Collaborative Blog Spam Filtering Using Adaptive Percolation Search

Seungyeop Han
Division of Computer Science
KAIST, Daejeon, Korea
syhan@an.kaist.ac.kr

Yong-yeol Ahn
Department of Physics
KAIST, Deajeon, Korea
yongyeol@gmail.com

Sue Moon
Division of Computer Science
KAIST, Daejeon, Korea
sbmoon@cs.kaist.ac.kr

Hawoong Jeong
Department of Physics
KAIST, Deajeon, Korea
hjeong@kaist.ac.kr

ABSTRACT

We propose a novel collaborative filtering method for link spams on blogs. The key idea is to rely on manual identification of spams and share this information about spams through a network of trust. The blogger who has identified a spam tells a small number of fellow bloggers (content implantation), and those who have not heard about it start a search using an adaptive percolation search, combined with content implantation, they contract the information about identified spam in only a fraction of the query period time without producing large volume of traffic.

1. INTRODUCTION

Spams or unsolicited bulk emails have been a pressing problem ever since the very beginning of the Internet [1]. According to the MessageLabs, 65% of the worldwide email traffic in July 2005 was spam [2]. Besides the email system, spams are also prevalent in instant messages, newsgroups, chat rooms, voice calls, and blogs.

A blog is a web-based publication consisting primarily of periodic articles [3]. Most of the blogs are used as personal diaries and also used by corporations, in media programs, or for political campaigns. The number of blogs is growing remarkably. By mid 2005, there were over 14.2 million blogs worldwide, and the population continues to double roughly every 5.5 months [4]. With such rapid growth, the number of blog spams has also increased to a problematic level.

The vulnerability of blog systems to spams lies in the openness of a *comment* or a *trackback*, which are the standard ways of communication among bloggers. A comment is a short reply to a writing in a blog; a trackback is a notification about a reply relevant to a blog, but written on some other blog. Both the comment and trackback are displayed along with the original post. Most blogs allow anyone to write comments and trackbacks, and they use a common trackback protocol. Hence, a spammer can easily write a comment or a trackback on most blogs.

As comments and trackbacks appear on the web page, they are good targets for *link spams* which contain URLs pointing to the spammers' intended web sites. In contrast to *content spams*, in which the content of the spam itself is annoying, the main goal of link spams is to mislead search engines in order to obtain higher-than-deserved ranking in search results [5]. The links to the web site of a spammer boost its ranking, as the number of incoming links plays an important role in the score of ranking algorithms many search engines use [6].

In 2005, major search engines, including Google, MSN Search, and Yahoo, came up with a partial solution for link spams: they recommend the "`rel=nofollow`" attribute to be added to hyperlinks in automatically generated parts of web pages (e.g., comments and trackbacks). When a search engine sees this attribute on hyperlinks, those links will not get any credit as they rank websites in their search results [7]. Although using the `nofollow` attribute may be effective in decreasing the level of pollution by link spams, it has not discouraged spammers from sending out spams to blogs, whether they employ the nofollow attribute or not. Another drawback of this solution is that not only the spammers, but also legitimate web pages, do not get justifiable benefits from the incoming links to their web pages. Whether to adopt `nofollow` or not is still under debate, and no conclusion has been drawn yet [8].

Various approaches to block spams have been proposed, mostly focusing on the email spams. Bayesian filters are widely deployed to block email spams (e.g., SpamProbe [9] and SpamAssassin [10]). They are quite effective in blocking content spams. However, a link spam may be filled with random words or some phrases people typically use in greetings that do not look like spam lexically; thus the contents of link spams and benign ones are indistinguishable in practice. Therefore they are not suitable in blocking link spams.

A collaborative spam filtering has been proposed for email spams [11–14]. Since a spam is typically sent to a very large number of recipients, anyone who first identifies it as a spam can share the knowledge with others, thus benefiting the community. However, such a scheme is not very effective against content spams, as spammers customize their spams by attaching *word salad* (i.e., random words) in order to avoid being matched as spams. Nevertheless, collaborative

WWW2006, May 22–26, 2006, Edinburgh, UK.
.

spam filtering scheme can effectively block link spams. Thus, we propose using a collaborative spam filter to block link spams in blog systems. We propose a new trust building scheme, which exploits existing social trust relations and a new search method to obtain information about identified spams.

