Scalable and Reliable Collaborative Spam Filters: Harnessing the Global Social Email Networks

J. S. Kong[*] P. O. Boykin[†] B. A. Rezaei[*] N. Sarshar[*] V. P. Roychowdhury[*]

Abstract

We introduce a collaborative anti-spam system that is based on pervasive global social email networks. Essentially, we provide a solution to this open research problem: given a network of N users who are willing to share information collaboratively (e.g. the digests or fingerprints of known spams), how do we search for each user's content efficiently and reliably in a distributed manner with minimal traffic cost on the network? As a solution to this open problem, our proposed system employs the percolation search process, which makes the traffic generated due to queries for spam digests scale sublinearly as a function of N. However, in order to reap the benefits of this novel percolation search algorithm, the node degree distribution of the underlying network must be heavy-tailed. Interestingly, latent global social email networks comprising of personal contacts possess a power-law heavy-tailed degree distribution, which renders itself an ideal natural platform to employ the percolation search algorithm. As a result, our proposed distributed spam filter requires no dedicated peer-to-peer (P2P) systems or centralized server-based systems. We have performed large-scale simulations and we find that the system achieves a spam detection rate close to 100%, while the false positive rate is kept around zero. The bandwidth cost per user as well as the system-wide bandwidth cost are shown to be very low.

[*] Electrical Engineering Deptartment, University of California, Los Angeles, CA 90095.

[†] Electrical and Computer Engineering Department, University of Florida, Gainesville, FL 32611

1 Introduction

Spam, or Unsolicited Bulk Email, is plaguing internet users around the world. It has been estimated that approximately 68% of the worldwide email traffic today is spam and up to 87% of the emails directed to US users is spam[3]. For the past few years, numerous spam filters have been proposed and deployed, and as briefly reviewed in the following section, while these filters hold considerable promise, they also suffer from several drawbacks, which render them inadequate to serve as the sole defense against the growing onslaught of the spam menace. The primary contribution of this paper is to introduce a **collective spam filtering system** that exploits the **pervasive and already-existing social email networking system,** and harnesses it efficiently and reliably to provide a multi-layered defense against spams. In particular, we demonstrate (i) how one can **leverage the topological properties** of the underlying social email networks and use **recent advances in network-based search algorithms**, to make global searches and queries **scalable**, (ii) how **trust and reputation implied by social email interactions** can be harvested from these email networks, and (iii) how a **multi-layered system that parallels decision making at the societal scale** can be designed to effectively combat spam. That is, such a system utilizes existing content-based and personal information-based schemes as the first-line of defense at the individual level, and then utilizes communal trust metrics and reliable searches at the higher levels, to effectively identify and eliminate spams. This work builds on the recent pioneering work by Boykin and Roychowdhury[7] that showed how the topological properties of personal email networks can be used to automatically generate highly accurate white lists and black lists at the level of individual users to filter spams.

1.1 Prior Work, Challenges, and Motivation

Of all the existing anti-spam solutions, two classes of spam filters have emerged as the most effective and widely-deployed: *Bayesian/rule-based* spam filters and *collaborative* spam filters. A Bayesian filter uses the entire context of an e-mail in looking for words or phrases that will identify the e-mail as spam based on the experience gained from the user's sets of legitimate emails and spams. One example of a widely deployed Bayesian spam filter is SpamAssassin[4]. Although the Bayesian anti-spam solutions offer very impressive performances, they suffer from several serious drawbacks: first, Bayesian filters require an initial training period and exhibit a lower performance in classifying messages composed of previously unknown words; second, Bayesian filters are unable to block messages that do not look like a typical spam such as messages that is consist of only a URL or messages that are padded with random words. Most recently a number of multifaceted approaches have been proposed[15, 6]. They consider combining various forms of filtering with infrastructure changes, financial changes, legal recourse, and more, to address shortcomings of regular statistical filters.

