Canning Spam in Wireless Gossip Networks

Daniela Gavidia
Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081HV
Amsterdam, The Netherlands
Email: daniela@cs.vu.nl

Gian Paolo Jesi
University of Bologna
Mura Anteo Zamboni 7
40127 Bologna, Italy
Email: jesi@cs.unibo.it

Chandana Gamage
Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081HV
Amsterdam, The Netherlands
Email: chandag@cs.vu.nl

Maarten van Steen
Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081HV
Amsterdam, The Netherlands
Email: steen@cs.vu.nl

***Abstract*— Once a problem associated only with email, spam is now affecting other media, such as instant messaging, blogs, newsgroups and mobile phone messaging. As wireless networks become more commonplace, we can expect that spam will find its way into upcoming wireless communication services. This paper studies the threat posed by malicious nodes inserting spam in a wireless network using gossiping as a method for information dissemination. We identify the security mechanisms needed to protect our gossip network against the proliferation of spam, reducing the problem to a matter of finding and removing corrupted messages. Finally, we propose a probabilistic method of integrity checking to contain the spread of spam which we evaluate through extensive simulations.**

I. Introduction

Being an extremely robust and scalable communication model, gossiping appears to be an ideal solution for information dissemination in highly dynamic environments, such as wireless networks. The simplicity and distributed nature of gossiping has already sparked interest for its use in wireless environments, ranging from sensor networks to MANETs. However, while gossip networks are often described as being robust to failures, their ability to cope with malicious behavior is rarely addressed. They can gracefully handle the departure of more than half of their members, but this strength would not be as impressive if a few malicious insiders could cause serious damage.

The effectiveness of gossiping is based on the collective effort by the nodes in the network, which results in the workload (and responsibility) being divided among the collection of nodes. With every node playing an equal role in the network, adhering to the agreed code-of-conduct is essential. However, assuming that every node will behave appropriately would be naive. It can be expected that the introduction of malicious nodes will disturb the balance in the gossip network. The extent of the disruption is the focus of this study.

In this paper, we explore the effect of having malicious nodes in a wireless gossip network used for information dissemination. The attack of choice for these malicious nodes is spamming. In the broadest sense of the word, spam is defined as *unsolicited email*. While spam often refers to unrequested emails of commercial nature that are sent in bulk, the term is also used to describe irrelevant or inappropriate messages in newsgroups or message boards, as well as non-commercial emails (religious, political, etc.) or junk mail. Nowadays, spam is not restricted to email anymore. It has made its way into other media, such as instant messaging, blogs, newsgroups, p2p networks and mobile phone messaging. For the purpose of this paper, we refer to any kind of message placed in the network as a result of malicious behavior as spam.

For the most part, nodes in a wireless network have limited resources compared to the average wired workstation making spam a serious threat and not just a nuisance. As our network is being used to disseminate information, it is only natural that selfish nodes would try to exploit the system by overloading it to suit their needs. The intent of these malicious nodes may be to achieve maximum exposure or even to destroy the system by polluting it with junk. Regardless of their motivation, our interest lies in determining the extent of the threat and minimizing the damage as much as possible without resorting to expensive and complex solutions.

A. The Problem with Spam in Wireless Gossip Networks

Gossiping as a method for information dissemination relies strongly on information being forwarded through randomly chosen paths. At each step, information is passed along to another peer selected on-the-go, making it virtually impossible to anticipate the path that a piece of information will travel. This random movement of information works in favor of dissemination as it ensures that information will find its way to all peers with certain probabilistic guarantees.

The problem with spam in a gossip network is intrinsically related to the dissemination properties of gossiping. As has been noted before [1], once a piece of information is gossiped it is extremely hard to remove it from the network unless special mechanisms for removal are in place. This makes the spam problem much more severe in a gossip network than spam email on the internet, from a theoretical point of view. Gossip networks used for data dissemination reduce the amount of work for the spammer to the bare minimum of injecting the spam and then sit back and watch as all other peers collaborate to deliver the spam. There is no need for the spammer to go through the process of trying to collect the addresses of potential targets. Knowing only one node in the network is enough for the spammer to start operating. After all, all other nodes will make sure that his/her message is delivered.

