

MessageReaper: Using Social Behavior to Reduce Malicious Activity in Networks

Matt Spear Juan Lang Xiamoing Lu Norman Matloff S. Felix Wu

University of California, Davis

{mjspear jilang lu}@ucdavis.edu {matloff wu}@cs.ucdavis.edu

ABSTRACT

Every communications medium can be abused for unwanted messages, e.g. email is dominated by spam messages, and Peer-to-Peer (P2P) file sharing systems have high proportions of invalid files. At the same time, the interest in Online Social Networks (OSNs) has grown. OSNs attempt to reduce unwanted messages by restricting communication to approved individuals. OSNs require centralized management, and are too restrictive, disallowing any-to-any communication. This paper introduces a novel, anonymous, economical approach to providing any-to-any communication, utilizing an underlying social network in both distributed and centralized settings that reduces spam. Another issue in distributed systems is users who do not contribute to the network (freeloaders). MessageReaper provides strong incentives for nodes not to send spam and not to freeload. MessageReaper requires only local knowledge and no automatic content analysis (and thus can be combined with existing spam blocking algorithms). It uses a simple greedy routing algorithm, trust structures, and an outcome propagation phase to drastically reduce the amount of spam and penalize freeloading nodes—suppressing these at the source. The outcome of each interaction is identified by the end-users, and only the majority of these need to be correct. MessageReaper achieves fast stabilization, incentives to neither send spam nor freeload, and achieves low false negatives (22%) and negligible false positives (< 1%) even with a large fraction of the network misbehaving (30%). This architecture is applicable to a variety of applications, from email to Instant Messaging (IM) to P2P file sharing to OSNs.

1. INTRODUCTION

Every communications medium can be abused for unwanted messages. These messages can take on many forms: connection requests on Online Social Networks (OSNs) may be unwanted by users, email spam is unwanted by the recipients, and invalid file content is unwanted by its downloader, just to name a few examples. In this paper, we concentrate on messages that are unwanted by their recipients, and we classify all such messages as *spam*. Examples of this broad definition of spam include email spam [40], blog spam [31], and invalid content downloaded online [25].

Though the problem of email spam has been studied

extensively, no solution seems to have addressed it adequately. Symantec [40] reported that as much as 75% of Internet email was spam in 2007. Whitelists are one of the most effective spam filtering techniques, but they are too restrictive—only people who are on each others’ whitelists may communicate, so any-to-any communication is impossible. Another proposed solution is to use fixed global identities [38].

Centralized OSNs, e.g. Facebook, MySpace and Orkut, attempt to address the spam problem by restricting communication. For example, in Facebook users are only able to talk to their immediate friends or those in the same community. In MySpace and Orkut, users can choose to restrict communication to immediate friends only, to everyone within a certain topological distance, or choose not to restrict communication at all. These mechanisms are overly restrictive, yet have not managed to avoid spam: e.g. MySpace and Orkut have been used extensively for spammers [10, 20]. Furthermore, if the social network is attacked by a worm then these restrictions become ineffective [36, 27].

In distributed networks another problem emerges, that of *freeloaders*. Freeloaders do not provide any resources to the system, but only consume. In many Peer-to-Peer (P2P) file sharing applications, freeloaders are a big problem, e.g. in Gnutella it was estimated that as much as 70% of the network was freeloading [1]. BitTorrent [7] attempts to curb freeloading by using a tit-for-tat mechanism wherein users’ download throughput is proportional to their upload throughput.

MessageReaper’s main objectives are twofold: (1) provide any-to-any communication whilst reducing spam, and (2) provide strong incentives to discourage freeloading. We also seek to have a negligible impact on well-behaved nodes. These goals are accomplished without the need of global identifiers. This algorithm can be utilized either in a centralized environment, e.g. Facebook or MySpace, or in a distributed system, e.g. P2P file sharing.

MessageReaper achieves the above objectives by applying trust structures to the forwarding layer, and can be applied in either anonymous networks (those providing both sender and receiver anonymity) or non-anonymous networks. Our key insight is that by al-

lowing communication only along existing friendship links, nodes may influence each other to behave well. This collaboration suppresses misbehaving nodes *at the source*. In addition, MessageReaper achieves fast stabilization, incentives to neither send spam nor freeloading, and achieves low false negatives (22%) and negligible false positives ($< 1\%$) even with a large fraction of the network is misbehaving (30%). This architecture is applicable to a variety of applications, from email to Instant Messaging (IM) to P2P file sharing to OSNs.

The remainder of this paper is organized as follows: Section 2 discusses the threat model. Section 3 describes our system. Section 4 provides an analysis of the system’s false positive and false negative behavior. Section 5 presents simulation results. Section 6 discusses some future work. Section 7 describes related work. Finally, we conclude this paper in Section 8.

2. THREAT MODEL

The first main threat we are considering is spam. One important question is the distribution of spammers in the network. For this, we first consider the distribution of spammers in different domains.

Ramachandran and Feamster [33] showed that email spammers are not isolated to any single region of the Internet, rather, their distribution appears to be uniform. Determining the distribution on social networks is more difficult. One strategy for spam in social networks is to make as many friends as possible in order to spam them [21, 35]. This strategy may have arisen because social networking sites typically allow users to restrict messages to direct friends. On a system in which messages are often not restricted—e.g. Orkut—spammers might use a different strategy, in which case their distribution in the network would be different. Since we allow any-to-any communications, it’s difficult to say *a priori* what the distribution of spammers will be. We suspect that botnets and compromised accounts will continue to be a large source of spam, so we assume that spammers are uniformly distributed throughout the network, in a manner similar to email spammers.

Another question is the level of power of the spammers. Our system does not rely on content analysis or up-to-date signatures, but on the recipient of the message to judge its outcome. While the recipient may employ tools to help her decide whether a message is good or bad, the ultimate judge is a human being. Thus we do not expect spammers to be able to defeat detection. We also assume that each spammer is acting independently—that groups of spammers do not collude in order to mount an attack.

The second main threat we are considering is freeloaders. In the context of our system, a freeloading node is node that wants its messages to be forwarded, but does not forward anyone else’s messages. We also assume freeloading nodes are uniformly distributed throughout the network, and that they do not collude. Please note that, in a typical social network, it is practically not

very easy to perform large-scale collusion attacks that will significantly weaken the defense mechanisms proposed by MessageReaper. We will make an informal argument later in Section 6.

3. MessageReaper

For each high-level interaction—e.g. email message sent, file downloaded, or blog comment posted—MessageReaper allows the recipient to decide an *outcome*, good or bad. This outcome is sent by the recipient to the sender, and noted by every node on the path. A path is a series of nodes connected by friendship relationships. Every node also tracks whether the message reaches its destination (it can infer that it did when it receives the good/bad outcome.)

3.1 Assumptions

MessageReaper makes the following assumptions:

1. **The network is a small-world network.** That is, the path length between nodes using only social links is short. Email networks have been shown to have small-world behavior [14], as has the Internet [2].
2. **Nodes maintain a list of neighbors** and communicate directly only with their neighbors. Direct neighbors may always communicate.
3. **Nodes can identify a node with which they wish to communicate.** This can either be using a global identifier, e.g. an IP address, or using an anonymous identification system, e.g. DSL [3] or hidden services in Tor [11].
4. **Nodes can estimate the distance via a neighbor to any node in the network.** This distance could be calculated directly on a social networking site, or could be provided by a routing protocol, or could be estimated using response times via each neighbor.
5. **The outcome of each interaction is correct with high probability.** Email and instant messaging clients have the ability to mark a message as spam, which is a way to mark an outcome as bad. A similar capability exists on social networking sites—deleting a blog entry is a form of giving it negative feedback.
6. **Outcomes can’t be modified, forged, or replayed.** In a centralized setting (e.g. Facebook) the outcome would be recorded directly and as such this is guaranteed. In a P2P environment a protocol would be used whereby every node on the path could verify the outcome but could not forge it. We leave for future work the actual design of an efficient, anonymous, unforgeable authentication protocol¹.