Our approach is based on a simple peer-to-peer trust building process and a novel information search method, called **adaptive percolation search (APS)**. Under our scheme, each blog sends out queries to its neighbors in the trusted network to see if anyone has already identified it as a spam. The basic idea of our APS is that the query is forwarded with the probability adjusted at every peer according to the peer's degree (i.e., the number of neighbors it has). Through this strategy, our algorithm always percolates the network without producing broadcast-like traffic. We use a periodic or an asynchronous query scheme to collect information from network to identify spams. We also present rigorous simulation results to show the effectiveness of our approach.

The rest of the paper is organized as follows. In Section 2, we survey related works in the area of blocking blog spams and present backgrounds on the blog networks. In Section 3, we describe our collaborative spam filtering approach in detail. In Section 4, we perform a simulation of our method based on a real-world blog network. We conclude in Section 5.

2. PRELIMINARIES

In this section, we introduce backgrounds on the blog networks and survey related works in the area of blocking blog spams.

2.1 Collaborative Spam Filtering Scheme

Collaborative spam filtering can be an effective way to exploit the massiveness of spams in the fast evolution of spam robots against various spam filters.

There are three key features in collaborative spam filtering: where the information of spam is stored; how to manage the trust relations; and how to effectively share and search the information.

Collaborative spam filtering may operate in a *centralized* or a *distributed* manner. Existing collaborative email spam filters mostly use a centralized approach (e.g., SpamNet [13]). A centralized server model is known to scale poorly as the number of spams increases, and has a single point of failure. Moreover, determining whom to trust is a difficult problem in the centralized model. SpamWatch [12] is the first known spam filter adopting a totally decentralized approach. Its operation is based on a distributed text similarity engine. More recently, Damiani *et al.* suggested a peer-to-peer (P2P) based collaborative filter with a hierarchical network topology [11], and Kong *et al.* proposed collaboration using existing email social networks [14]. Our approach is motivated by these collaborative spam filters. However, our focus is on blog systems.

In any collaborative endeavor, determining whom to trust is an important issue. When a peer cooperates with other peers in order to ferret out spams, one needs to evaluate and manage how trustworthy other peers are. Depending on the scope of the trust, two evaluation schemes are possible: *global* and *local*. In a global trust scheme, each peer has a single reputational value for one's own trustworthiness, and all the other peers refer to the single reputation value. In a local trust scheme, each peer could be rated differently by different peers.

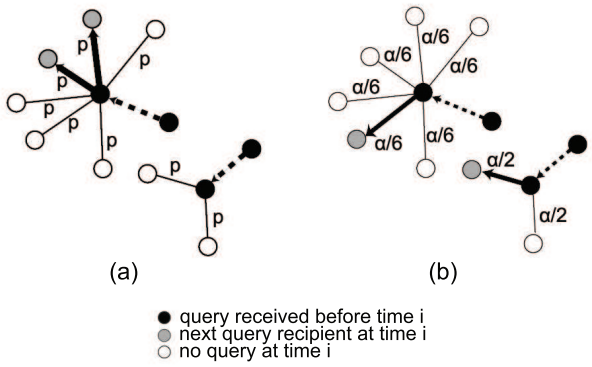

Figure 1: Forwarding probabilities of a node: (a) PS and (b) APS. In (b), α is tunable constant, here is set to 1.

There are several studies on building a global trust in distributed systems. Kamvar *et al.* propose a distributed and secure method, called EigenTrust, to compute global trust values based on local information [15]. They also show that their reputation system works well, even when malicious peers cooperate under various scenarios. Golbeck *et al.* propose a reputation inference algorithm used in scoring emails [16].

In order to share information in a collaborative scheme, efficient search and dissemination mechanisms are crucial. Since the performance of a search algorithm heavily depends on the network's topology, it is important to know of the topology information in order to understand search algorithms. It has been reported that Gnutella networks and email networks have power-law degree distributions. Recent analysis of a blog network in Poland also reveals that the degree distributions of blog networks follow power-law with exponents between 2 and 3 for both the incoming and outgoing edges [17].

