A Trust Overlay Architecture and Protocol for Enhanced Protection against Spam

Jimmy McGibney, Dmitri Botvich
Telecommunications Software & Systems Group, Waterford Institute of Technology
{jmcgibney, dbotvich}@tssg.org

Abstract

The effectiveness of current anti-spam systems is limited by the ability of spammers to adapt to new filtering techniques and the lack of incentive for mail domains to filter outgoing spam. This paper describes a new approach to spam protection based on distributed trust management. This is motivated by the fact that the SMTP mail infrastructure is managed in a distributed way by a community of mail domain administrators. A trust overlay architecture and a new protocol are presented. The TOPAS protocol specifies how experiences and recommendations are communicated between a spam filter at each mail domain and its associated trust manager, and between trust managers of different mail servers. A technique for improving mail filtering using these trust measures is also described. Initial simulations indicate the potential of this approach to improve rates of false positives and false negatives in anti-spam systems.

1. Introduction

As well as being annoying, spam introduces many serious security risks and is often used to conduct fraud, as a conduit for malicious software and to carry out denial of service attacks on mail servers.

The principal current anti-spam techniques focus on filtering incoming email, either based on parsing message content, Bayesian filtering, maintaining Domain Name System (DNS) blocklists, and/or the use of collaborative filtering databases. Spammers tend to be resourceful though, and quickly find ways to get around most countermeasures. There are various attempts to also make anti-spam more adaptive, with the availability of reporting services that allow spam filters and blocklists to keep up to date with new spam techniques. There is a feeling though that spammers are always a step ahead, with the anti-spam community following with countermeasures some time afterwards.

In addition to technical countermeasures, several governments have introduced legislation outlawing spam and providing for stiff penalties. Despite some prosecutions, such legislation has had little or no effect, for several reasons such as the inter-jurisdictional nature of email traffic and spam.

Internet email is transferred using the Simple Mail Transfer Protocol (SMTP) [1]. SMTP was first introduced [2] at a time when the total number of Internet hosts was just a few hundred and trust between these could be assumed, and was designed to be lightweight and open. Attempts to introduce authentication via extensions or with new protocols that can sit on top of SMTP (e.g. Pretty Good Privacy, PGP [3]) have proven useful in some situations but are very far from universal adoption.

In this paper, we explore the use of decentralised trust to provide a more robust spam filtering system. Trust is primarily a social concept and is personalised by the subject. In this paper, we make use of trust by taking the socially-inspired notion of trust based on experience and recommendation.

The peer-to-peer nature of the Internet mail infrastructure suggests that it should be well suited to a distributed approach to trust management.

This paper proposes an architecture and protocol for establishing and maintaining trust between mail servers. The architecture is effectively a closed loop control system that can be used to adaptively improve spam filtering. In this approach, mail servers dynamically record trust scores for other mail servers; trust by one mail server in another is influenced by direct experience of the server (i.e. based on mail relayed by that server) as well as recommendations issued by collaborating mail servers. As well as modelling trust interactions between mail servers, we explore how mail filtering can combine trust values with existing mail filtering techniques.

The remainder of this paper is organised as follows. The next section (section 2) outlines the reasons that spam is prevalent and the main anti-spam techniques. A trust overlay architecture is presented in section 3, followed by the new TOPAS protocol in section 4. Section 5 describes some algorithms for handling trust

Second International Conference on Availability, Reliability and Security (ARES'07)
0-7695-2775-2/07 $20.00 © 2007 IEEE

measures received with TOPAS. Section 6 describes how trust measures can be used to enhance spam filtering. Section 7 presents an illustrative example of the use of our approach. Finally, section 8 discusses related work and section 9 concludes the paper.

2. Anti-spam background

The Internet mail infrastructure, based on SMTP, has in general been very successful. Much of this success has been due to its simplicity. SMTP is a simple protocol wherein mail is transferred from a client system to a server system using a limited set of text commands. The focus of SMTP is on effective mail transport across heterogeneous systems, with the assumption that security or other desired features are provided at a higher layer or by means of extensions. The main security effect is in a lack of requirement for authentication of the source of an email or the path that it has taken.

