has to set the retransmis-

sion probability to relatively high values. Consequently, such schemes still generate a large number of redundant messages. Other approaches [5, 12, 22, 31, 32] 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 to understand them. Finally, as we discuss later in this section, 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 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). 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 insight, we have deduced the following principles for obtaining an efficient and reliable probabilistic broadcast protocol: The retransmission probability of each node is set to 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 probabil-

ity is chosen so that the expected number of retransmitting nodes will be the most cost effective 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 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. 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, thus greatly reducing the number of messages generated by the protocol. The timer based corrections 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 resulting protocol 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. In particular, it does not rely on any 2-hop neighborhood information. The paper includes a detailed performance evaluation carried by simulation, validating our claims.

Notice that without a gossip and recovery mechanism, one cannot guarantee reliable delivery of all messages even if all nodes broadcast all the messages with very high probability, and even if a deterministic scheme is employed. This is due to the possibility of collisions in wireless networks. Moreover, in probabilistic and counter based schemes, 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 or message counting, it is possible that p will decide not to rebroadcast m due to rebroadcasts by n_i nodes. Consequently, q will never receive it. However, if we use the gossip and recovery mechanism, eventually q will find out that it is missing m and will ask its neighbors that have m to rebroadcast it.

2 System Model and Definitions

Assume a collection of *nodes* placed in a given finite size area. A node in the system is a device owning an omnidirectional 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

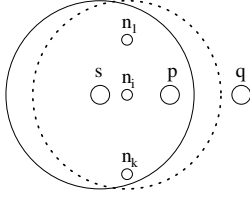


Figure 1. A transmission by a node s can be received by p, n_1, \dots, n_k

transmission power (called *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 a node p by r_p . 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 . Messages can be lost. For example, if two nodes p and q transmit a message at the same time, and there exists a node z that is a direct neighbor of both, then z will receive neither message, and we say that there was a *collision*. Yet, we assume that a message is delivered with a positive probability.

Finally, in Section 4.2, 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*. Throughout this work, 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.

Our theoretical analysis in this section relies on the famous *Random Geometric Graph* (RGG) model, which is

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.

often used to model the network connectivity graph of 2-dimensional wireless ad hoc networks and sensor networks [11]. 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 within Euclidean distance r of each other [23]. 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 [11]) 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} w.h.p.

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. Let Y_p be a random variable corresponding to the number of times that a node p has received a given message. We calculate below an upper bound on the probability that an arbitrary node will not receive m , or in other words, $\Pr(Y_p = 0)$.

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

Proof: S is the set of all nodes that transmit message m . The size of S is a binomial random variable with mean $n\mathcal{P}$. For each $q \in S$ and any node p , let $X_{p,q}$ 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

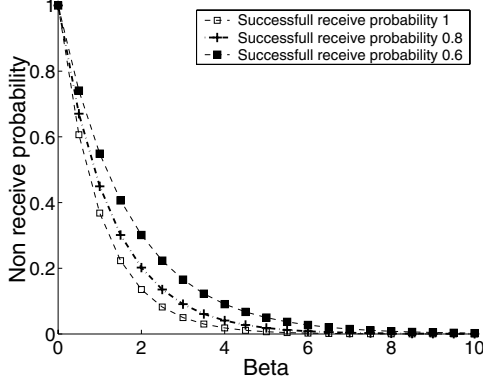


Figure 2. An upper bound on the probability that an arbitrary node does not receive a message m

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 $\Pr(X_{p,q} = 1) = Q \frac{\pi r^2}{ab}$.

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

$$\begin{aligned} \Pr(Y_p = 0) &= \sum_{i=0}^n \Pr(Y_p = 0 || S| = i) \Pr(|S| = i) = \\ &= \sum_{i=0}^n \prod_{q \in S, |S|=i} \Pr(X_{p,q} = 0) \Pr(|S| = i) = \\ &= \sum_{i=0}^n (1 - Q \frac{\pi r^2}{ab})^i \binom{n}{i} \mathcal{P}^i (1 - \mathcal{P})^{n-i} = \\ &= \sum_{i=0}^n \binom{n}{i} (\mathcal{P} - \mathcal{P} Q \frac{\pi r^2}{ab})^i (1 - \mathcal{P})^{n-i} = (1 - \mathcal{P} Q \frac{\pi r^2}{ab})^n \\ &\leq (e^{-\mathcal{P} Q \frac{\pi r^2}{ab}})^n = e^{-\frac{\beta}{d_{\text{avg}}} Q \frac{\pi r^2}{ab} n} \leq e^{-\frac{\beta ab}{\pi r^2 (n-1)} Q \frac{\pi r^2}{ab} n} \leq e^{-\beta Q} \end{aligned}$$

In the fourth line we have used the binomial coefficients formula and in the last line the inequality $1 - x < e^{-x}$, which holds for all $x > 0$. ■

Figure 2 depicts an upper bound $e^{-\beta Q}$ on the value of $\Pr(Y_p = 0)$ for an arbitrary node p as a function of β and Q . It can be seen that the probability that a given node does not receive a message m is small for quite small values of β . For example, for $Q = 0.9$, $\Pr(Y_p = 0)$ is bounded by 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 $Q = 0.9$, approximately 90% of all nodes will receive m .