Various search algorithms have been suggested for P2P systems. Gnutella, one of the oldest P2P file sharing programs, operates on a query flooding protocol, and scales poorly, as the network size grows. Alternatively, a random walk, iterative deepening, and a percolation search method have been suggested as a mitigating solution to heavy traffic from flooding [18–21]. The random walk algorithm generates much less traffic than flooding, but the success rate is rather low and also has a large variance. The iterative deepening, a variant of flooding, certainly reduces the traffic of original flooding, but is ineffective in that it has to visit a large number of peers. The percolation search algorithm utilizes the content replication strategy in P2P systems for both the content and the query [21, 22]. It is also known to exploit the property of scale-free networks that percolation takes place with a very low percolation probability [23].

The percolation search consists of three key concepts: content implantation, query implantation, and bond percolation. Every node in a network takes a short random walk and caches desired information to be shared on the visited nodes (i.e., *content implantation*). When a node initiates

a query, it first executes a short random walk and implants the query to each visited node (i.e., *query implantation*). Finally, parallel probabilistic broadcasts are started with the implanted queries (i.e., *bond percolation*); when a peer receives a query, it forwards the received query to all its neighbors with probability p (see Figure 1(a)), except to the one who sent the query. When p is larger than the *percolation threshold*, p_c, the query propagates through the entire network; when $p < p_c$, the query dies out before reaching the entire network. In power-law networks of a finite size, the percolation threshold approaches 0. The percolation threshold, p_c, can be calculated from a degree distribution, as $p_c = \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle}$. Here, k stands for the degree of a node, and the notation $\langle ... \rangle$ means the average over the degree distribution [24].

2.2 Existing Solutions to Fight Blog Spam

There are more than 20 plug-ins to identify spams in WordPress [25] and several solutions for MovableType [26], two of the most popular blog platforms. For example, requiring login before writing comments, Captcha turing test, HashCash [27], a Bayesian filter, blacklisting or whitelisting, and language model approach [28] have been proposed to block blog spam.

Some of them, such as Captcha turing test, are very successful for now, but the bottom line for these stand-alone spam filters is that it is possible to break down these system in principle. Moreover, since there are ambiguous spams that should be determined by a human, although a stand-alone spam filters is almost perfect, people will not be freed completely from the spam deletion process. In contrast, each user's intervention can be reduced to negligible amount if the number of users in the collaboration becomes very large in collaborative scheme. In ideal case, only one user's manual identification of a spam should set all others free. In other words, the collaborative method adds another dimension that cannot be achieved by stand-alone methods.

3. SYSTEM DESIGN & PROTOCOL

Our blog spam filtering method is based on manual identification of link spam and sharing this newly acquired information with others through a network of trust.

3.1 Link Spam Identification

In our scheme, a spam is first identified by a user manually and then stored in individual databases of blogs. When blog owners sees spams in their blog, they select spam links in the message, and then the IP addresses of spam links is stored.

3.2 Adaptive Percolation Search (APS)

When a new comment or trackback is added to a blog, it triggers the blog system to send out a query to its neighbors. This query propagates the blog network according to our new adaptive percolation search. The performance of the original percolation search (PS) is very dependant on the percolation probability. If the percolation probability is set too high above the percolation threshold, the entire network is flooded with search traffic. In order to prevent overloading the network with search traffic, the complete node degree distribution of the network must be known before the search, and percolation threshold must be calculated. A relevant work is done by Kong *et al.* [14], where they search for the correct percolation probability in unit increment starting from a very small percolation probability. However, their algorithm is dependent on the magnitude of unit increment for speed and accuracy, and cannot achieve both.

We suggest adaptive modification called **adaptive percolation search (APS)**. It takes advantage of well-controlled traffic in the random walk based algorithms and the effectiveness and tunability of the percolation search algorithms. In contrast to a static percolation probability as in the simple PS, a peer with degree k in APS forwards a query with probability $\frac{\alpha}{k-1}$ (α is a tunable constant usually set to 1) to its neighbors except to the one who has sent the query. Each peer receiving a query will forward it to its own peers. However, the query forwarding probability is adjusted at every node according to the node degree. This adaptive nature of our algorithm allows percolation without global knowledge about the network and with zero probability for broadcast-like traffic. We compare the PS with our method experimentally later in §4.2.