The main anti-spam techniques in practical use are based on message content and DNS blocklists, and/or use of collaborative filtering databases. *SpamAssassin* [4], for example, processes each incoming mail and assigns a "score" to it based on a combination of values attributed to possible spam indicators. The higher the score, the more likely it is that the mail is spam. A threshold is then used to filter mail – a mail that scores below the threshold is accepted and one that scores above the threshold is flagged as spam. The alternative, or complementary, approach is blocklisting – mail coming from unreliable sources is simply blocked.

These techniques have significant limitations though. With content filtering using a threshold, some genuine mail messages may score above the threshold and be flagged as spam (false positives) and some spam may score below the threshold and be accepted (false negatives). Careful tuning of the threshold can optimise the balance between the incidence of false positives and false negatives. For example, many email users will tolerate a modest level of spam to reduce the risk of some important mails being blocked. In practice, spammers are quite resourceful and adapt to content filtering advances – by, for example, using images rather than text, mutating text to avoid keywords that cause high filter scores, or spoofing source address to make it more acceptable.

DNS blocklists are also problematic. These effectively identify spam sources in a binary fashion (i.e. a mail source is either on the list or it is not). This becomes problematic when "good" mail servers are attacked and exploited. Even with good system administration, this can happen and may not be noticed immediately.

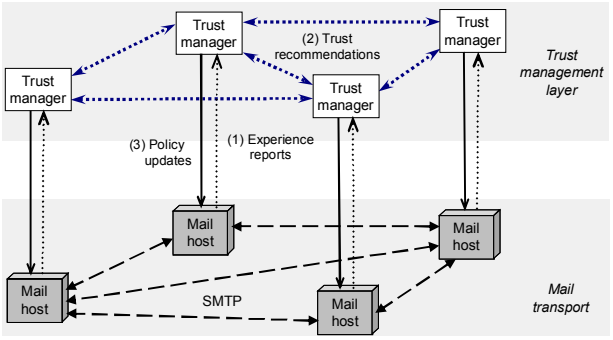

Figure 1. Mail infrastructure trust overlay

Though innovative new anti-spam techniques have been proposed, such as the use of micro-payments [5] and a requirement for the sender to solve a computational challenge for each message, spam remains a significant problem for users and system administrators.

3. Trust overlay architecture

In this section, we propose the overlay of a distributed trust management infrastructure on the mail infrastructure, with a view to using trust information at nodes to assist with spam filtering. Figure 1 illustrates the relationship between SMTP and this new infrastructure. Mail transport operates as normal – we do not propose any changes or extensions to SMTP or other existing mail protocols. With our proposed trust overlay architecture, a trust management layer operates separately from mail transport. Two message passing interfaces are defined between the mail transport layer and the trust management layer and another between the trust managers of individual nodes. The interfaces are as follows (Figure 1):

(1) Experience reports: Mail host → Trust manager
(2) Recommendation: Trust manager ↔ Trust manager
(3) Policy updates: Trust manager → Mail host

3.1. Data structures: representing trust between mail servers

We can assume, without loss of generality, that the Internet mail infrastructure is made up of a (large) number of SMTP-capable hosts. Each of these hosts can send and receive mail using SMTP. For the remainder of this discussion, we simply refer to these mail server hosts as *nodes*.

Trust is defined as being between two nodes – i.e. node A has a certain level of trust in node B. Each node records a set of trust-related parameters for each other node. In the general case, there could be many

Table 1. Representation of trust scores, from perspective of a specific node (smtp1.foo-inc.com)

Node name	Trust score	Confidence	Recency
smtp2.foo-inc.com	0.99	0.99	100
mail.barfoo.org	0.95	0.95	200
labserver.uxy.edu	0.80	0.80	700
webmailprov.com	0.50	0.50	30
lazyconfig.net	0.25	0.25	10
phishysite1342.com	0.01	0.01	8000

such parameters corresponding to a variety of services for which trust measures are used. In our experiments, trust in another mail server is represented by a score in the range (0,1) – each node then records this score for each other node of which it is aware. A trust score of 0 means that there is no trust in this node and a trust score of 1 means that this node is fully trusted. Some trust scores will be very stable and may result from plenty of experience and corroboration by several peers and thus there may be a high level of confidence in these scores. Other trust scores may be less reliable. Thus we also record a *confidence* level for each trust score, again in the range (0,1). A third parameter that we use is *recency*, indicating how recently this trust score was updated. Recency is important as mail servers come and go; good mail servers become bad (e.g. due to a hijack) and bad mail servers become good (e.g. due to removal of malware). Recency can be recorded in any appropriate units of time.