As for accountability, the spammer is in an enviable position. In gossip networks, the nodes themselves act as routers for the delivery of data. As a result, nodes can not be

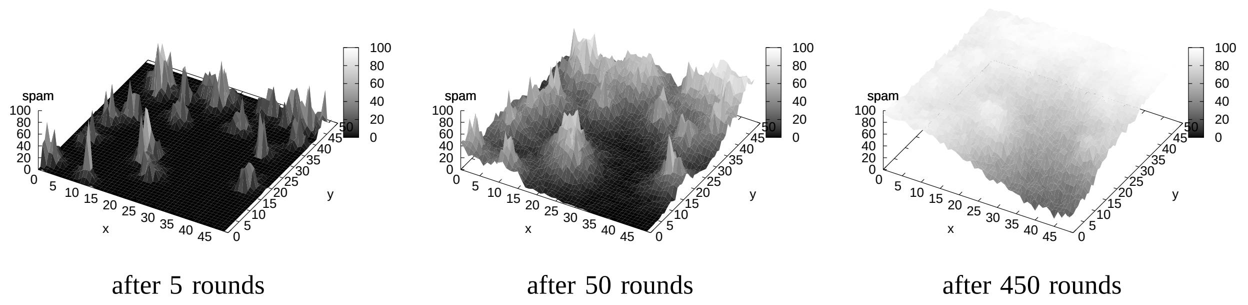

Fig. 1. Spread of spam after 5, 50 and 450 rounds of gossiping. Nodes are arranged in a 50 x 50 grid with 1% of nodes being spammers. The level of pollution of their caches is indicated by the height and brightness of the surface.

held accountable for the data they deliver. This makes tracking down the source of the spam (or any other piece of data for that matter) very difficult. In addition, gossip networks are often promoted on the merits of being decentralized (no central authority) and robust (being able to deal with nodes coming and going gracefully). This works in favor of a malicious node too, as there is no central authority to keep track of its behavior and its sudden joining or leaving will not disrupt the network or be seen as suspicious.

B. *Motivating example*

The magnitude of the damage that can be caused by a few malicious nodes in a gossip network can best be illustrated through an example. Consider a collection of nodes arranged in a grid, gossiping with their four neighbors to the North, South, East and West. They gossip according to the shuffle protocol, which will be explained in detail later on. For now, the most important observation to make is that any two nodes that engage in a shuffle essentially *swap* a number of data entries from their caches. In doing so, they not only preserve the data that are collectively stored in the network, but also "move" these data around in a seemingly random fashion. The underlying idea is that by randomly shuffling data entries between nodes, all nodes will be able to see all data eventually.

Nodes gossip periodically, swapping half of the contents of their caches with a randomly chosen neighbor. In our scenario shown in Figure 1, after nodes have been shuffling for some time (50 cycles or rounds), malicious nodes appear. These spammers account for 1% of the network, but the effect of their actions is devastating to the network. Instead of forwarding the messages from their peers, they drop them and replace them with spam. Figure 1 shows how the caches of the nodes become polluted with spam. In just 5 rounds their presence can be felt. Gradually, they replace valid items with their own, filling up the network with spam. Eventually, the network will be saturated with spam at 100% and all valid items will have been lost.

C. *Contribution*

Our contribution is twofold. First, we show that wireless gossip networks are highly vulnerable to the proliferation of spam. In fact, we claim that without any security mechanisms, spammers could easily take advantage of the dissemination properties of gossiping to overwhelm the network, consuming valuable resources (storage space, bandwidth and processor cycles) at the same time. Second, by means of well-established security measures, we reduce the spam problem to a matter of integrity checking. Additionally, we propose a probabilistic solution for verifying the integrity of messages which succeeds not only in reducing the amount of spam in the network, but also in restricting its dissemination.

The remainder of this paper is organized as follows. In the next section, we describe the system model for our gossip network, specifying the assumptions we make and describing the gossip protocol used. Section III details the basic mechanisms that need to be in place to establish a line of defense against malicious nodes. In Section IV, the method of attack by malicious insiders is explained, as well as the obvious (and more expensive) ways to counter the attack. Section V describes the probabilistic solution we propose as a way to fight malicious insiders. An experimental evaluation based on simulations is presented in Section VI. Related work is discussed in Section VII followed by conclusions and future work in the last two sections.