The increasing realization that the *dynamic of spam constitutes a complex phenomenon brewed, fostered and propagated in the interconnected realm of the cyberspace*, has prompted the use of collaborative spam filters, where the basic idea is to use the collective memory of, and feedback from, the users to reliably identify spams. That is, *for every new spam that is sent out, some user must be the first one to identify it* upon receiving this spam (e.g., by using a Bayesian filter or locally generated white and black lists); now, any subsequent user that receives a suspect email can query the community of email users to find out if it has been already tagged as spam or not. In contrast to Bayesian type filters, collaborative spam filters do not suffer from the drawbacks just mentioned above, and it has been shown that they are also capable of superior spam detection performance. **The existing collaborative filtering schemes mostly ignore the already present and pervasive social communities in the cyberspace** and try to create new communities of their own to facilitate the sharing of information. This unenviable task of creating new social communities is beset with several difficulties that have limited the deployment and effective use of most collaborative filtering schemes proposed so far. The challenges include:

(i) *How to find users to participate?:* In order for a collaborative spam filter to be highly effective, a large number of users (on the order of hundreds of thousands or millions) must participate. However, effectively finding and interconnecting a large number of willing participants is non-trivial. In other words making any artificially established community acceptable and popular is an unpredictable and difficult task at best, and impossible at worst.

(ii) *How to make the search scalable?*: The power of a collaborative spam filter lies in the fact that spam databases from a large number of users are pooled together and utilized to fight spam. In order to avoid high server cost, the spam databases are typically stored locally on users' computer. Finding a way to do efficient searches on a network of distributed databases is very challenging.

(iii) *Who to trust*: Inevitably, there would be malicious users who try to subvert the collaborative anti-spam system by providing false information regarding spam. Therefore, a trust scheme must be devised to place more weights on the opinions of some provably trustworthy users than on some unknown users who can be potentially malicious.

The different proposed schemes for collaborative filtering attempt to address the above challenges to different degrees of effectiveness. For example, SpamNet[5] employs the following mechanisms to address the challenges stated above: It *uses a central server model* to connect all the willing participants of this collaborative spam filter. *The central server solution is not scalable* as the system scales and the server becomes a single point of attack or failure. SpamWatch[18] is a totally distributed spam filter based on the Distributed Hash Table (DHT) system Tapestry[17]. Most recently Gray *et. al.* have proposed CASSANDRA, a collaborative spam filter where the network is formed as clusters of trusted and similar peers. Finally a new reputation analysis have been proposed by Golbeck *et. al.* [11] where reputation relationships are inferred from the structure and are used as a method to score emails.

1.2 Harnessing The Global Social Email Network

Recently, Boykin and Roychowdhury investigated the notion of utilizing social networks to do spam filtering[7]. In their work, it was shown that just by looking at the clustering coefficient of an email user's personal contact networks, their algorithm is able to achieve a spam detection rate of 53% **with zero false positives**. Although this algorithm is very attractive, it ignores the larger social email network and focuses only on a projection as witnessed by an individual user, and it begs the question whether the larger social email networks can be harnessed. In this paper, we show that a high-performance, scalable and secure spam filter that exploits global email networks can indeed be designed. The fundamental idea is as follows: every user of the system is connected to each other through a chain of email contact links; for every new spam that is sent out, some user of the system must be the first one to identify it upon receiving this spam; now, any subsequent user that receives the same spam can query the network by following personal email contact links to locate a user that has already identified the message as spam, since all users are connected through the global social email network.

The main contributions of this paper lie in showing that *the three challenges outlined in the preceding discussions* can be *effectively addressed* using the topological properties of the underlying social email networks and *recent advances in complex networks theory*. **First,** no specially designed network has to be created for collaborative filtering. Hence, one of the main features of this system is that *all queries and communications are exchanged via email through personal contacts and that no server or a traditional P2P system with TCP/IP connections is needed.* **Second,** we observe that social email networks correspond to Power-Law (PL) graphs[10], with a PL coefficient around 2. *So the underlying network naturally possesses a scale-free structure that is a key hall-mark of many unstructured P2P systems that have organically grown for file-sharing on the Internet.* One can then **utilize a scalable global search query recently proposed by Sarshar et. al. [16] on this naturally scale-free graph** of social contacts to enable peers to exchange their spam signature data. **Finally,** the third main feature of the system is that we can **harvest and utilize the trust that is embedded in the web of email contacts**. By regarding contact links as local measures of trust and using a distributed Power Iteration algorithm, we can obtain a trust score called *mailtrust.* In fact, the famous Google PageRank[8] is computed in a similar fashion.