This can be implemented as a table that might look something like that shown in Table 1.

Nodes can control the size of this table by choosing to "forget" a trust score and deleting its entry. For scalability reasons, trust scores could just be recorded for nodes that have been encountered fairly recently.

3.2. Mail host identification

Trust scores are recorded per IP address (IPv4 or IPv6). Table 1 shows hostnames only for reasons of readability. A trust score thus is just meaningful when applied to mails from the IP address to which it is attached. As SMTP does not protect against modification of sending SMTP server hostname or IP address by intermediate mail relays, processing of trust values is always based on the last hop. Mail received from an SMTP relay affects the trust score of that SMTP relay rather than the originating host. This provides the relay with a strong incentive to filter outgoing spam. The IP address of the last hop is collected from the source IP of packet(s) containing the SMTP HELO/ELHO message (to which a reply will have been sent).

3.3. Modelling centralised trust references

The architecture described here is highly distributed, with each node autonomously managing its own view of the trustworthiness of other nodes. In practice, however, some nodes might prefer to defer to a centralised trust reference. Such a centralised trust reference can be modelled as a virtual node with a very high trust value (perhaps 1.0).

4. TOPAS protocol

This section describes a protocol that we call TOPAS (Trust Overlay Protocol for Anti Spam). The TOPAS protocol is for collecting spam statistics to calculate trust, sharing this trust information between nodes and feeding it back to mail hosts to allow them to more effectively filter spam.

The protocol executes using asynchronous message passing in the case of interfaces (1) and (2) of Figure 1. Interface (3) uses a simple synchronous request-reply technique. An underlying set of services is assumed, including message delivery, failure detection and timeout. It is assumed that each process receives queued messages of the form (`tag`, Arg_1, ..., Arg_n). This outline style of protocol specification is influenced by the Generic Aggregation Protocol (GAP, [6]).

(1) Experience report: Mail host → Trust manager

The mail host has an associated spam filter. Each item of mail that arrives at the server is processed by the spam filter, with the result that it is either accepted or flagged as spam. This information needs to be available to the trust manager. The following two messages, from the mail host to the trust manager, provide this:

- (`singleMail`, *i, s*) may be sent from the local mail host to the trust manager to report on a single mail from node *i*. The value of *s* is 1 if the mail has been determined by the mail host to be spam and 0 otherwise. *i* identifies the sending node.
- (`bulkMail`, *i*, *n*, *s*) may be sent from the local mail host to the trust manager to report on a sequence of mails from node *i*. *n* is the number of mails received from node *i* (since the last report) and *s* the number of these determined by the mail host to be spam.

Note that in both messages above, the node identifier *i* could be an IP address, a hostname, a mail domain name or any other identifier supported by the mail host. It is up to the trust manager to process this.

Sending a `bulkMail` message conveys the same information as would several `singleMail` messages.

It is included in the protocol for performance reasons. Where mail volumes are heavy, it is more efficient to send periodic `bulkMail` updates rather than burden the server with generating a `singleMail` message per mail received.

(2) Trust recommendation: Trust manager ↔ Trust manager

Nodes collaborate to share trust information with one another. This is done by the trust managers, using the primitives outlined here. Trust managers may issue recommendations spontaneously (for example on occurrence of some event like sudden appearance of spam) or in response to a request from another node. Note that a request may be issued by one trust manager to another, but a reply is not guaranteed. As mentioned earlier, message passing is asynchronous and the protocol requires no state information to be maintained by the corresponding entities. The following five messages may be sent from one trust manager to another:

- (`getTrust`, *k*) allows node *i* to ask another node *j* to report its trust in node *k*. This message should be interpreted as an indication from node *i* of a desire to receive a recommendation regarding node *k*. Node *j* may respond with a trust report. This might be expected to be used as follows. On receipt of some volume of mail from node *k*, node *i* could issue a series of `getTrust` messages to neighbouring nodes requesting recommendations regarding node *k*. Only those neighbours with some experience or knowledge of *k* might reply, with the remainder staying silent. Note though that nodes are not obliged to respond, even if they do have knowledge or experience of *k*. Nodes could have other reasons (e.g. lack of trust in the requester, *i*) to not reply. Nodes that have no information are expected to stay silent.
- (`getTrustResponseRequested`, *k*) is a variation on `getTrust` that expects the recipient to respond (with a null value) even if it has no trust information on node *k*. Again, there is no strict obligation to respond.
- (`getTrustAll`) allows one node *i* to ask another node *j* to report its trust in all known nodes. This message should be interpreted as an indication from node *i* of a desire to receive a recommendation regarding all nodes of which the recipient is aware. There is no obligation on the recipient to respond. Nodes that have no information on any nodes should issue a null reply.
- (`setTrustReportingPreferences`, *t, c, r, f*) allows node *j* to specify to node *i* how spontaneous trust reports are sent to it (see discussion on "push" model later). This message also indicates a desire by node *j* to receive trust advertisements from node *i* (i.e. to be included in its neighbourhood).
  t is a list of zero or more trust level thresholds. Trust updates are requested whenever trust exceeds or falls below any of these threshold values.
  c is a confidence level threshold. Trust updates are only desired if the confidence level of the sender is at least *c*.
  r is a recency threshold. Trust updates are only desired if the trust information has been updated by the sender within the previous *r* time units.
  f indicates the maximum frequency of update.
- (`trustReport`, *k, T*) allows node *j* to send a recommendation to another node *i*, in relation to node *k*. *T* is an object that encapsulates sender *j*'s trust in node *k*. *T* is set to null in the case where the sender *j* has no trust information regarding node *k*. Note that a `trustReport` may be issued either spontaneously or in response to a `getTrust` or `getTrustResponseRequested` message.
- (`bulkTrustReport`, *l*) allows node *j* to send a recommendation to another node *i*, in relation to a set of nodes. Parameter *l* is a set of pairs (*k, T*) where *k* is the node identifier and *T* is an object that encapsulates sender *j*'s trust in node *k*. *l* is the empty set in cases where the local node has no trust scores to share. Note that a `bulkTrustReport` may be issued either spontaneously or in response to a `getTrustAll` request.

(3) Policy update: Trust manager → Mail host

The third part of this collaboration architecture is responsible for closing the loop. Direct experience is recorded by nodes and shared among them. The result of this experience and collaboration is then used to inform the mail host to allow it to operate more effectively. Specifically, the mail host, on receiving a mail from node *i* needs to be able to access the current trust information that its trust manager has on node *i*. Although the mail host may be able to use local storage to, for example, cache trust values, we do not place any such requirements on it. Thus we need the ability for the mail host to request trust information from the trust manager and receive a timely reply.

- (`getTrustLocal`, *i*) allows the mail host to request the trust score for a particular node, *i*.
- (`trustReportLocal`, *i, T*) allows the trust manager to respond to a `getTrustLocal` request. *T* is an object that encapsulates the trust manager's trust in node *i*. *T* is set to null in the case where the trust manager has no trust information regarding node *i*.

5. Using the TOPAS protocol

The Internet mail architecture is highly dynamic – new nodes appear all the time, and existing nodes will quite frequently receive mail from previously unknown senders. In distributed trust management generally, choosing an initial value of trust to assign to such new arrivals is a non-trivial task. There are essentially two options, where the range of trust is (0,1):

(1) Initialise the trust level at zero (motivated by the *Sybil attack* [7]).

(2) Assume some "default trust" exists and initialise at some value above zero.

Treating previously unknown senders the same as spammers would restrict the utility of email as it tends to block new entrants; this points to the second option as perhaps the best.

Issuing inter-node trust advertisements

The functions specified at interface (2) above allow for either a "pull" or a "push" model for sharing trust information. Messaging is asynchronous for experience reports and recommendations.

