On Reliable Dissemination in Wireless Ad-Hoc Networks

Roy Friedman Vadim Drabkin
Computer Science Department
Technion - Israel Institute of Technology
Haifa 32000, Israel
{dvadim,roy}@cs.technion.ac.il
Gabriel Kliot*
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
Gabriel.Kliot@microsoft.com
Marc Segal
Computer Science Department
Technion - Israel Institute of Technology
Haifa 32000, Israel
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 trade-offs 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.

I. 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) [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., [25], [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 [1] [8].

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., [14], [26], in which the decision of a node to rebroadcast depends on some locally computable probabilistic mechanism, *counter based* approaches (and its derivatives

*Work done while the author was a PhD student at the Technion.

[1]By selfish we mean nodes that only send their own messages.

such as distance-based and location-based forwarding), e.g., [6], [14], [40], [41], in which rebroadcasting a message depends on the number of retransmissions the node hears in its neighborhood, and *lazy gossip* [19], [22], in which nodes periodically gossip with their neighbors about the ids of messages they have received and request missing messages from them.[2]

Previous analysis of probabilistic flooding [14], [33] has taught us that in order to obtain reasonable reliability level in a fixed probability protocol, one has to set the forwarding probability to very high values. The latter means that for very high reliability, this scheme becomes almost as wasteful as flooding. Moreover, as we discuss in this paper, even with very high retransmission probability (yet, strictly smaller than 1), no node failures, and no message loss, probabilistic flooding cannot ensure absolute reliability.

Counter based schemes and their derivatives [6], [14], [40], [41] can obtain high reliability level while generating much fewer messages than probabilistic schemes. Yet, as we show in this paper, the counter based approach inherently imposes longer latencies than probabilistic flooding and flooding. Additionally, we show that counter based schemes cannot ensure 100% reliability either, even when there are no node failures and no message is ever lost.

Finally, when pure pull based gossip is used alone, it requires very large buffer spaces to ensure good reliability. Also, it either imposes extremely long delivery latencies, or needs to be activated very frequently, thereby generating much traffic.

Hence, we come to the conclusion that in order to obtain a solution that can ensure reliable delivery at a reasonable cost and with low latency, the three method should be combined. The main question this paper addresses is how to obtain this? In other words, when and how to orchestrate each of these three schemes in order to obtain an effective protocol that can deliver messages reliably, economically, and fast. We present such a protocol, called *RAPID*, and discuss the rational behind it.

Specifically, RAPID has a dynamically adaptable probabilistic facet in which a node p retransmits a message it receives for the first time with probability $\beta/|N(p)|$, where $N(p)$ are the set of neighbors of p and β is a parameter. The purpose of β is to control how many retransmissions of the same message will appear on average in each neighborhood. We perform a formal analysis of β. This analysis allows us to determine the value of β that yields the most efficient use of probabilistic flooding in terms of the tradeoff between the communication cost of the probabilistic phase and the reliability level attained through it.

Boosting the reliability beyond what is achieved through the probabilistic phase is obtained through a counter based facet coupled with a lazy pull based gossip mechanism. The combination of these three methods in RAPID creates a fast and highly reliable, yet economical, dissemination protocol. Notice, again, that the reliability of the combined protocol is ensured by the use of the pull based gossip and counter based techniques, while the probabilistic forwarding mechanism mainly serves to reduce the latency, by delivering the messages fast to the majority of nodes.

The paper also includes a detailed performance study, performed by simulations. It indicates that RAPID 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 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.

[2]Notice that probabilistic flooding protocols, such as GOSSIP3 [14], are often referred to as *(active) push based gossip*.

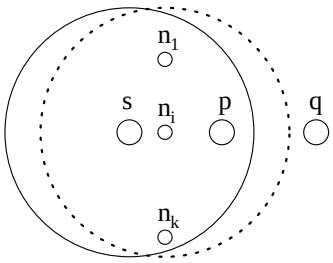

Fig. 1. A transmission by a node s can be received by all nodes within its transmission range: p, n_1, ...,n_k

Paper's road-map: The model and basic definitions and assumptions are described in Section II. The theoretical results of this work and description of the three dissemination techniques are presented in Section III. Section IV describes the RAPID protocol. The results of the performance evaluation are given in Section V. Section VI compares our work with related work, and we conclude with a discussion in Section VII.

II. 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*.[3] The combination of the nodes and the transitive closure of their transmission disks forms a wireless ad-hoc network. Nodes can physically move across the network; new nodes may join and existing nodes may leave the network at any time, either gracefully or by suffering a crash failure. Nodes that crash or leave the network may rejoin it later.

We denote the transmission range of device p by r_p. This means that a node q can only receive messages sent by p if the distance between p and q is smaller than r_p. A node q is a *direct neighbor* of another node p if q is located within the transmission disk of p as illustrated in Figure 1. In the following, $N(p)$ refers to the set of direct neighbors of a node p. Additionally, messages can be lost. For example, if two nodes p and q transmit a message at the same time, and there exists a node 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

[3]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.

form a connected sub-network. More severe *malicious* behavior is discussed in the Appendix.