A broadcasting algorithm that sets the retransmission probability \mathcal{P} inversely proportional to the average degree has the following appealing property: Let S be the set of

nodes that (re)broadcast a given message. The number of transmissions is constant with respect to the number of nodes n and to the nodes' density. Specifically,

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

4 The RAPID Protocol

In RAPID, 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/hello message (unless it has already sent another message during a predefined time interval). As already mentioned in Section 3 (Figure 2), setting $\beta = 2.5$ results in a very cost effective algorithm in terms of the tradeoff between the rebroadcasting probability and the reliability level.

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 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.

When examining the graph in Figure 2, it can be seen that the reliability level obtained is highly dependant on the probability that a transmission will not be lost. Specifically, in wireless networks, most message losses are caused by collisions. Hence, to reduce the chance of collision, and thereby be able to obtain reliability levels similar to the bottom most line of Figure 2, RAPID employs jitter. When a node decides to rebroadcast a message, it waits for a short random time before doing so. Hence, the small probability of rebroadcasting and the short jitter before rebroadcasting means that RAPID very rarely causes collisions.

The ideas described above have 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 mechanism, which is slow.

Thus, we slightly change the protocol by adding two complementing corrective measures that are based on each node monitoring 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 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 and [31], 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 also retransmit.

4.1 RAPID in Details

The pseudo-code of RAPID is listed in Figure 3. In the code, 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. We also use a queue called `cast_queue`, with the `add` method that accepts as parameters the sending probability, a time, the message itself and the type of the message. The protocol includes the dissemination of messages as well as gossip and recovery to overcome message loss. In addition, the protocol includes a monitoring and jitter mechanism to reduce collisions as well as quickly correct situations in which none of the neighbors decided to rebroadcast a message. We explain these issues below.

The Dissemination Task in Detail. Message dissemination in our protocol consists of the following steps: (1) The originator p of a message m sends $m||header(m)$ to all

```

Upon send(msg) by application do
(01) header := msg_id||node_id;
(02) data_msg := header||msg;
(03) gos_msg := header;
(04) prob_bcast(prob = 1, data_msg, DATA);
(05) lazycast(gos_msg, GOSSIP);

Upon receive(msg, DATA) sent by  $p_j$  do
(06) if (have not received this msg before) then
(07)   Accept( $p_j$ , msg); /*forward it to the application*/
(08)   cast_queue.add(prob = min(1,  $\frac{\beta}{|N(p)|}$ ),
(09)     time=random(0, short_jitter), msg, DATA);
(10)   lazycast(gos_msg, GOSSIP);
(11) endif;

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

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

Interceptor
(21) if (msg that appears in cast_queue was received) then
(22)   cast_queue.remove(msg);
(23) endif;

Upon Expiration of timer of msg in cast_queue do
(24) cast_queue.remove(msg);
(25)  $pr$  = the probability attached to msg;
(26)  $type$  = the message type associated with msg;
(27) prob_bcast(prob =  $pr$ , msg, type);
(28) if (msg was not broadcasted) then
(29)   cast_queue.add(prob = 1, time=long_jitter, msg, type);
(30) endif;

```

Figure 3. The RAPID Protocol

nodes in $N(p)$ (Lines 01–04 in Figure 3). The header part of m includes a 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 05). (3) When a node p receives a message m for the first time, p accepts m (Lines 06–07). (4) p schedules a broadcast of m with probability $\min(1, \frac{\beta}{|N(p)|})$ after some random short jitter (Line 08 – our protocol was simulated with β equals to 2.5). (5) If a node p receives a message m it has already received beforehand, then m is ignored.

Whenever p schedules a rebroadcast in Step (4) to occur after some random jitter, if during this time p overhears a retransmission of m by another node, then p cancels its own retransmission of m (Lines 18–20). 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 25–27).

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 09). p does not forward gossips about messages it has not received yet, 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 11–14). 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 schedules a broadcast of the missing message (Lines 15–17) after a random short jitter.