"pull" model

The trust manager receives a `getTrust`, `getTrustResponseRequested`, or `getTrustAll` request for trust information about another node, or all nodes, and subsequently replies with a `trustReport` or `bulkTrustTeport` message. The sender of the request will need to use its own timeout mechanism. In the case of a `getTrust` message, there is no obligation on the recipient to reply at all. With `getTrustResponseRequested` and `getTrustAll`, the recipient should issue a reply even if this contains no trust information.

"push" model

The trust manager may be configured to spontaneously issue trust updates, either to other nodes or to its local mail server. This is typically for reasons of performance and efficiency. It is wasteful for nodes to repeatedly poll each other unless there is useful new information. In the "push" case, each node decides when and to whom to issue trust advertisements.

When to issue trust advertisements?

Each individual node controls the frequency of issuance of trust advertisements, and may even decide to issue none. Issuing a trust advertisement uses processing, memory and network resources of both the sender and recipient. The sender needs to find a balance – the more collaboration the more effective the system, but sending too many updates may put a strain on resources. It is also desirable if the recipient can control the number or frequency of advertisements from the sender. A sender trust advertisement strategy might have them sent periodically or on significant change in trust value or confidence level in this value. For example, the sender may issue trust advertisements relating to a node when its trust in that node exceeds a certain threshold *and* when its confidence level in this trust is high. If the trust level exceeds the threshold but its confidence level in this value is low, it may choose not to send anything.

The TOPAS protocol allows this to be based on the recipient's specified trust reporting preferences.

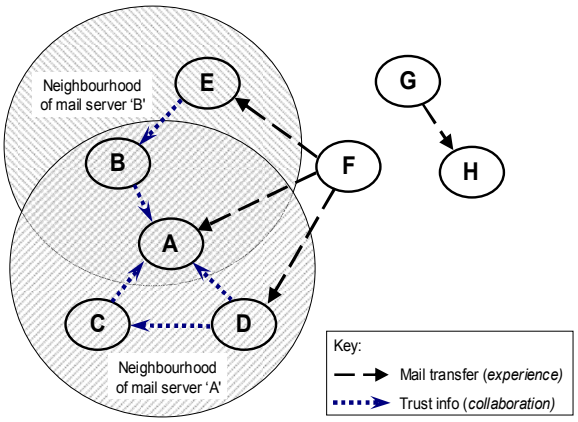

Figure 2. Distribution of trust information

To whom to issue trust advertisements?

Each individual node also controls the set of nodes that forms its "neighbourhood" – those nodes to which it sends trust advertisements. The set of neighbours could be defined, for example, as containing nodes that are nearby, most trusted, most collaborative, and/or have specifically requested trust reports (using `setTrustReportingPreferences`),

Figure 2 illustrates the distribution of trust information. There are eight nodes (mail servers) in the example shown. Each node has its own neighbourhood, two of which are shown. In some short time duration, node 'F' sends mail to 'A', 'D' and 'E'. Node 'A' establishes a trust score for node 'F' based on (i) mail it receives directly from 'F' and (ii) reputation information from its neighbours, 'B', 'C' and 'D'. The reputation information provided by node 'B' relates the experience of node 'E' that 'E' has shared with it. The reputation information provided by node 'C' derives its trust score relates the experience of node 'D' that 'D' has shared with it. Note that trust transitivity may allow a trust score to propagate quite some distance – in this example, the score that 'E' records for 'F' is propagated to 'B' and onward to 'A'.

Handling experience reports

On receipt of a `singleMail` message from the mail host relating to node *j*, the trust manager at node *i* updates its trust value in node *j*, $T_{i,j}$, as follows:

$$T_{i,j} \leftarrow f_e(T_{i,j}, S),$$

where f_e is a function defining how trust is updated. Further work is required to recommend suitable algorithms for updating trust on experience. In our initial simulations, we use an exponential average function.

