On Reliable 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. This paper studies three common approaches for achieving scalable reliable broadcast in ad-hoc networks, namely probabilistic flooding, counter based broadcast, and lazy gossip. The strength and weaknesses of each scheme are analyzed, and a new protocol that combines these three techniques, called RAPID, is developed.

Specifically, the analysis in this paper focuses on the tradeoffs between reliability (percentage of nodes that receive each message), latency, and the message overhead of the protocol. Each of these methods excel in some of these parameters, but no single method wins in all of them. This motivates the need for a combined protocol that benefits from all of these methods and allows to trade between them smoothly. Interestingly, since the RAPID protocol only relies on local computations and probability, it is highly resilient to mobility and failures and even selfish behavior. By adding authentication, it can even be made malicious tolerant.

Additionally, the paper includes a detailed performance evaluation by simulation. The simulations confirm that RAPID obtains higher reliability with low latency and good communication overhead compared with each of the individual methods.

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 [39]. 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) [17, 18].

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 be both efficient and 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 [38]. 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 [40].

A common alternative to flooding is to perform a constrained flooding on top of a deterministic overlay, e.g., [24, 37, 45, 46]. 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 and expensive task. Finally, it is hard to make overlays resilient to malicious or even selfish behavior.\footnote{By selfish we mean nodes that only send their own messages.}

Hence, in this work we are interested in non-overlay based methods for reliable dissemination. Loosely speaking, the three most common techniques for obtaining this in ad-hoc networks are probabilistic flooding, e.g., [13, 25], in which the decision of a node to rebroadcast a message depends on some locally computable probabilistic mechanism, counter based approaches (and its derivatives such as distance-based and location-based forwarding), e.g., [6, 13, 40, 41], in which rebroadcasting a message depends on the number of retransmissions the node hears in its neighborhood, and lazy gossip [19, 21], in which nodes periodically gossip with their neighbors about the ids of messages they have received and request missing messages from them.

Contributions of this Work We start this work with an analysis of the three most common techniques for performing reliable dissemination in ad-hoc networks that do not rely on overlays or geographical knowledge. As mentioned above, these techniques include probabilistic flooding, counter-based forwarding, and lazy gossip.

In the case of probabilistic flooding, we show the following theoretical results: First, in order to obtain the best results in terms of reliability vs. efficiency, the forwarding probability should be set proportionally to the number of nodes in each one-hop neighborhood. This way, a probabilistic protocol can self adapt to the network’s density and avoid sending many redundant messages in dense networks. This is unlike many leading works that used constant probability, e.g. [13]. Second, it is possible to obtain an average reliability level of between 80%–90% with relatively low forwarding probability, or in other words, in a very efficient
manner. However, in order to increase the average reliability beyond that point, one has to increase the forwarding probability dramatically, to the point that the protocol becomes quite wasteful. Consequently, probabilistic forwarding should be complemented with some deterministic corrective measures in order to obtain both efficiency and very high reliability.

Counter based dissemination, on the other hand, is very hard to analyze formally. Yet, practically, with empirical fine tuning, it can obtain delivery ratios of about 95% while generating only a small amount of overhead. Still, as we discuss later in this paper, this scheme inherently suffers from long latencies and neither this scheme nor probabilistic flooding alone can obtain near 100% reliability. Therefore, as an additional measure to guarantee extremely high reliability, lazy gossip can be used. However, using only lazy gossip is inherently slow.

Armed with this insight, we have developed the RAPID protocol, which combines the benefits of these three approaches and our formal analysis about probabilistic flooding. That is, RAPID includes three phases: probabilistic forwarding, a counter based corrective measure, and lazy gossip. In the probabilistic forwarding phase, the retransmission probability of each node should be inversely proportional to the number of neighbors it observes at a given moment. Also, this probability is chosen to match the best tradeoff between the number of retransmissions and the reliability, according to our formal analysis. The counter based phase can be viewed as a corrective measure that helps in boosting the reliability to much higher numbers. Finally, the lazy gossip kicks in whenever even the counter based corrective measure fails to deliver some messages. The majority of the messages is received by probabilistic forwarding and only a small portion is received due to the counter based phase or through lazy gossip. Thus, the high latency of the last two measures is only seldom paid.

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, failures, and selfishness (and even some forms of malicious behavior), due to its probabilistic nature, the reliance on local information only, and the gossip mechanism. 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.