II. System Model

A. *General Description*

We focus on a system where a heterogeneous mix of fixed and mobile nodes, ranging from mobile devices such as PDAs and smart phones to PCs with internet access, collaborate by volunteering storage space for the creation of a collective data space. Users in the system are able to publish events, which we call *items*, of interest to other users. The nodes in the system devote a limited amount of space, which we refer to as their *caches*, to store items. The collection of caches of all nodes in the network makes up a *collective data space*.

The caches are updated periodically using the gossip protocol first introduced in [2]. As a result, the items in a node's cache are in transit, which means that they could be exchanged for other items at any moment. Items are not purposefully routed. Once published, they become part of the collective data space, replicating themselves (the number of replicas is dictated by the storage capacity of the network) and

```
/*** Active thread ***/
// Runs periodically every T time units
Q = selectPeer()
buff_send = selectItemsToSend()
send buff_send to Q
receive buff_recv from Q
cache = selectItemsToKeep()
```

(a)

```
/*** Passive thread ***/
// Runs when contacted by another node
receive buff_recv from any P
buff_send = selectItemsToSend()
send buff_send to P
cache = selectItemsToKeep()
```

(b)

Fig. 2. Skeleton of an epidemic protocol.

moving freely through the network (geographic restrictions for the dissemination of items are also possible).

Taking part in gossip exchanges results in a node populating its cache with a collection of items. As the cache size is limited, the contents of a cache constitute a sample of the totality of items available in the network.

Users of the system can discover items of interest by going through the items in the local cache. Depending on the number of items in the network, the local cache may not contain all items of interest to a user at a particular time. Nevertheless, previous experiments have shown that all items of interest can be discovered after participating in enough gossip exchanges [3]. Items of interest can then be stored separately in the node's private data store.

B. Assumptions

Items can be published by any user of the system and are propagated through the network in the form of *entries*. While an item is a piece of information, an entry is the representation of the item in the network and for each item several entries may exist. The dissemination of entries occurs between neighboring nodes that exchange entries. As entries are gossiped, replication may occur naturally if a node has available storage space to keep a copy of an entry. As a result, after an item is published and gossiped, many entries for this item may be present in the network, The number of entries per item is dictated by the capacity of the network and the number of items published, as explained in [3].

A unique *id* is associated with each node. The entries that a node inserts into the network can be uniquely identified by a combination of the node id and a sequence number. In its most basic form, an entry contains a unique id, a timestamp and a time-to-live. There may be other fields of information depending on the application. A limited number of these entries can be stored by each node in its local *cache*. A node can store, at most, c entries in its cache. For our experiments, all nodes have the same cache size c. Nodes in the network gossip periodically, exchanging the entries in their caches. We define a *round* as a gossiping interval in which each node initiates an exchange once.

C. The Shuffle Protocol

The data exchange between nodes follows a predefined structure, with each node initiating an exchange once per round. Figure 2 shows the skeleton of the push-pull epidemic protocol we use for communication. Three methods, `selectPeer()`, `selectItemsToSend()` and `selectItemsToKeep()` represent the core of the protocol. By implementing different policies in these methods, various epidemic protocols, each with its own distinctive characteristics, can be instantiated.

In the shuffle protocol, each node agrees to keep the entries received from a neighbor for the next round. This might seem trivial, but given the limited storage space available in each node, keeping the entries received during an exchange implies discarding some entries that the node has in its cache. By picking the entries to be discarded from the ones that have been sent to the neighbor, we ensure the conservation of data in the network. The policies for the shuffle protocol are summarized as follows:

- `selectPeer()`: Select a neighbor randomly
- `selectItemsToSend()`: Randomly select $s \leq c$ entries from the local cache and send a copy of those entries (`buff_send`) to the selected peer.
- `selectItemsToKeep()`: Add received entries (`buff_recv`) to the local cache and remove repeated entries. If the number of entries exceeds c, remove entries among the ones that were previously sent (unless they were also in `buff_recv`) until the cache contains c entries.

An example of an application of the shuffle protocol is presented in our earlier work on a *news service* for wireless mesh networks [3]. The service is provided by a mesh backbone composed of a large number of wireless routers which communicate through gossiping. Users in charge of the routers running the news service are able to publish events, which we call *news items*, of interest to other (mobile) users. These mobile users carry around *clients*, which are portable devices (such as smart phones, laptops or PDAs) capable of connecting to the mesh backbone to retrieve news items. Essentially, the clients poll the routers for news items matching the interests of users. By specifying their preferences in advance and communicating them to a nearby router, users are able to receive in their portable devices only relevant news items.