III. Common Reliable Dissemination Techniques

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

A. 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], [14], [26]. 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 [14], [26], [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: We provide a model for analyzing an upper bound on the tradeoff in probabilistic flooding between the retransmission probability and reliability. In other words, this analysis formally captures the tradeoff between efficiency and reliability offered by pure probabilistic flooding. This 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. This means that the forwarding probability of each node should be set in reverse proportion to the size of its neighborhood. This probability can be expressed as β/n_i, where n_i is the neighborhood size of node i and β 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 β 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.

1) Formal Analysis of Probabilistic Flooding Probability: The theoretical analysis in this section relies on a formal graph model of wireless ad hoc networks. The network connectivity graph $G = (V, E)$ of an ad hoc network is a special case of a 2-dimensional *Unit Disk* graph, in which n nodes are embedded in the surface of a 2-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 $G^2(n, r)$. 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 [13]. In our case we assume n nodes are placed uniformly at random in the rectangular area $[a, b]$ and form a connected graph.

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 IV will ensure reliable delivery in any case, alas possibly with higher communication cost.

In this analysis we aim to estimate the reliability that can be provided by probabilistic forwarding alone. Specifically, we ask the following question: Suppose that each node in the system is given an independent opportunity to broadcast the same message m with probability $\min(1, \frac{\beta}{n_q})$. How many nodes will receive the message m? Formally, 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, i.e., $\Pr(Y_p = 0)$. In the analysis, we assume that a message that was sent has a probability $\mathcal{Q}$ to be successfully received by a neighboring node.

Notice that here we analyze a situation in which every node receives a message m at least once and attempts to probabilistically rebroadcast it exactly once. Hence, another way of looking at the analysis is as follows: Suppose we have a mixed protocol that employs multiple techniques to ensure reliable delivery of messages, one of which is probabilistic flooding. In this case, what percentage of the nodes will receive each message due to probabilistic flooding for a given forwarding probability.

Lemma 3.1: Denote by $\mathcal{P}_{send}$ the probability that a random node rebroadcasts a message m. Then, for $n \geq 50$ and $c_1 = 1 - \frac{\beta}{29}$

$$\mathcal{P}_{send} \geq c_1 \beta \frac{ab}{\pi r^2 (n-1)}$$

Proof Idea. We calculate a lower bound on $\mathcal{P}_{send}$ by conditioning on the number of neighbors of node q (denoted n_q) and by applying a probabilistic version of Jensen's inequality. The full proof is deferred to the appendix.

Claim 3.2: For any node p, the probability that p does not receive a message m is upper bounded by $e^{-c_1 \beta \mathcal{Q}}$, for $c_1 = 1 - \frac{\beta}{29}$.

Proof: For every two nodes p and q, let $X_{p,q}$ be a 0-1 random variable indicating whether the node p receives a message m from node the q or not. Node p can receive a message m from q if and only if q has probabilistically decided to broadcast m, q is a neighbor of p in $G^2(n, r)$ and m has not collided with other messages. The first event happens with probability $\mathcal{P}_{send}$, the second event with probability $\mathcal{R} = \frac{\pi r^2}{ab}$, and the third event with probability $\mathcal{Q}$.

Let Y_p be the random variable indicating the number of times node p has received m.

$$\Pr(Y_p = 0) = \Pr(X_{p,1} = 0 \cap X_{p,2} = 0 \cap \ldots \cap X_{p,n} = 0) =$$

$$\prod_{q \in N, q \neq p} \Pr(X_{p,q} = 0) = \Pr(X_{p,q} = 0)^{n-1} = (1 - \mathcal{Q}\mathcal{R}\mathcal{P}_{send})^{n-1} =$$

$$(1 - \mathcal{Q}\mathcal{R}\mathcal{P}_{send})^{n-1} \leq \left(1 - \mathcal{Q}\mathcal{R}\frac{c_1\beta}{\mathcal{R}(n-1)}\right)^{n-1}$$

$$= \left(1 - \mathcal{Q}\frac{c_1\beta}{(n-1)}\right)^{n-1} \leq e^{-c_1\beta\mathcal{Q}}$$

In the last line we have used the inequality $(1 - \frac{x}{n})^n \leq e^{-x}$.

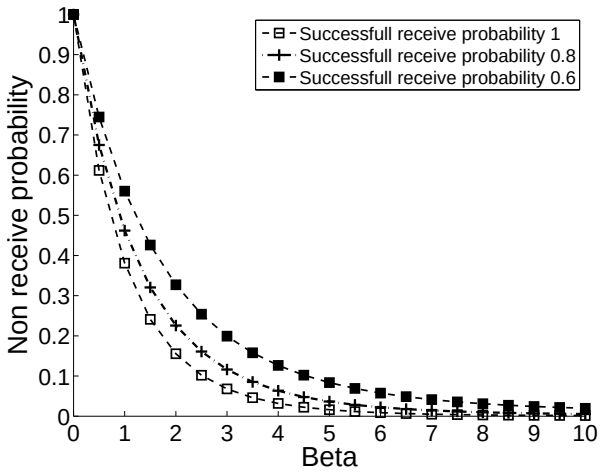

Fig. 2. An upper bound on the probability that an arbitrary node does not receive a message m

■

Corollary 3.3: $c_1 = 1 - \frac{\beta}{29} \geq 0.8$ for $\beta \leq 6$.