Paper’s road-map The model and basic definitions and assumptions are described in Section 2. The theoretical results of this work and description of the three dissemination techniques are presented in Section 3. Section 4 describes the RAPID 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 following as the transmission disk; the radius of the transmission disk is called the transmission range.$^2$

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

$^2$ 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.
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 $z$ that is a direct neighbor of both, then $z$ 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, we also consider a case in which some nodes act selfishly, i.e., they refuse to forward messages of other nodes. Such nodes are called selfish whereas the others are called correct. We assume that the correct nodes in the system continuously form a connected sub-network. More severe malicious behavior is discussed in the Appendix.

### 3 Common Reliable Dissemination Techniques

In this section we present the various techniques used for dissemination in wireless ad hoc networks and discuss their properties.

#### 3.1 Probabilistic Flooding

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 [6, 13, 25]. Probabilistic protocols are appealing since they are very simple and are inherently robust to failures and mobility. Moreover, these protocols enable messages to advance asynchronously, and therefore they exhibit very low latency in delivering messages. Yet, as was empirically discovered in [13, 25, 33], in order to obtain very high reliability levels with pure probabilistic broadcasting, one has to set the retransmission probability to high values. This in turn translates into a very large number of redundant messages.

Below, we obtain the following results about probabilistic flooding: We provide a model for analyzing the tradeoff in probabilistic flooding between the number of randomly selected nodes that retransmit each message in a given one hop neighborhood and the expected reliability level. In other words, this analysis...
formally captures the tradeoff between efficiency and reliability offered by pure probabilistic flooding. This, for example, enables designers to decide on a forwarding probability based on their goals w.r.t. this tradeoff.

Second, our formal analysis shows that in order to achieve a given tradeoff point between reliability and efficiency, it is enough that a constant number of nodes in each one hop neighborhood will retransmit a message. Constant here means independent of the nodes density. In other words, the forwarding probability of each node should be set in reverse proportion to the size of its neighborhood. This probability can be expressed as \( \beta / n_i \), where \( n_i \) is the neighborhood size of node \( i \) and \( \beta \) is the required constant of forwarders. Further, the behavior of the reliability w.r.t. forwarding probability is concaved with a knee at values of \( \beta \) between 2.5 and 3.5 (Figure 2). Setting the forwarding probability to these values results in delivery to 80%-90% of the nodes very quickly and very efficiently. However, for boosting the reliability beyond these levels, it makes more sense to utilize some complementing measures.

Finally, we show that regardless of the forwarding probability, pure probabilistic protocols cannot ensure 100% reliability. This again hints that probabilistic flooding should be aided by another mechanism if one wishes to ensure extremely high levels of reliability. We now turn to the details of the analysis.

### 3.1.1 Formal Analysis of Probabilistic Flooding Probability

Our theoretical analysis in this section relies on a formal graph model of 2-dimensional wireless ad hoc networks. The network connectivity graph \( G = (V, E) \) of an ad hoc network is a special case of a \( d \)-dimensional Unit Disk graph, in which \( n \) nodes are embedded in the surface of a \( d \)-dimensional unit torus, and any two nodes within Euclidean distance \( r \) of each other are connected. When the nodes are placed uniformly at random on the surface the graph is known as a Random Geometric Graph (RGG) [30] and is denoted by \( G^d(n, r) \). Specifically, the \( G^2(n, r) \) graph is often used to model the network connectivity graph of 2-dimensional wireless ad hoc networks and sensor networks [12]. In our case we assume \( n \) nodes are placed uniformly at random in the rectangular area \([a, b] \) and the transmission radius \( r \) is scaled accordingly such as \( G^2(n, r) \) is connected with high probability. It has been shown (by Gupta and Kumar [12] and [29]) that for \( r = \pi r^2 \geq ab \log n + o(n) \), \( G^2(n, r) \) is asymptotically connected with probability one if and only if \( c(n) \to \infty \) as \( n \to \infty \). 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 the 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 RAPID 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 \( d_{\text{avg}} \) the average number of neighbors of any node in \( G^2(n, r) \). It is well known that \( d_{\text{avg}} \leq \pi r^2 (n-1) / ab \) and for large networks, when the edges effect is negligible, \( d_{\text{avg}} \sim \pi r^2 / ab \). We have previously shown in [2] that the maximal and minimal degrees are of the order of \( d_{\text{avg}} \) with high probability. That is, the actual degree of any node in \( G^2(n, r) \) is close to \( d_{\text{avg}} \) 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 \( P = \beta / d_{\text{avg}} \) from all network nodes, independently from all other nodes, where \( \beta \) is a parameter called the reliability factor of algorithm \( A \). Informally, \( \beta \) 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 \( Q \) to be successfully received by a neighboring node. Let \( Y_p \) be a random variable corresponding to the number of times that 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 \( e^{-\beta Q} \).
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 $nP$.

