

블로그 스팸 분석과 적응적 스미기 검색을
이용한 블로그 스팸 협동 필터

Analysis of Blog Spams and Collaborative Blog
Spam Filtering Using Adaptive Percolation
Search



Analysis of Blog Spams and Collaborative Blog
Spam Filtering Using Adaptive Percolation
Search

Advisor : Professor Moon, Sue Bok

by

Han, Seungyeop

Department of Electrical Engineering and Computer Science

Division of Computer Science

Korea Advanced Institute of Science and Technology

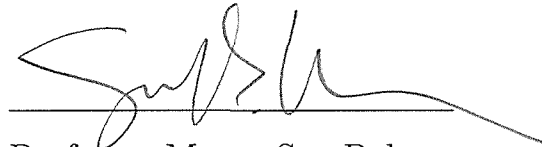
KAIST

A thesis submitted to the faculty of the Korea Advanced
Institute of Science and Technology in partial fulfillment of the
requirements for the degree of Master of Engineering in the
Department of Electrical Engineering and Computer Science,
Division of Computer Science

Daejeon, Korea

2006. 12. 12.

Approved by



Professor Moon, Sue Bok

Advisor

블로그 스팸 분석과 적응적 스팸기 검색을 이용한 블로그 스팸 협동 필터

한 승 엽

위 논문은 한국과학기술원 석사학위논문으로 학위논문심사
위원회에서 심사 통과하였음.

2006년 12월 12일

심사위원장 문 수 복

심사위원 송 준 화

심사위원 Cheong, Otfried



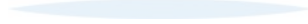
MCS 한 승 엽. Han, Seungyeop. Analysis of Blog Spams and Collaborative Blog
20053633 Spam Filtering Using Adaptive Percolation Search. 블로그 스팸 분석과 적응
적 스미기 검색을 이용한 블로그 스팸 협동 필터. Department of Electrical
Engineering and Computer Science, Division of Computer Science . 2007. 22p.
Advisor Prof. Moon, Sue Bok. Text in English.

Abstract

We provide basic analysis on blog spams, which little is known about before. Then, 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.



KAIST



Contents

Abstract	i
Contents	iii
List of Tables	v
List of Figures	vi
1 Introduction	1
2 Preliminaries	3
2.1 Collaborative Spam Filtering Scheme	3
2.2 Existing Solutions to Fight Blog Spam	5
3 Analysis of Blog Spam	7
3.1 Dataset	7
3.2 Analysis	8
4 System Design & Protocol	11
4.1 Link Spam Identification	11
4.2 Adaptive Percolation Search (APS)	11
4.3 Trust Management	12
4.4 Periodic and Asynchronous Query Schemes	13
5 Simulation	14
5.1 Simulation Setup	15
5.2 Analysis	16
6 Conclusion and Discussion	19
Summary (in Korean)	20



List of Tables

3.1	Summary of the dataset	7
3.2	Top 10 ASes which sent most spams	9
3.3	Top 10 ASes which have most spamming IPs	9
5.1	Parameter settings for simulation	16
5.2	Maximum and average times to detect spam (min)	17



List of Figures

2.1	Forwarding probabilities of a node: (a) PS and (b) APS. In (b), α is tunable constant, here is set to 1.	4
3.1	The number of spams during collection period.	8
3.2	Distribution of the number of spams that each host sent.	10
3.3	Distribution of the number of spamming IPs for each target web host.	10
5.1	Degree distribution of the Egloos blogroll network.	14
5.2	CDF of number of users versus the number of logins in the telnet-based private board system in KAIST from 2003 to 2005.	15
5.3	Comparison between PS and APS in terms of the average spam detecting time and the estimated traffic overhead versus percolation perobability.	17
5.4	(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.	18

1. Introduction

Spams or unsolicited bulk emails have been a pressing problem ever since the very beginning of the Internet [19]. 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 articles¹. 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 months². With such rapid growth, the number of blog spams has also increased to a problematic level. According to Akismet³, as of November 2006, 93% of messages to blogs using Akismet spam filtering plugin are spams.