¹A Public Key Infrastructure (PKI) can also be used, though doing so removes anonymity.

7. **The graph is static.** Modeling the dynamics of adding and removing edges in a social graph with attackers is beyond the scope of this paper.

3.2 Notation

MessageReaper uses a local definition of trust, i.e., each node makes independent decisions based only on local knowledge. Throughout this paper we use the subscript to indicate that a value is from a particular node’s point of view. For example, the node u ’s opinion of the *routing value* (to be defined shortly) via neighbor v to destination t is written as $R_u(v, t)$.

3.3 Trust definition

As we described above, each node records the good/bad outcome of each interaction, in addition to whether a message reached its destination. More formally, we define a set $\mathbf{A} = \{\text{MESSAGE}, \text{ROUTE}\}$ as the set of actions for which each node records an outcome. Given a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ and a node $u \in \mathbf{V}$, we define a trust function for a node u that takes an action and a neighboring vertex and produces a pair:

$$T_u : \mathbf{A} \times \mathbf{V} \rightarrow \mathbb{N}^2 \quad (1)$$

The pair represents the count of good, bad interactions for that action, respectively. That is, given $a \in \mathbf{A}$,

$$T_u(a, v) \stackrel{\text{def}}{=} \begin{cases} \langle m, n \rangle & \text{if } u \text{ and } v \text{ are neighbors} \\ \text{undefined} & \text{otherwise} \end{cases} \quad (2)$$

Thus, $T_u(\text{MESSAGE}, v)$ is u ’s record of v ’s prior messages, with m being a count of the good messages u received from v and n being a count of the bad messages u received from v . Similarly $T_u(\text{ROUTE}, v)$ is u ’s record of v ’s routing history, with m being a count of messages sent via v that reached their destination, and n being a count of messages sent via v that did not reach their destination.

To evaluate the likelihood of an interaction being beneficial to a node we define the probability that the next outcome is good given the number of previous good/bad interactions as:

$$\sigma(\langle m, n \rangle) \stackrel{\text{def}}{=} \frac{m + 1/2}{m + n + 1} \quad (3)$$

The intuition for this function is as follows: If a node has no prior interactions with a neighbor, $\sigma(\langle 0, 0 \rangle) = .5$: the next interaction has equal probability of being good or bad. If $m \gg n$, $\sigma(\langle m, n \rangle) \approx 1$: the next interaction is likely to be good. If $m \ll n$, $\sigma(\langle m, n \rangle) \approx 0$: the next interaction is likely to be bad. The formula can be derived in a straightforward way from Bayes’ theorem.

Both records of prior interactions (Equation (2) for both message and route interactions) and the trust function (Equation (3)) are used for routing messages.

3.4 Message Forwarding

Assume that node s wants to send a message M to node t that is not a direct friend of s . s chooses an ID

for the message and appends it to the message. It then chooses among its neighbors the best neighbor u for sending to t , and forwards the message to u . u decides whether it should forward the message. If it should, it appends an (anonymized) ID for itself to the message that will be used to exclude u from future neighbor selections. u chooses among its neighbors the best neighbor for sending to t , and this process continues until the message is either dropped or reaches its destination.

3.4.1 Preparing the message for sending

In order to be able to associate an outcome with the message, s creates a unique message ID for the message and appends it to the message. The message ID should be probabilistically unique—a collision-resistant hash of the message and a nonce would suffice.

3.4.2 Choosing the best neighbor

Assume node u wants to send a message to node t that is not a direct friend of u . u computes the *routing value* for each neighbor:

$$R_u(v, t) \stackrel{\text{def}}{=} \alpha \cdot \hat{\delta}_u(v, t) + (1 - \alpha) \cdot \sigma(T_u(\text{ROUTE}, v)) \quad (4)$$

where $\hat{\delta}_u(v, t)$ is a scaled estimate of the distance from u to t via neighbor v , and α is the *path weight* parameter.

The path weight parameter α is a system parameter that determines to what extent each node should favor path length vs. trust when considering the next hop. When $\alpha = 1$, the shortest path is always chosen, regardless of how good each neighbor is at forwarding messages. When $\alpha = 0$, the path becomes a directed self-avoiding walk, where edges that have successfully forwarded messages in the past are explored before other edges. The best value for α is explored further in Section 4 and Section 5.

The scaled distance estimate $\hat{\delta}_u(v, t)$ is defined as:

$$\hat{\delta}_u(v, t) \stackrel{\text{def}}{=} \frac{\delta_u(\max, t) - \delta_u(v, t)}{\delta_u(\max, t)} \quad (5)$$

where $\delta_u(\max, t)$ is the maximum distance via any of u ’s neighbors, and $\delta_u(v, t)$ is the unscaled distance estimate from u to t via v . The range of the scaled distance estimate is therefore $[0, 1]$.

u chooses the best next hop by choosing the neighbor v whose routing value is maximal:

$$\text{next_hop}_u(t) \leftarrow \max_{v \in \mathcal{N}(u) \setminus \mathcal{P}(M)} R_u(v, t) \quad (6)$$

where $\mathcal{P}(M)$ is the existing path that M has traversed, which is initially empty.

3.4.3 Deciding whether to forward a message

When a node v receives a message from a friend u destined to t , it decides whether to forward the message via Algorithm (1). There are two key elements to this algorithm:

1. A node decides whether to forward a message based on the lower of the two trust values. This achieves

both main aims of this paper: it penalizes nodes for sending bad messages, and it provides incentives for forwarding messages to penalize freeloaders.

2. If the minimum trust value is above the system *Threshold* parameter, the message is forwarded. This is the result of the built-in bias against false positives: we are less concerned with spam reaching its destination than we are about legitimate messages getting dropped. Good values for *Threshold* are also explored further in Section 5.

If Algorithm (1) calls `Forward()`, node v chooses the next hop according to Equation (6). If there is no neighbor the message has not yet visited, v drops the message. (Note that v does not need to know the entire path— it is sufficient for v to only recognize its neighbors). If Equation (6) returns a next hop, node v inserts an (anonymized) identifier for itself into $\mathcal{P}(M)$, and sends $(M, \mathcal{P}(M))$ to the next hop.

Algorithm 1 Decide whether to forward a message from u at v

- 1: $\text{Pr}_f \leftarrow \sigma[\min(T_v(\text{ROUTE}, u), T_v(\text{ROUTE}, u))]$
 - 2: **If** $\text{Pr}_f > \text{Threshold}$ **Then**
 - 3: `Forward()`
 - 4: **ElseIf** $\text{Pr}_f \geq \text{Random}(0, 1)$ **Then**
 - 5: `Forward()`
 - 6: **Else**
 - 7: `Drop()`
 - 8: **EndIf**
-

3.5 Outcome propagation

When a message is received, the recipient decides its outcome, o , and propagates it along the path by which it came:

$$\forall_{\{t, \dots, i, \dots, s\} \setminus \{s\}} T_i(\text{ROUTE}, i-1) \leftarrow T_i(\text{ROUTE}, i-1) + o \quad (7)$$

That is, each node in the path adds to its trust value for the prior node in the path with the outcome of the message. Upon receipt of an outcome message, each node on the path updates its trust value for routing with a good interaction ($o = \langle 1, 0 \rangle$). If no outcome message is received within a timeout value, each node on the path updates its trust value for routing with a bad interaction ($o = \langle 0, 1 \rangle$).

3.6 Example

Two small examples illustrate the effectiveness of our system. Figure 1(a) shows a 20 node network, in which 25% of the nodes (in black) are freeloaders, and the remaining nodes (in white) are ordinary nodes. The edges between the nodes show the direction and degree of trust between nodes. A dark edge is not very well trusted, while a lighter edge is more trusted. The thickness of the edge shows how many messages are

sent along an edge. The size of the node indicates its betweenness.