III. Preparing for the fight

Securing a gossip-based system like the one we propose against malicious nodes requires 1) regulating the entry of nodes into the system (access control), 2) being able to accurately identify the source of an item (source node authentication) 3) ensuring the integrity of messages and 4) enforcing fair use of the system (rate control).

A. Access Control

To ensure that only authorized nodes can join the network, issuing *credentials* for these nodes is required. One possible solution is to have a *Certification Authority (CA)* certify a public key for each node. This procedure would only take place once, establishing the identity of each node and allowing nodes to refuse communication with outsiders.

B. Source Node Authentication

By the time an item arrives at a particular node's cache, it has most likely been shuffled around several times by other nodes. As a result, when a node receives an item from a neighbor, it can not make any assumptions about the item's origin. In order to be able to identify the source of a item, it is necessary for the item to be digitally signed by the original node who published it.

C. Integrity

Given that most likely an item has been forwarded several times before reaching an interested user, the item has to be protected against malicious insiders who may want to modify its contents. By having the source sign the item, a user can check if the item has been modified along the way.

An ideal solution for preserving the integrity of items in the network would be to verify the integrity of each item at every hop. This would require that every node executes a public key signature verification operation for every item it receives from a neighbor. The computational workload of such a solution could be prohibitive. Another more effective way of ensuring the network remains free of forged items is for every node to do a batch verification of the signatures on items received from a neighbor. Verifying multiple signatures in batches is less expensive than verifying each signature at a time. We elaborate more on this and propose our own solution for ensuring the integrity of items in Sections IV and V, respectively.

D. Rate Control

The shuffle protocol ensures that, on average, each item has the same number of entries in the network (this and other properties of the shuffle protocol are explained in detail in [3]). Given that the collective storage space is limited, a larger number of different items in the network results in a smaller number of entries per item. Therefore, a node producing an excessive number of items would occupy a large portion of the storage space with its items reducing the number of entries that other nodes can place. To ensure that nodes do not abuse the system by flooding the network with their own items, a mechanism for rate control is needed.

This flooding of items by a node is a form of a denial-of-service attack. As shown in [3] , the dissemination speed of items through the network is inversely proportional to the number of items published. It follows that the insertion of excessive amounts of items by one node has a negative effect on the performance of the system, given that dissemination speed is sensitive to the amount of items in the network. In essence, more items in the network (due to one node's excessive publishing) result in the dissemination speed of all items slowing down. For this reason, it is necessary to prevent a node from publishing an excessive number of items. Otherwise, a single "overactive" node could cause the service to slow down to the point of not being useful anymore.

Rate control can be enforced by restricting the id space of items per node. This way, a node would be allowed to have at most x items in the network at any point in time, where x is the size of the id space of items per node. For example, the id space of items per node could be restricted to n bits resulting in 2^n items. After a node has published 2^n items with different ids, the next published item will have the same id as one of the previously published items. Since nodes are only allowed to hold one entry per item based on the item id, the more recently published item will overwrite the older item in the network resulting in an upper bound for the amount of storage space occupied by a node's items. With d published items in the network, a node could only occupy at most x/d of the collective storage space.

IV. Spamming through the Corruption of Messages

The shuffle protocol ensures fairness, meaning that each node can use up the same fraction of collective storage space for its items. As a result, a malicious node can insert only so much spam under its own identity. In order to place more spam in the network, a malicious node would have to utilize the ID space of items per node assigned to other nodes. Analogous to the way an email spammer uses false email identities to increase the likelihood that his spam makes its way into our inboxes, a malicious node in our gossip network can place more spam by corrupting the content of the entries that pass through its cache. In essence, a malicious node would be replacing the content of other nodes' entries with its own while keeping the entries' metadata (ID, signatures, ...) intact. This way, the spammer can steal the storage space of other nodes and create more instances of its messages. The spam problem then becomes a problem of preserving the integrity of messages.

A. The threat of malicious insiders

With the measures to prevent unauthorized nodes from infiltrating the network in place, being able to cope with attacks from malicious insiders becomes the biggest challenge. Our gossip network obtains its desirable properties from the periodic execution of a specific gossip protocol at every node. Having nodes in the network behave differently can pose a major threat to the system.