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 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 $Pr(X_{p,q} = 1) = \frac{\pi r^2}{4ab}$.

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,

$$Pr(Y_p = 0) = \sum_{i=0}^{n} Pr(Y_p = 0 | |S| = i) Pr(|S| = i) =$$

$$\sum_{i=0}^{n} \prod_{q \in S, |S| = i} Pr(X_{p,q} = 0) Pr(|S| = i) =$$

$$\sum_{i=0}^{n} \binom{n}{i} (1 - \frac{\pi r^2}{4ab})^i \left(\frac{P}{i}ight)^i (1 - P)^{n-i} =$$

$$\sum_{i=0}^{n} \binom{n}{i} (P - \frac{\pi r^2}{4ab})^i (1 - P)^{n-i} = (1 - \frac{\pi r^2}{4ab})^n$$

$$\leq (e^{-\frac{\pi r^2}{4ab}})^n = e^{-\frac{\pi r^2}{4ab}n} \leq e^{-\frac{\beta ab}{\pi r^2(n-1)}} \leq e^{-\beta Q}$$

In the forth 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 $\beta$ and $Q$.

![Figure 2: An upper bound on the probability that an arbitrary node does not receive a message $m$](image-url)
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 \( \beta \). For example, for \( Q = 0.8 \), \( \Pr(Y_p = 0) \) is less than 0.06 for \( \beta = 3.5 \). That is, if there are only \( \beta = 3.5 \) nodes in every one-hop neighborhood that transmit \( m \) and \( Q = 0.8 \), approximately 94% of all nodes will receive \( m \).

**Discussion:** A broadcasting algorithm that sets the retransmission probability \( P \) 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.

\[
E(|S|) = \frac{n\beta}{d_{avg}} = \frac{n\beta}{\pi r^2 \frac{(n-1)}{ab}} \leq \frac{ab\beta}{\pi r^2}.
\]

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 \( S \).

### 3.1.2 On the Impossibility of Absolute Reliability

Notice that no pure probabilistic protocol can ensure absolute dissemination reliability. Consider an example of a node \( q \) with neighbors \( n_1, \ldots, n_k \), each of which forwards each message they receive with probability \( P_{n_i} \). Hence, with probability \( \text{Prob} = \prod_{i=1,\ldots,k}(1 - P_{n_i}) \), no node will retransmit the message and therefore \( q \) will not receive the message. No matter how high the probabilities \( P_{n_i} \) are, \( \text{Prob} \) is non zero and can sometimes be non negligible. In particular, even if the average density across the whole network is high, if nodes are scattered in a somewhat random manner, there is a likelihood that some parts of the network will have low density. In those parts \( k \) can even be less than 2. Thus, the probability that there will be some node \( q \) that will not receive some messages is non-negligible in any pure probabilistic protocol.

### 3.2 Counter Based Broadcast

The shortcomings of probabilistic flooding has led to the development of the counter-based approach [6, 13, 40, 41] and its distance-based and location-based derivatives (and their combinations). The idea in these schemes is that rather than placing the randomness directly on the retransmission probability, the randomness is placed on the timing of the rebroadcasting. That is, every node \( p \) that receives a message \( m \) for the first time, decides to rebroadcast the message after some random time. If during this chosen period \( p \) hears \( k \) (the counter) retransmissions of \( m \), then \( p \) decides to abort its retransmission.

Interestingly, this is another way to ensure a constant number of retransmissions in each neighborhood. But, as opposed to the probabilistic method, the number of retransmissions is deterministically guaranteed by the protocol. Despite this, as we show in Section 3.2.2, even counter based approach cannot guarantee reliable delivery of all messages on an arbitrary topology. In fact, if we assume that the nodes are uniformly
distributed in the network, and that the random function used for setting the retransmission time is independent of the node’s location, then we can utilize our formal analysis from Section 3.1.1 to calculate the reliability level of a counter-based protocol for a given value of $k$.