Another property is self-avoiding dynamics. If a query arrives at a blog peer, which has already received the same query before, the peer drops the query. Thus, we prevent our network from flooding without resorting to Time To Live (TTL)[1]-based approach. One drawback of the self-avoiding walk is its attrition of paths [29], but we believe the implantation process and periodic query compensates for this weakness. Furthermore, our algorithm's performance can be finely tuned using α, in contrast to random walk based algorithms. Here, α represents the average number of new queries forwarded from a single query. We can increase the reliability of query delivery in the presence of off-line peers. APS might require more hops than PS until it reaches a node that has information about the identified spam, which may result in increased time to detect a spam. However, such a delay in blog spam identification is permissible and not as detrimental as in the case of file sharing which needs immediate responses.

3.3 Trust Management

For our APS, we have assumed a undirected (or bidirectional) network of trusted users. In cyber communities, such as Orkut[2] and Cyworld[3], online friendships are bidirectional; both users must acknowledge the friendship.

Most blog platforms provide a feature called *blogroll* (or *blog link*), which enables a blogger to add a link toward a friendly blog in his or her blog. A blogroll reflects a real trust relation unlike comments or trackbacks where a random set of blogs may have links between them. However, percolation search schemes are known to work poorly in directed graphs. Moreover, trust in a collaborative system should be mutual, i.e., if A believes B, then B should also believes A.

Thus we propose a new relation, *trustroll*. Unlike blogroll which is directional, trustroll is undirectional, established manually by each other's acceptance. Additionally, each blogger can manage the number of neighbors and the amount of traffic because of undirectionality of the relation. Although there is no such relation as a trustroll currently, we expect that it can be easily constructed between peers that already have a blogroll, as existing online relationships al-

[1]The TTL value can be thought as an upper bound of hops on the forwarding of query.
[2]Orkut. http://www.orkut.com
[3]Cyworld. http://cyworld.nate.com

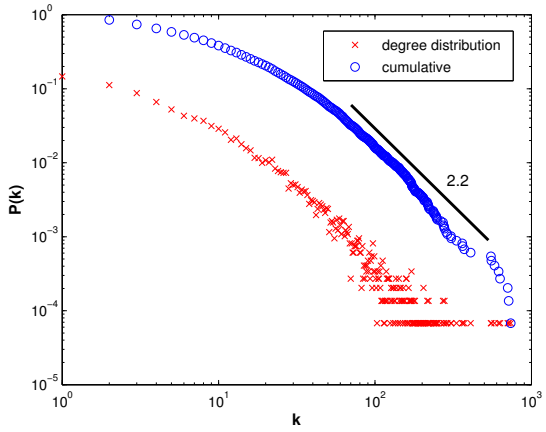

Figure 2: Degree distribution of the Egloos blogroll network.

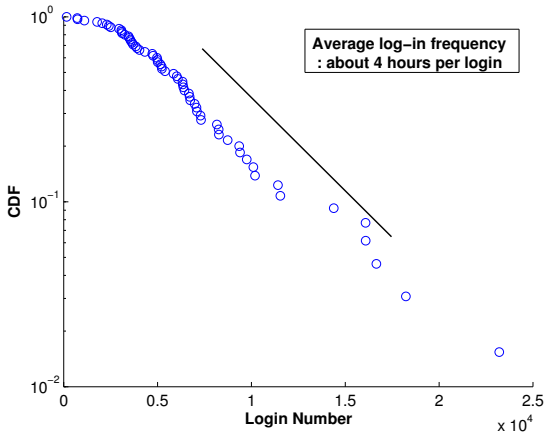

Figure 3: CDF of number of users versus the number of logins in the telnet-based private board system in KAIST from 2003 to 2005.

ready manifest the characteristics.

Trustroll relations are assumed to be transitive: if A trusts B and B trusts C, then A does C. Note that the transitive relation cannot be assured to be concrete for the distant pairs. Several approaches may assist our trustroll. We may use whitelist or blacklist approach to allow a predefined set of links as benign (or malicious), or introduce a higher threshold in determining spams. In this paper, we limit ourselves to the simplest case where we trust any blogs from the trustroll network.

3.4 Periodic and Asynchronous Query Schemes

A blog system is always on, thus collaboration among the peers is possible at all times. If a peer sends a query only once upon the arrival of a message, the query might reach other blogs before any blogger has identified the identical message as a spam.