After 10 rounds of simulation, we see that the freeloading nodes are avoided when sending messages. This is especially evident with node 5, the large, centrally located freeloading node. Even though it is centrally located, it is avoided by its neighbors, as shown by the thin edges to it, and the thicker edges that avoid it. The darkness of the edges to it also indicates that its neighbors do not trust it.

Figure 1(b) shows a similar network in which 25% of the nodes are spammers (the darker nodes.) The edges to spammers are quite dark, indicating their neighbors do not consider them to be very trustworthy. The edges from the spammers to their neighbors are very thin, indicating they are able to send few messages. The edges from the normal nodes to the spammers are sometimes thick, which indicates that normal nodes may still send messages to spamming nodes, even if they do not accept any from them.

4. ANALYSIS

4.1 Overhead

4.1.1 Path length

Our system’s efficiency depends on the path length of message transmission. We already assume a small world network, in which the diameter is $O(\log N)$ (where $N = |\mathbf{V}|$.) However, if the average path length is long, the paths may still be inefficient. Mislove et al [29] showed that the average path length is small, much less than the diameter, for a number of online social networks.

4.1.2 Message overhead

In order to support distinguishing one message from another, messages have a message ID appended. This size is some fixed constant c . Messages also have a small identifier added by each node the message is forwarded by. This adds $O(\log N)$ size to each message. The total overhead to each message is thus $O(\log N)$.

For each interaction, we add a single outcome message. This message is of a small fixed size.

4.1.3 Memory consumption

We have described our system as though every interaction has an outcome associated with it. This is of course unlikely, and nodes have to remember a message ID at least until an outcome is received for it. This imposes a potentially unbounded memory requirement on each node. The usual techniques for dealing with limited memory can be applied here: assuming some messages will never be replied to, older message IDs can be discarded either after enough time has elapsed or after some space bound is reached. For such cases, an outcome of $\langle 0, 0 \rangle$ is recorded.

4.2 Predicting the probability of success

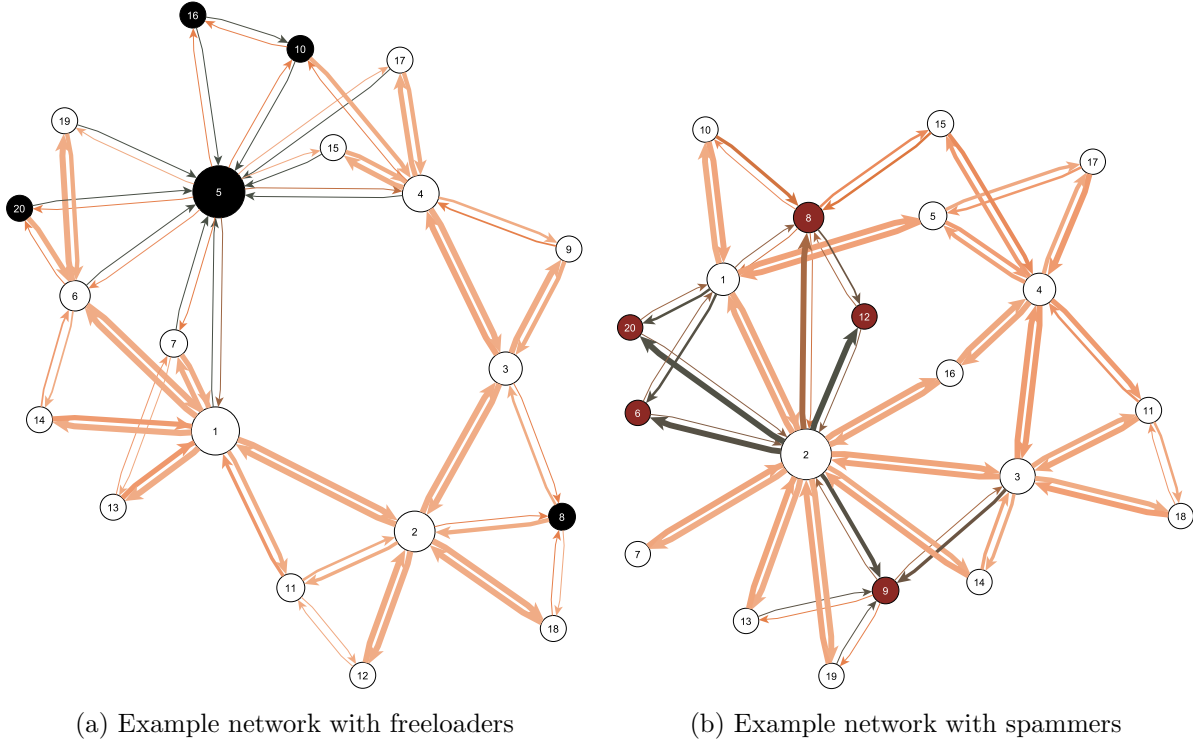


Figure 1

We now derive an estimator for the probability that a node will be able to send a message to its intended recipient given the presence of spammers in the network. To do so, we:

1. Determine the conditions under which a good node is chosen over a spammer on a path.
2. Estimate the probability that a message succeeds given a path of a given length.
3. Estimate the probability that a spammer is chosen.
4. Derive an expression for the probability that a message succeeds given the sending node is either a good node or a spammer.

4.2.1 Choosing a good node over a spammer on a path

Assume we are at a node u with at least one good neighbor and at least one spamming neighbor. We wish to calculate the circumstances under which the good neighbor will be chosen to forward a message.

Recall from Equation (6) that the node chosen has the maximum of all its neighbors' routing values. If we assume that all good nodes send equally many messages, if node u has more than one good neighbor, only the good neighbor with the shortest path to t need be considered, as it will always be chosen over u 's other good neighbors. Let this neighbor be v .

Parameter	Description
N	The number of nodes in the network
k	The number of spammers in the network
α	The path weight parameter
x	A path length
b	The probability that a spammer is chosen using the routing algorithm
p	The probability that a message from a good node is forwarded
q	The probability that a message from a spammer is forwarded
$d(i)$	The probability that a node has degree i
$\ell(x)$	The probability that a path of length x is formed via the routing algorithm
λ	The exponential parameter for $\ell(x)$
$\ell'(x)$	The probability distribution of the difference in path length for routing around a spammer
λ'	The exponential parameter for $\ell'(x)$
Δ	A random variable drawn from $\ell'(x)$

Table 1: Symbols used in the analysis

Similarly only the spammer with the shortest path to t need be considered. Let this neighbor be v' .

We wish to calculate under what circumstances

$$R_u(v, t) > R_u(v', t) \quad (8)$$

In order to do so, we first estimate the routing trust value of both v and v' , that is, $T_u(\text{ROUTE}, v)$ and $T_u(\text{ROUTE}, v')$. Each depends on the probability of the message reaching its destination, which is what we are trying to determine. For simplicity of analysis, we assume that the system has run long enough that for good nodes, $m \gg n$, such that

$$\sigma(T_u(\text{ROUTE}, v)) \approx \frac{m + 1/2}{m + 1}$$

Similarly, we assume that a spammer neighbor has been ostracized by its neighbors, and $n' \gg m'$. Hence

$$\sigma(T_u(\text{ROUTE}, v')) \approx \frac{1/2}{n' + 1}$$

Assume that the number of good messages sent through v is approximately equal to the number of messages dropped along the path through v' , i.e., $m \approx n'$. Let

$$r = m = n'$$

We use a simplified version of the routing value that makes use of a global distance between nodes:

$$R_u(v, t) \stackrel{\text{def}}{=} \alpha \cdot \delta(v, t)^{-1} + (1 - \alpha) \cdot \sigma(T_u(\text{ROUTE}, v)) \quad (9)$$

where $\delta(v, t)$ is the distance in hops between nodes v and t .