We then show that our proposed system not only has innovative and interesting structural features, but is also **capable of delivering high performances while incurring minimal costs.** Under the assumption that there would be a large number of users (on the order of hundreds of thousands or millions), the system can offer a spam detection rate around 99%; in fact, the detection rate can reach close to 100% when the number of users approach the internet scale. At the same time, the number of false positives in our system can be tightly controlled to a level very close to zero. Meanwhile, *as the number of users of the system scales, the communication cost of the system would be kept at a sublinear scale and the memory storage cost would grow only at a logarithmic scale.* In addition, **due to the fact that no TCP/IP connection is required and all communications in the system is done via background email exchanges**, less computational and networking burden would be placed on local computers. Lastly, the system is designed to be secure and rigorously protective of users' privacy and confidentiality.

The rest of the paper is organized as follows. In section 2, we present the background theory and the important concepts vital to this paper, such as email network theory and the percolation search algorithm. In section 3, we describe the protocol of our social network based collaborative anti-spam system in detail. In section 4, we use a real world email network to perform large-scale simulations of the system. Finally, in section 5, we address several important topics such as the protection of privacy and the system's resilience against random user failure.

2 Background Theory

Our system is motivated by a number of recent advances in complex networks theory and systems, Eigen-methods based computation of trust and relevance, and the proven efficacy of the spam digest system as signatures of emails.

<u>**Topology of Social Email Networks**</u>: A particular email network comprising 56,969 nodes (i.e., email addresses) has been studied by Ebel et. al.[10] Based on the statistics reported in Ebel's work, we identify three desirable properties that would make social email networks an attractive platform for building a collaborative spam filter:

(i) An email network has been found to possess a scale-free topology. More precisely, for the email network examined in [10], the node degree distribution follows a power law (PL): $P(k) \propto k^{-1.81}$, where k is the node degree, and $P(k)$ denotes the probability that a randomly chosen node has degree equal to k. One of the consequences of this property is that of very low percolation threhold[16]; in other words, *the network is extremely resilient to random deletions of nodes.* One can also show that even if high-degree nodes are deleted preferentially, *one has to remove almost all the high-degree nodes*, before the network gets fragmented.

(ii) A large fraction of the nodes ($\tilde{9}5.2\%$) in a social email network is connected to the giant connected component (GCC). This means that any node can reach almost any other arbitrary node by simply following email links.

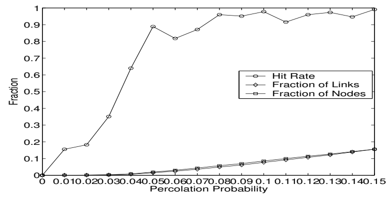
(iii) The email network has a low diameter (i.e. there exist short paths between almost any pair of two nodes in the network). In fact, for the email network investigated by Ebel et. al.[10], the mean shortest path length in the giant connected component was found to be $l = 4.95$ for a component size of $56,969$ nodes. This short-diameter property allows any email user to efficiently communicate with any other email user in the network by crossing only a few email contact links.

The above properties of the social email network should not come as a surprise, since it reflects the same social dynamics that we practice in our everyday life.

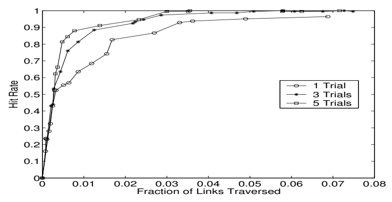
Scalability via Percolation Search: We can utilize the percolation search algorithm on a Power Law (PL) network proposed by Sarshar et. al.[16], which is based on the classical percolation theory. **It is shown that unstructured search in PL networks can be made highly scalable using the percolation search.** The key steps of the algorithm are as follows: **(i)** *Caching or Content Implantation*: Each node performs a short random walk in the network and caches its content list on each of the visited nodes. The length of this short random walk is specified later. **(ii)** *Query Implantation*: When a node intends to make a query, it first executes a short random walk of the same length as step 1 and implants its query requests on the nodes visited. **(iii)** *Bond Percolation*: All the implanted query requests are propagated through the network in a probabilistic manner; upon receiving the query, a node would relay to each of its neighboring nodes with percolation probability p, which is vanishingly greater than the percolation threshold, p_c, of the underlying PL network.