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 [10]. 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 link-based ranking algorithms, such as PageRank [4], many search engines use.

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 [24] and SpamAssassin [22]). 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

¹<http://en.wikipedia.org/wiki/blog>

²<http://www.technorati.com/weblog/2005/08/34.html>

³<http://akismet.com/stats/>

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 [5, 26, 23, 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 spam filtering scheme can effectively block link spams by sharing the information of the spam link.

Recently, there has been research on the statistics of spam web pages [6], the spam farms [10], and method for identifying spam farms targeting PageRank [9]. Those spam web pages can be target web pages of blog spams to boost their rank. However, it has been still little known about blog spams themselves.

In this thesis, we provide characteristics of link spams, such as, a distribution of IP addresses from where these link spams came and the characteristics of target webpages of link spams. Then, we propose using a collaborative spam filter to block link spams in blog systems and 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 Chapter 2, we survey related works in the area of blocking blog spams and present backgrounds on the blog networks. In Chapter 3, we provide analysis on blog spams. Then, we describe our collaborative spam filtering approach in detail in Chapter 4. In Chapter 5, we perform a simulation of our method based on a real-world blog network. Finally, we conclude in Chapter 6.

2. Preliminaries

In this chapter, 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 [23]). 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 [26] 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 [5], 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.

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 [13]. They also show that their reputation system works well, even when malicious peers cooperate under various scenarios. Golbeck *et al.* propose

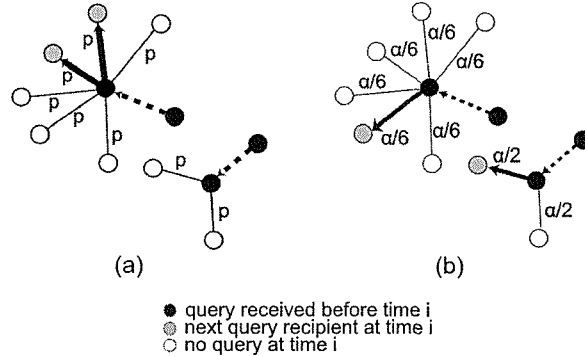


Figure 2.1: Forwarding probabilities of a node: (a) PS and (b) APS. In (b), α is tunable constant, here is set to 1.

a reputation inference algorithm used in scoring emails [8].

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 [2].

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 [1, 15, 25, 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, 12]. It is also known to exploit the property of scale-free networks that percolation takes place with a very low percolation probability [3].

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 2.1(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 $\langle \dots \rangle$ means the average over the degree distribution [17].

2.2 Existing Solutions to Fight Blog Spam

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 results¹. Although using the nofollow attribute may be effective in decreasing the level of pollution by link spams in search results, 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 debate².

There are several plug-ins to identify spams in WordPress³ and several solutions for MovableType⁴, two of the most popular blog platforms. For example, requiring login before writing comments, Captcha turing test, a Bayesian filter, blacklisting or whitelisting, 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. One of the most popular

¹<http://googleblog.blogspot.com/2005/01/preventing-comment-spam.html>

²<http://nonofollow.net>

³<http://wordpress.com>

⁴<http://movabletype.com>

plug-ins is Akismet, which sends new comments and trackbacks to the Akismet server when they come and identify them by the results from the server. Although Akismet shows good performance, it has a risk due to single-point of failure, since it uses centralized approach.

Mishne *et al.* proposed a method using language model disagreement [16]. It compares the language models of comments and the blog posts, then if they disagree each other, then those comments are classified as spams. They show some promising results, however, false positive of the method cannot be negligible.



3. Analysis of Blog Spam

In this chapter, we provide preliminary result of analysis on the blog spams collected at the border router of KAIST campus network.

3.1 Dataset

Inside the campus, there are several web servers which contain blogs. Especially, isloco.com¹ serves more than 200 blogs using TatterTools², one of the most popular blog tools in South Korea. As those blogs expose to search engines, they also have become targets of the blog spams.

For the purpose of campus area network measurement, KAIST has a network monitoring tool box, DAGMON³, at the border router connecting to Internet. Every packet which comes into or goes out from KAIST passing the border router can be captured by the monitoring tool.