Substituting into Equation (8) and solving for $\delta(v, t)$ yields

$$\delta(v, t) \leq \frac{\alpha \delta(v', t)r + \alpha \delta(v', t)}{((\alpha - 1)\delta(v', t) + \alpha)r + \alpha} \quad (10)$$

This is the maximum path length a good node can have and still be chosen over a spammer. We define the right side of the inequality as $\delta_{max}(v, t)$ – the maximum path length from a good neighbor v to a destination t . Note that when the denominator is ≤ 0 , the good node will always be chosen, irrespective of path length. This occurs when

$$\alpha \geq \alpha_{min} = \frac{\delta(v', t)r}{(\delta(v', t) + 1)r + 1} \quad (11)$$

α_{min} is independent of k , the number of spammers in the network.

4.2.2 Probability of success for a path of a given length

Suppose we have a path of x nodes from source to destination. Each node has probability b of being a spammer, with the good/spammer states of the nodes being statistically independent. Furthermore, assume that $x \ll k$, i.e. the diameter of the graph is much less than the number of spammers. (This is true for all but the smallest number of spammers.)

When a message arrives at a node, it will be forwarded with a probability that depends on whether the

upstream neighbor is good (probability p) or is a spammer (probability q), under the assumption that all good nodes have the same probability p , and all spammers have the same probability q .

Let B denote the number of spammers on a path of length x , then the probability that $B = i$ is given by the binomial distribution:

$$\Pr(B = i) = \binom{x}{i} b^i (1 - b)^{x-i} \quad (12)$$

Now, given $B = i$, the probability that a message reaches the destination is

$$\Pr(\text{success} | x) \stackrel{\text{def}}{=} p^{x-i} q^i \quad (13)$$

So, the unconditional probability that the message reaches the destination is

$$\Pr(\text{success} | x) \stackrel{\text{def}}{=} \sum_{i=0}^x \binom{x}{i} b^i (1 - b)^{x-i} p^{x-i} q^i \quad (14)$$

But this expression can be rewritten as

$$\sum_{i=0}^x \binom{x}{i} (bq)^i [(1 - b)p]^{x-i} = \quad (15)$$

$$\sum_{i=0}^x \binom{x}{i} (bq)^i (1 - bq)^{x-i} \left[\frac{(1 - b)p}{1 - bq} \right]^{x-i} = \quad (16)$$

$$\left[\frac{(1 - b)p}{1 - bq} \right]^x \sum_{i=0}^x \binom{x}{i} (bq)^i (1 - bq)^{x-i} \left[\frac{1 - bq}{(1 - b)p} \right]^i \quad (17)$$

The sum on the right-hand side is the generating function of a binomial distribution with x trials and success probability bq , evaluated at $j = (1 - bq)/[(1 - b) \cdot p]$. The generating function is $g(j) = (1 - bq + bqj)^x$. So, the probability of reaching the destination is

$$\Pr(\text{success} | x) \stackrel{\text{def}}{=} [(1 - b)p + bq]^x \quad (18)$$

4.2.3 Probability a spammer is chosen

A spammer can be chosen as the next hop in one of two ways: either all of u 's neighbors are spammers, or the paths that avoid a spammer neighbor are too long. Let *stuck* be the event that all of u 's neighbors are spammers, and *too long* be the event that the paths that avoid a spammer neighbor are too long. Then

$$b = \Pr(\text{stuck}) + (1 - \Pr(\text{stuck})) \cdot \Pr(\text{too long}) \quad (19)$$

The probability that all u 's neighbors are spammers depends on the degree distribution of the graph. We are working in a small-world graph with a power-law degree distribution whose maximum degree is

$$d_{max} = 2 \lfloor \sqrt{N} \rfloor \quad (20)$$

The maximum degree is a result of the way in which our graphs were grown [4]. (Such a limit on the maximum degree is common in social networks [41].) Thus, the

probability that any node has degree i is:

$$d(i) \stackrel{\text{def}}{=} \frac{i^{-1}}{\sum_{2 \leq j \leq d_{max}} j^{-1}} \quad (21)$$

The average probability that all neighbors are spammers over all nodes can be computed by summing the probability of the degree occurring and the probability all the neighbors are spammers. We assume the probability of a single neighbor being a spammer is dependent only on k and N , which is valid as we assume the spammers are uniformly distributed throughout the network. That is,

$$\Pr(stuck) \stackrel{\text{def}}{=} \sum_{2 \leq i \leq d_{max}} d(i) \frac{\binom{k}{i}}{\binom{N}{i}} \quad (22)$$

Now we wish to compute the probability that the path through a good node is too long, that is, calculate $\Pr(too\ long)$. This occurs when the spammer occurs on the shortest path ($\delta(v', t) < \delta(v, t)$) and the difference in path lengths is greater than $\delta_{max}(v, t)$.

Given a path of length x , the probability that at least one spammer is on that path is 1 minus the probability of choosing all good nodes on a path of length x :

$$\Pr(spammer\ on\ SP) \stackrel{\text{def}}{=} 1 - \frac{\binom{N-k}{x}}{\binom{N}{x}} \quad (23)$$

In order to determine the probability that the difference in path length is greater than $\delta_{max}(v, t)$, let $\ell'(x)$ be the distribution of increases in path length due to avoiding spammers, and let Δ be a random variable from that distribution. Then

$$\begin{aligned} \Pr(spammer\ chosen) &\stackrel{\text{def}}{=} \\ \Pr(\delta(v, t) + \Delta > \delta_{max}(v, t)) &= \\ 1 - \Pr(\Delta \geq \delta_{max}(v, t) - \delta(v', t)) &= \\ &= 1 - \text{CDF}(\Delta) \end{aligned} \quad (24)$$

In order to determine the CDF of Δ , we grew a number of graphs in the same manner that we did for our simulation. We then removed arbitrary nodes, and calculated the mean change in path lengths between pairs of nodes. The resulting change in path length was exponentially distributed with parameter λ' . Then

$$\Pr(spammer\ chosen) = e^{-\lambda'[\delta_{max}(v, t) - \delta(v', t)]} \quad (25)$$

We may now define $\Pr(too\ long)$:

$$\Pr(too\ long) \stackrel{\text{def}}{=} \Pr(spammer\ on\ SP) \cdot \Pr(spammer\ chosen) \quad (26)$$

Given $\Pr(stuck)$ and $\Pr(too\ long)$, we have now calculated b , the probability that a message will succeed for a path of length x . We are finally armed to calculate the probability that a message will succeed.

4.2.4 Probability a message succeeds

Finally, we derive the probability that a message from a given node will reach its destination. Let $\sigma_0(s)$ be the

probability that a node s 's message will be forwarded by its first neighbor—this allows us to distinguish the probability of success for good nodes and spammers independently. Then

$$\Pr(success|\sigma_0(s)) \stackrel{\text{def}}{=} \sigma_0(s) \sum_{1 \leq x \leq \lfloor \log(N) \rfloor} \ell(x) \cdot \Pr(success|x) \quad (27)$$

where $\ell(x)$ is the probability that a path of length x is created by our routing algorithm. In order to estimate $\ell(x)$, we grew a number of graphs and calculated the distribution of path lengths in the graph. The path lengths were approximately a Beta(2,5) distribution, with the most probable path length being 2. However, our system always allows communication between neighboring nodes, so we are only interested in paths of length ≥ 2 . We therefore approximated $\ell(x)$ as an exponential distribution with parameter λ .

Equation (27) allows us to estimate the probability that a message will reach its intended recipient for both good nodes and spammers. We compare this to our simulation results in Section 5.

5. EVALUATION

We ran simulations of our system that create a random graph with 100 nodes² according to the Holme-Kim model [18] with $m_0 = 5$ initial nodes, $m = 2$ minimum edges per node, and a probability of triad formation $p = .75$. We then chose k spam nodes at random and f freeloaders, also at random, where k and f are parameters to the simulation.

