Title
Collaborative spam filtering using e-mail networks

Journal
Computer, 39(8)

ISSN
0018-9162

Authors
Kong, Joseph S
Rezaei, B A
Sarshar, N
et al.

Publication Date
2006-08-01

Peer reviewed

Collaborative Spam Filtering Using E-Mail Networks

Joseph S. Kong, Behnam A. Rezaei, Nima Sarshar, and Vwani P. Roychowdhury
University of California, Los Angeles
P. Oscar Boykin
University of Florida

A distributed spam-filtering system that leverages e-mail networks' topological properties is more efficient and scalable than client-server-based solutions. Large-scale simulations of a prototype system reveal that this approach achieves a near-perfect spam-detection rate while minimizing bandwidth cost.

Widespread Internet use has led to the emergence of several kinds of large-scale social networks in cyberspace, most notably e-mail networks. Unfortunately, the tremendous convenience that e-mail affords comes at a high price: Although the network's underlying connectivity is hidden, making it almost impossible to build a comprehensive directory of every individual's contact lists, e-mail addresses themselves can be easily obtained from publicly available documents.

Spammers have exploited this vulnerability to inundate users with unsolicited bulk e-mail. Yet, despite public outcry over spam, the problem continues to grow and plague Internet users worldwide. In a recent study, 35 percent of e-mail users reported that more than 60 percent of their inbox messages were spam, and 28 percent said they spend more than 15 minutes a day dealing with junk e-mail (www.pewinternet.org/reports/toc.asp? Report=102).

Researchers in both industry and academia generally agree that there is no silver bullet and that researchers must develop new filtering techniques and integrate them with existing ones to build a layered system that, as a whole, can manage and restrict spam.

One promising approach is to beat spammers at their own game—to identify and stop spam by harnessing the same e-mail network and service infrastructure that they exploit. Spammers send the same or similar messages to thousands of users; we have developed a system that lets users query all of their e-mail clients to determine if another user in the system has already labeled a suspect message as spam. Because the network is latent, the system is message-based and distributed, enabling users to query for information without flooding the network.

SPAM FILTERS

During the past few years, *Bayesian* and *collaborative* spam filters have emerged as the two most effective antispam solutions. A Bayesian or rule-based filter uses the entire context of an e-mail to look for words or phrases that will identify it as spam based on the user's sets of legitimate e-mails and spam. One example of a widely deployed Bayesian spam filter is SpamAssassin (http://spamassassin.apache.org).

Collaborative spam filters use the collective memory of, and feedback from, users to reliably identify spam.[1-3] That is, for every new spam sent out, some user must first identify it as spam—for example, via locally generated blacklists or human inspection; any subsequent user who receives a suspect e-mail can then query the user community to determine whether the message is already tagged as spam. Some collaborative spam filters, such

0018-9162/06/$20.00 © 2006 IEEE

as SpamNet (www.cloudmark.com), deliver performance comparable to that of Bayesian filters.

A major drawback of collaborative filtering schemes is that they ignore the already present and pervasive social communities in cyberspace and instead try to create new ones of their own to facilitate information sharing.

For collaborative systems to scale up and serve truly large-scale communities, they must address three key challenges:

- *Performance.* For a collaborative spam filter to be effective, a large number of users—on the order of millions—must participate. However, targeting and interconnecting this many users is a difficult task with unpredictable outcomes.
- *Scalability.* The power of a collaborative spam filter lies in the ability to query spam data from numerous users. To avoid high server costs, spam databases are typically stored locally on users' computers. Efficiently searching a network of distributed databases is difficult.
- *Trust.* Inevitably, hackers will try to subvert the system by providing false information regarding spam. Therefore, a trust scheme must be devised to weigh the opinions of provably trustworthy users more heavily than those of unknown, potentially malicious, users.

A collaborative spam-filtering system must meet the challenges of performance, scalability, and trust.