Table 3.1: Summary of the dataset

	Trace 1	Trace 2
Start time	2006-11-10 18:00	2006-11-18 15:15
End time	2006-11-16 20:30	2006-11-25 03:12
Duration	148 hrs 30 mins	155 hrs 57 mins
# of Legitimate comments	130	88
# of Spam comments	10106	13497
# of Legitimate trackbacks	1	3
# of Spam trackbacks	17019	41992
# of Distinct spamming IPs	6317	8381

We collected two traces including all packets heading to isloco.com blog server, about one week for each trace. Then, we filtered only flows which start with 'POST /comment/add' or

¹<http://isloco.com>

²<http://tattertools.com>

³<http://endace.com>

‘POST /trackback’, methods for writing a comment and sending a trackback to TatterTools blog, respectively.

Table 3.1 describes the summary of two traces. Note that these traces do not contain packets transmitting between hosts both inside the campus, because they have been captured at the border router. As most users of isloco are inside KAIST campus, these traces do not contain legitimate messages from those users. In order to identify spams, we have checked first trace manually. Because there is no English legitimate message in the first trace, (i.e., comments and trackbacks which have no Korean character were all spams, owing to the fact that all of users in the blog server are Korean,) we check only messages contain Korean character for the second trace. It might miss some legitimate messages in English, however, missings would not affect analysis on blog spams significantly.

3.2 Analysis

As we described in Table 3.1, only 0.9% of comments and 0.006% of trackbacks are legitimate among all message from outside of campus. Figure 3.1 depicts the number of spams during collection period.

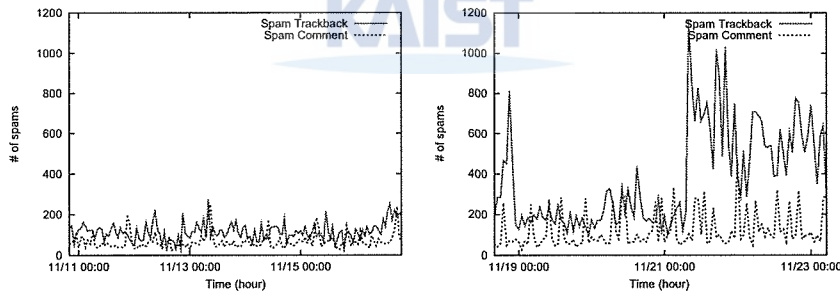


Figure 3.1: The number of spams during collection period.

In order to figure out where these spams come from, we surveyed top 10-ASes which sent most spams with their primary countries. We used BGP snapshots from Route View Project [18] to associate IP addresses with AS numbers. Table 3.2 shows top 10 ASes which sent most spams, and Table 3.3 shows top 10 ASes which have most spamming IP addresses. In contrast to the case of email spam [20], which mostly originate from ASes in United States, blog spams are sent from diverse countries.

As shown in Figure 3.2, more than 50% of spamming hosts sent spams at least twice. Notably, top-10 hosts sent 21264 spams, it’s more than 25% of total number of spams.

Table 3.2: Top 10 ASes which sent most spams

AS Number	# of Spam	AS Name	Primary Country
3786	4936	DACOM Corporation	Korea
7470	4621	ASIA INFONET Co.,Ltd	Thailand
31103	3206	Keyweb AG	Germany
9105	2367	Tiscali UK	United Kingdom
9583	1980	Sify Limited	India
30315	1915	Everyones Internet, Inc	United States
4766	1681	Korea Telecom	Korea
7132	1454	SBC Internet Services	United states
17676	1388	BLOCK Japan Network Information Center	Japan
8551	1308	BEZEQINT	Israel

Table 3.3: Top 10 ASes which have most spamming IPs

AS Number	# of Unique Spamming IPs	AS Name	Primary Country
7132	736	SBC Internet Services	United States
4766	436	Korea Telecom	Korea
3320	401	Deutsche Telecom AG	Germany
20115	344	Charter Communications	United States
28573	299	NET Servicos de Comunicacao S.A.	Brazil
8551	294	BEZEQINT	Israel
17676	260	BLOCK Japan Network Information Center	Japan
8167	238	Brasil Telecom S.A.	Brazil
9829	234	BSNL-NIB	India
9318	217	Hanaro Telecom	Korea