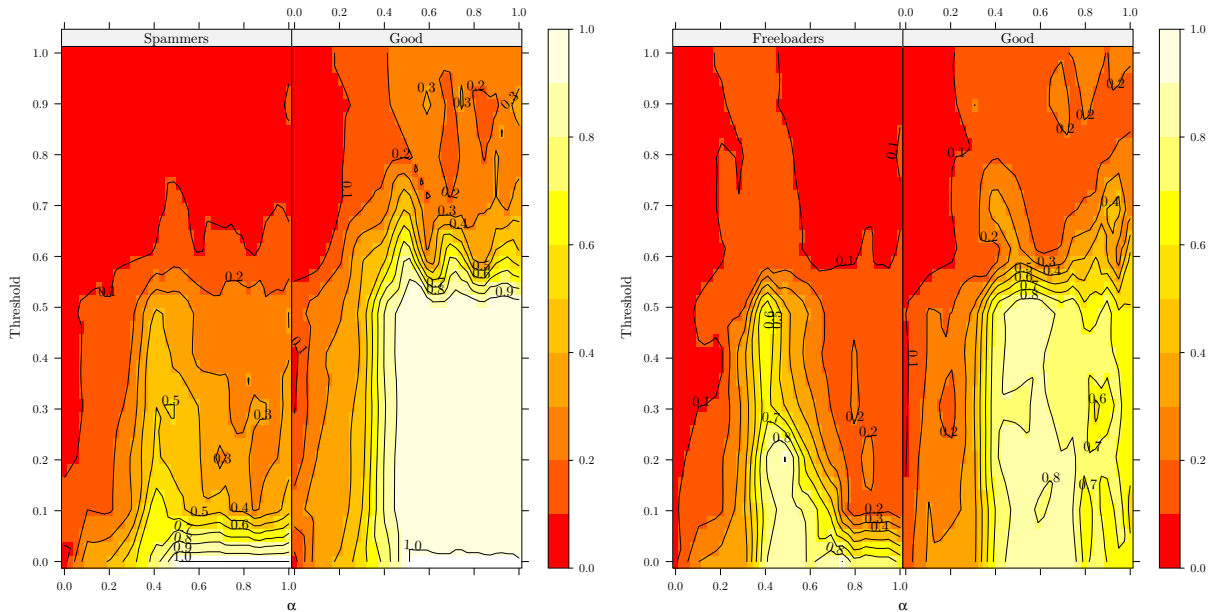
Each node in the network sent a constant number of messages per round of the simulation. Each recipient was chosen at random for every message. Finally, freeloaders send just as many (good) messages as ordinary nodes, but do not forward any messages.

5.1 Determining system parameter values

Our system's effectiveness depends on good choices of the parameters *Threshold* and α . We wish to choose values such that the probability of success for a good node is high, while the probability of success for malicious nodes (both spammers and freeloaders) is low. That is, we wish to minimize the difference between the two.

Figure 2(a) shows the probability of success for both spammers (the left half of the graph) and for good nodes (the right half of the graph) for varying values of *Threshold* and α when 20% of the nodes are spammers. Regions with a light color indicate a high probability of success, while darker regions indicate a low probability of success. The lower-right quadrant of both graphs, with the exception of when *Threshold* is very low, has the desired properties: good nodes have high probabilities of success, while spammers have lower probabilities

²We ran some simulations with 500 nodes as well, and their results were similar. We used smaller graphs for these results to reduce computation time.



(a) Probability of success for spammers and good nodes (b) Probability of success for freeloaders and good nodes

Figure 2: Effect of system parameters with 20% malicious nodes

of success. Note the sharp cliff for both spammers and good nodes at $\alpha \approx 3.5$: for values of α below this, the probability of success is very low. This corresponds to the point predicted by α_{min} in Section 4, below which routing is no longer guided by distance, and unlikely to succeed.

Figure 2(b) is a similar graph showing the probability of success for both freeloaders and good nodes when 20% of the nodes are freeloaders. Again, the lower-right quadrant of both graphs has the desired properties: good nodes have high probabilities of success, while freeloaders have lower probabilities of success. The cliff when $\alpha < \alpha_{min} \approx 3.5$ is still evident.

For the remainder of this paper we use $Threshold = .4$ and $\alpha = .6$, as these maximized the difference between the probability of success for good nodes and malicious nodes.

5.2 Convergence time

One of the key determining factors of a routing algorithm’s effectiveness is its convergence time. Figure 3(a) shows the probability of success for both good nodes and spammer in a network with 20% spammers compared to the number of messages sent. The good nodes enjoy almost immediate success, while the bad nodes have a probability of success below 50% after having sent 10 spams. Their probability of success stabilizes at around 30%. (We note that spammers have the highest probability of success when they comprise 20% of the network, as we shall show shortly.)

Figure 3(b) shows the probability of success for both

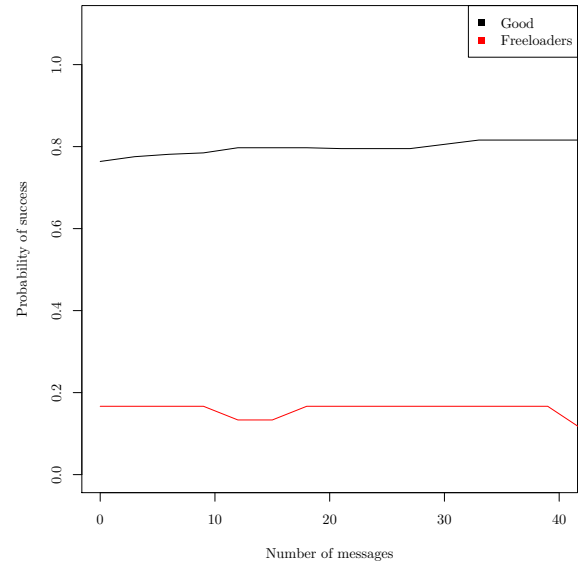
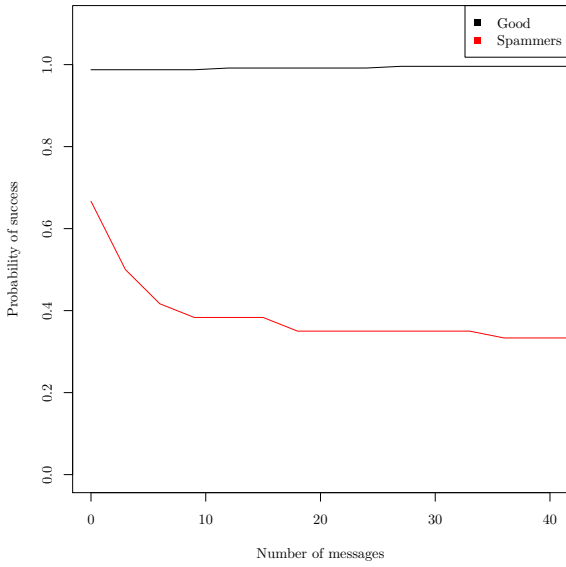
good nodes and spammers in a network with 20% freeloaders over time. In this simulation the good nodes do not fare as well, stabilizing around an 80% success rate. The freeloaders never have a success rate above 20%, however, though the variance in their success rate is higher. We revisit the low performance of our system with high numbers of freeloaders shortly.

5.3 Performance vs. spammers

Another measure of our algorithm’s success is how well it fares against large numbers of spammers. In order to evaluate this we simulated the system with varying percents of spammers. For each value of k (the number of spammers), we ran the simulation 10 times in order to generate 95% confidence intervals for the probability of success for both good nodes and spammers.