Different collaborative systems address these challenges with varying degrees of effectiveness. For example, SpamNet uses a central server model: Users upload spam information to a central database, which all users wanting to check whether a suspect e-mail is tagged as spam can query. Because this type of system has access to every user's data, it can also detect potential attackers and build trust scores based on user feedback. However, this approach is not scalable—for example, it is doubtful that SpamNet, which has around a million customers, can cost-effectively serve tens of millions of users. In addition, the central server becomes a single point of attack or failure.

Proposed distributed spam filters employ a distributed hash table and multiagent protocols[1-3] to provide scalable query performance. However, these systems must build new communities and effective algorithms for determining meaningful user trust scores.

HARNESSING SOCIAL E-MAIL NETWORKS

A recent study[4] proposed an attractive algorithm for using personal e-mail networks to filter spam but largely ignored the question of whether entire social e-mail networks can be harnessed. Advances in *complex networks theory*—an emerging field that uses statistical mechanics to analyze network dynamics—now make it possible to leverage such networks' topological properties to create a collaborative spam-filtering system that meets the challenges of performance, scalability, and trust.

A major advantage of using social e-mail networks to filter spam is that it is not necessary to specifically design a network for this purpose. Because all queries and communications are exchanged via e-mail through personal contacts, neither a server nor a traditional peer-to-peer (P2P) system is needed.

A 2002 analysis[5] of an e-mail network comprising nearly 57,000 nodes (e-mail addresses) revealed that such networks possess a scale-free topology. More precisely, the node-degree distribution follows a power law $P(k) \propto k^{-1.81}$, where k is the node degree and $P(k)$ denotes the probability that a randomly chosen node has a degree equal to k. One of the consequences of this PL property is a very low percolation threshold, which makes the e-mail network an ideal platform for the efficient *percolation search* algorithm.[6] Thus, it is possible to overlay a scalable global-search system on this naturally scale-free graph of social contacts to enable peers to exchange their spam signature data efficiently.

In social e-mail networks, trust is embedded in the web of e-mail contacts. By regarding contact links as local measures of trust and using a distributed power iteration algorithm, it is possible to obtain trust scores for all participating users. In addition, such collaborative spam-filtering mechanisms can be implemented as plug-ins to popular e-mail programs such as Microsoft Outlook.

COLLABORATIVE SPAM-FILTERING MECHANISMS

Our spam-filtering system uses two key mechanisms to exploit the topological properties of social e-mail networks: the novel percolation search algorithm, which reliably retrieves content in an unstructured network by looking through only a fraction of the network, and the well-known digest-based indexing scheme.

Percolation search

Finding a provably scalable method to search for unique content on unstructured networks has remained an open research problem for years. It originated in the tremendous query traffic generated by the popular Gnutella P2P file-sharing program. The initial Gnutella search protocol used scoped broadcast mechanisms to discover content, which scales as $O(N)$. Since then, researchers have proposed numerous solutions, such as Ultrapeer and various random-walk methods, but most of these reduce query traffic cost only by a constant fac-