It is shown in [16] that the percolation threshold of any random network is given as $p_c = \langle k \rangle / \langle k^2 \rangle$. For a PL network with exponent τ and maximum degree k_{max}, we have $\langle k^2 \rangle = O(k_{max}^{3-\tau})$ and $\langle k \rangle = O(k_{max}^{2-\tau})$, and hence, we get a percolation threshold of $p_c = O(k_{max}^{-1})$, which is vanishingly small if k_{max} increases with the size of the network, which is usually the case. Thus, if we percolate at a multiple γ of p_c, then the total traffic generated would be, $\mathcal{C}_\tau = \gamma p_c \langle k \rangle N = O(\frac{\langle k \rangle^2 N}{\langle k^2 \rangle}) = O\left(k_{max}^{-\tau+1} N\right)$. In real world networks, k_{max} typically scales sublinearly as a function of the network size. For $k_{max} = O(N^{1/\tau})$, we have: $\mathcal{C}_\tau = O\left(k_{max}^{-\tau+1} N\right) = O(N^{\frac{1}{\tau}})$. For a detailed analysis of the hit rate and how it behaves as one performs multiple searches see [16].

The simulation plots for performing percolation search on a real email dataset [10, 1] are provided in Fig. 1(a) and Fig. 1(b).



(a)



(b)

Figure 1: **Percolation On Social Email Networks:** (a)The hit rate, fraction of links and fraction of nodes traversed as a function of the percolation probability. Notice that there is a sudden jump in the hit rate above the percolation threshold, while the fraction of links and nodes processing the search query increases only linearly. The network used in this percolation search simulation is a real-world email contact network. The number of nodes is 56,969, $\tau \approx 1.81$, the TTL is 50 for both query and content implants and only one unique content exists in the network. (b) Hit rate for percolation on email contact network with TTL of 50. Repeating the percolation trial multiple times pushes the hit rate exponentially closed to 1.

The MailTrust Algorithm

Just as in the case of WWW, where the PageRank captures the relevance of a particular web page, the topological structure of the social email networks can be used to assign trust or reputation to individual users. First, we model each email contact as placing a unit of trust on the recipient. Thus, for a node that contacts k_{out} other nodes, we can compute the fraction of trust that this node places on each of his out-neighbors as followed: the trust for neighbor i, t_i, is equal to the number of emails sent to neighbor i divided by the total number of emails sent. Note that the collection of t_i's forms a probability vector, called the personal trust vector $\overrightarrow{t}$. Thus, if we model the entire email network as a discrete time Markov chain, the local trust vector, $\overrightarrow{t}$, becomes the transition probability function for each node. We then compute the steady state probability vector using Power Iteration method which is the the same algorithm adopted to compute pagerank score of documents on web [12, 8]. As discussed in the literature, one needs to make sure that this Markov chain is ergodic and this can be achieved by having nodes with zero out-degree assign uniform trust to a set of pre-trusted nodes who have been carefully picked. We will refer to this trust score as *MailTrust* in the rest of this paper. A plot of the MailTrust scores obtained from [10, 1] is shown in Fig. 3. It is evident that the distribution of the MailTrust scores is heavy-tailed, which is in good agreement with the distribution the web documents' PageRank scores [14].