Handling recommendations

On receipt of a `trustReport` from node *j*, indicating a level of trust in node *k*, the trust manager at node *i* updates its trust value in node *k* as follows:

$$T_{i,k} \leftarrow f_r(T_{i,k}, T_{i,j}, T_{j,k}),$$

where f_r is a function defining how trust is updated. Further work is required to find suitable algorithms for updating trust on recommendation. In our initial simulations, we use an exponential average function.

6. Using trust scores to filter email

Assume that a spam filtering system applies some test to each incoming email. In each case, a decision is made whether to accept the mail. A negative result (in the spam test) means that the mail is accepted and a positive result means that the mail is rejected (or at least marked as spam).

Most spam filters combine a variety of measures into a suspicion score and compare this with a *pre-defined* threshold. This threshold is a fixed value that may be tuned manually. Mail resulting in a score above the threshold is marked as spam and the remainder (under the threshold) is accepted.

In our system, we attempt to improve spam filtering by allowing the threshold to vary. The threshold level depends on the trustworthiness of the node that sent the message. So, we use the sender's trust score (as perceived by the receiver) to define the threshold – i.e. the more trusted a node is, the higher we set the threshold for marking a new mail message as spam. Conversely, if a node is untrusted then the threshold is set to a lower value. There are several ways to cause this threshold to vary. In our initial experiments, the threshold for mail received from a server is simply a linear function of the trust score of that server (as the mean of the trust score range is 0.5 and the default threshold for *SpamAssassin* is 5, we set the threshold to be simply ten times the trust score).

In practice, the dynamics of trust applied to spam filtering allows an organisational mail server to process email in a way that depends on its trust in the sending node. In many cases, this trust level will be somewhere in the middle, between "trusted" and "untrusted".

In the current (fixed threshold) situation, consider where the spam filter threshold is 5.0. This is the setting on the authors' *SpamAssassin* implementation. All incoming mail is given a spam score. If a genuine messages scores 5.1, it is diverted to the spam box. If a spam message scores 4.9, it is allowed into the user's inbox.

If we instead have a dynamically tuned threshold, however, a genuine message from a trusted source scoring 5.1 will be accepted as the source should have a higher threshold. Likewise, spam received from an untrusted server scoring 4.9 will be filtered out as the threshold should be lower.

7. Illustrative example

This section reports on an illustrative implementation of the TOPAS protocol with a simulated network of mail servers generating traffic, some of which is spam. We show how improvements in spam filtering (in terms of reduced false positive and false negative rates) can be achieved through closed loop control based on sharing trust scores.

7.1. Simulation set up

In our simulations, each node has a *neighbourhood* defined. This is a set of nodes that are somehow "close" to the node in question, with the expectation of above average frequency of communication with them.

Neighbourhoods are defined randomly for each mail server. This is done as follows: Initially, the neighbourhood of each node is the empty set. Choose two nodes at random. Add one to the neighbourhood of the other. Repeat until the total set of neighbour relations is equivalent to a connected graph. Then trust will (eventually) propagate throughout the network.

Email traffic volumes also vary randomly, with spammers tending to produce email in greater quantities than regular email users. For each mail sent, the recipient can be anywhere on the network, but is more likely to be a neighbour than any random node.

Each email contains a value S' that models aggregated indicators of spam, in the style of *SpamAssassin*. This value is used by the receiving node to test for spam.

S' has a Gaussian (normal) probability density function with mean μ and standard deviation σ. Mail that is actually spam tends to have a high value of μ. Normal mail tends to have a lower value of μ. The standard deviation determines the tendency for the filtering system that analyses such mail to be prone to false positives and false negatives.

For the purposes of our experiments, whether or not an email is actually spam is indicated by a binary value that is communicated separately to the receiver. This is not used in spam detection, but is used afterwards in the evaluation of how well the spam filter worked.

7.2. Using Trust to Enhance Mail filtering

In our experiments, we examine two different approaches to mail filtering:

- Use of a pre-defined threshold. The value of the threshold is selected to be half way between the means of the probability density functions of S' for spam and non-spam respectively. Our experiments are designed so that this mean is 5.0 (a common *SpamAssassin* threshold).
- Use of an automatic threshold that is directly related to trust in the sending node.

The first approach takes no account of trust information. The second uses trust information to tune the spam filter.