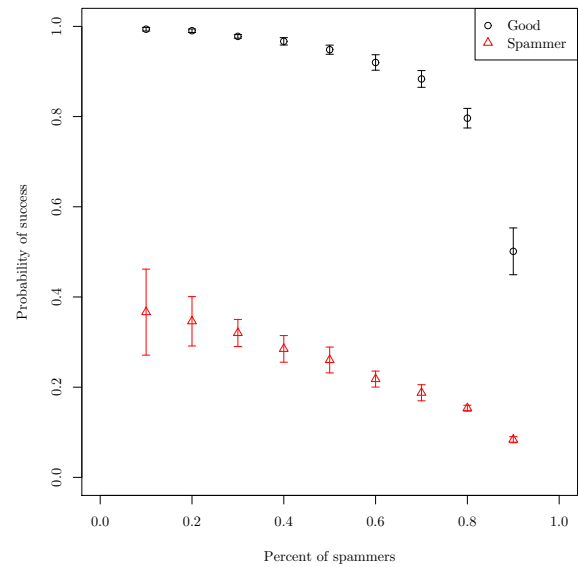
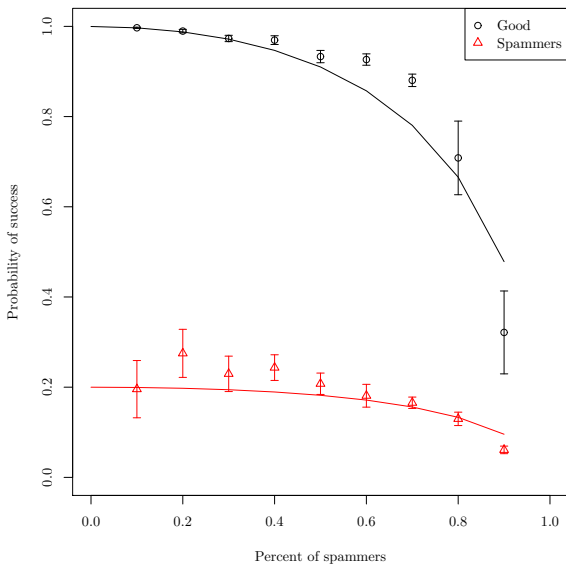
Figure 4(a) shows the results when both good messages and spams are identified with 100% accuracy. The graph shows the probability of success for both good and spam nodes compared to the percent of spammers in the network, where the probability of success for a good node is the probability that it sends a (good) message to its intended recipient, while the probability of success for a spammer is the probability that it sends a spam to its intended recipient. The solid lines indicate the probability of success predicted using our analysis from Section 4.

The system performs quite well with as many as 60% spammers in the network—the probability of success for good nodes remains above 90% in all such scenarios,



(a) Probability of success vs. messages sent with 20% spammers (b) Probability of success vs. messages sent with 20% freeloaders

Figure 3: Convergence



(a) Outcome is marked with 100% accuracy (b) Outcome is marked with 95% accuracy

Figure 4: Probability of success vs. percent spammers

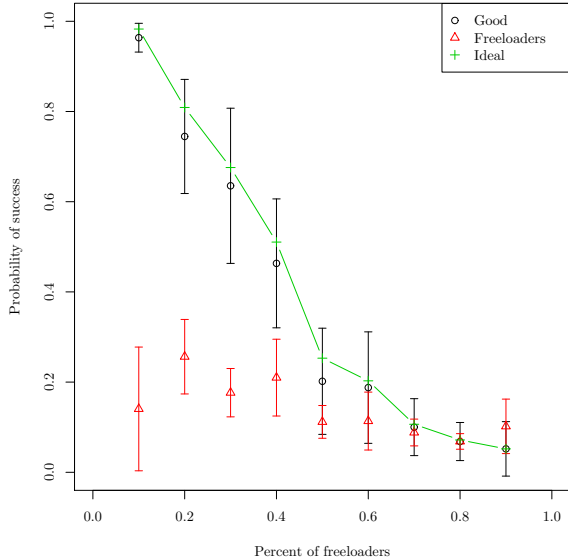


Figure 5: Probability of success vs. percent freeloaders

with the probability of success for spammers also decreasing as the number of spammers increases. Not until 70% of the network is a spammer does the probability of success for good nodes drop below 90%. One interesting data point is that the peak probability of success for spammers occurs when they comprise 20% of the network.

One question is whether populations of spammers this large are reasonable. Since botnets seem to provide the largest spammer population on the Internet, it is worthwhile to estimate the size of botnets. Estimating them has its difficulties. Rajab et al [32] pointed out that the largest estimates of botnets indicate they could occupy as many as 11% of the nodes on the Internet, but that such estimates are likely overestimating the true botnet population. Therefore we feel confident that our performance at a much larger percentage of spammers indicates the effectiveness of our system.

Figure 4(b) shows the results when the outcomes are only 95% accurate—5% of good messages are misidentified as spam, and 5% of spams are misidentified as good messages. It is apparent that good nodes’ probability of success is affected little—in fact, it improves slightly compared to perfectly accurate outcomes. The spammers’ probability of success also improves: when they comprise 10% of the network, their mean probability of success in sending a spam is around .35. This recognition rate is very low, as email classifiers routinely achieve a 99.9% accuracy rate [43], and we assume human beings judge the outcomes. (Our analysis does not account for recognition accuracy, so we do not show analytical results for this case.)

5.4 Performance vs. freeloaders

Another measure of our algorithm’s success is how well it fares against large numbers of freeloaders. Figure 5 shows the probability of success for both good and freeloading nodes compared to the percent of freeloaders in the network. Again we ran the simulation 10 times at each value of f (the number of freeloaders) in order to generate confidence intervals for the probability of success for both good nodes and freeloaders.

Unfortunately our system does not fare so well with large populations of freeloaders, as the probability of success for good nodes decreases quickly as the population of freeloaders increases. This is somewhat of a concern given Adar and Huberman’s estimate that 70% of the Gnutella network is a freeloader [1].

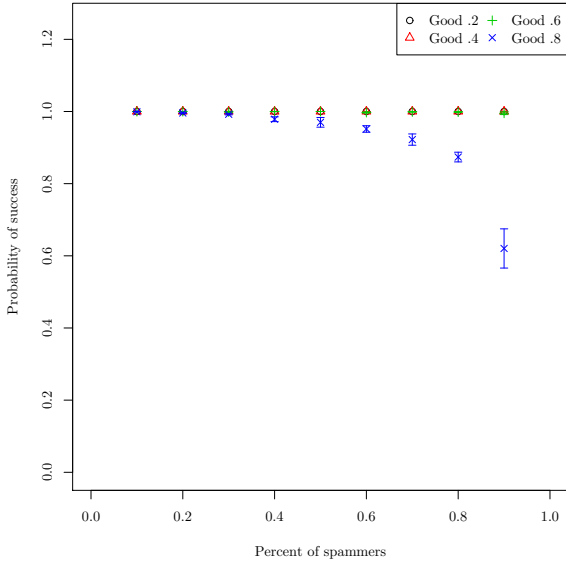
As Figure 5 shows, the freeloader’s probability of success is less than half that for a good node, at up to 50% of the network being a freeloader. Since the freeloader’s aim is to maintain his own probability of success, a node would have to act irrationally to be a freeloader. We claim therefore that such large populations of freeloaders are unlikely, since there is a strong disincentive (a greatly decreased probability of success) for freeloading.

We were also interested in how much the structure of the graph contributed to the probability of failure. A freeloader, which forwards no messages, is a node through which no messages can be routed. Thus a large number of freeloaders can disconnect the network. We calculated the probability that a path existed between all pairs of nodes after freeloaders were removed from the network, and plotted this probability as Ideal in Figure 5. It is ideal in the sense that two nodes can only communicate if a path between them exists. We see that the mean probability of success for good nodes is very near the mean probability that a path exists, and that therefore we route around freeloaders whenever possible. (We do not plot confidence intervals for the probability that a path exists in order to avoid cluttering the graph.)