Therefore, queries should be sent periodically, or all the blogs should keep received queries in their own database for a while. We choose to use periodic queries, intermediate peers drop queries after checking and forwarding it.

In an asynchronous scheme, intermediate peers keep every query for a certain period of time or until the answer to the query is made available.

Each blog which received a query, only if the link of the query is identified as spam, sends reply to the blog who sent out the query originally. Then, the blog which sent the query originally, collects those replies, and if the number of replies exceeds a threshold, classify the queried message as a spam.

In order to reduce the communication overhead, multiple queries can be put together into a single request.

4. SIMULATION

In this section, we evaluate our blog spam filtering method based on the following three metrics: the mean and the maximum times to identify and delete a spam, the percentage of spams deleted automatically by our method without bloggers' intervention, and the communication overhead from collaboration.

As there does not exist a good publicly available blog network, we have used a crawler to capture a popular portal-site blog network called Egloos[4] in South Korea. The crawling is based on the snowball sampling technique. The basic idea of snowball sampling is to randomly select a seed node and follow blogrolls of the seed node, then their neighbors, until all the neighbors are visited. We only focus on the blogs that are in the same cluster as the seed node.

As the blogroll network is a directed graph and we need an undirected graph for our collaborative blog spam filtering method, we convert the directional edges to undirected edges and create a trustroll network. To check whether this procedure is relevant or not, we calculate the *link reciprocity*. Link reciprocity estimates the extent of the network's links which are bidirectional in comparison with a random network with the same link density [30]. The link reciprocity of captured network is 0.4. Although it is not near 1.0, it is larger than that of the email network made by the address books.

The numbers of blogs and undirected edges in the captured Egloos blog network are $14,738$ and $109,531$, respectively. The node degree distribution is shown in Figure 2, which roughly follows a power-law distribution. Such a finding is consistent with the result from a blog network in Poland [17].

While we are able to capture a blog network in use, we have no data on bloggers' behavior (e.g., how often bloggers log in, check updates on their blogs, and remove spam comments or trackbacks). For realistic representation of bloggers' behavior, we use login statistics of a telnet-based bulletin board system (BBS), loco.kaist.ac.kr, from KAIST. In the pre-blog days, the Loco BBS provided private bulletin boards to thousands of individuals and played a similar role as today's blog system. Figure 3 plots the cumulative distribution function of the number of logins per each user during a two-year time period from 2003 to 2005. The figure shows that the login frequency follows an exponential distribution, which is later used in our simulation.

4.1 Simulation Setup

We begin with a blog network of size $N = 14,738$. We assume that a certain ratio of blogs, P_s, are initially spammed

[4]OnNet Co., http://www.egloos.com

Table 1: Parameter settings for simulation

Parameter	Notation	Default Value
Density of identical spams	P_s	0.05
Average checking time	h	4 hours
Query period	q	20 minutes
Threshold for hit counts	th	1
Limit of checking time	T	24 hours

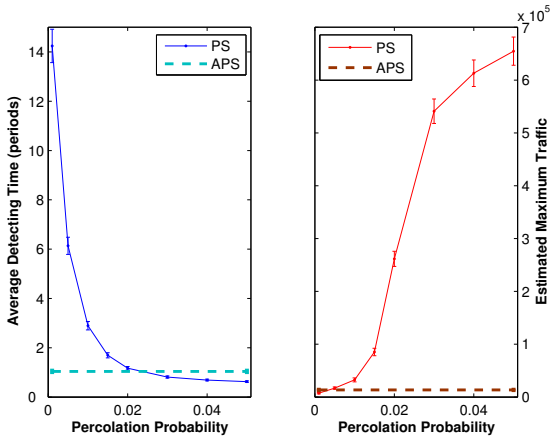

Figure 4: Comparison between PS and APS in terms of the average spam detecting time and the estimated traffic overhead versus percolation perobability.

with an identical link spam; spams are only assigned at time $t = 0$ and are not assigned after then. There are two spam deletion processes. One is a manual deletion by individual bloggers, and the other is by our collaborative spam filter automatically.

At every time tick (1 minute in our simulation), each blogger i is assumed to log in and delete spams at regular intervals, h_i, of which period is determined from the empirical exponential distribution in Figure 3. This process mimics manual deletion of spams.