Figure 2 depicts an upper bound $e^{-(1-\frac{\beta}{29})\beta\mathcal{Q}}$ on the value of $\Pr(Y_p = 0)$ for an arbitrary node p as a function of β and $\mathcal{Q}$. It can be seen from the figure that the probability that a given node does not receive a message m is small for quite small values of β. For example, for $\mathcal{Q} = 0.8$, $\Pr(Y_p = 0)$ is less than 0.09 for $\beta = 3.5$. That is, if there are only $\beta = 3.5$ nodes in every one-hop neighborhood that transmit m and $\mathcal{Q} = 0.8$, approximately 91% of all nodes will receive m.

Discussion: A broadcasting algorithm that sets the retransmission probability $\mathcal{P}_{send}$ inversely proportional to a node's degree has a number of advantages. First, the number of transmissions is constant with respect to the number of nodes n and to the nodes' density. Formally, define by S_m the set of nodes that retransmit a given message m. We have:

$$E(|S|) = n\mathcal{P}_{send} \approx n(1 - \frac{\beta}{29})\beta\frac{ab}{\pi r^2(n-1)} \approx \frac{ab\beta}{\pi r^2}.$$

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 nodes' 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 retransmission set.

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 $Prob = \Pi_{i=1,\ldots,k}(1 - P_{n_i})$, no node will retransmit the message and therefore q will not receive the message. No matter how high the probabilities P_{n_i} are (as long as they are strictly smaller than 1), $Prob$ 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.

B. Counter Based Broadcast

The shortcomings of probabilistic flooding has led to the development of the counter-based approach [6], [14], [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 III-B.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 III-A.1 to calculate the reliability level of a counter-based protocol for a given k.

Empirical studies have shown that counter-based schemes can obtain high delivery ratios with relative efficiency [6], [14], [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.

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 [16], [17]. 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 network 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.

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 many n_i nodes as needed, such that n_i 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 m.

C. Lazy Gossip

In lazy gossip [19], [22], 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 than 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 comes 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 gossips, and these only occur periodically. In order to keep the message overhead reasonable, gossips 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.

IV. 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 `prob_bcast` denotes an immediate broadcast to all the direct neighbors of the sender with a given probability. The primitive `lazycast` initiates periodic broadcasting of the given message to the direct neighbors of the sender.

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

The rebroadcasting probability used by RAPID is set to $\min(1, \frac{\beta}{|N(k)|})$. β is a parameter of the protocol and corresponds directly to the communication overhead. For bigger β 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 β is set to around 3.5. We further explore the effect of parameter β on RAPID in the simulation section.

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

Upon `send`(msg) by application **do**
(A01) *header* $:= msg\_id||node\_id$;
(A02) *data_msg* $:= header||msg$;
(A03) *gos_msg* $:= header$;
(A04) `prob_bcast`($prob = 1$, *data_msg, DATA*);
(A05) `lazycast`(*gos_msg, GOSSIP*);

Upon `receive`(*msg, DATA 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`($prob = \min(1, \frac{\beta}{|N(p)|})$, *msg, DATA*);
(A09) `lazycast`(*gos_msg, GOSSIP*);
(A10) **endif**;

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

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

Fig. 3. Basic RAPID (executed by node p)

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 V.

A. Basic RAPID

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 $m||header(m)$ 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 $header(m)$ to all nodes in $N(p)$ (Line A05). (3) When a node p receives a message m for the first time, p accepts m (Lines A06–A07). (4) p broadcasts m with probability $\min(1, \frac{\beta}{|N(p)|})$ (Line A08 – our protocol was simulated with β equals to 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.

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 [12], [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.

B. 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.

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 IV-A.1) except for step 4 of Section IV-A.1. In Enhanced RAPID, whenever a node p receives a message m for the first time, it schedules a rebroadcast of m with probability $\min(1, \frac{\beta}{|N(p)|})$ to occur after some random jitter (Line B08 in Figure 4). If a received message has never been rebroadcasted,

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

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

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

Interceptor
(B18) **if** (*msg* that appears in `cast_queue` was received) **then**
(B19) **if** (*msg.type==REQUEST or msg.type==DATA_REPLY*) **then**
(B20) `cast_queue.remove`(msg);
(B21) **endif**;
(B22) **endif**;

Upon `Expiration of timer of msg in cast_queue` **do**
(B23) `cast_queue.remove`(*msg*);
(B24) $pr =$ the probability attached to *msg*;
(B25) *type* $=$ the message type associated with *msg*;
(B26) `prob_bcast`($prob = pr$, *msg, type*);
(B27) **if** (*msg* was not broadcasted) **then**
(B28) `cast_queue.add`($prob = 1$, time=long_jitter, *msg, type*);
(B29) **endif**;

Fig. 4. Enhanced RAPID (lines that were modified w.r.t Figure 3 are boxed while lines B18–B29 were added)

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 B27–B29).