Figure 3.3 depicts the distribution of the number of spamming IPs for each target web host. Here, target web host means addresses which belong in spam messages to be exposed as hyperlinks when spams are written in blogs. More than 30% of target hosts associated with at least two spamming IP addresses. Note that there is a spam web host associated

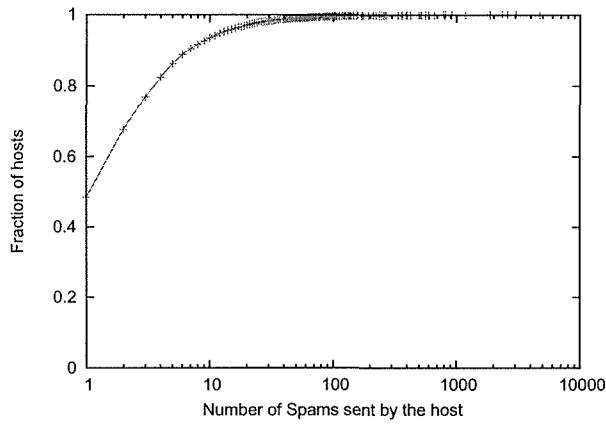


Figure 3.2: Distribution of the number of spams that each host sent.

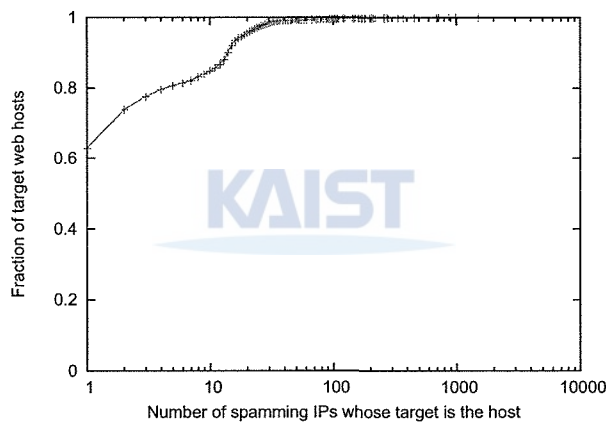


Figure 3.3: Distribution of the number of spamming IPs for each target web host.

to more than 100. Since the purpose of link spams is to boost the ranking of the target web pages, information about target web pages are valuable to share, in addition to the IP addresses of senders. While spammers can easily change spamming IP addresses, the target web page should remain in order to have boosted effect. Hosts which have more than 200 IP addresses are usually benign hosting servers, such as blogspot.com, who offer blogs to general users. In order to avoid blocking every message from those benign hosting servers, a user id of the target page in the hosting server should be shared, too.

4. 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.

4.1 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 see spams in their blog, they select spam links in the message, and then the IP addresses of spam links is stored.

4.2 Adaptive Percolation Search (APS)

When a new comment or traceback 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 §5.2.

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 [11], 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.

4.3 Trust Management

For our APS, we have assumed a undirected (or bidirectional) network of trusted users. In cyber communities, such as Orkut² and Cyworld³, 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 already manifest the characteristics.

Trustroll relations are assumed to be transitive: if A trusts B and B trusts C , then A does

¹The TTL value can be thought as an upper bound of hops on the forwarding of query.

²orkut. <http://www.orkut.com>

³Cyworld. <http://cyworld.nate.com>

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.

4.4 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.

5. Simulation

In this chapter, 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.