On the other hand, our collaborative spam filter works as follows. At time $t = 0$, blogs, upon receiving a suspicious comment or a trackback, initiate a periodic query to its neighbors to verify whether the received message is indeed a spam. To verify a message as a spam, our spam filter requires at least th number of peers who return acknowledgements (hit messages). Initial starting time of the periodic query is randomly chosen between 0 and $q - 1$, where q is the query period. The message is considered as a non-spam message and the spam filter stops generating queries about it, if it is undetermined until time T passes. At the end of the simulation, we count the number of nodes with any spam message in their blogs, which are false negative cases (i.e., automatic spam filter has not yet identified and deleted the spam). The smaller this number is, the more effective our spam filter.

Every simulation is averaged 100 to 1,000 runs. The default parameters used in our simulation are shown in Table 1.

4.2 Analysis

Table 2: Maximum and average times to detect spam (min)

Period	**w/o col**	5	10	20	30	40
Max	**78935**	60.0	109.8	196.8	273.4	324.1
Avg	**2160**	7.4	12.2	20.8	28.6	35.7

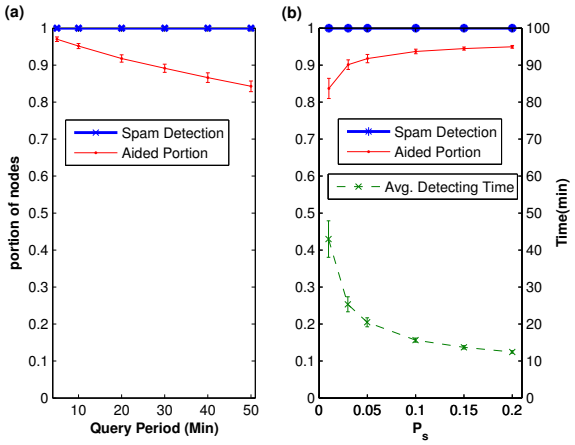

Figure 5: (a) The covering performance of our APS algorithm vs. quering period. We measure the spam detection ratio, the number of detected spams divided by total number of spams, and the portion of aided nodes, the portion of blogs in which the spam is deleted by our system not by the blogger of the blog. (b) Our algorithm's performance under various number of identical spams in the network. Note that the spam detection rate is always 1, and the performance also increases while the number of spams increases.

Figure 4 shows the contrast between APS and the sensitivity of PS to the percolation probability p. In Table 2, we compare our scheme's performance to the case that no collaboration is used and each blogger has to identify a spam individually. We note that average spam detection time is reduced by three orders of magnitude when the periodic query is sent out every five minutes. Even when the query is sent out every 40 minutes, we still observe two orders of magnitude reduction in the average and maximum time of detection.

Figure 5 presents the percentage of total detected spams and automatically detected spams, as we vary the query period and the spam density. Throughout wide parameter range, most of the spams are detected and at least 80% of spams are automatically deleted by our spam filter. The rest of the spams are deleted manually. This is due to the fact that there are users who visit their blogs very frequently, leaving no time for our automatic spam filter to delete the spam. Since collaborative scheme exploits the abundance of spams, our methods works better for spams with higher density.

As we assume that everyone in our trustroll network reports spams correctly, there is no false positive case in our simulation results.

In order to estimate the communication overhead, we count

the number of messages including queries and hit messages that are used by collaborative spam filter. When 18.6%[5] nodes send out queries at the same time, the number of messages at the most crowded node was 158.85 per second at the peak. If we assume the query size is 1KB, this upper bound traffic load is equivalent to 0.158Mb/s. Since the length of IP address is 32bits, the query size might be a lot less than 1KB. For a typical fast internet connection of 100Mb/s, this represent 0.16% bandwidth cost. The average traffic was less than 1Kb/s, substantially lower than the upper bound.

5. CONCLUSION AND DISCUSSION

In this paper, we have introduced a new collaborative spam filtering scheme to block link spams in user-hosted blog systems, which is based on a simple trust management scheme and (periodic or asynchronous) querying called by an effective adaptive search algorithm. All collaborative schemes including our approach can work as an augmentation of any other stand-alone methods. We have shown the efficiency of our scheme by numerical simulations carried on a real-world blog network constructed by blogroll. Our approach is easy and straightforward to implement as a plug-in to any existing blog platforms.