Empirical studies have shown that counter-based schemes can obtain high delivery ratios with relative efficiency [6, 13, 40, 41]. Yet, these works do not include a formal analysis of this behavior. Moreover, as we now discuss, counter-based schemes are inherently slower than probabilistic schemes.

3.2.1 Latency

As mentioned before, the rebroadcasting time of each node is set randomly. However, in order for the protocol to succeed, the values should be set from a sufficiently large range so that the number of collisions will be small, or even zero [15, 16]. In other words, the range from which the rebroadcast timing is chosen must be proportional to the number of nodes in each neighborhood. For ensuring zero collisions, by using the birthday paradox, we can deduce that the range should be roughly $sl \times n_i^2$, where $sl$ is the minimal slot required for a message transmitted by one node to be heard by any other node in its neighborhood and $n_i$ is the size of the neighborhood of node $i$. For example, routing protocols in ad hoc networks usually apply a random delay uniformly distributed between 0 and 10 milliseconds [4].

On the other hand, with probabilistic flooding as we suggest, and assuming $\beta \leq 3.5$, at most $3.5$ nodes might retransmit simultaneously in each neighborhood. Hence, the jitter applied to probabilistic forwarding can be much shorter than for counter-based schemes.

3.2.2 On the Impossibility of Absolute Reliability

We claim that no counter-based scheme can guarantee reliable delivery of all messages on an arbitrary topology. Consider a scenario w.r.t Figure 1. When node $s$ broadcasts a message $m$, nodes $p$ and $n_1, ..., n_k$ receive it. If some of $n_i$ nodes rebroadcasts the message before node $p$, $p$ will refrain from rebroadcasting $m$ and therefore $q$ will not receive $m$. For any counter-based scheme and for any value of the counter in $p$, there could be as much $n_1$ nodes as needed, such that $n_1$ is a neighbor of $s$ and $p$, but not of $q$. Then, all $n_i$ nodes might rebroadcast $m$ before $p$, by this satisfying the counter in $p$ and preventing $p$ from rebroadcasting the message $m$.

3.3 Lazy Gossip

In lazy gossip [21], nodes periodically gossip with their neighbors about the ids of messages they have received. Yet, this gossiping is performed in a deterministic manner, in the sense that each node sends such a gossip message as a broadcast to all its neighbors. Whenever a node $q$ learns that one of its neighbors $p$ has a message that $q$ has missed, $q$ explicitly asks $p$ to retransmit this message. Here, there can be a few optimizations such as broadcasting requests for retransmissions, etc.

Lazy gossip incurs a constant per node message overhead due to the need to periodically gossip about messages. The overall network overhead grows with the network density. However, due to its deterministic nature, lazy gossip can obtain absolute reliability.

The shortcomings of lazy gossip mainly come from its very high latency and the fact that for reliability, it must gossip multiple times for each message. The latency stems from the fact that messages are propagated only due to gossip, and these only occur periodically. In order to keep the message overhead reasonable, gossip might be performed once every several seconds, in which case forwarding a message across multiple
hops can take dozens of seconds. Also, due to message loss, obtaining absolute reliability involves unlimited memory consumption and unbounded message sizes, at least in theory.

4 The RAPID Protocol

For didactic purposes, we develop our protocol in two steps. The basic version of our protocol appears in Figure 3 whereas an enhanced version of the protocol that sends even fewer messages and provides higher delivery ratio is depicted in Figure 4 (a malicious resilient version of our protocol appears in the Appendix). In all figures we make use of two primitives. The primitive probcast 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/hello message (unless it has already sent another message during a predefined time interval).

The rebroadcasting probability used by RAPID is set to \(\min(1, \frac{\beta}{|N(p)|})\). \(\beta\) is a parameter of the protocol and corresponds directly to the communication overhead. For bigger \(\beta\) higher reliability level is achieved, however with larger communication cost. As can be seen in Figure 2 (the knee in the graph), a good tradeoff between the number of retransmissions and the reliability level is achieved when \(\beta\) is set to around 3.5. We further explore the effect of parameter \(\beta\) on RAPID in the simulation section.

In parallel, every node \(p\) periodically broadcasts to its neighbors the headers of messages it 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.