Unlike fixed networks, wireless networks rely on nodes forwarding messages for their neighbors. Without a trusted routing infrastructure available, a great deal of responsibility is placed on the forwarding nodes to deliver a message.

Assuming that malicious nodes are present, at every hop there is a chance that the message might be tampered with. In addition to this, we are dealing with a gossiping system which relies heavily on randomness to forward its messages. The combination of these two aspects result in malicious insiders having plenty of opportunities to corrupt the content of messages and be safe from detection due to the random nature of gossiping.

B. *Checking all messages at every hop*

A conventional approach to security can be applied to ensure the integrity of items. Under this scheme, all entries in the network are required to be signed by their publisher and are subject to integrity checks. Integrity checks can be used to fight attacks based on replays of old entries and modification of entries, as the checks would discover that the content of the entry has been tampered with. However, given that in our system items are constantly being gossiped, verifying all entries received during a gossip exchange would be computationally very expensive. In essence, an entry would have to be verified at every hop. Even though doing this would permit the identification of malicious nodes as soon as they appear, the cost of following this approach would be prohibitive.

C. *Batch Verification*

An alternative to verifying the signatures of the entries received one-by-one is to do a batch verification [4]. Verifying multiple digital signatures simultaneously, instead of verifying each one individually, can be done at lower costs with different schemes for fast verification of digital signatures in batches. These schemes test the validity of all signatures in a batch and the test would succeed only if all signatures are valid. The drawback is that batch verification does not identify which signatures are invalid in the batch. This is, however, not critical since discovering any invalid signature in the batch would be enough to conclude that we have come in contact with a malicious node.

Having all nodes do batch verification of signatures after receiving items from a neighbor would allow nodes to discard any invalid batch and take measures against the neighbor who forwarded the dubious entries. However, the benefits of using batch verification are only evident when a large number of signatures are tested. Therefore, unless nodes are exchanging a large number of items at a time, batch verification could still be expensive. Furthermore, it is necessary to use a digital signature scheme where its batch verification algorithm allows a batch of signatures from different signers.

V. Probabilistic Verification

As an alternative solution to checking all entries at every hop, we propose a more flexible and cost efficient approach to combating malicious nodes. Our solution is based on a probabilistic selection of the entries to be checked.

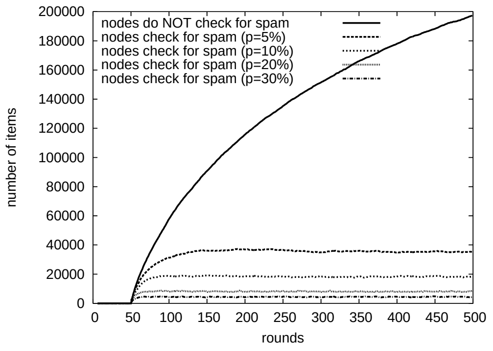

Fig. 3. Number of corrupted entries in the network when 1% of nodes are malicious.

A. *Selection of entries to verify*

The verification phase is incorporated into our gossip protocol in the following way:

- In `selectItemsToKeep()`, each node decides how to merge the entries in its cache with the entries received from the selected peer. Before merging, a probabilistic verification phase is executed.
- Each of the received entries is checked with a probability P_{check}. The integrity of an entry is checked by verifying its digital signature. If the entry is valid, then it is marked as `checked`. Otherwise, the entry is discarded.
- The entries that were not selected for checking and the ones that passed the check are merged into the local cache.

B. *Attack model*

To test the validity of probabilistic verification as a technique to counter malicious behavior in the form of corruption of entries, we assume that a small percentage of the nodes in the network are malicious insiders while the rest behaves according to our gossip protocol.

Malicious nodes execute a slightly different version of the shuffle protocol. In `selectItemsToSend()`, the selected entries are corrupted, with the exception of the entries marked as `checked`. The reason for this is that sending a corrupted entry marked as `checked` will raise suspicion if the receiving node executes an integrity check on that entry. In this paper, other nodes do not make an effort to detect malicious nodes and take measures against them (this is the subject of current study). Nevertheless, we assume that malicious nodes are cautious. As malicious nodes do not want to be trivially discovered, they will execute integrity checks with a P_{check} probability and will only corrupt entries that are not marked as `checked`, thus avoiding direct responsibility for any corrupted entry they have forwarded.