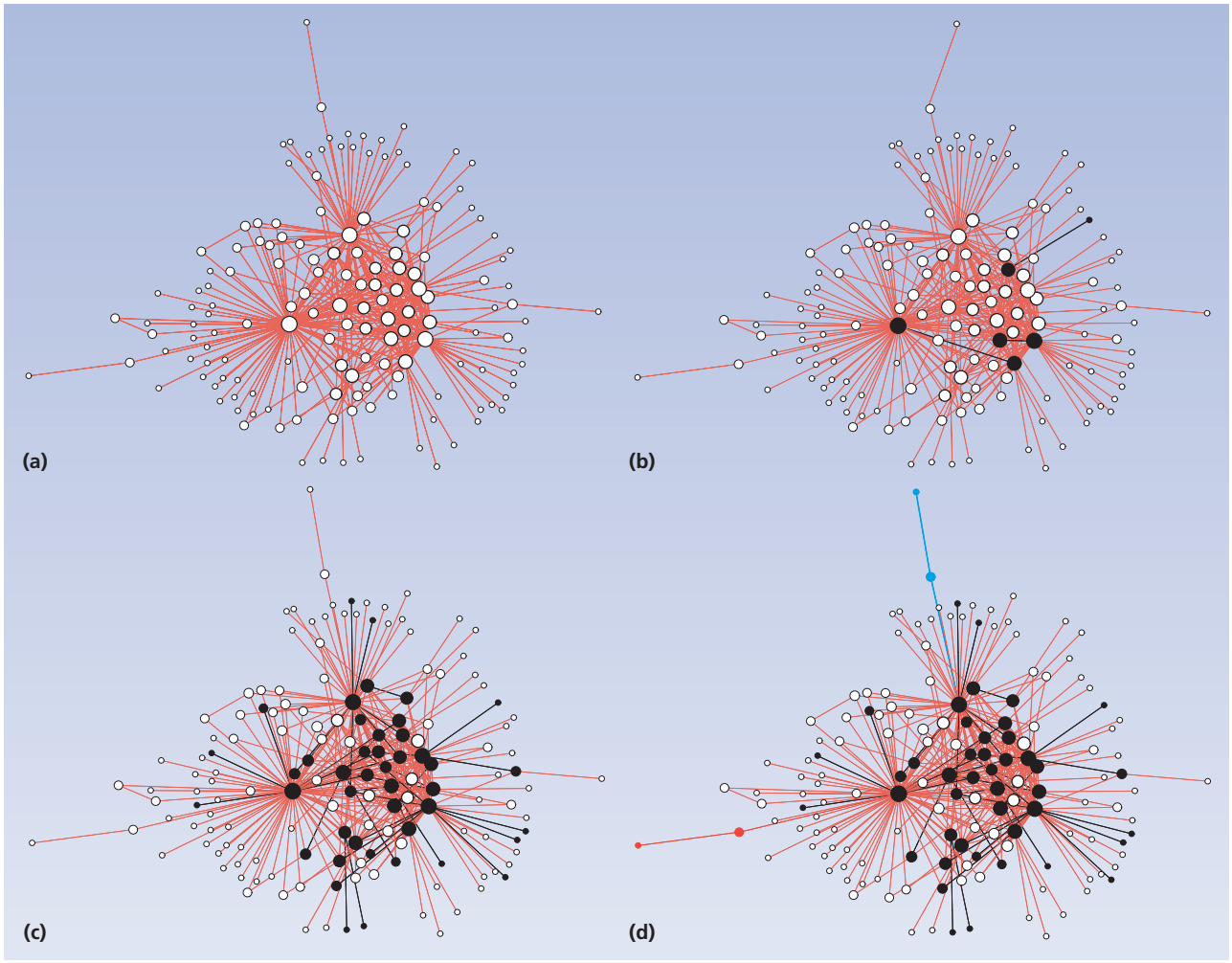

Figure 1. Percolation search. Percolated networks are obtained by keeping each edge in the underlying network with probability p. *(a) Random power-law network with 153 nodes, 366 edges, and a PL coefficient of 1.81; the percolation threshold is approximately 0.0359. (b) Bond percolation at* p = *0.0144, which is below the threshold, results in small-size, disconnected components. (c) Bond percolation at* p = *0.0898, which is 2.5 times the threshold, produces a giant connected component containing all high-degree nodes. (d) Sample search: Using random walks, one node implants a query request (blue path) while another publishes its content (red path); the search returns a hit through bond percolation at* p = *0.0898.*

tor. In contrast, the query traffic cost using percolation search scales sublinearly as a function of the system size.[6]

Percolated networks. To understand how percolation search works, consider a random PL network like that shown in Figure 1a. Bond percolation removes each edge in a graph with probability $1 - p$ (each edge is kept with probability p), where p is the percolation probability. The surviving edges and nodes form the percolated network.

Figures 1b, 1c, and 1d show three realizations of a percolation process on an identical underlying network. If the percolation probability is less than the percolation threshold p_c, the percolated network consists of small-size connected components and lacks a giant connected component as in Figure 1b; if $p > p_c$, a giant connected component emerges in the percolated network as in Figures 1c and 1d. Note that the percolation threshold, p_c, is extremely small for any PL network. As a result, the size of the percolated network core is extremely small, since the size is proportional to p_c.