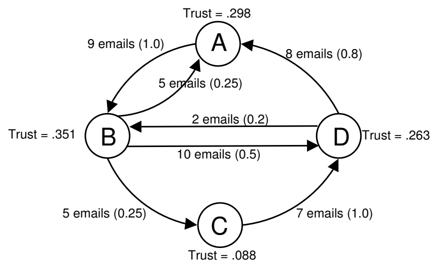


Figure 2: **MailTrust:** A simple illustration of the MailTrust algorithm. The numbers in parentheses represent the local trust values that each node places on his/her neighbors. The MailTrust scores for each node is then obtained by computing steady state probability vector of the Markov chain.

An alternate way to compute the MailTrust score in a

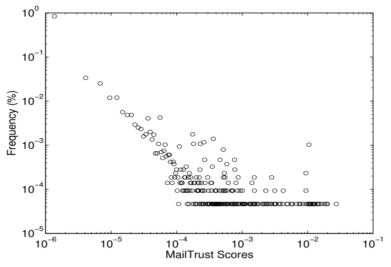


Figure 3: **MailTrust Distribution:** The probability density function of MailTrust scores using 10^4 bins. These scores are obtained by applying the MailTrust algorithm on a real-world email contact dataset. Notice that this probability density function is heavy-tailed, indicating that a few nodes are much more trustworthy than most nodes.

distributed fashion can be found in [12], along with a scheme on how the trust scores can be kept securely in the system even with the presence of malicious users.

Digest-based Spam Indexing In a collaborative spam filtering system, it is important to have an effective mechanism to index known spams so that subsequent arrivals of the same spam can be correctly identified. The collaborative design of the system does not depend on any specific algorithm but for initial experimental results we have adopted the well known *digest-based indexing* mechanism [9] to share spam information between users. Damiani et. al. have rigorously demonstrated that the digest algorithm described in [9] is highly resilient against possible forms of automatic modifications such as randomizing the spam content by switching the order of words and sentences. The digest algorithm is further shown to satisfy both the privacy preserving and zero positive requirements. There are also other mechanisms to index spams such as the Approximate Text Addressing technique used by SpamWatch [18] and the fingerprint generation algorithm used by SpamNet [5].

We are aware that finding an effective mechanism to index spam that is resilient to all forms of automatic modifications still remains to be an open research problem. However, one has to acknowledge that the accuracy and popularity of existing collaborative spam filtering systems such as SpamNet [5] proves that current spam indexing solutions are already effective and sophisticated enough to be deployed in large scale.

3 Implementation and System Protocol

In order to use our proposed collaborative spam filtering system, an interested individual must first obtain a simple client program that works as a plug-in to an email program such as MS Outlook, Eudora, Sendmail, etc[1]. This simple client will only need to provide the following features: first, the client must come with a digest-generating function as specified in section 2; second, the client is reponsible for keeping a personal blacklist of spams for the end-user as well as caching blacklists of spams for other nodes as described in the section on the percolation search algorithm, (see section 2); third, the client would have access to the list of social email contacts (both inbound and outbound) of the end-user. The pseudo code of the distributed client is given in Algorithm 1.

Algorithm 1 PROCESS-MAIL(Email E)

```
 1: if $E.From$ is in Contact list then
 2:     Mark $E$ as no Spam detected
 3: else
 4:     $D_e$ = Digest(E);
 5:     Implant percolation of $D_e$ on a random walk of
        length $l$
 6:     Wait(T);
 7:     $H_e$ = HitScore();
 8:     if $H_e < threshold$ then
 9:         Mark $E$ as no Spam detected
10:     else
11:         Mark $E$ as spam
12:     end if
13: end if
```

Algorithm 2 Publish-Spam(Email E)

```
1: $D_e$ = Digest(E);
2: Implant $D_e$ on a random walk of length $l$
```

System Maintenance: If the EigenTrust algorithm from section 2 is implemented, we would need to up-

[1]However, implementing the client program as an email program plug-in is not the only option; large email providers can also implement this system on the email server ends.