(4) As in the dissemination task, 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 (Lines 18–20 and 21–27).

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 [10, 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.

Latency of RAPID. In both RAPID and *counter-based* protocols [5, 12, 31, 32], 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 3 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. In order to be effective, the duration of the jitter must be proportional to the number of expected concurrent transmissions. The expected number of concurrent transmitters competing for transmission due to the probabilistic mechanism is quite small (β). On the other hand, in the situations in which *long_jitter* is used, and similarly in counter based protocols, all nodes in the neighborhood might transmit concurrently. Hence, *long_jitter* must be long enough to accommodate for that. Consequently,

short_jitter is much shorter than *long_jitter*. For example, to completely eliminate collisions with high probability, then following the birthday paradox, the length of the jitter must be proportional to s^2 , where s is the expected number of concurrent senders. Most times in RAPID the timer-based corrective measure will not be used, so average latency is mostly dominated by *short_jitter*.

4.2 Maliciousness Resilient RAPID

It is possible to slightly adjust RAPID in order to make it resilient to malicious attacks. These changes mainly relate to adding signatures on messages, and having each node monitor its neighbors in order to discount irresponsible nodes as neighbors. This way nodes cannot forge messages or gossip about non-existent messages, and the probability of rebroadcasting a message is only computed w.r.t. the number of well behaving nodes. Due to lack of space, the details appear in the full version of this paper.

5 Simulations

We evaluate below the performance of RAPID and compare it with the performance of flooding and with the performance of the GOSSIP3 protocol [12]. 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 nodes (GOSSIP3 was simulated with probability 0.65 and $M=1$). GOSSIP3 is chosen since 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 [12]. 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.

We have used the JiST/SWANS simulator [33] 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]. The transmission range was set to roughly 200 meters. The nodes were placed at uniformly random locations in a square area of $3500 \times 3500 m^2$, 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 ($2500 \times 2500 \text{ m}^2$ and $4500 \times 4500 \text{ m}^2$) 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 5.1. Mobility was modelled by the Random-Waypoint model [16] with the speed of movement picked from the range 1-10 m/s. Due to 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. 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.

We have used the following notation: FLOODING denotes the flooding protocol; our probabilistic dissemination protocol from Figure 3 is denoted RAPID; a restricted version of RAPID in which the gossip and recovery mechanism was disabled is denoted RAPID-No-Gossip; GOSSIP3 is the probabilistic protocol by Haas et al. [12]. 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.

5.1 Results

Changing the Number of Broadcasting Nodes. Figures 4 and 5 present results for mobile networks. Figure 4 shows the percentage of nodes that received all messages vs. the number of nodes that initiate one new broadcast per second. FLOODING delivers all messages to all the nodes, due to the extremely high redundancy in FLOODING, 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 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. 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. Since 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

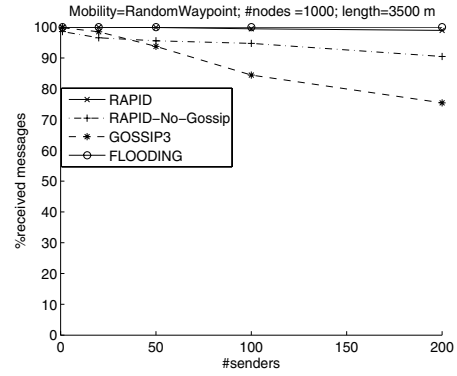


Figure 4. Message delivery ratio

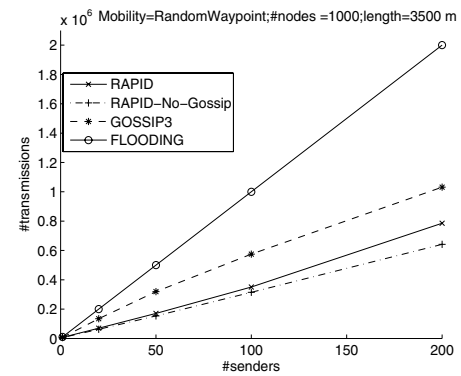


Figure 5. Network load

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 re-

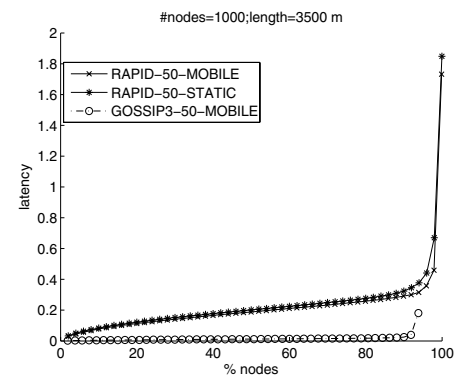


Figure 6. Latency to deliver a message to X% of the nodes (with 50 broadcasting nodes)