The percolation threshold is the lowest percolation probability in which the expected size of the largest connected component goes to infinity for an infinite-size graph. When percolation occurs above p_c, as in Figures 1c and 1d, high-degree nodes always survive the process and become part of the giant connected component. Thus, in an unstructured network search, if contents are cached in random high-degree nodes and every query starts from a random high-degree node, percolation can reliably find all content with high probability.

Starting from any node in a PL network with exponent 2, a query is almost certain to encounter a high-degree node with a random-walk path of $O(\log N)$ hops.[6] Thus, if every node implants its content on

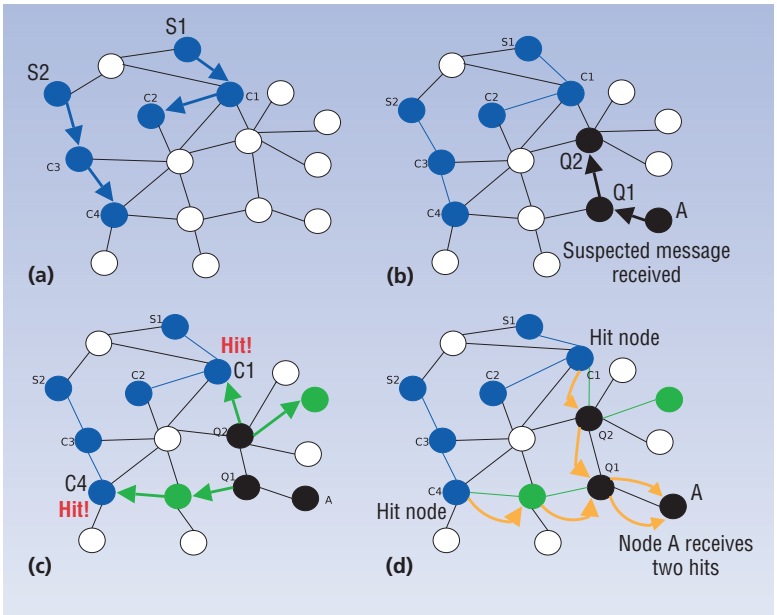

Figure 2. Key steps in spam-filtering system protocol. (a) Digest publication. (b) Query implantation. (c) Bond percolation. (d) Hit routeback.

$O(\log N)$ nodes via a random walk, then all contents will be cached in high-degree nodes. Similarly, queries can be implanted on high-degree nodes via a random walk of length $O(\log N)$. Thus, bond percolation starting from high-degree nodes will find the contents, as in Figure 1d.

Search algorithm. Using the percolation search algorithm can make unstructured searches in PL networks highly scalable. This algorithm passes messages on direct links only and includes three key steps:

- *Cache* or *content implantation*: Each node performs a short random walk in the network and caches its content list on each visited node. The length of this short random walk is referred to as the *time to live* (TTL).
- *Query implantation*: A node making a query executes a short random walk of the same length as the TTL used in the content implantation process and implants its query requests on the nodes visited.
- *Bond percolation*: The algorithm propagates all implanted query requests through the network in a probabilistic manner; upon receiving the query, a node relays it to each neighboring node with percolation probability p, which is a constant multiple of the percolation threshold, p_c, of the underlying network.

Note that query traffic is proportional to the size of the percolated network. For PL networks, the percolated network is small in size. Because social e-mail networks have a PL degree distribution, this algorithm is ideally suited for reaping the benefits of a percolation search.

Digest-based spam indexing

A collaborative spam-filtering system needs an effective mechanism to index known spam to correctly identify subsequent arrivals of the same spam. Although our system does not depend on a specific algorithm, we recommend one such digest-based indexing mechanism developed by Ernesto Damiani and colleagues[7] to share spam information among users. This algorithm is highly resilient to automatic modification of spam, preserves privacy, and produces zero false positives in which the digest of a legitimate e-mail matches that of an unrelated spam.