We note that our approach has a limitation in that it is only adequate for user-hosted blogs (e.g., WordPress or Movabletype), but not for developer-hosted blogs (e.g., Bloggers or LiveJournals). A developer-hosted blog system can be assisted with spam filters that directly check the central blog database and detect spams based on the occurrence of identical messages in the hosted blogs. However, we think that our trust building scheme can be also applied to those blog systems.

As future work, we plan to do more explicit mathematical analysis of adaptive percolation search and implement our algorithm for popular blog systems.

6. ACKNOWLEDGEMENTS

This work was supported by grant No. R01-2005-000-1112-0 from the Basic Research Program of the Korea Science & Engineering Foundation. We thank Meeyoung Cha for valuable comments and careful proofreading.

7. REFERENCES

[1] J. Postel. On the junk mail problem. RFC706, 1975.
[2] Messagelabs. http://messagelabs.com.
[3] Wikipedia. http://en.wikipedia.org/wiki/blog.
[4] Technorati. http://technorati.com.
[5] Z. Gyöngyi and H. Garcia-Molina. Link spam alliances. In *Proceedings of VLDB*, 2005.
[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
[7] Nofollow. http://googleblog.blogspot.com/2005/01/preventing-comment-spam.html
[8] Nonofollow. http://nonofollow.net.
[9] SpamProbe. http://spamprobe.sourceforge.net.
[10] SpamAssassin. http://spamassassin.apache.org.
[11] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. P2P-based collaborative spam detection and filtering. In *Proceedings of Peer-to-Peer Computing*, 2004.
[12] F. Zhou, L. Zhuang, B. Y. Zhao, L. Huang, A. D. Joseph, and J. Kubiatowicz. Approximate object location and spam filtering on peer-to-peer systems. In *Proceedings of Middleware*, 2003.
[13] SpamNet. http://cloudmark.com.
[14] J. S. Kong, P. O. Boykin, B. A. Rezaei, N. Sarshar, and V. P. Roychowdhury. Let your cyberalter ego share information and manage spam, 2005.
[15] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *Proceedings of WWW*, 2003.
[16] J. Golbeck and J. Hendler. Reputation network analysis for email filtering. In *Proceedings of the First Conference on Email and Anti-Spam* , 2004.
[17] W. Bachnik, S. Szymczyk, P. Leszczynski, R. Podsiadlo, E. Rymszewicz, L. Kurylo, D. Makowiec, and B. Bykowska. Quantitive and sociological analysis of blog networks, in preprint, 2005.
[18] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Physics Review E*, 64, 2001.
[19] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the international conference on Supercomputing*, New York, NY, USA, 2002. ACM Press.
[20] B. Yang and H. Garcia-Molina. Efficient search in peer-topeer networks. In *Proceedings of IEEE ICDCS*, 2002.
[21] N. Sarshar, P. O. Boykin, and V. P. Roychowdhury. Percolation search in power law networks: Making unstructured peer-to-peer networks scalable. In *Proceedings of Peer-to-Peer Computing*, 2004.
[22] Z. Jia, B. Pei, M. Li, and J. You. A comparison of spread methods in unstructured P2P networks. *ICCSA (3)*, 2005.
[23] S. Bornholdt and H. G. Shuster. *Handbook of Graphs and Networks*. WILEY-VCH, 2003.
[24] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.
[25] WordPress. http://wordparess.org.
[26] MovableType. http://www.sixapart.com/movabletype/.
[27] HashCash. http://elliottback.com/wp/archives/2005/10/23/wordpress-hashcash-30-beta/.
[28] G. Mishne, D. Carmel, and R. Lempel. Blocking blog spam with language model disagreement. In *Proceedings of AIRWeb*, 2005.
[29] C. P. Herrero. Self-avoiding walks on scale-free networks. *Physical Review E*, 71, 2005.
[30] D. Garlaschelli and M. I. Loffredo. Patterns of link reciprocity in directed networks. *Physics Review Letter*, 93, 2004.

[5]From a survey on Egloos blogs, 18.6% nodes received at least one comment or trackback in a day.