When examining the graph in Figure 2, it can be seen that the reliability level obtained depends on the probability that a transmission will not be lost. Specifically, in wireless networks, most message losses are caused due to 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. That is, when a node decides to rebroadcast a message, it waits for a short random time before doing so. Hence, the small probability of rebroadcasting plus the short jitter before rebroadcasting means that RAPID very rarely causes message collisions. The value of jitter is discussed in Section 5.

4.1 Basic RAPID

4.1.1 The Dissemination Task in Details

This protocol is a combination of probabilistic flooding with lazy gossip. Hence, its message dissemination consists of the following steps: (1) The originator \(p\) of a message \(m\) sends \(m||\text{header}(m)\) to all nodes in \(N(p)\) (Lines A01–A04 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 \(\text{header}(m)\) 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 \(\beta\) equals to
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) lazymcast(gos_msg, GOSSIP);

Upon receive(msg, DATA or DATA_REPLY) sent by p_j do 
(A06) if (have not received this msg before) then
(A07) Accept(p_j, msg); /*forward to the application*/
(A08) prob_bcast(prob = min(1, β|N(p)|), msg, DATA);
(A09) lazymcast(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, β|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, β|N(p)|), msg, DATA_REPLY);
(A17) endif.

Figure 3: Basic RAPID (executed by node p)

3.5. (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 [11, 36]. 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 an important drawback: 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.

In order to deal with this drawback and improve the reliability and the latency of RAPID, we slightly change the protocol by adding a complementing counter-based like mechanism that is based on having each node monitor its neighbors. That is, 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 \( m \). This 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.

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.

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 \( \min(1, \frac{1}{|N(p)|}) \) to occur after some random jitter (Line B08 in Figure 4). 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

The main difference between gossiping in Basic RAPID vs. Enhanced RAPID is in the cancelling of `REQUEST` and `DATA_REPLY` messages. That is, in the enhanced protocol every node \( p \) monitors its neighbors and if \( p \) planned to broadcast such a message \( m \), but \( p \) heard a transmission of \( m \) by its neighbor node, then \( p \) cancels the transmission of \( m \). This cancelling is done in order to eliminate redundant `REQUEST` and `DATA_REPLY` messages due to the broadcast nature of wireless networks. 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 [6, 13, 40, 41], 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. Notice that 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 (\( \beta \)). On the other hand, in the situations in which `long_jitter` is used in our protocol, and similarly in counter based protocols,
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 or DATA_REPLY) sent by pj do
(B06) if (have not received this msg before) then
(B07) Accept(pj, msg); /* forward it to the application */
(B08) cast_queue.add(prob = min(1, |N(pj)|), time=random(0, short_jitter), msg, DATA);
(B09) lazycast(gos_msg, GOSSIP);
(B10) endif;

Upon receive(gos_msg, GOSSIP) sent by pj 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, |N(pj)|), time=random(0, short_jitter), gos_msg, REQUEST);
(B14) endif;

Upon receive(gos_msg, REQUEST) sent by pj do
(B15) if (I have the msg that matches gos_msg) then
(B16) cast_queue.add(prob = min(1, |N(pj)|), time=random(0, short_jitter), msg, DATA_REPLY);
(B17) endif;

Interceptor
(B18) if (msg that appears in cast_queue was received) and (msg.type==REQUEST or msg.type==DATA_REPLY) 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)

all nodes in the neighborhood might transmit concurrently. Hence, long_jitter must be long enough to accommodate for that. Consequently, short_jitter can be much shorter than long_jitter. For example, if the target is 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. Moreover, most times in RAPID the timer-based corrective measure will not be used, so average latency is mostly dominated by short_jitter. The actual values used for both jitters are described in Section 5.
5 Simulations

In this section, we evaluate the performance of RAPID and compare it with the performance of the counter-based protocol of Tseng et al. [40] and with the performance of the GOSSIP3 protocol [13]. In GOSSIP3, when a node \( q \) receives a message, it broadcasts the message to its neighbors with probability \( P \) and with probability \( 1 - P \) 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\(^3\). 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 [13]. 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.

5.1 Setup

We have used the JiST/SWANS simulator [42] 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, denoted by \( \text{short jitter} \), which was set to 3 milliseconds. In addition, in the TSENG protocol and in the counter based mechanism in RAPID we have used a \( \text{long jitter} \) of \( 0.33 \times s^2 \) millisecond (where \( s \) is the expected number of concurrent senders).