SYSTEM IMPLEMENTATION

To implement our collaborative spam-filtering system, the typical user must first obtain a simple client that works as a plug-in to an e-mail program such as MS Outlook, Eudora, or Sendmail (large e-mail providers can also implement this system on the e-mail server). This simple client must include a digest-generating function, keep a personal blacklist of spam for the user as well as cache blacklists of spam implanted by other nodes, and have access to the user's inbound and outbound e-mail contacts.

Initial processing. When the user receives an e-mail, the client program first attempts to determine whether it is DefinitelyNotSpam or DefinitelySpam. The client program can use any traditional spam-filtering method—such as whitelist, blacklist, or Bayesian filters—to perform this function. For example, DefinitelyNotSpam can be a whitelist of addresses in the contact list and DefinitelySpam can be Bayesian filter output indicating a high probability that e-mail is spam.

Digest publication. If the client program determines that the e-mail is definitely spam, it calls the digest function to generate a digest, D_e, for the message and caches the digest on a short random walk of length l, which is the TTL. In Figure 2a, nodes S1 and S2 implant their blacklists of known spam through random walks with a TTL equal to 2.

Query implantation. If the client program suspects that the e-mail is spam, it can query the system to determine whether any other user in the network already has D_e on its spam list. It implants each query message for this digest via a random walk of length l. In Figure 2b, node A receives a suspected message and implants a query via a random walk with a TTL equal to 2.

Bond percolation. Nodes with an implanted query request percolate the query message containing D_e through the e-mail contact network. Each node that the query visits declares a hit if the digest matches any messages cached on that node.[7] In Figure 2c, nodes Q1 and

Q2 initiate the bond percolation process; the query messages find hits at nodes C1 and C4.

Hit routeback. The client program routes all hits back to the node that originated the query through the same path by which the query message arrived at the hit node. In Figure 2d, the system routes the hits at nodes C1 and C2 back to node A through the same path.

Hit processing. After routing all hits back, the client program calculates the number of hits received. If this HitScore exceeds a constant threshold value, the program declares the message in question as spam; otherwise, it determines the message not to be spam.

The client program places all e-mail messages declared as spam in the user's spam folder. It then calls the function that generates the digest of the spam message, D_e, and caches this on a short random walk, taking the process back to the digest publication step.

SIMULATION AND SYSTEM PERFORMANCE

We simulated our spam-filtering system on a real-world e-mail network[5] consisting of 56,969 nodes and 84,190 edges. It has a power-law node degree distribution with a PL exponent of approximately 1.8, a mean node degree of 2.96, a node degree second moment of 174.937, and an approximate percolation threshold of 0.0169 (note that the percolation threshold of a network is approximately equal to the ratio of the first and second moment of the degree distribution).

We modeled spam detection performance as a function of the number of copies of similar spam messages that arrive at the system. Assuming that every unique spam is sent uniformly at random to approximately 5 million Internet users among about 600 million users worldwide (www.itu.int/ITU-D/ict/statistics/at_glance/Internet02.pdf), the probability that any individual would receive a copy of a given spam is 0.8 percent. Thus, about 500 replicas of a unique spam arrive uniformly at random to the network.

Percolation probabilities

Because the underlying network's percolation threshold might not be known, our scheme adaptively searches for spam using an increasing sequence of percolation probabilities, thereby ensuring a high hit rate for queries and a low communication cost for the system.

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 twice that of the first one. If still not enough hits are routed back, we repeat the searches by increasing the percolation probability in this twofold fashion until the probability value reaches a maximum value, p_{max}. Once this maximum is reached, we repeat the query with the maximum percolation probability for a constant number of trials, n_{max_stop}.