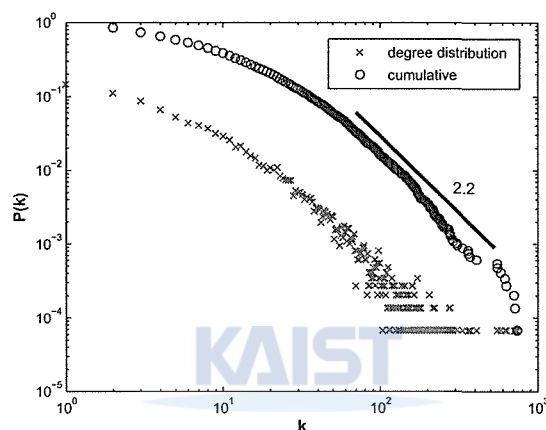


Figure 5.1: Degree distribution of the Egloos blogroll network.

As there does not exist a good publicly available blog network, we have used a crawler to capture network from a popular portal-site blog, called Egloos¹ 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 [7].

¹OnNet Co., <http://www.egloos.com>

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 5.1, which roughly follows a power-law distribution. Such a finding is consistent with the result from a blog network in Poland [2].

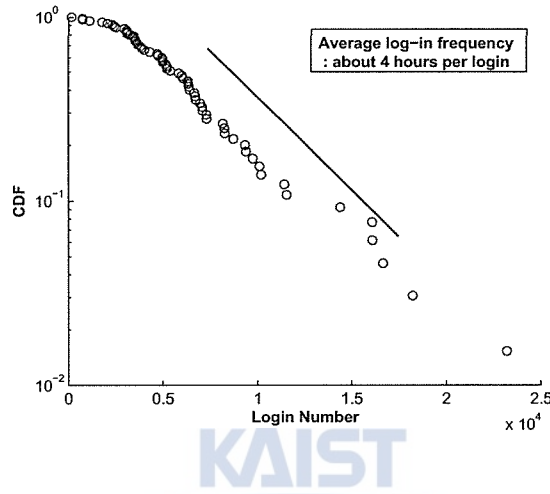


Figure 5.2: CDF of number of users versus the number of logins in the telnet-based private board system in KAIST from 2003 to 2005.

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 5.2 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.

5.1 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 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 5.2. 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 traceback, 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 5.1.



Table 5.1: 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

5.2 Analysis

Figure 5.3 shows the contrast between APS and the sensitivity of PS to the percolation probability p . In Table 5.2, 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,

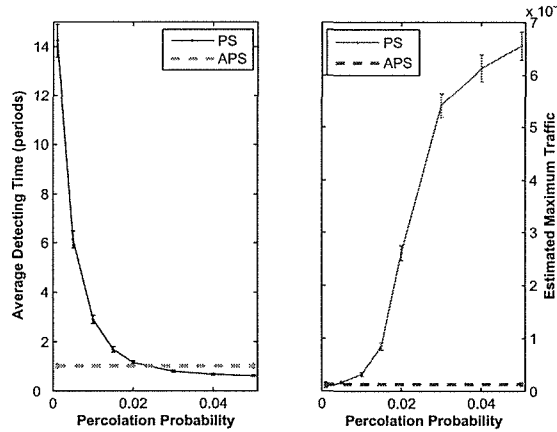


Figure 5.3: Comparison between PS and APS in terms of the average spam detecting time and the estimated traffic overhead versus percolation probability.

Table 5.2: 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

we still observe two orders of magnitude reduction in the average and maximum time of detection.

Figure 5.4 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%²

²From a survey on Egloos blogs, 18.6% nodes received at least one comment or trackback in a day.

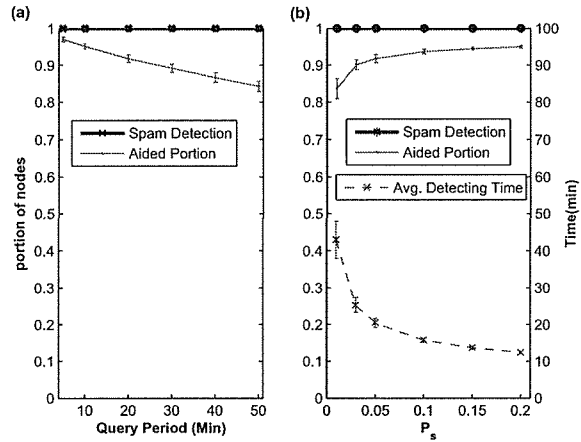


Figure 5.4: (a) The covering performance of our APS algorithm vs. querying period. We measure the spam detection ratio, the number of detected spams divided by total number of spams, and the portion of aided nodes, the portion of blogs in which the spam is deleted by our system not by the blogger of 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.