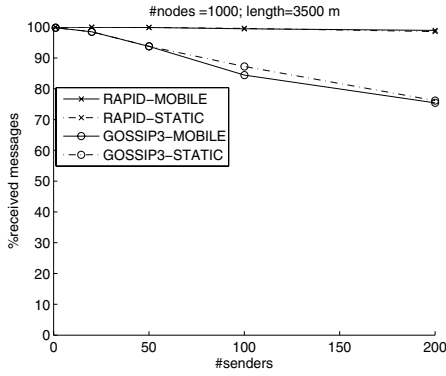


Figure 7. Message delivery ratio

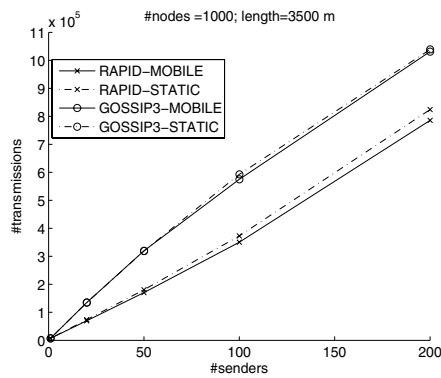


Figure 8. Network load

quired to ensure continued dissemination with high probability, depending on the number of observed neighbors of each node.

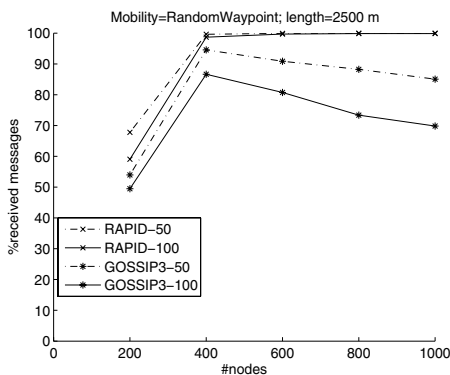


Figure 9. Message delivery ratio with varying density



Figure 10. Network load with varying density

Figure 6 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.

Impact of Mobility. Figures 6, 7, and 8 present the impacts of mobility. We have run simulations while varying the speed of nodes 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 6, 7, and 8, 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.

Network Density. Figures 9 and 10 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. This is due to the very poor network connectivity. Also, 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 [24]. Moreover, we know from Gupta and Kumar’s connectivity bound for ad hoc networks [11] 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 gossip.

When looking at the total number of transmissions in Figure 10, 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.

6 Related Work

A comprehensive study of broadcasting and multicasting protocols for wireless ad hoc networks can be found in [30, 35]. Here, we only discuss the most relevant protocols to our work.

The simplest probabilistic broadcast protocol is probabilistic flooding [12, 31]. In this scheme, each node rebroadcasts a message with a fixed probability \mathcal{P} . Works by Haas et al. [12] and Sasson et al. [25] study the rebroadcasting probability \mathcal{P} with regard to the *phase transition phenomena*. A generic epidemic model for information diffusion in MANETs has appeared in [18].

Other probabilistic approaches [5, 12, 17, 22, 31, 32] include *counter-based*, *distance-based*, *location-based*, and *color-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. Yet, those protocols suffer from increased latency due to the packet delay introduced at each hop (as explained in

Section 4.1) and none of them guarantees reliable dissemination of messages to all nodes (as explained in Section 4).

The works in [26, 38] utilize an adapted probabilistic flooding that makes use of local density. Their approaches are based on the observation that the retransmission probability \mathcal{P} should be adjusted relatively to the local nodes density. In [38] 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 for calculating \mathcal{P} based on nodes density.

Demers et al. were the first to use gossip in the context of replicated databases in [6]. This idea was later adopted and extended in works such as the MNAK layer of the Ensemble system, as well as the PBCast/Bimodal work and its variants [3, 8]. 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.

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, 9]. An example of a protocol for ad-hoc networks that uses a pull-push approach and is easily expressed in that framework is [21]. Our protocol can also be seen as an instance of pull-push dissemination.

Scribble is a protocol for reliable broadcast and many-cast in ad hoc networks [34]. In Scribble, the responsibility for dissemination initially rests with the 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.

7 Discussion and Conclusions

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 additionally employs a deterministic gossip based mechanism to recover messages that were not delivered by the probabilistic dissemination. We have also presented theoretical results, which motivate our protocol, and explained why our design choices are in-

herent to the environment. According to this analysis, if the retransmission probability is set inversely proportional to the node's density, then the actual number of retransmissions required to guarantee high node coverage is constant with regards to the overall number of nodes.