Table 1. Spam-filtering simulation results.

n_{max_stop}	Percentage of links crossed per query	Spam detection rate
1	0.086	99.3 ±0.3
2	0.099	99.5 ±0.2
3	0.104	99.5 ±0.1
4	0.109	99.5 ±0.1
5	0.117	99.6 ±0.1

Simulation execution and results

We conducted five simulation sessions, with each session consisting of 30 runs and a different n_{max_stop} value ranging from 1 to 5. We ran the simulation with TTL = 50, percolation probability starting at 0.00625, and p_{max} = 0.05. As Table 1 shows, the system achieves superior spam detection rates while incurring minimal traffic cost.[8]

The required traffic for each query is about 0.1 percent of the number of edges, which corresponds to roughly 84 background e-mails. Bandwidth cost per query is thus approximately 84 + 50 (the TTL) = 134 e-mail exchanges. Moreover, every background e-mail containing a message digest is about 1 Kbyte, and every e-mail incurs bandwidth cost—a multiplicative constant of 2—on both the sender and receiver. Thus, bandwidth cost per query is $134 \times 1 \times 2 = 268$ Kbytes. Assuming that each user receives one spam per hour, the average bandwidth cost per user is a modest 268 Kbytes/hour, which will minimally impact other network applications.

Note that the query traffic that our system generates will scale sublinearly as a function of the number of spam arrivals. As more copies of the same spam arrive, more nodes cache this information—thus, fewer message exchanges are needed to identify the spam. In addition, as Figure 3 on the next page shows, the high-degree nodes handle more queries. Such processing demands are inherently fair, as it is natural to ask a user who sends e-mails to twice as many contacts to resolve twice as many queries.

INFILTRATION AND COUNTERMEASURES

By nature, spammers will attempt to defeat all spam-filtering techniques. One common strategy is to join the system and blacklist well-known legitimate e-mails such as those sent from popular mailing lists. To combat spammer infiltration, our system uses an EigenTrust scheme. Such schemes have proven to be effective in ranking documents on the Web and in assigning trust to peers in P2P systems.[9,10] A recently proposed algorithm similarly uses the EigenTrust method to assign trust to mail senders.[11]

Just as Google's PageRank captures the importance of particular Web pages based on their hyperlink struc-

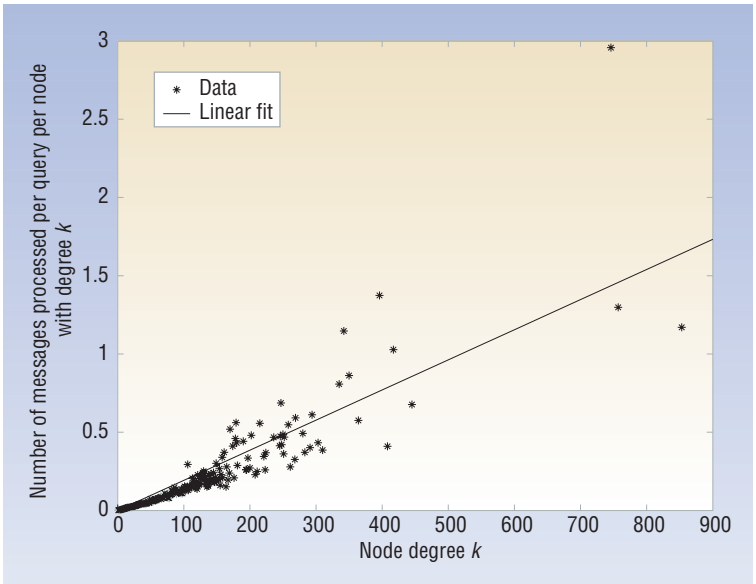

Figure 3. Message processing. The average number of messages processed at a node increases linearly with its e-mail activity level—that is, its degree in the network.