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.

6. Conclusion and Discussion

In this paper, we have provided the result of analysis on blog spams. In contrast to email spams, blog spams originate from various countries.

Then, 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.

요 약 문

블로그 스팸 분석과 적응적 스미기 검색을 이용한 블로그 스팸 협동 필터

블로그 스팸은 링크스팸의 일종으로 최근 그 수가 크게 증가하고 있다. 이 논문에서는 지금까지 알려지지 않았던 블로그스팸에 대한 초기 분석결과를 제시한다. 블로그 스팸은 다양한 나라에서 오고 있으며 단순한 블랙리스트만으로는 효율적이지 않음을 보인다. 또한 링크 스팸을 막기위한 새로운 협동필터를 제안하고 있다. 이 방식은 스팸을 수동으로 분류한 후에 이 정보를 믿을 수 있는 사람들간의 네트워크를 통해 공유하도록 한다. 새로운 메시지를 받으면 적응적 스미기 검색을 통해 많은 트래픽을 유발하지 않고서도 빠른 시간 내에 스팸정보를 네트워크로 부터 찾아낼 수 있다.



References

- [1] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *CoRR*, cs.NI/0103016, 2001.
- [2] W. Bachnik, S. Szymczyk, P. Leszczynski, R. Podsiadlo, E. Rymaszewicz, L. Kurylo, D. Makowiec, and B. Bykowska. Quantitative and sociological analysis of blog networks. *ACTA PHYSICA POLONICA B*, 36:2435, 2005.
- [3] S. Bornholdt and H. G. Shuster. *Handbook of Graphs and Networks*. WILEY-VCH, 2003.
- [4] 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.
- [5] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. P2P-based collaborative spam detection and filtering. In *Peer-to-Peer Computing*, pages 176-183, 2004.
- [6] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In *WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases*, 2004.
- [7] D. Garlaschelli and M. I. Loffredo. Patterns of link reciprocity in directed networks. *Physical Review Letters*, 93:268701, 2004.
- [8] J. Golbeck and J. Hendler. Reputation network analysis for email filtering. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004.
- [9] Z. Gyöngyi, P. Berkhin, H. Garcia-Molina, and J. Pedersen. Link spam detection based on mass estimation. In *VLDB*, 2006.
- [10] Z. Gyöngyi and H. Garcia-Molina. Link spam alliances. In *VLDB*, 2005.
- [11] C. P. Herrero. Self-avoiding walks on scale-free networks. *Physical Review E*, 71, 2005.
- [12] Z. Jia, B. Pei, M. Li, and J. You. A comparison of spread methods in unstructured P2P networks. In *ICCSA (3)*, pages 10-18, 2005.

- [13] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *WWW*, pages 640–651, 2003.
- [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] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proceedings of the 16th international conference on Supercomputing*, pages 84–95, New York, NY, USA, 2002. ACM Press.
- [16] G. Mishne, D. Carmel, and R. Lempel. Blocking blog spam with language model disagreement. In *AirWEB*, 2005.
- [17] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167, 2003.
- [18] U. of Oregon Route Views Project. <http://www.routeviews.org/>.
- [19] J. Postel. On the junk mail problem, 1975. rfc706.
- [20] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *SIGCOMM*, 2006.
- [21] N. Sarshar, P. O. Boykin, and V. P. Roychowdhury. Percolation search in power law networks: Making unstructured peer-to-peer networks scalable. In *Peer-to-Peer Computing*, pages 2–9, 2004.
- [22] SpamAssassin. <http://spamassassin.apache.org>.
- [23] SpamNet. <http://cloudmark.com>.
- [24] SpamProbe. <http://spamprobe.sourceforge.net>.
- [25] B. Yang and H. Garcia-Molina. Efficient search in peer-to-peer networks. In *Proc. of the 22nd IEEE ICDCS*, 2002.
- [26] F. Zhou, L. Zhuang, B. Y. Zhao, L. Huang, A. D. Joseph, and J. Kubiawicz. Approximate object location and spam filtering on peer-to-peer systems. In *Middleware*, pages 1–20, 2003.