VI. Experimental Results

The results presented in this section correspond to a network of 2500 nodes with a cache size of 100. The nodes were arranged in a square grid topology, with 50 nodes on each side over an area of 50$\times$50 units. The range of each node was

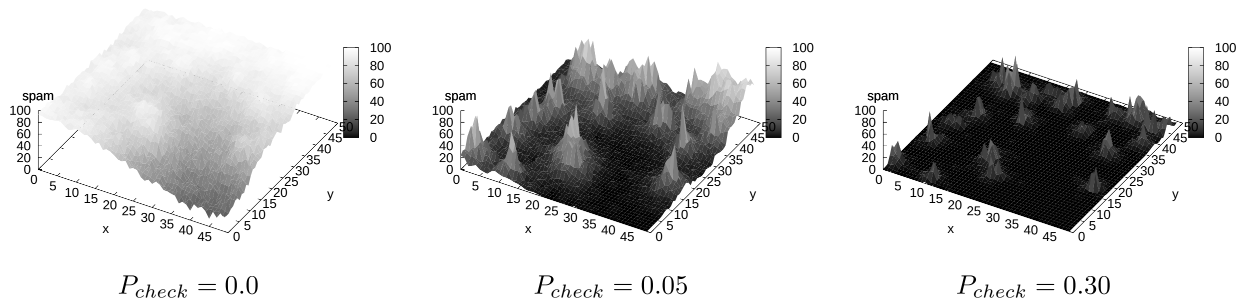

Fig. 5. Spread of spam after 450 rounds since the appearance of 25 spammers (1% of nodes in the network).

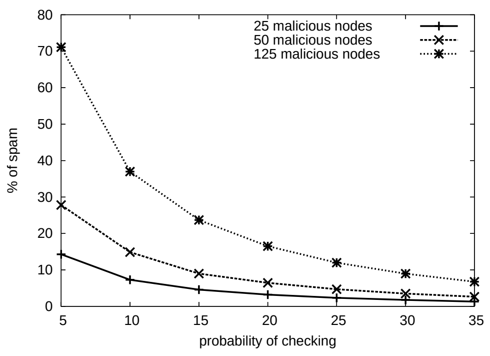

Fig. 4. The percentage of corrupted messages in the network decreases with the probability of checking an entry, P_{check}.

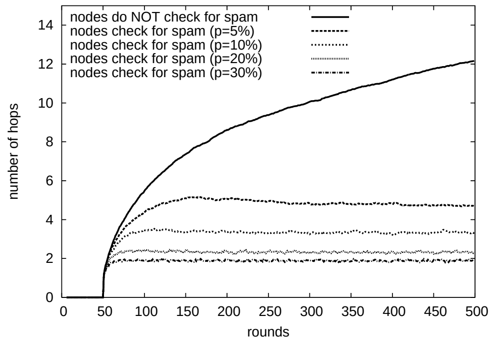

Fig. 6. Average distance (in hops) of corrupted entries from their source over time (in rounds) , with 1% of malicious nodes in the network.

set to 1 unit, making communication possible with the node's immediate neighbors to the North, South, East and West. Nodes placed at random locations in the grid were selected to be malicious. Experiments were conducted for different concentrations of malicious nodes (1%, 2% and 5%).

A. Amount of Spam in the Network

Malicious nodes carry out a very simple attack: corrupt as many entries that pass through as possible, taking into account that some entries will need to be checked. In the absence of any measures to counter the pollution of the network with corrupted entries, this kind of attack is extremely effective.

Figure 3 shows the spread of corrupted entries through the network over time. In the experiment, 50 malicious nodes (1% of the network) appear at round 50 and from that moment start corrupting entries. Entries do a random walk through the network which leads to each item eventually visiting every node in the network, including the malicious ones. As a result, without any integrity checks, the number of corrupted entries keeps increasing until all entries in the network are corrupted. On the other hand, when nodes execute probabilistic checks, the number of corrupted entries soon reaches an equilibrium where the amount of spam generated matches the amount of spam dropped by non-malicious nodes. Experiments with 50 and 125 malicious nodes (2% and 5% of the network, respectively) show similar behavior, but converging to different levels of spam. In all cases, spam spreads through the network after the appearance of malicious nodes, but after an initial period of growth, the amount of spam settles at a level inversely proportional to P_{check}.