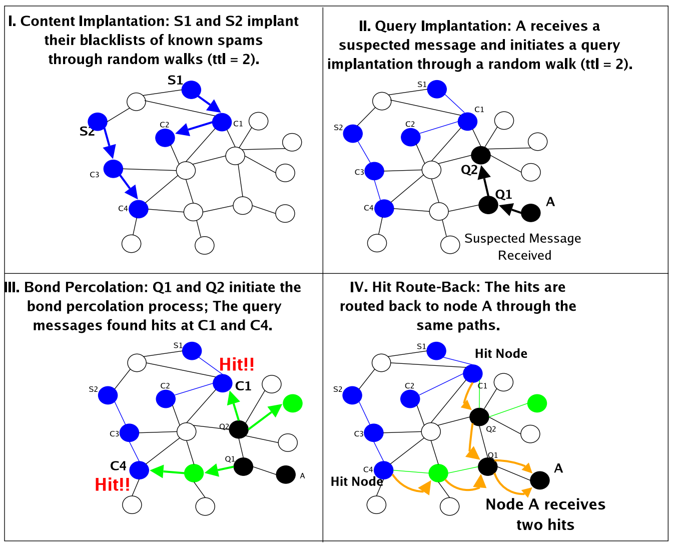

I. Content Implantation: S1 and S2 implant their blacklists of known spams through random walks (ttl = 2).
II. Query Implantation: A receives a suspected message and initiates a query implantation through a random walk (ttl = 2).
Suspected Message Received
III. Bond Percolation: Q1 and Q2 initiate the bond percolation process; The query messages found hits at C1 and C4.
Hit!!
IV. Hit Route-Back: The hits are routed back to node A through the same paths.
Hit Node
Node A receives two hits
S1
S2
C1
C2
C3
C4
Q1
Q2
A

Figure 4: An illustration of the protocol of the system.

Algorithm 3 HitScore(Hits)

```
1: if Using MailTrust then
2:     HitScore = $\Sigma_{h \in \text{Hits}} mailtrust(h)$
3: else
4:     HitScore = $|Hits|$
5: end if
6: Return HitScore;
```

date the trust scores of the nodes on a periodic basis. Since most people's amount of email contacts change more or less on a daily basis, we propose that the distributed EigenTrust computation should be performed about once a day to obtain new trust scores for all nodes.

Whenever a client program logs on or logs off from the system, a simple background message declaring join/leave should be sent to each of the user's contacts. In addition, every node in the network should send a periodic *ping* message to each of its neighbors to ensure that all contacts are online and available.

4 Simulation and System Performance

Network Model: In this section, all simulations are performed on a real-world email network investigated in Ebel et.al.'s work[10]. (The email network data can be obtained via this url [1].) In the following simulations, only the *giant connected component* is used, which contains 95.2% of all nodes in the original dataset[2]. Thus, all nodes are interconnected in this email network. Please see table 1 for the specific values of this email network's parameters.

Spam Arrival Model: The spam detection performance of a collaborate spam filter is usually a function of the number of copies of the same spam message that arrive to the system. It can be easily demonstrated that holding all other parameters constant, the greater is the number of identical copies arrived at the system, the better will the system performs in spam detection. On the other hand, in the extreme case that every spam arrived to the system is unique, one can easily see that a collaborative filter would be totally futile, since no user can benefit from the prior identifications of others. Assuming every unique spam is sent to approximately 5 million internet users uniformly at random among about 600 million internet users worldwide[2], the probability that any individual would receive a copy of a given spam is 0.8%. Thus about 500 replicas of a unique spam come uniformly at random to our network of 56,969 users.

Specification of Percolation Probabilities: In order to ensure a high hit rate for queries and a low communication cost for the system, we propose the following scheme to perform query searches: we start the first query with very low percolation probability; if not enough hits are returned, we send out a second query with a percolation probability that is twice of the first one; if still not enough hits are routed back, we repeat the searches by increasing the percolation probability in this two-fold fashion until the probability value reaches a maximum value, p_{max}; once this maximum is reached, we repeat the query with the maximum

[2]Only the giant connected component of the network is of interest, since all participants of this collaborative system can join the giant component through simple mechanisms, e.g. sending dummy emails to a group of designated system nodes

percolation probability for a constant number of trials and stop. We want to re-emphasize that the query search is terminated as soon as the total number of distinct hits routed back reaches the threshold after any given trial. The exact values for these parameters are up to the system designer to decide.