The transmission range was set to roughly 200 meters\(^4\). 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 5.2.4.

Mobility was modelled by the Random-Waypoint model [20]. 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 [5], 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 enhanced version of our probabilistic dissemination protocol from Figure 4 is denoted RAPID; a restricted version of the enhanced RAPID in which the

\( ^3 \) GOSSIP3 was simulated with probability \( P = 0.65 \) and \( M=1 \).

\( ^4 \) In SWANS one can choose the transmission power which translates into a transmission range based on power degradation and background noise.
gossip and the recovery mechanism were disabled is denoted RAPID-NO-GOSSIP; the counter-based protocol of Tseng et al. [40] is denoted TSENG; GOSSIP3 is the probabilistic protocol by Haas et al. [13]. 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 Broadcasting Probability - exploring $\beta$

Figures 5, 6, 7 and 8 explore the delivery ratio, the number of transmissions and the latency (in seconds) against the broadcast probability of nodes in RAPID. Since the broadcasting probability of node $i$ is expressed as $\beta/n_i$, where $n_i$ is the neighborhood size of $i$, increasing the value of $\beta$ leads to an increase in the broadcasting probability of $i$. The following discussion and simulations analyze the influence of $\beta$ values on the latency, the delivery ratio and the number of transmissions of RAPID.

We can see in Figures 7 and 8 that when we increase the $\beta$ value, the latency of RAPID decreases. It can be explained by the fact that more nodes decide to broadcast the received message and therefore more nodes receive messages from their neighbors by the probabilistic mechanism and not due to the completion or recovery mechanisms. Yet, when the value of $\beta$ increases, more messages are injected into the network, as can be seen in Figure 6. In addition, the value of $\beta$ has hardly any influence on the reliability of RAPID and even for $\beta = 1.5$, RAPID delivers all messages to more than 99% of nodes. In this case more messages are delivered via the completion and the recovery phases, which increases the latency, but still keeps the reliability of RAPID as high as 99%, as can be seen in Figure 5. Hence, the decision of whether to use RAPID with low or high value of $\beta$ can be made based on the tradeoff between the latency and the message load for a given application. In the following sections, we present RAPID with $\beta = 3.5$ since it gives a good tradeoff between throughput and latency.

5.2.2 Changing the Number of Broadcasting Nodes

Figures 9 and 10 present a comparison of RAPID with other protocols in mobile networks. Figure 9 shows the percentage of nodes that received all messages vs. the number of nodes that initiate one new broadcast per second. RAPID delivers a very high percentage of messages (99.9%), even when the number of broadcasting nodes is as high as 200. RAPID-NO-GOSSIP, GOSSIP3 and TSENG also deliver high percentage of 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 RAPID-NO-GOSSIP, GOSSIP3 and TSENG deliver to all the nodes decreases substantially. 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 RAPID-NO-GOSSIP, GOSSIP3 and TSENG only employ a probabilistic dissemination mechanism, they cannot recover these lost messages.

Interestingly, the gap between the reliability of RAPID-NO-GOSSIP and TSENG and the reliability of GOSSIP3 grows as the number of broadcasting nodes is increased. This is because RAPID-NO-GOSSIP and TSENG generates significantly fewer messages than GOSSIP3 and therefore there are fewer collisions. 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 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 10.
We can also observe in Figure 10 that RAPID sends more messages than RAPID-NO-GOSSIP and TSENG, in order to overcome the collisions and message loss. Hence, the decision of whether to use RAPID or RAPID-NO-GOSSIP (or TSENG) can be made based on the tradeoff between reliability and load for a given application.

Figure 11 explores the latency to deliver messages to a varying percentage of the nodes when the number of broadcasting nodes is 100. As can be seen by the graphs, GOSSIP3 is significantly faster than all other protocols. Yet, GOSSIP3 delivers messages only to 95% of the nodes, while RAPID delivers the messages to 99.6% of the nodes within 0.15 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. As expected, RAPID is much faster than TSENG due to the fact that the timeout between broadcasts of nodes in RAPID is smaller than the timeout of broadcast in TSENG.
as it was explained in 4.2.3 and the recovery of missing messages in RAPID is faster than the completion protocol in TSENG. Finally, RAPID (with gossip) is faster than RAPID-NO-GOSSIP due to the recovery protocol that it runs in parallel to probabilistic dissemination.