Practically, since networks are seldomly perfectly uniform, 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. Friedmann, and G. Kliot. RaWMS - Random Walk based Lightweight Membership Service for Wireless Ad Hoc Networks. In *MobiHoc*, 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. *WCMC*, 2(5):483–502, 2002.
- [5] J. Cartigny and D. Simplot. Border Node Retransmission Based Probabilistic Broadcast Protocols in Ad-Hoc Networks. *Telecommunication Systems*, 22(1–4):189–204, 2003.
- [6] 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 *PODC*, pages 1–12, 1987.
- [7] V. Drabkin, R. Friedman, and M. Segal. Efficient Byzantine Broadcast in Wireless Ad-Hoc Networks. In *DSN*, pages 160–169, June 2005.
- [8] 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.
- [9] 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.
- [10] K. Guo and I. Rhee. Message Stability Detection for Reliable Multicast. In *Proc. of IEEE INFOCOM'2000*, March 2000.
- [11] 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.
- [12] Z. Haas, J. Halpern, and L. Li. Gossip-Based Ad Hoc Routing. In *INFOCOM*, pages 1707–1716, June 2002.
- [13] F. Ingelrest, D. Simplot-Ryl, and I. Stojmenovic. Broadcasting in Hybrid Ad Hoc Networks. In *WONS*, 2005.
- [14] I. Ioannidis and B. Carburnar. Scalable Routing in Hybrid Cellular and Ad-Hoc Networks. In *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 *Middleware*, pages 79–98, 2004.
- [16] D.B. Johnson and D.A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, 1996.
- [17] A. Keshavarz-Haddad, V. J. Ribeiro, and R. H. Riedi. Color-based broadcasting for ad hoc networks. In *WiOpt*, pages 49–58, 2006.
- [18] A. Khelil, C. Becker, J. Tian, and K. Rothermel. An Epidemic Model for Information Diffusion in MANETs. In *MSWiM*, pages 54–60, 2002.
- [19] A. Laouiti, A. Qayyum, and L. Viennot. Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks. In *HICSS*, 2001.
- [20] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks, 2004.
- [21] 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, 2004.
- [22] H. Miranda, S. Leggio, L. Rodrigues, and K. Raatikainen. A Power-Aware Broadcasting Algorithm. In *PIMRC*, September 2006.
- [23] M. D. Penrose. *Random Geometric Graphs*. Oxford Press, 2003.
- [24] E. Royer, P. Melliar-Smith, and L. Moser. An Analysis of the Optimum Node Density for Ad hoc Mobile Networks. In *ICC*, 2001.
- [25] Y. Sasson, D. Cavin, and A. Schiper. Probabilistic Broadcast for Flooding in Wireless Mobile Ad hoc Networks. In *WCNC*, 2003.
- [26] D. Scott and A. Yasinsac. Dynamic probabilistic retransmission in ad hoc networks. In *ICWN*, pages 158–164, June 2004.
- [27] K. Singh, A. Nedos, G. Gaertner, and S. Clarke. Message Stability and Reliable Broadcasts in Mobile Ad-Hoc Networks. In *ADHOC-NOW*, pages 297–310, October 2005.
- [28] I. Stojmenovic, M. Seddigh, and J. Zunic. Dominating Sets and Neighbor Elimination Based Broadcasting Algorithms in Wireless Networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(1):14–25, January 2002.
- [29] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, 1996. 3rd Ed.
- [30] C.K. Toh. *Ad Hoc Mobile Wireless Networks*. Prentice Hall, 2002.
- [31] 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.
- [32] 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 *ICDCS*, pages 481–488, 2001.
- [33] Cornell University. JiST/SWANS Java in Simulation Time / Scalable Wireless Ad Hoc Network Simulator. Available at <http://jist.ece.cornell.edu/>.
- [34] E. Vollset and P. Ezhilchelvan. Enabling reliable many-to-many communication in ad-hoc pervasive environments. In *MP2P*, 2005.
- [35] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *MobiHoc*, pages 194–205, 2002.
- [36] C.W. Wu and Y.C. Tay. AMRIS: A Multicast Protocol for Ad-Hoc Wireless Networks. In *Proc. of MILCOMM*, 1999.
- [37] J. Wu and H. Li. On Calculating Connected Dominating Sets for Efficient Routing in Ad Hoc Wireless Networks. In *DialM*, pages 7–14, 1999.
- [38] Q. Zhang and D. P. Agrawal. Dynamic probabilistic broadcasting in MANETs. *J. of Parallel Distr. Computing*, 65(2):220–233, 2005.