Scaling of System Bandwidth Cost: One main criticism of a collaborative spam filtering system is its traffic cost on the internet. In this section, we will show that the total query traffic cost caused by users receiving multiple copies of the same spam is upper bounded by a constant value. As specified in Algorithm 2 in section 3, each email that is filtered as spam is published in the system on a random walk of length ttl, which is $O(log\ N)$ with N being the number of nodes in the system. Given the assumption that copies of the same spam email E arrive at nodes of the system uniformly randomly, all nodes in the system will have cached at least one digest of E almost surely after $O(\frac{N}{ttl})$ or $O(\frac{N}{logN})$ copies of E have arrived. Using the same assumption from the **Spam Arrival Model** section above that the number of copies of each distinct spam arrived at the system is $N * 0.8\%$, it is easy to see that the total bandwidth cost is bounded by a constant as long as the inequality $O(\frac{1}{ttl}) < 0.8\%$ is satisfied. In contrast, central server solutions such as SpamNet[5] has a total bandwith cost that is directly proportional to the number of copies received by system users, which is highly unscalable.

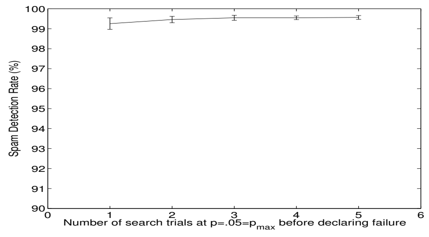


Figure 5: **Spam Detection Performance:** This figure plots the simulated spam detection rate (in percentage) as a function of the number of query trials repeated with percolation probability set at p_{max} before declaring failure. Note that all the average detection rates are well above 99%. The results are averaged over 30 runs and the error bar plots one standard deviation above and below the mean.

5 Concluding Remarks

Our fairly comprehensive simulation results show that global social email networks possess several properties that can be exploited using recent advances in complex networks theory to provide an efficient collaborative spam filter. **Due to space limitations, we could not present more detailed simulation results where we have implemented our MailTrust scheme, and showed that malicious attacks can indeed be thwarted,** and false positives can be cut down to a bare minimum. Clearly, the bare-bones and proof-of-concept systems discussed here can be vastly improved and augmented with schemes that have proven successful at various levels. For example, one can use white lists and black lists as well as Bayesian filters at local levels to decide whether an incoming message is a spam or not. Only if the message is found to be suspicious and not resolvable by local means then it is searched for in the global system. Similarly, an email client can use its local schemes to tag an email as a spam and publish its digest. As mentioned, in our work we have done extensive simulations of attacks by malicious users and shown that the trust scheme can be used to keep their attempts to tag non-spam emails as spam under control. We end the paper by noting that all omitted simulation results as well as detail discussion on several important topics, such as the system's measure of privacy protection and the system's resilience against user unreliability, are presented in the journal version of the paper [13].

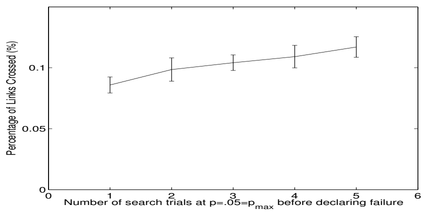


Figure 6: **Communication Cost Over The Whole Network:** This figure plots the percentage of edges crossed per query as a function of n_{reps}, average over 30 runs. Note that traffic cost is extremely low (only around 0.1% of network links need to be crossed per query). The error bar plots one standard deviation plus and minus the mean.

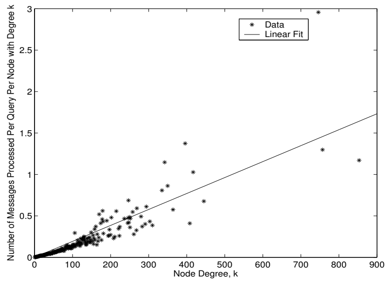


Figure 7: **Communication Cost As A Function of The Degree of a Node:** The data points show the average number of messages processed per percolation query for a node with degree k (i.e. it is the total number of messages processed per query for all nodes in the network with degree k divided by the number of nodes with degree k.) This plot is obtained by using an n_{reps} value of 3 for every percolation query. The slope of the linear fit is 0.0019 query/degree. Since each node forwards a query to a link with a fixed percolation probability, we naturally expect that high-degree nodes handle more messages.