5.2.3 Impact of Mobility

Figures 13 and 14 explore 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 13 and 14, when nodes are mobile, the performance of RAPID (in terms of 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.

5.2.4 Network Density

Figures 15 and 16 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 $2500 \times 2500$ m$^2$ (the average density is about 4 nodes per neighborhood), RAPID with 100 broadcasting nodes delivers all messages to 52.4% of the nodes, while GOSSIP3 delivers all messages to 38.04% of the nodes and TSENG delivers all messages to
42.5% 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 100 broadcasting nodes delivers all messages to 94.9% of the nodes, Tseng delivers all messages to 95.4% of the nodes, and the delivery ratio of RAPID is above 98%.

Interestingly, this echoes the results of [32]. Moreover, we know from Gupta and Kumar’s connectivity bound for ad hoc networks [12] 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, \( C \) 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 looking at the total number of transmissions in Figure 16, 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 gossip 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 3.1.1. RAPID achieves a slightly better delivery ratio than RAPID-NO-GOSSIP. Yet, if the number of messages is more important, we may use RAPID-NO-GOSSIP that sends even less messages than RAPID (this is since RAPID-NO-GOSSIP tunes its rebroadcast probability according to the number of observed neighbors just like RAPID, yet it does not gossip about the existing messages).

5.2.5 Selfish Nodes

Figure 12 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. Surprisingly, the latency does not grow with the number of selfish nodes. This is since on one hand selfish nodes do not rebroadcast other’s messages, but on the other hand they do not send gossip messages and therefore cause fewer collisions. We can also see that even when the number of selfish nodes is 200 (20% of all nodes), RAPID delivers the messages to 98.99% of the nodes within 0.14 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 even further.

Table 1 presents the delivery ratio and the message overhead in mobile networks for varying numbers of selfish nodes. We can see that the number of selfish nodes hardly 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 gossip 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 [39, 44]. Here, we only discuss the most relevant protocols to our work.
The simplest probabilistic broadcast protocol is probabilistic flooding [13, 40]. In this scheme, each node rebroadcasts a message with a fixed probability $P$. Works by Haas et al. [13] and Sasson et al. [33] study the rebroadcasting probability $P$ 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 $\overline{P}$, in a sense that for any $P > \overline{P}$ almost all nodes will receive the message and for $P < \overline{P}$ almost none. Both works show that the threshold probability $\overline{P}$ is around $0.59 - 0.65$; in [33] this is done analytically based on percolation theory while in [13] it is obtained by simulations. It is also noted in [13] 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 a few probabilistic assumptions, the delivery distribution function behaves in a concave manner rather than being bimodal. That is, nodes coverage initially grows fast with $P$. Then, at some critical point, the added coverage becomes negligible with further increase of $P$. Our protocol is designed with corrective measures that compensate for situations in which the simplifying assumptions do not hold. A generic epidemic model for information diffusion in MANETs has appeared in [23].

Other probabilistic approaches [13, 28, 40, 41] 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, [13] 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 guarantees reliable dissemination of messages to all nodes (as explained in Section 4.2). The works in [27, 35, 47] 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 $P$ should be adjusted relatively to the local nodes density. In [47] this is done through counters, while in [35] the uniform density is assumed. In [27] a local nodes density is compared to a network-wide average nodes’ density (which is is assumed to be known) and the retransmission probability is set to a higher/lower value if the number of neighbors of a retransmitting node is less/more than the network average number of neighbors. However, those works contain little theoretical analysis of the proposed schemes and like other counter-based or probabilistic 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 $P$.

The work in [6] studies three variants of the above ideas. The first is to retransmit with probability $k/n_i$, where $k$ is some constant and $n_i$ is the size of the neighborhood. The second method is based on having each node learn its 2-hop neighborhood and then computing the rebroadcasting probability based on 1-hop neighborhoods intersections. The final scheme in [6] also computes the probability according to $k/n$, but adds a mechanism in which if a node suspects that some of its neighbors did not receive the message, it rebroadcast the message regardless of its initial decision. Unlike the work in [6], we formally analyze the