The number of spammers present during an experiment directly affects the amount of spam in the network, as we show in Figure 4, which summarizes our experiments regarding the amount of spam in the network. For each number of spammers (25, 50 and 125) and value of P_{check} (from 5% to 35%, with increments of 5), the level to which the amount of spam converges was recorded (by averaging the last 200 rounds) in order to show the relationship between the amount of spam in the network and the probability of checking a received entry. We observe that the amount of spam is inversely proportional to P_{check}. As can be expected, it is also proportional to the number of spammers in the network. This is due to each spammer creating an independent “spam heap” in its surroundings.

The effect of probabilistic verification is that corrupted entries are restricted from spreading too far away from the source, as they become more likely to be removed by a non-malicious node with every hop. Figure 5 shows how spam is contained within an area surrounding the spammer. The snapshots, taken at round 500 for different values of P_{check}, clearly illustrate the benefit of probabilistic verification not only in reducing the amount of spam, but also in decreasing its reach.

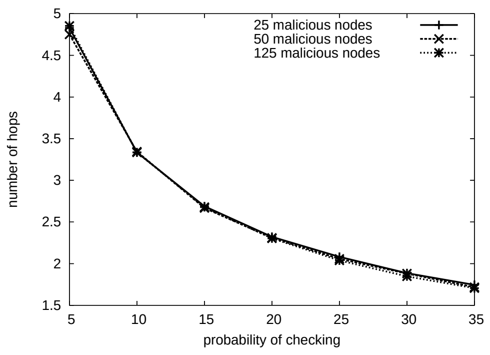

Fig. 7. Average distance (in hops) of corrupted entries from their source for different values of P_{check}. Results for spammers accounting for 1%, 2% and 5% of nodes in the network are shown.

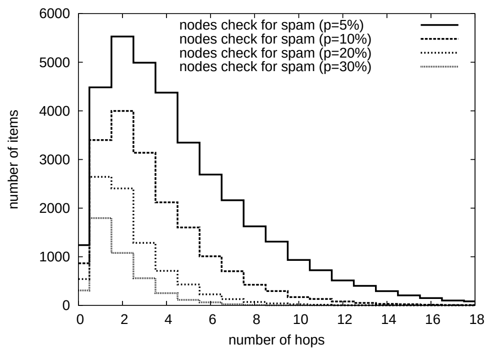

Fig. 8. Distribution of corrupted entries according to their average distance (in hops) from the source.

B. Reach of Spam in the Network

In the same way as the amount of spam reaches a particular equilibrium point depending on the checking probability, the average distance (in hops) from the source that spam travels also reaches a stable state, as can be seen in Figure 6. In this experiment, for each corrupted entry we record the distance (in hops) from its source and calculate an average distance at every round. This average distance serves as an indicator of how far away spam travels before being discovered and removed. After an initial period where corrupted entries find their way into the caches of nodes in the vicinity of a spammer, the corrupted entries start being dropped, preventing their dissemination any further. Notice how the steady state to which the average distance converges in Figure 6 is inversely proportional to P_{check}. Experiments with 2% and 5% of malicious nodes converge to very similar values.

By measuring the value to which the average distance converges (by averaging the last 200 rounds), we can observe its relationship with the checking probability as well as the amount of spammers. Figure 7 shows the average number of hops away from the source that spam travels with respect to the probability of checking for 25, 50 and 125 malicious nodes. An important observation depicted in this graph is that the distance traveled by the spam is independent of the number of malicious nodes present. In fact, the heaps of spam generated by spammers may overlap, as seen previously in Figure 5.

Figure 8, which shows the number of spam entries that have traveled a certain number of hops, summarizes the effectiveness of probabilistic verification as a way of reducing and containing spam. In this graph, we observe (after 500 rounds) the distribution of spam according to the distance from the source for various values of P_{check}. It is evident from the area covered by each curve that higher values of P_{check} reduce the amount of spam in the network as well as reduce the area affected by spam.

VII. Related Work

Previous work has looked at malicious behavior in wireless ad hoc networks as a problem of lack of cooperation and selfish behavior (for example, not forwarding messages to save energy). Security in wireless environments has concentrated mostly on the routing layer by modifying existing routing protocols, such as DSR and AODV.