We now illustrate how improvements in spam filtering (in terms of reduced false positive and false negative rates) can be achieved through closed loop control based on sharing trust scores. We simulate a network of fifty nodes, of which a single one is a spammer. The spammer is responsible for 50% of all email generated in the system. Trust convergence for "good" nodes is based on exponential averaging, with a parameter of 0.03 (chosen experimentally for stable but moderately fast convergence). The randomly generated neighbourhood of each node consists on average of one-seventh of all nodes.

For this experiment, we choose relatively flat (but distinct) probability density functions for spam indicators for both spam and non-spam email. Both have the Gaussian (normal) distributions shown in Table 2. Note the overlap implied by the relatively large standard deviation values.

Table 2. Parameters for Gaussian (normal) distributions used in experiments

	Mean	Std Deviation
Spam	8.0	4.0
Non-spam	2.0	4.0

As already mentioned, most spam filters combine a variety of measures into a suspicion score and compare this score with a pre-defined threshold. For our experiments, a fixed threshold of 5.0 is chosen (*SpamAssassin* default) and used as a benchmark. As can be seen in Figures 3 and 4, a significant reduction in both false positives and false negatives can be achieved with auto-tuning of the threshold (based on trust values). Auto-tuning is of course most effective in a steady-state situation when trust values are quite stable. A range of other predefined threshold values were also tried, but with no better results than the value of 5.0 shown. Choosing a higher predefined threshold causes an increase in false negatives and choosing a lower predefined threshold causes an increase in false positives.

8. Related work

Specific spam filtering techniques are not of direct concern to us in this paper and can effectively be plugged in as needed. Our emphasis is on using trust information to tune such filters, and our initial implementation focuses on filters that are based on thresholds. This section overviews other work that uses collaborative techniques to fight spam, and points out similarities and differences in approach to ours.

There has been some other work on applying trust and reputation information to spam filtering.

Golbeck and Hendler [8] present a technique based

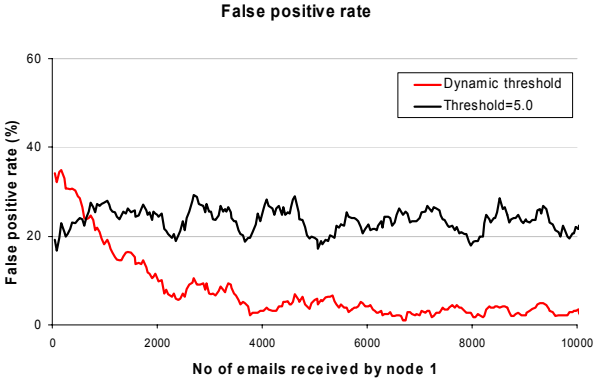

Figure 3. Comparison of dynamic vs fixed threshold: impact on rate of false positives.

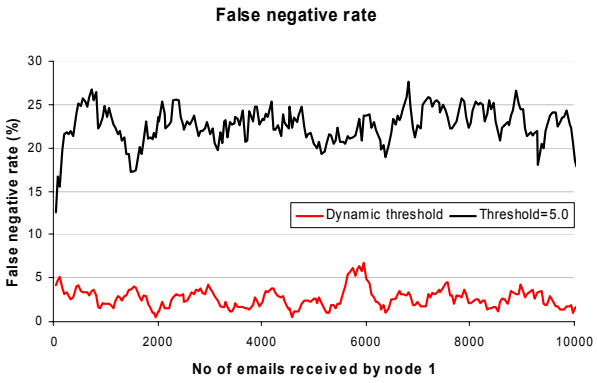

Figure 4. Comparison of dynamic vs fixed threshold: impact on rate of false negatives.

on social networks for sharing reputation information among email users. This allows users to sort received messages based on a sender rating value. This rating value is set by the user or is inferred from neighbours' ratings using a weighted average technique. Our approach differs from this in two main ways: firstly, we focus on mail servers rather than individual email addresses; secondly, our system provides closed loop feedback without direct user involvement.