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–B21, and B23–B29.

3) Latency of RAPID: In both RAPID and *counter-based* protocols [6], [14], [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 (β). On the other hand, in the situations in which *long_jitter* is used in our protocol, 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* 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 V.

V. 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 [14]. 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[4]. 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 [14]. 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.

A. 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 *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 *long_jitter* of $0.33 \times s^2$ millisecond (where s is the expected number of concurrent senders).

[4]GOSSIP3 was simulated with probability $\mathcal{P} = 0.65$ and M=1.

Signal Propagation model	Two-Ray ground reflection
MAC	802.11
Bandwidth	54 Mb/sec
Transmission range	200 meter
Simulation area	3500x3500 m^2 default. Also 2500x2500, 4500x4500
Node count	1000
Density (# of one hop neighbors)	10 default
Mobility	Random-Waypoint, speed 1-10 m/s, pause time 0s
# broadcasting nodes	1 to 200
# messages per broadcasting nodes	10
Message size	512 bytes

Fig. 5. Simulation setup

The transmission range was set to roughly 200 meters[5]. 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 V-B.4. The simulation setup is summarized in Figure 5

Mobility was modelled by the Random-Waypoint model [21]. 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 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. [14]. 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.

B. Results

1) Broadcasting Probability - exploring β: Figures 6, 7, 8, and 9 explore the delivery ratio, the number of transmissions

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

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

We can see in Figures 8 and 9 that when we increase β, the latency of RAPID decreases. This is since 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 β increases, more messages are injected into the network, as can be seen in Figure 7. In addition, the value of β 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 6. Hence, the decision of whether to use RAPID with low or high value of β 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.

2) Changing the Number of Broadcasting Nodes: Figures 10 and 11 present a comparison of RAPID with other protocols in mobile networks. Figure 10 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 11.

We can also observe in Figure 11 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 12 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, 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 IV-B.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.

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.

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 2500x2500 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 [13] 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 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

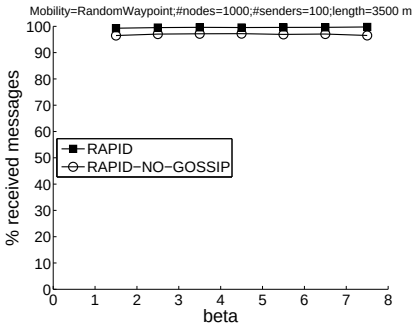

Fig. 6. Message delivery ratio when all nodes are mobile (comparing RAPID with different values of β)

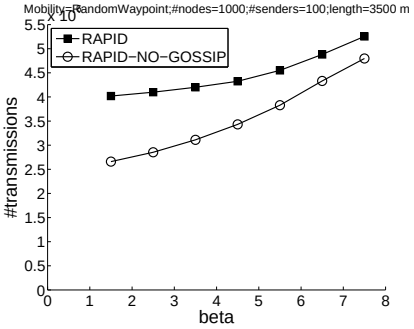

Fig. 7. Network load in terms of total number of transmissions when all nodes are mobile (comparing RAPID with different values of β)

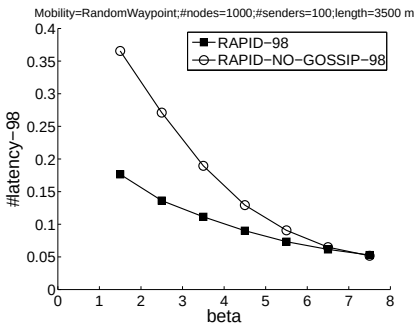

Fig. 8. Latency to deliver a message to 98% of the nodes when all nodes are mobile with varying values of β (with 100 broadcasting nodes)

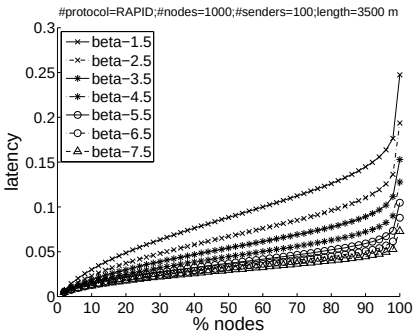

Fig. 9. Latency to deliver a message to X% of nodes when all nodes are mobile with varying values of β (with 100 broadcasting nodes)

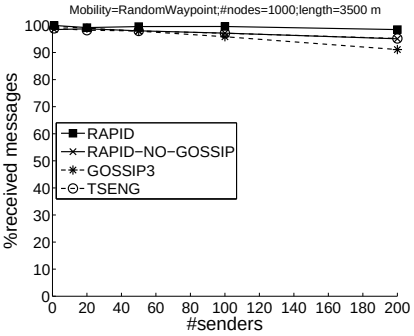

Fig. 10. Message delivery ratio when all nodes are mobile

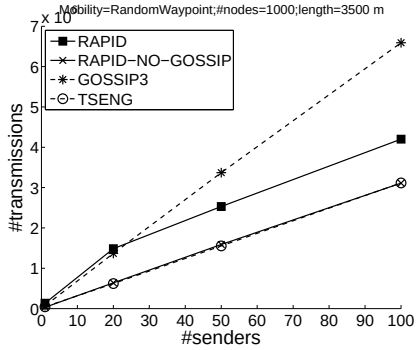

Fig. 11. Network load in terms of total number of transmissions when all nodes are mobile