Efforts to alleviate the problem of malicious behavior by enforcing cooperation include payment systems [5] and reputation systems [6]. Payment schemes assume that a node can be swayed away from his selfish behavior through economic incentives, while reputation systems usually rely on second-hand reputation reports (which could be false). [7] avoids issues of trust by relying only on first-hand observations to build the reputation of a node. In any case, reputation systems aim to isolate the malicious node. In our work, we are interested in reducing the effectiveness of spamming instead of detecting the misbehaving node. We believe that containing the dissemination of spam to the malicious node's neighborhood will discourage malicious behavior in the network. By focusing on the authenticity of the messages in the network (without judging other nodes), our work is more closely related to the efforts to counteract content pollution in peer-to-peer networks [8].

VIII. Conclusions

In this paper, we explored the vulnerability of wireless gossip networks to spamming attacks. We showed that the probabilistic nature of information dissemination in gossip networks makes these networks specially susceptible to the proliferation of spam. In an effort to secure the network, we proposed that only accredited nodes be allowed to gossip. With only authorized nodes gossiping, our efforts focused on dealing with malicious insiders, as these malicious nodes could only spam by taking over the identity of other nodes. This resulted in our proposal of probabilistic verification of messages as a way to fight spam. We evaluated this technique through extensive simulations showing that the amount of spam is effectively reduced and its spread restricted.

IX. Future Work

In the future, we intend to focus on doing a probabilistic analysis of the messages received from a neighbor. With this information, we expect to be able to dynamically adjust the checking probability for each individual node and be able to detect suspicious behavior which will allow us to take action against suspicious nodes.

As mentioned earlier when describing the attack model, malicious behavior could raise suspicion in neighboring nodes. However, the current protocol does not include a mechanism to react when faced with changes in the amount of spam received. In that sense, the current protocol takes a proactive approach to fighting spam. The problem with this approach lies in the constant toll it takes on the nodes, requiring a fixed amount of checks to be performed regardless of the threat.

It is clear that during periods when the threat is low, it would be desirable to lower the amount of checks performed. Likewise, when faced with heavy spamming, an increase in the checking would be appropriate. Our current research focuses on making this possible by observing the traffic from neighbors. The goal of the study is to set the value of P_{check} dynamically according to the changing conditions in the network. To further refine the study, two approaches are being tested: a) maintaining a different P_{check} a) for each node and b) for each link in each node.

In each round, a node only checks a fraction of the entries it receives. We use this sample to estimate the level of pollution in the network. With this information, a new value of P_{check} is calculated based on its previous value and the level of pollution. Preliminary results are encouraging, showing that setting P_{check} dynamically greatly reduces the overall number of integrity checks performed in the network, with the majority of the work being done by the nodes surrounding the spammers. The burden on the neighbors of the spammers can be high, so the logical next step for this work would be to device a set of rules to identify a spammer and reduce (or cut) communication with it.

References

[1] 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 *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC '87)*, August 1987, pp. 1–12.

[2] D. Gavidia, S. Voulgaris, and M. van Steen, "Epidemic-style monitoring in large-scale wireless sensor networks," Vrije Universiteit Amsterdam, Tech. Rep. IR-CS-012.05, 2005.

[3] D. Gavidia, S. Voulgaris, and M. van Steen, "A gossip-based distributed news service for wireless mesh networks," in *Proceedings 3rd IEEE Conference on Wireless On demand Network Systems and Services (WONS)*, Les Menuires, France, January 2006.

[4] M. Bellare, J. A. Garay, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures." in *EUROCRYPT*, 1998, pp. 236–250.

[5] L. Buttyan and J.-P. Hubaux, "Enforcing service availability in mobile ad-hoc wans," in *MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*. Piscataway, NJ, USA: IEEE Press, 2000, pp. 87–96.

[6] S. Buchegger and J.-Y. Le Boudec, "Self-Policing Mobile Ad-hoc Networks," in *P2PEcon 2004*. CRC Press, 2004, p. 6, handbook on Mobile Computing.

[7] S. Bansal and M. Baker, "Observation-based cooperation enforcement in ad hoc networks. Technical Report," 2003. [Online]. Available: citeseer.ist.psu.edu/bansal03observationbased.html

[8] K. Walsh and E. G. Sirer, "Fighting peer-to-peer spam and decoys with object reputation," in *Proceedings of P2PECON Workshop, Philadelphia, Pennsylvania*, August 2005. [Online]. Available: http://trust.eecs.berkeley.edu/pubs/59.html