감사의 글

우선 지도교수님이신 문수복 교수님께 깊은 감사를 드립니다. 학부때부터 많은 조언과 격려를 해주셨고 세심한 가르침으로 이끌어주신 교수님 덕분에 즐겁게 연구하고 공부할 수 있었습니다. 연구에 있어서 많은 도움과 조언을 주신 정하웅 교수님께도 이 자리를 빌어 감사 드립니다. 그리고 논문 심사를 맡아주시고 좋은 지적과 충고를 해주신 송준화 교수님과 오프리드 정 교수님께 감사드립니다. 전산학 연구에 대해서 많은 것을 깨닫게 해주신 홍세준 교수님께도 감사의 말씀을 드립니다.

지난 2년간 함께 연구한 용열형에게 깊은 감사를 드립니다. 처음 이 연구도 형의 제안으로 시작하게 되었고, 같이 연구하면서 정말로 많은 것을 배울 수 있었습니다. 석사생활 2년은 좋은 연구실 사람들을 만난 덕에 즐겁게 보낼 수 있었습니다. 연구실 동기로 많은 시간을 함께하고 서로를 위로하고 격려해준 태희누나와 두영이형에게 우선 고마움을 전하고 싶습니다. 항상 멋진 미영누나, 늘 모범이 되어준 종건이형, 만물박사 동기형, 세심하게 챙겨주시는 은진누나에게 감사를 드립니다. 부족한 저를 잘 따라주고 함께 공부하는 현우와 다양한 표정의 똑똑한 해운형, 든든한 몽남형과 항상 열심히 Toan에게도 감사를 드립니다. 지금은 졸업한 태호형과 승준누나, 그리고 미국에서 열심히 공부하고 있을 민경형에게도 감사의 마음을 전합니다. 짧은 기간이었지만 함께 연구실 생활을 했던 Frederik과 Nicolas에게도 감사를 드립니다. 연구실 여러 행사들을 함께하고 연구에 조언도 주고 토론도 하는 등 많은 도움을 주신 전길남 교수님 연구실의 준복형, 유성형, 덕희형, 상호형, 정호형, 건, 영재에게도 감사의 말을 전합니다.

전산과 생활은 학부때부터 함께 해온 동기들이 있어서 많은 위로가 되고 즐거웠습니다. 중근, 호성, 정은, 진성, 주혁, 상운, 승환형, 준희형, 장환, 영재, 정현, 윤섭, 지호, 희진, 은호, 동일, 미경에게 감사의 마음을 전합니다.

마지막으로 세상에서 가장 존경하고 사랑하는 부모님, 하나뿐인 동생 봉엽이, 그리고 늘 고맙고 사랑하는 유희에게 깊은 감사와 사랑으로 이 논문을 바칩니다.

Curriculum Vitae

Name : Seungyeop Han
Date of Birth : March 18, 1983
Birthplace : Seoul
E-mail : syhan@an.kaist.ac.kr

Educations

2005. 3. – 2007. 2. KAIST, Division of Computer Science (M.S.)
2001. 3. – 2005. 2. KAIST, Division of Computer Science (B.S.)
1999. 3. – 2001. 2. Kyounggi Science High School

Academic Activities



1. Haewoon Kwak, **Seungyeop Han**, Yong-Yeol Ahn, Sue Moon, Hawoong Jeong, *Impact of snowball sampling ratios on network characteristics estimation: A case study of Cyworld*, 정보과학회 제 33회 추계 학술대회, Seoul (Korea), Oct., 2006
2. **Seungyeop Han**, Yong-Yeol Ahn, Sue Moon, Hawoong Jeong, *Collaborative Blog Spam Filtering Using Adaptive Percolation Search*, Workshop on the Weblogging Ecosystem, Edinburgh (UK), May, 2006