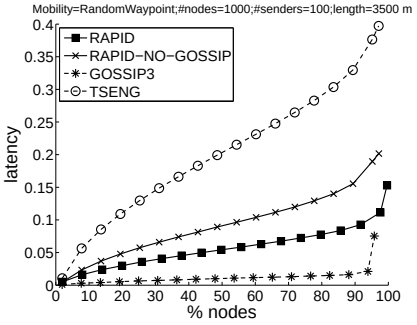

Fig. 12. Latency to deliver a message to X% of the nodes (with 100 broadcasting nodes)

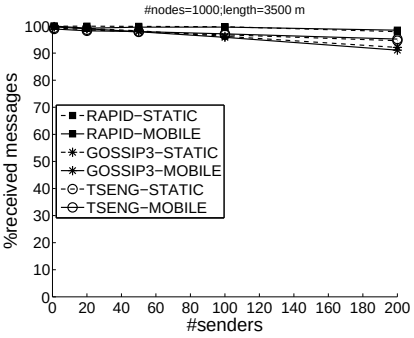

Fig. 13. Message delivery ratio (compare protocols both in static and mobile environments)

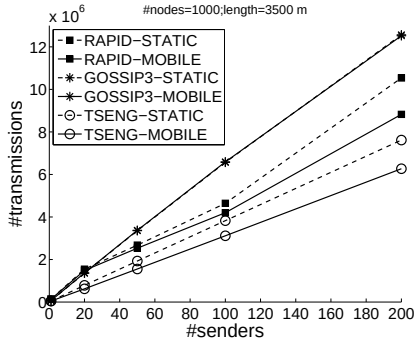

Fig. 14. Network load in terms of total number of transmissions (compare protocols both in static and mobile environments)

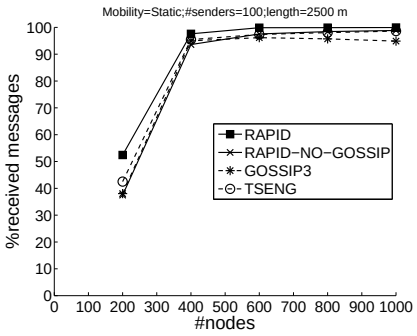

Fig. 15. Message delivery ratio with varying density while all nodes are static

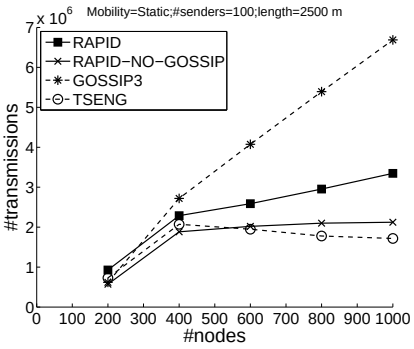

Fig. 16. Message overhead with varying density while all nodes are static

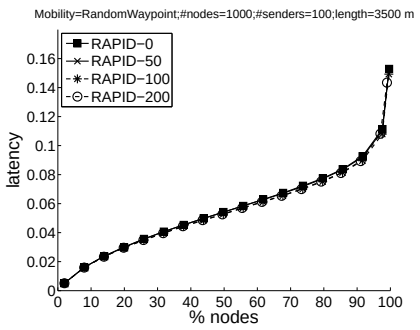

Fig. 17. Latency to deliver a message to X% of the modes when all nodes are mobile with varying number of selfish nodes (with 100 broadcasting nodes)

# Selfish	Delivery ratio	Message overhead
0	99.61%	4200144
50	99.57%	4071811.7
100	99.55%	4006395.7
200	98.99%	3816120.6

TABLE I
DELIVERY RATIO AND MESSAGE COUNT VS. THE NUMBER OF SELFISH NODES (WITH 100 BROADCASTING NODES)

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 III-A.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 send gossip messages).

5) Selfish Nodes: Figure 17 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 I 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.

VI. 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 [14], [40]. In this scheme, each node rebroadcasts a message with a fixed probability $\mathcal{P}$. Works by Haas et al. [14] and Sasson et al. [33] study the rebroadcasting probability $\mathcal{P}$ with regard to the so called *phase transition phenomena*. Both works establish that the delivery distribution has a bimodal behavior with regard to some threshold probability $\overline{\mathcal{P}}$, in a sense that for any $\mathcal{P} > \overline{\mathcal{P}}$ almost all nodes will receive the message and for $\mathcal{P} < \overline{\mathcal{P}}$ almost none. Both works show that the threshold probability $\overline{\mathcal{P}}$ is around $0.59 - 0.65$; in [33] this is done analytically based on percolation theory while in [14] it is obtained by simulations. It is also noted in [14] 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 III. 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 $\mathcal{P}$. Then, at some critical point, the added coverage becomes negligible with further increase of $\mathcal{P}$. 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 [24].

Other probabilistic approaches [14], [29], [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, [14] 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 IV-B.3) and none of them guarantees reliable dissemination of messages to all nodes (as explained in Section IV-B).