Kong et al [9] also focus on end user email addresses, but provide for the anonymous sharing of information with a wider community. When a user flags a mail as spam, this is made available to other users' spam filters, which is useful as the same spam messages are usually sent to a large number of users. Damiani et al [10] similarly present a system of sharing spam information, wherein a cryptographic digest of each spam mail encountered is shared.

Seigneur et al [11] use "hard" cryptographic authentication to support whitelists of known good guys in conjunction with "soft" trust management for new contacts. Trust derives from stored *evidence*, which includes recommendations, observations, certificates and reputations. Though our trust model is similar to this ("observations" are called "experience" in our model, and we consider certificates and reputations to be the same as recommendations), our work uses this information for filter tuning and investigating the dynamics of the resulting system. Also, we focus on mail servers rather than individual email users, as mentioned above.

Foukia et al [12] are, like us, motivated to encourage mail servers to restrict output of spam. Their approach is agent-based – each participating mail server has an associated Federated Security Context Agent that contributes to, and draws on, an aggregated community view (the Federated Security Context). They also use quotas to control the volume of mail output by a server in an attempt to prevent temporary traffic bursts that are typical of spammer activity.

In summary, we can say that some elements of our anti-spam approach (using trust, closed loop control, etc) have already been proposed in some form by others, but our approach is unique in the way in which we combine them.

9. Conclusions

A new approach to improving spam filtering, based on collaboration between mail servers to manage trust, has been described in this paper. A trust management overlay architecture and a new lightweight protocol, called TOPAS (Trust Overlay Protocol for Anti Spam), have been presented. In this system, each mail server records trust measures relating to each other mail server of which it is aware. Trust by one mail server in another is influenced by direct experience as well as recommendations issued by collaborating mail servers. The TOPAS protocol specifies how these experiences and recommendations are communicated between each spam filter and its associated trust manager, and between trust managers of different mail servers. A technique for improving mail filtering performance and the TOPAS protocol using these trust measures has also been described. Experimental results have illustrated use of the protocol in a simulated network scenario, and indicate the potential of this approach to significantly improve rates of false positives and false negatives in anti-spam systems.

10. References

[1] J. Klensin (ed.), *Simple mail transfer protocol*, RFC 2821, Internet Engineering Task Force, 2001.

[2] J.B. Postel, *Simple mail transfer protocol*, RFC 821, Internet Engineering Task Force, 1982.

[3] P.R. Zimmermann, *The official PGP user's guide*, MIT Press, 1995.

[4] A. Schwartz, *SpamAssassin*, O'Reilly, 2004

[5] J. Goodman and R. Rounthwaite, "Stopping outgoing spam", *Proc. ACM Conference on E-Commerce*, 2004.

[6] M. Dam and R. Stadler, "A generic protocol for network state aggregation", *Proc. Radio Science and Communication conference* (RVK), Linköping, 2005.

[7] J. Douceur, "The Sybil attack", *Proc. 1st Int'l Workshop on Peer-to-Peer Systems*, 2002.

[8] J. Golbeck and J. Hendler, "Reputation network analysis for email filtering", *Proc. 1st Conference on Email and Anti-Spam (CEAS)*, 2004.

[9] J.S. Kong, B.A. Rezaei, N. Sarshar, V.P. Roychowdhury and P. Oscar Boykin, "Collaborative spam filtering using e-mail networks," *IEEE Computer*, vol. 39, no. 8, pp. 67-73, Aug. 2006.

[10] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi and P. Samarati, "P2P-based collaborative spam detection and filtering," *4th IEEE Int'l Conf. Peer-to-Peer Computing*, 2004.

[11] J.-M. Seigneur, N. Dimmock, C. Bryce and C. D. Jensen, "Combating Spam with TEA (Trustworthy Email Addresses)", *Proc. 2nd Annual Conference on Privacy, Security and Trust (PST)*, 2004.

[12] N. Foukia, L. Zhou and C. Neuman, "Multilateral decisions for collaborative defense against unsolicited bulk e-mail", *K. Stolen et al (Eds.): iTrust 2006,* LNCS 3986, pp. 77-92, 2006.

Acknowledgement

The authors' work is supported, respectively, by the European Commission FP6 project *OPAALS* and by Science Foundation Ireland (*Foundations of Autonomics* project).