Network	# of nodes	56,969
	# of edges	84,190
	Node degree distribution	Power-Law (PL)
	PL exponent	≈ 1.8
	mean node degree $< k >$	2.96
	node degree 2nd moment $< k^2 >$	174.937
	approximate percolation threshold $(q_c) \approx \frac{<k>}{<k^2>}$	.0169
	time-to-live (ttl)	50
Simulation Param.	# of arrivals of the same spam	500
	threshold (# of hits needed to identify spam)	2
	percolation probability trials	[.00625 .0125 .025 .05 .05 ...]
	# of runs	30
Threat Model	# of time steps	25
	# of malicious nodes inserted per time step	10
	total # of mailing lists	50,000
	Zipf coefficient	0.8
	# of non-spams queried per time step (x)	1,000
	m, number of items on a blacklist	10
	% of user's non-spam to be queried	5%

Table 1: Simulation Settings

References

[1] Email network data. http://www.theo-physik.uni-kiel.de/~ebel/email-net/email_net.html.

[2] Nua internet surveys. http://www.nua.ie/surveys/howmanyonline/world.html.

[3] Spam statistics. http://www.theregister.co.uk.

[4] Spamassassin. http://spamassassin.apache.org/.

[5] Spamnet. http://www.cloudmark.com.

[6] N. B. Barry Leiba. A multifaceted approach to spam reduction. In *Proc. of First Conference on Email and Anti-Spam (CEAS)*, 2004.

[7] P. O. Boykin and V. P. Roychowdhury. Leveraging social networks to fight spam. *IEEE Computer*, 38(4):61–68, April 2005.

[8] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[9] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. An open digest-based technique for spam detection. In *Proceedings of the 4th IEEE international conference on peer-to-peer computing*, 2004.

[10] H. Ebel, L.-I. Mielsch, and S. Bornholdt. Scale-free topology of e-mail networks. *Phys. Rev. E*, 66(035103), 2002. http://xxx.lanl.gov/abs/cond-mat/0201476.

[11] J. H. Jennifer Golbeck. Reputation network analysis for email filtering. In *Proc. of First Conference on Email and Anti-Spam (CEAS)*, 2004.

[12] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the twelfth international conference on World Wide Web*, pages 640–651. ACM Press, 2003.

[13] J. S. Kong, P. O. Boykin, B. Rezaei, N. Sarshar, and V. Roychowdhury. Let your cyberalter ego share information and manage spam. 2005. pre-print http://xxx.lanl.gov/abs/physics/0504026.

[14] G. Pandurangan, P. Raghavan, and E. Upfal. Using pagerank to characterize web structure. In *COCOON '02: Proceedings of the 8th Annual International Conference on Computing and Combinatorics*, pages 330–339, London, UK, 2002. Springer-Verlag.

[15] J. K. B. L. Richard Segal, Jason Crawford. Spamguru: An enterprise anti-spam filtering system. In *Proc. of First Conference on Email and Anti-Spam (CEAS)*, 2004.

[16] N. Sarshar, P. Boykin, and V. Roychowdhury. Percolation search in power law networks: Making unstructured peer-to-peer networks scalable. In *Proceedings of the 4th IEEE international conference on peer-to-peer computing*, 2004.

[17] B. Y. Zhao, L. Huang, S. C. Rhea, J. Stribling, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A global-scale overlay for rapid service deployment. *IEEE J-SAC*, 22(1):41–53, January 2004.

[18] F. Zhou, L. Zhuang, B. Y. Zhao, L. Huang, A. D. Joseph, and J. D. Kubiatowicz. Approximate object location and spam filtering on peer-to-peer systems. In *Proc. of Middleware*, pages 1–20, Rio de Janeiro, Brazil, June 2003. ACM.