The works in [28], [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 $\mathcal{P}$ 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 [28] 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 $\mathcal{P}$.

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 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 [23]. 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 [23] are similar to our analysis in Section III-A.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.

Some works, such as NAPS [11], use probabilistic schemes to define which nodes can "nap" and which ones should be awake. The probability is set s.t. node's sleeping time is proportional to its degree. Yet, the goal is to save energy while still maintaining connectivity in a sensor network, whereas our goal is to disseminate data to all nodes of an ad-hoc network efficiently.

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 for message recovery 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 [15]. 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 [27]. Our protocol can also be seen as a specific instantiation of pull-push dissemination.

Another 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 suffers from increased latency and is more message consuming.

VII. 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

[1] D. Allen. Hidden terminal problems in Wireless LAN's. In *IEEE 802.11 Working Group Papers*, 1993.

[2] Z. Bar-Yossef, R. Friedman, and G. Kliot. RaWMS - Random Walk based Lightweight Membership Service for Wireless Ad Hoc Networks. In *Proc. of the 7th ACM Intr. Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 238–249, 2006.

[3] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, , and Y. Minsky. Bimodal Multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, May 1999.

[4] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. of the 4th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 85–97, 1998.

[5] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing (WCMC):*, 2(5):483–502, 2002.

[6] J. Cartigny and D. Simplot. Border Node Retransmission Based Probabilistic Broadcast Protocols in Ad-Hoc Networks. *Telecommunication Systems*, 22(1–4):189–204, 2003.

[7] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proc. of the 6th annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–12, New York, NY, USA, 1987. ACM Press.

[8] V. Drabkin, R. Friedman, and M. Segal. Efficient Byzantine Broadcast in Wireless Ad-Hoc Networks. In *Proc. of the 6th IEEE Conference on Dependable Systems and Networks (DSN)*, pages 160–169, June 2005.

[9] 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.

[10] 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.

[11] P.B. Godfrey and D. Ratajczak. Naps: Scalable, Robust Topology Management in Wireless Ad Hoc Networks. In *IPSN*, April 2004.

[12] K. Guo and I. Rhee. Message Stability Detection for Reliable Multicast. In *Proc. of IEEE INFOCOM'2000*, March 2000.

[13] 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.

[14] Z. Haas, J. Halpern, and L. Li. Gossip-Based Ad Hoc Routing. In *Proc. of the 21st Conference of the IEEE Communication Society (INFOCOM)*, pages 1707–1716, June 2002.

[15] M. Hayden. The Ensemble System. Technical Report TR98-1662, Department of Computer Science, Cornell University, January 1998.

[16] IETF Mobile Ad hoc Networks Working Group. Jitter considerations in Mobile Ad Hoc Networks (MANETs). Available at `http://www.ietf.org/internet-drafts/draft-ietf-manet-jitter-04.txt`.

[17] IETF Mobile Ad hoc Networks Working Group. The Optimized Link State Routing Protocol version 2. Available at `http://www.ietf.org/internet-drafts/draft-ietf-manet-olsrv2-04.txt`.

[18] F. Ingelrest, D. Simplot-Ryl, and I. Stojmenovic. Broadcasting in Hybrid Ad Hoc Networks. In *Proc. 2nd Annual Conference on Wireless On demand Network Systems and Services (WONS)*, 2005.

[19] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM Transactions on Computer Systems (TOCS)*, 25(3):8, 2007.

[20] J. Jensen. Jensen's inequality. `http://planetmath.org/encyclopedia/JensensInequality.html`.

[21] D.B. Johnson and D.A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, volume 353. 1996.

[22] A.-M. Kermarrec and M. van Steen. Gossiping in Distributed Systems. *SIGOPS Operating Systems Review*, 41(5):2–7, 2007.

[23] A. Keshavarz-Haddad, V. J. Ribeiro, and R. H. Riedi. Color-based broadcasting for ad hoc networks. In *Proc. of the 4th IEEE Int. Symposium on Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks (WiOpt)*, pages 49–58, April 2006.

[24] A. Khelil, C. Becker, J. Tian, and K. Rothermel. An Epidemic Model for Information Diffusion in MANETs. In *Proc. of the 5th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWIM)*, pages 54–60, 2002.

[25] A. Laouiti, A. Qayyum, and L. Viennot. Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks. In *Proc. 35th IEEE Annual Hawaii International Conference on System Sciences (HICSS)*, 2001.

[26] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks, 2004.

[27] J. Luo, P. Eugster, and J.-P. Hubaux. PILOT: ProbabilistIc lightweight group communication system for mobile ad hoc networks. *IEEE Trans. on Mobile Computing*, 3(2):164–179, April–June 2004.

[28] M. O. Khaoua M. B. Yassein, L. M. Mackenzie, and S. Papanastasiou. Improving the Performance of Probabilistic Flooding in MANETs. In *Proc. of Int. Workshop on Wireless Ad-hoc Networks (IWWAN)*, 2005.