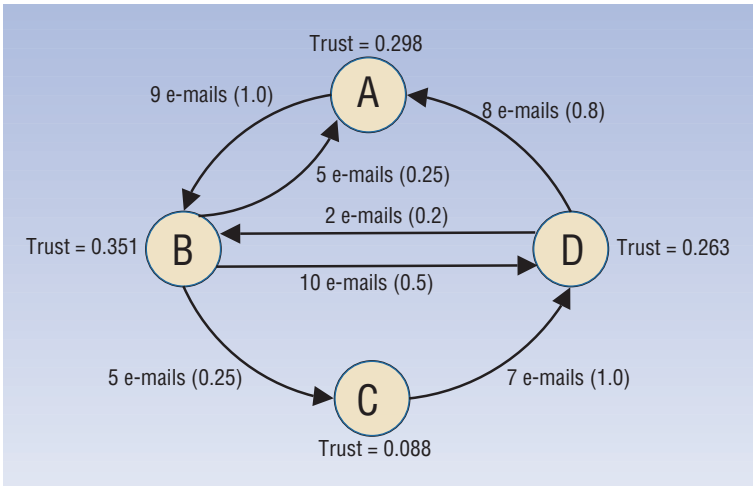

Figure 4. The EigenTrust scheme obtains the trust score for each node by computing the steady-state probability vector of the Markov chain.

ture, our proposed system can use the topological structure of social e-mail networks to assign trust to individual users. In our scheme, each e-mail contact places a unit of trust on the recipient. For a node that contacts k_{out} other nodes, the fraction of trust, t_i, that this node places on each out-neighbor i is equal to the number of e-mails sent to i divided by the total number of e-mails sent.

The collection of t_is forms a *personal trust vector*, $\vec{t}$. We next model the entire e-mail network as a discrete-time Markov chain, where entries of the personal trust vectors define the state-transition probabilities. As Figure 4 shows, the EigenTrust scheme then computes the steady-state probability vector using the same power iteration algorithm PageRank employs to score Web documents.[9] To ensure that the Markov chain is ergodic, nodes with zero out-degree assign uniform trust to a set of carefully chosen pretrusted nodes.

Because our proposed antispam system is social-network-based, it is important to protect users' privacy by preventing anybody from using the network to map out social links. To address this problem, nodes must forward all messages exchanged in the system anonymously. The basic idea is that when a node forwards a message, any information pertaining to the nodes that the message has visited must be deleted before forwarding. This keeps all system communications to an acquaintance-acquaintance level.

In addition to privacy concerns, users might be worried that our system provides an additional channel for spreading computer viruses. Recall, however, that the system exchanges all messages via background e-mails. Users are not required to click and open any system message or file. Moreover, the system can program clients to reject all messages that do not match a predefined format and thus are potentially malicious. Finally, we recommend adding a personalization feature that lets the user blacklist only spam addressed to the public, such as drug ads, while keeping a separate filter for personal spam—messages targeted specifically to the user and unlikely to have been mass mailed.

Our simulation results show that global social e-mail networks possess several properties that researchers can exploit using recent advances in complex networks theory, such as the percolation search algorithm, to provide an efficient collaborative spam filter. Clearly, our proof-of-concept system can be vastly improved and augmented with other spam-filtering schemes proven successful at various levels.

In addition, a collaborative spam-filtering system such as ours can be overlaid on top of any PL network, not just social e-mail networks. For example, the contact network of SMTP servers can be modeled after the Internet router network, which is a natural PL network.[12] Consequently, developers can implement our distributed system on SMTP relay servers for earlier spam detection and filtering. Blocking spam during the early stages of spam injection can save significant Internet bandwidth.

Finally, there is nothing special about searching for and caching spam digests, and administrators can use our pervasive message-passing system to manage a general distributed information system. The primary requirement is to be able to provide enough benefits to users to encourage their participation, which is relatively easy when it comes to spam management. If users become accustomed to a spam-filtering system, queries for other information will follow.

Acknowledgments

The authors thank Linling He, Andrew Nguyen, and Jinbo Cao for their many helpful comments and suggestions. This work was supported in part by NSF grants ITR:ECF0300635 and BIC:EMT0524843.