<table>
<thead>
<tr>
<th># Selfish</th>
<th>Delivery ratio</th>
<th>Message overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>99.61%</td>
<td>4200144</td>
</tr>
<tr>
<td>50</td>
<td>99.57%</td>
<td>4071811.7</td>
</tr>
<tr>
<td>100</td>
<td>99.55%</td>
<td>4006395.7</td>
</tr>
<tr>
<td>200</td>
<td>98.99%</td>
<td>3816120.6</td>
</tr>
</tbody>
</table>

Table 1: Delivery ratio and message count vs. the number of selfish nodes (with 100 broadcasting nodes)
value of $k$. Also, we include a gossip and recovery mechanism, whereas none of the protocols in [6] do so. Consequently, RAPID is more reliable than any of the schemes of [6]. Moreover, RAPID has a variant that can deal with many forms of malicious behavior while the other protocols do not.

The color-based scheme has been recently proposed in [22]. 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 [22] are similar to our analysis in Section 3.1.1. 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 [31]. In contrast, our approach to recovery of messages is based on gossip. Also, Mistral cannot cope with malicious or selfish 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 [14]. 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., [9]. 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 [7]. A generic framework for presenting gossip protocols was proposed in [19], 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, 10]. An example of a protocol for ad-hoc networks that uses a pull-push approach and is easily expressed in the above framework is [26]. Our protocol can also be seen as a specific instantiation of pull-push dissemination.

An additional protocol for reliable broadcast and manycast in ad hoc networks called Scribble has been proposed in [43]. In Scribble, the responsibility for dissemination initially rests with the manycast 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 studied the three most common techniques for obtaining reliable broadcast in ad-hoc networks, which do not rely on an overlay. By analyzing these techniques, we came to the conclusion that a protocol that combines the benefits of each technique is needed. We have described such a protocol called RAPID. The protocol includes a probabilistic flooding phase that is complemented by two corrective measures, namely, counter based forwarding and a deterministic gossip based mechanism. The latter enable
recovering messages that were not delivered by the probabilistic dissemination process while maintaining low communication overhead.

The probabilistic flooding part of the protocol takes advantage of the locally observed network’s density in order to send a small number of messages, yet one that is still sufficient to deliver the message to most nodes. This is in accordance with our formal analysis. This provides very rapid dissemination of the message to most nodes in the system with low message overhead, and in a way that is scalable in the network density.

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

References


A 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.

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||msg_id|node_id);
(C03)  prob_bcast(prob = 1, data_msg, DATA);
(C04)  lazy_cast(gos_msg, GOSSIP);

Upon receive(msg, DATA or DATA_REPLY) sent by pj do
(C05)  if (verify_signature(msg) = TRUE) then
(C06)  if (have not received this msg before) then
(C07)  Accept(pj, msg); /*forward it to the application*/
(C08)  cast_queue.add(prob = min(1, |trusted_neighbors|), time=random(0, short_jitter), msg, DATA);
(C09)  lazy_cast(gos_msg, GOSSIP);
(C10)  endif;
(C11)  else /*the message is not correct*/
(C12)  suspect(pj);
(C13)  endif;

Upon receive(gos_msg, GOSSIP) sent by pj: 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, pj);
(C17)  /*The node asks from the node that sent the gossip to send the real message*/
(C18)  send(gos_msg, REQUEST, pj);
(C19)  endif;
(C20)  else /*the message is not correct*/
(C21)  suspect(pj);
(C22)  endif;

Upon receive(gos_msg, REQUEST, pk) sent by pj do
(C23)  if (verify_signature(gos_msg) = TRUE) then
(C24)  if (I am pk and I have the msg that matches gos_msg) then
(C25)  prob_bcast(prob = 1, msg, DATA_REPLY);
(C26)  endif;
(C27)  else /*the message is not correct*/
(C28)  suspect(pj);
(C29)  endif;

Figure 17: Maliciousness Resilient RAPID (lines that were modified w.r.t Figure 4 are boxed)
A.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 [34]. 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 $m$, $q$ 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 17. Here we introduce four new primitives: send, verify_signature, suspect and expect, and the retransmission probability 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 $\text{sig}(m)$ matches $m$. If it does not then $m$ is ignored and the node 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.

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 17 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 17 does not include this optimization for simplicity.

### A.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 behavior\(^5\)), (3) sending gossip messages without ever supplying the real 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

---

\(^5\) Giving 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.
probability. Hence, by assumption, the fourth attack is not possible. The rational 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.