[29] H. Miranda, S. Leggio, L. Rodrigues, and K. Raatikainen. A Power-Aware Broadcasting Algorithm. In *In Proc. of The 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'06)*, September 2006.

[30] M. D. Penrose. *Random Geometric Graphs*. Oxford Press, 2003.

[31] S. Pleisch, M. Balakrishnan, K. Birman, and R. van Renesse. MISTRAL: Efficient Flooding in Mobile Ad-hoc networks. In *Proc. of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 1–12, 2006.

[32] E. Royer, P. Melliar-Smith, and L. Moser. An Analysis of the Optimum Node Density for Ad hoc Mobile Networks. In *Proc. of the IEEE International Conference on Communications*, June 2001.

[33] Y. Sasson, D. Cavin, and A. Schiper. Probabilistic Broadcast for Flooding in Wireless Mobile Ad hoc Networks. In *Proc. of the IEEE Wireless Comm. and Networking Conference (WCNC)*, March 2003.

[34] B. Schneier. *Applied Cryptography*. Wiley, 1996.

[35] D. Scott and A. Yasinsac. Dynamic probabilistic retransmission in ad hoc networks. In *Proc. of the Int. Conference on Wireless Networks (ICWN)*, pages 158–164, Las Vegas, Nevada, June 2004.

[36] K. Singh, A. Nedos, G. Gaertner, and S. Clarke. Message Stability and Reliable Broadcasts in Mobile Ad-Hoc Networks. In *Proc. of the 4th ADHOC-NOW*, pages 297–310, October 2005.

[37] I. Stojmenovic, M. Seddigh, and J. Zunic. Dominating Sets and Neighbor Elimination Based Broadcasting Algorithms in Wireless Networks. *IEEE Trans. on Parallel and Distributed Systems*, 13(1):14–25, January 2002.

[38] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, 1996. 3rd Ed.

[39] C.K. Toh. *Ad Hoc Mobile Wireless Networks*. Prentice Hall, 2002.

[40] 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.

[41] Y.-C. Tseng, S.-Y. Ni, and E.-Y. Shih. Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc networks. In *Proc. of the 21st International Conference on Distributed Computing Systems (ICDCS)*, pages 481–488, 2001.

[42] Cornell University. JiST/SWANS Java in Simulation Time / Scalable Wireless Ad Hoc Network Simulator. Available at `http://jist.ece.cornell.edu/`.

[43] E. Vollset and P. Ezhilchelvan. Enabling reliable many-to-many communication in ad-hoc pervasive environments. In *Proc. of the 2nd Intr. Workshop on Mobile Peer-to-Peer Computing (MP2P)*, 2005.

[44] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proc. of the 3rd Intr. Symp. on Mobile Ad Hoc Networking & Computing (MobiHoc)*, pages 194–205, 2002.

[45] C.W. Wu and Y.C. Tay. AMRIS: A Multicast Protocol for Ad-Hoc Wireless Networks. In *Proc. of the IEEE MILCOMM*, Nov. 1999.

[46] J. Wu and H. Li. On Calculating Connected Dominating Sets for Efficient Routing in Ad Hoc Wireless Networks. In *Proc. of the 3rd DialM*, pages 7–14, 1999.

[47] Q. Zhang and D. P. Agrawal. Dynamic probabilistic broadcasting in MANETs. *Journal of Parallel Distributed Computing*, 65(2):220–233, 2005.

APPENDIX

A. Bounding a broadcast probability $\mathcal{P}_{send}$

Lemma 3.1 (restated) *Denote by $\mathcal{P}_{send}$ the probability that a random node rebroadcasts a message m. Then, for $n \geq 50$ and $c_1 = 1 - \frac{\beta}{29}$*

$$\mathcal{P}_{send} \geq c_1 \beta \frac{ab}{\pi r^2 (n-1)}$$

Proof: According to the algorithm, $\mathcal{P}_{send} = \min(1, \frac{\beta}{n_q})$, while n_q is the number of neighbors of q. We have assumed that the network is connected and therefore $n_q > 0$.

$$\mathcal{P}_{send} = \sum_{k=1}^{n-1} (\mathcal{P}_{send} | n_q = k) \Pr(n_q = k) =$$

$$\sum_{k=\beta}^{n-1} \frac{\beta}{k} \Pr(n_q = k) + \sum_{k=1}^{\beta-1} 1 \Pr(n_q = k) =$$

$$\sum_{k=1}^{n-1} \frac{\beta}{k} \Pr(n_q = k) - \sum_{k=1}^{\beta-1} \frac{\beta}{k} \Pr(n_q = k) + \sum_{k=1}^{\beta-1} 1 \Pr(n_q = k).$$

For every two nodes p and q, let $R_{p,q}$ be a 0-1 random variable indicating whether nodes p and q are neighbors. Two nodes are neighbors if and only if they are at distance at most r from each other. Therefore, $\Pr(R_{p,q} = 1) = \frac{\pi r^2}{ab}$. For simplicity, denote $\frac{\pi r^2}{ab} = \mathcal{R}$. We have:

$$\Pr(n_q = k) = \binom{n-1}{k} R^k (1-\mathcal{R})^{n-1-k}$$

and

$$E[n_q] = \sum_{k=0}^{n-1} k \Pr(n_q = k) = \sum_{k=0}^{n-1} k \binom{n-1}{k} R^k (1-\mathcal{R})^{n-1-k} = (n-1)\mathcal{R}$$

We separate the sum in $\mathcal{P}_{send}$ into two components and bound each one separately:

1) $\mathcal{A}_1 = \sum_{k=1}^{n-1} \frac{\beta}{k} \Pr(n_q = k) \geq \frac{\beta}{(n-1)\mathcal{R}}$
2) $\mathcal{A}_2 = \sum_{k=1}^{\beta-1} (\frac{\beta}{k} - 1) \Pr(n_q = k) \leq \frac{\beta}{29} \mathcal{A}_1$, for $n \geq 50$.