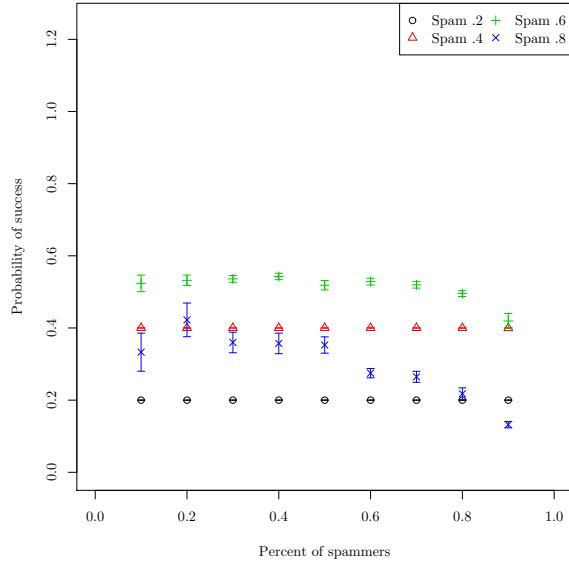
5.5 Performance vs. partial spammers

One attack specific to our system is that a spammer might vary the amount of spam it sends in a period of time. If the spammer can generate good messages as well as spam, it can improve its trust score, and have a higher probability of having spam accepted. While we do not believe a spammer will be able to generate arbitrary quantities of good messages, a spammer might be able to keep track of the number of good messages it forwards, and send a fraction of this as spam. We modeled this by specifying an additional parameter for spammers: β is the fraction of the messages a spammer sends per round that are spam. (Recall that we simulated constant quantities of messages per node.) When $\beta = 1$, a spammer sends only spam, in which case the results we have already presented apply.

Figure 6(a) shows the effect of such “partial spammers” with varying values of β on good nodes. The



(a) Probability of success of good nodes vs. partial spammers



(b) Probability of success of partial spammers

Figure 6: Probability of success vs. partial spammers

lowest probability of success for good nodes occurs when $\beta = .8$, and is similar to the results when $\beta = 1$. Decreasing values of β only improve the probability of success for good nodes.

Figure 6(b) shows the effect of varying β on the probability of success for a spammer. The highest probability of success occurs when β is $.6$ —precisely $1 - Threshold$. That is, a spammer has the best probability of success (sending a spam) when it sends approximately $Threshold$ of its total messages as good messages. The best a spammer manages in this case is approximately a 50% probability of success, no matter the number of spammers. The good nodes have no significant probability of failure in this circumstance.

6. DISCUSSION AND FUTURE WORK

We saw in our results that the highest probability of success for a spammer occurred when 20% of the network was a spammer, with approximately a 30% probability of success. If this point represents a “natural” fraction of spammers—that is, if the economic incentives for spam would tend to favor a system with the highest probability of success for spammers—it’s natural to determine how much spam would occur at this point. With a negligible probability that a good node’s message will be dropped, the total amount of spam received in this scenario is $< 7\%$.

On the other hand, spammers could improve their probability of success under our system by sending good messages as well as bad, and keeping their fraction of

good messages near $Threshold$ generated the highest level of spam in our system. Unlike when spammers sent only spam, there was no point at which the probability of success for a partial spammer was highest. Instead, their probability of success remained relatively constant (near 50%) until they comprised a very large portion of the network. (The good nodes’ probability of success in this case remains 100%.) If 90% of the network is a partial spammer, each of which sends 60% of its messages as spam, the probability of success in sending a spam is approximately 50%. In this case the total amount of spam received by a good node is about 52%. Getting to this level requires a very large population of spammers indeed, each capable of generating arbitrary quantities of good messages. Thus, even with an extraordinarily powerful attacker, the level of spam in our system does not approach that seen on the Internet.

We did not discuss sudden changes in the trustworthiness of a node. If a node had acted trustworthily for a long period of time, it would take a long time for neighbors to learn its new behavior and begin shunning it. Adding an exponential decay to our trust functions would address this.

We assumed that attacking nodes do not collude. An attack by a set of colluding nodes could be quite damaging to our system: the nodes could send random messages to one another, each of which they rate highly. Doing so would artificially raise their trustworthiness, and the intermediate nodes would wish to forward their

messages. Such nodes would then have either an increased ability to send spam along the paths between them, or be able to avoid forwarding any legitimate messages from nodes along that path with decreased penalty. This is partially the consequence of our system that provides partial anonymity: if we maintained trust values for every node (and removed anonymity) such an attack could be mitigated. We do not believe that we need to remove anonymity in order to mitigate the attack, however. We note first that in order to carry out such an attack, the attacking nodes would have to connect at different points in the network. This is similar to (but distinct from) a Sybil attack [13], in which an attacker creates many identities, each of whom rate him highly. The distinction is that, in a Sybil attack, the attacker does not control where he connects to the network. Such an attack would not succeed in our system, because while the attacking node and his friends would rate each other highly, that would not impact their ability to send messages to anyone else, or block messages from anyone else. We note that successfully carrying out an attack with colluding nodes thus depends on the ability of the attacker to gain friends. We also note that the attack depends on the ability to send unlimited quantities of messages along a path, and that simply limiting the rate at which messages can be sent could limit the impact of the attack. We leave improving MessageReaper to combat colluding attackers for future work.

Finally, we assume that the parameters *Threshold* and α are fixed for the entire system, but they need not be so. We intend to allow nodes to choose these independently, and to allow spammers to vary the amount of spam they send, in order to investigate our system with nodes with a variety of competing aims.

7. RELATED WORK

MessageReaper utilizes trust structures to handle two classes of attack: spam and freeloaders. Both areas have received a large amount of attention; therefore, we highlight a small number of works in each area.

Email spam (and spam in general) has been extensively studied. Many algorithms operate at the application layer, filtering spam once it reaches the destination, either the Internet Service Provider (ISP)’s or the user’s machine. The most common of these filters are white/black-lists [17], filters applying Bayesian analysis [28], and rule-based filters [37]. With whitelists, mail from users on the whitelist are allowed passage into the inbox, while the remaining users’ mails are marked as spam. Blacklists operate oppositely, i.e. they filter the “definitely bad”. One interesting take on white/black-lists utilizes the communication network of the message [5]: Boykin and Roychowdhury noticed that spam messages tended to have a low clustering coefficient while hams tended to have a high clustering coefficient. By marking those messages whose clustering coefficient is below a threshold, about half of all spams were elimi-

nated, with no effect on hams. Recently, Kong et al [22] showed how one could build a global digest system utilizing a percolation search in small-world networks and reputation to determine a messages spam-state (“definitely spam”, “definitely not spam”, or “unknown”). Golbeck and Hendler [17] introduced a system wherein a trust value of the sender could be determined by executing a breadth-first search of the graph using the min composition function; once this trust value was obtained the message could be marked correctly with high probability.

Freeloaders are a large problem in P2P networks, and a number of techniques for mitigating the problem have been investigated. The foremost mechanism is incentives [8, 9] wherein nodes must provide service to other nodes in the network before they are able to utilize resources. Similar in concept to incentive schemes is credit systems. In credit systems nodes take risks offering credit to neighbors in return for future rewards. For example, in Scrivener [30] nodes give credit to debtors until a limit is reached while the creditors continue to make requests on the debtors in order to mitigate the debt. TrustDavis [12] has the creditors assume some liability for a transaction. If the transaction is bad, the creditors must provide services to the requestor to pay off the liability. Many papers [15, 26, 16] have examined the effectiveness of incentives under a game-theoretic model. These models are generally based upon the generalized Prisoner’s Dilemma, and have the requirement that mutual cooperation has the highest payoff, and nodes do not forget credit. Under these assumptions mutual cooperation quickly dominates as it results in the highest utility for individual nodes.

When utilizing trust systems one has to decide between a system to achieve a global trust value for every pair of nodes, or a local trust system giving a trust value only for neighbors. Both have been extensively studied. One example of computing a global trust state is EigenTrust [19]. EigenTrust is developed so that malicious trust values can be discovered and ignored; this is accomplished via repeating the computation of a single node’s trust value on a set of random nodes and taking the majority value. An algorithm operating with local trust values and multi-hop message propagation was presented in [34]. This protocol has nodes issue flooded queries. Each time the query reaches its destination, the destination replies to the neighbor from which it heard the query, and the neighbor appends its trust value for the destination. This reply and reputation are propagated back to the source, which decides whether to request the data. Once the exchange is completed an outcome is flooded out to one hop beyond the destination, informing all neighbors of the destination of the increase in the trust value.