References

1. F. Zhou et al., "Approximate Object Location and Spam Filtering on Peer-to-Peer Systems," *Proc. ACM/IFIP/Usenix Int'l Middleware Conf.*, LNCS 2672, Springer, 2003, pp. 1-20.
2. A. Gray and M. Haahr, "Personalised, Collaborative Spam Filtering," *Proc. 1st Conf. E-Mail and Anti-Spam*, 2004; www.ceas.cc/papers-2004/132.pdf.
3. J. Metzger, M. Schillo, and K. Fischer, "A Multiagent-Based Peer-to-Peer Network in Java for Distributed Spam Filtering," *Proc. 3rd Int'l Central and Eastern European Conf. Multi-Agent Systems*, LNCS 2691, Springer, 2003, pp. 616-625.
4. P.O. Boykin and V.P. Roychowdhury, "Leveraging Social Networks to Fight Spam," *Computer*, Apr. 2005, pp. 61-68.
5. H. Ebel, L-I. Mielsch, and S. Bornholdt, "Scale-Free Topology of E-Mail Networks," *Phys. Rev. E*, vol. 66, no. 035103(R), 2002; www.theo-physik.uni-kiel.de/~bornhol/pre035103.pdf.
6. N. Sarshar, P.O. Boykin, and V.P. Roychowdhury, "Percolation Search in Power Law Networks: Making Unstructured Peer-to-Peer Networks Scalable," *Proc. 4th Int'l Conf. Peer-to-Peer Computing*, IEEE CS Press, 2004, pp. 2-9.
7. E. Damiani et al., "P2P-Based Collaborative Spam Detection and Filtering," *Proc. 4th IEEE Int'l Conf. Peer-to-Peer Computing*, IEEE CS Press, 2004, pp. 176-183.
8. J.S. Kong et al., "Let Your CyberAlter Ego Share Information and Manage Spam," 2005; http://xxx.lanl.gov/abs/physics/0504026.
9. S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks," *Proc. 12th Int'l Conf. World Wide Web*, ACM Press, 2003, pp. 640-651.
10. S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Computer Networks and ISDN Systems*, vol. 30, nos. 1-7, 1998, pp. 107-117.
11. P-A. Chirita, J. Diederich, and W. Nejdl, "MailRank: Using Ranking for Spam Detection," *Proc. 14th ACM Int'l Conf. Information and Knowledge Management*, ACM Press, 2005, pp. 373-380.
12. G. Siganos et al., "Power Laws and the As-Level Internet Topology," *IEEE/ACM Trans. Networking*, vol. 11, no. 4, 2003, pp. 514-524.

Joseph S. Kong *is a PhD student in the Department of Electrical Engineering at the University of California, Los Angeles. His research interests include large-scale distributed computing, P2P overlay network design, and complex networks theory and its application to social and technological networks. Kong received an MS in electrical engineering from the University of California, Los Angeles. Contact him at jskong@ee.ucla.edu.*

Behnam A. Rezaei *is a PhD student in the Department of Electrical Engineering at the University of California, Los Angeles. His research focuses on general complex networks, in particular P2P network design, and structural analysis of complex networks and their application in social and biological networks. Rezaei received an MS in electrical engineering from the University of California, Los Angeles. Contact him at behnam@ee.ucla.edu.*

Nima Sarshar *is a PhD candidate at the Department of Electrical and Computer Engineering at McMaster University, Hamilton, Ontario, Canada. His research interests include distributed computation in complex networks and statistical mechanics of large-scale communication networks, in particular P2P data mining. Sarshar received an MS in electrical engineering from the University of California, Los Angeles. Contact him at nima@ee.ucla.edu.*

Vwani P. Roychowdhury *is a professor of electrical engineering at the University of California, Los Angeles. His research focuses on computation models, including parallel and distributed processing systems, quantum computation and information processing, and circuits and computing paradigms for nanoelectronics and molecular electronics. Roychowdhury received a PhD in electrical engineering from Stanford University. Contact him at vwani@ee.ucla.edu.*

P. Oscar Boykin *is an assistant professor in the Department of Electrical and Computer Engineering at the University of Florida. His research interests include quantum cryptography, quantum information and computation, P2P computing, and neural coding. Boykin received a PhD in physics from the University of California, Los Angeles. Contact him at boykin@ece.ufl.edu.*