Preliminary 1.1: Jensen's inequality in probabilistic setting [20]: Let X be some random variable, and let $f(x)$ be a convex function (defined at least on a segment containing the range of X). Then the expected value of $f(X)$ is at least the value of f at the mean of X:

$$\mathrm{E}[f(X)] \geq f(\mathrm{E}[X]).$$

We can use Jensen's inequality to bound $\mathcal{A}_1$, since in our case $f(n_q) = \frac{\beta}{n_q}$ is indeed a convex function[6]:

$$\mathcal{A}_1 = \sum_{k=1}^{n-1} \frac{\beta}{k} \Pr(n_q = k) = E[\frac{\beta}{n_q}] \geq \frac{\beta}{E[n_q]} = \frac{\beta}{(n-1)\mathcal{R}}$$

As for $\mathcal{A}_2$, we will show that it is very small compared to $\mathcal{A}_1$, for the values of n that we consider. Formally,

$$\frac{\mathcal{A}_2}{\mathcal{A}_1} = \frac{\sum_{k=1}^{\beta-1} \left(\frac{\beta}{k} - 1\right) \Pr(n_q = k)}{\sum_{k=1}^{n-1} \frac{\beta}{k} \Pr(n_q = k)} \approx \frac{\sum_{k=1}^{\beta-1} \left(\frac{\beta}{k} - 1\right)}{\sum_{k=1}^{n-1} \frac{\beta}{k}} =$$

$$\frac{\sum_{k=1}^{\beta-1} \frac{\beta}{k} - (\beta - 1)}{\sum_{k=1}^{n-1} \frac{\beta}{k}} = \frac{\sum_{k=1}^{\beta-1} \frac{1}{k} - (1 - 1/\beta)}{\sum_{k=1}^{n-1} \frac{1}{k}}$$

$$\approx \frac{\ln(\beta - 1) - 1 + 1/\beta}{\ln(n-1)} \leq \frac{\beta}{29}$$

The first approximation is due to the fact that n_q is binomially distributed random variable with many trials (we assume n to be large). As such, it has a small variance, and therefore all the significant contributors to this sum are very close to each other, i.e., they all lie within a small segment. We can thus say that $\Pr(n_q = k)$ are approximately equal for all k and reduce the numerator and the denominator by $\Pr(n_q = k)$. In the second approximation we have used the formula of harmonic numbers, i.e., $H_n = \sum_{i=0}^{n} \frac{1}{n} \approx \ln n$.

Finally, $\frac{\ln(\beta-1)-1+1/\beta}{\ln(n-1)} \leq \beta/29$ holds any β and $n \geq 50$.

We can now conclude the proof by substituting the two components into $\mathcal{P}_{send}$:

$$\Pr(Send_q = 1) = \mathcal{A}_1 - \mathcal{A}_2 \geq \mathcal{A}_1 - \frac{\beta}{29}\mathcal{A}_1 \geq \frac{(1 - \frac{\beta}{29})\beta}{(n-1)\mathcal{R}}$$

■

B. 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.

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 18. It introduces 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 neighbors of a node p that p has not suspected yet of being malicious form its set of trusted neighbors. The primitive `send` is a point to point send. The primitive `verify_signature` verifies that $sig(m)$

[6]This is also a case of weighted arithmetic and harmonic means inequality. That is, $\frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \geq \frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} \frac{w_i}{x_i}}$.

Upon `send`(msg) by application **do**
(C01) *gos_msg* $:= msg\_id||node\_id||sig(msg\_id||node\_id)$;
(C02) *data_msg* $:= msg\_id||node\_id||msg||sig(msg\_id||node\_id||msg)$ $||sig(msg\_id||node\_id)$;
(C03) `prob_bcast`($prob = 1$, *data_msg, DATA*);
(C04) `lazycast`(*gos_msg, GOSSIP*);

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

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

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

Fig. 18. Maliciousness Resilient RAPID (lines that were modified w.r.t Figure 4 are boxed)

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. A node p that executes `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 that sent 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 18 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 18 does not include this optimization for simplicity.

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[7]), (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 as follows. 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 m from q after the timeout, it starts suspecting q as being malicious. In this case, p stops counting q for calculating its rebroadcasting probability.

As for the fourth attack, in our model we assumed that all messages are delivered with a non-zero probability. Hence, by assumption, the fourth attack is not possible. The 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.

[7]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.