Finally, the formalization of trust structures is another abundant research topic [39, 6, 42, 23, 24]. These papers define a global trust state (which may or may not be explicitly known) wherein every pair of nodes

has a trust value. These trust values, in turn, have as many as two complete partial orders: one for trust-worthiness, another for information content. Utilizing these structures one is able to compare neighbors with more complicated ratings than a scalar value allows, e.g. \mathbb{N}^2 (which MessageReaper borrows from [23]). Furthermore, these assume a function mapping a trust structure to an accept or reject decision, e.g. $\sigma : \mathbb{N}^2 \rightarrow \{\top, \perp\}$. Together these form a complete system for reasoning about a node's reputation and predicting if a future outcome will be beneficial.

8. CONCLUSION

This paper introduced a novel, economical, anonymous networking protocol combining routing with trust to dramatically reduce the effect of spammers and freeloaders. To encourage cooperation, strong incentives were used reducing the probability of success for any excessive misbehavior. MessageReaper provided any-to-any communication over a social network, with fast stabilization, low false negatives and false positives even with a large portion of the network misbehaving. This protocol is applicable to both centralized and distributed communication settings. The system provided little incentive for spammers to mask their behavior by sending only a fraction of their messages as spam, and required only the majority of the outcomes to be correctly identified. We provided analysis of the probability of success under varying numbers of attackers, and did extensive simulations to measure the effect. Our analysis gave good correlation with our simulated results. This solution can be combined with existing spam blocking architectures to further reduce the number of unwanted messages. MessageReaper can potentially make significant impacts to our future communication systems/applications.

9. REFERENCES

- [1] E. Adar and B. A. Huberman. Free riding on gnutella. *First Monday*, Sept. 2000.
- [2] R. Albert, H. Jeong, and A.-L. Barabasi. The diameter of the world wide web. *Nature*, 401:130, 1999.
- [3] L. Banks, S. Ye, Y. Huang, and S. F. Wu. Davis social links: Integrating social networks with internet routing. *ACM SIGCOMM 2007 Workshop on Large-Scale Attack Defense (LSAD)*, August 2007.
- [4] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.
- [5] P. O. Boykin and V. Roychowdhury. Personal email networks: An effective anti-spam tool. *IEEE COMPUTER*, 38:61, 2004.
- [6] M. Carbone, M. Nielsen, and V. Sassone. A calculus for trust management.
- [7] B. Cohen. Incentives build robustness in BitTorrent, 2003.
- [8] B. Cohen. Incentives build robustness in bittorrent, 2003.
- [9] D. DeFigueiredo, B. Venkatachalam, and S. F. Wu. Bounds on the performance of p2p networks using tit-for-tat strategies. In *P2P '07: Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing (P2P 2007)*, pages 11–18, Washington, DC, USA, 2007. IEEE Computer Society.
- [10] Digital Inspiration. Spam inside Orkut Scrapbook. <http://www.labnol.org/internet/favorites/orkut-scrapbook-spam/1971/>.
- [11] R. Dingledine, N. Mathewson, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.
- [12] D. do B. DeFigueiredo and E. T. Barr. TrustDavis: A non-exploitable online reputation system. In *CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC'05)*, pages 274–283, Washington, DC, USA, 00 2005. IEEE Computer Society.
- [13] J. R. Douceur. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.
- [14] H. Ebel, L.-I. Mielsch, and S. Bornholdt. Scale-free topology of e-mail networks. *Physical Review E*, 66:035103, 2002.
- [15] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 102–111, New York, NY, USA, 00 2004. ACM.
- [16] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 102–111, New York, NY, USA, 00 2004. ACM.
- [17] J. Golbeck and J. Hendler. Reputation network analysis for email filtering, 2004.
- [18] P. Holme and B. J. Kim. Growing scale-free networks with tunable clustering. *Phys. Rev. E*, 65(2):026107, Jan 2002.
- [19] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM.
- [20] P. Kolari. Sploggers compromising MySpace. <http://ebiquity.umbc.edu/blogger/2006/11/27/sploggers-compromising-myspace/>.
- [21] P. Kolari, T. Finin, A. Java, and A. Joshi. Spam in Blogs and Social Media, Tutorial . In *ICWSM*

- 2007, March 2007.
- [22] J. S. Kong, P. O. Boykin, B. A. Rezaei, N. Sarshar, and V. P. Roychowdhury. Let your cyberalter ego share information and manage spam, 00 2005.
- [23] K. Krukow. *Towards a Theory of Trust for the Global Ubiquitous Computer*. PhD thesis, University of Aarhus.
- [24] K. Krukow and A. Twigg. Distributed approximation of fixed-points in trust structures. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 805–814, Washington, DC, USA, 00 2005. IEEE Computer Society.
- [25] J. Liang, R. Kumar, Y. Xi, and K. W. Ross. Pollution in P2P file sharing systems. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2:1174–1185 vol. 2, 13-17 March 2005.
- [26] R. T. B. Ma, S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau. A game theoretic approach to provide incentive and service differentiation in p2p networks. In *SIGMETRICS '04/Performance '04: Proceedings of the joint international conference on Measurement and modeling of computer systems*, pages 189–198, New York, NY, USA, 00 2004. ACM.
- [27] McAfee. Orkut spam worm spotted! <http://www.avertlabs.com/research/blog/index.php/2007/12/19/orkut-spam-worm-spotted/>.
- [28] T. A. Meyer and B. Whateley. Spambayes: Effective open-source, bayesian based, email classification system. In *CEAS*, 2004.
- [29] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42, New York, NY, USA, 2007. ACM.
- [30] A. Nandi, T.-W. Ngan, A. Singh, P. Druschel, and D. S. Wallach. Scrivener: Providing incentives in cooperative content distribution systems. In *Middleware*, pages 270–291, 00 2005.
- [31] Y. Niu, Y.-M. Wang, H. Chen, M. Ma, and F. Hsu. A quantitative study of forum spamming using context-based analysis. In *Proceedings of the 14th Annual Network and Distributed System Security Symposium (NDSS)*, pages 79–92, San Diego, CA, Feb. 2007.
- [32] M. A. Rajab, J. Zarfoss, F. Monroe, and A. Terzis. My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 5–5, Berkeley, CA, USA, 2007. USENIX Association.
- [33] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. *SIGCOMM Comput. Commun. Rev.*, 36(4):291–302, 2006.
- [34] T. Repantis and V. Kalogeraki. Decentralized trust management for ad-hoc peer-to-peer networks. In *MPAC '06: Proceedings of the 4th international workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC 2006)*, page 6, New York, NY, USA, 00 2006. ACM.
- [35] D. Robson. How to spot false friends on Facebook. *The New Scientist*, 195(2613):28, July 2007.
- [36] samy. Technical explanation of the MySpace worm.
- [37] A. Schwartz. *SpamAssassin*. O'Reilly Media, Inc., 2004.
- [38] J.-M. Seigneur, N. Dimmock, C. Bryce, and C. D. Jensen. Combating spam with TEA (trustworthy email addresses). In *Proceedings of the 2nd Conference on Privacy, Security and Trust*, 2004.
- [39] V. Shmatikov and C. Talcott. Reputation-based trust management. *J. Comput. Secur.*, 13(1):167–190, 00 2005.
- [40] Symantec. The State of Spam - A Monthly Report - January 2008. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/Symantec_Spam_Report_-_January_2008.pdf.
- [41] R. Toivonen, J.-P. Onnela, J. Saramki, J. Hyvnen, and K. Kaski. A model for social networks. *Physica A*, 371, Nov 2006.
- [42] W. Wagealla, M. Carbone, C. English, S. Terzis, and P. Nixon. A formal model of trust lifecycle management.
- [43] W. S. Yerazunis. The spam-filtering accuracy plateau at 99.9 percent accuracy and how to get past it. In *MIT Spam Conference 2004*, December 2004.