ELiSyR: Efficient, Lightweight and Sybil-Resilient File Search in P2P Networks

Hyeong S. Kim[1], Eunjin(EJ) Jung[2] and Heon Y. Yeom[1]
[1] School of Computer Science and Engineering, Seoul National University
Seoul, Korea
[e-mail: {hskim,yeom}@dcslab.snu.ac.kr]
[2] Department of Computer Science, University of San Francisco
San Francisco, USA
[e-mail: ejung@cs.usfca.edu]
*Corresponding author: Hyeong S. Kim

Received February 5, 2010; revised July 9, 2010; accepted October 27, 2010; published December 23, 2010

Abstract

Peer-to-peer (P2P) networks consume the most bandwidth in the current Internet and file sharing accounts for the majority of the P2P traffic. Thus it is important for a P2P file sharing application to be efficient in bandwidth consumption. Bandwidth consumption as much as downloaded file sizes is inevitable, but those in file search and bad downloads, e.g. wrong, corrupted, or malicious file downloads, are overheads. In this paper, we target to reduce these overheads even in the presence of high volume of malicious users and their bad files. Sybil attacks are the example of such hostile environment. Sybil attacker creates a large number of identities (Sybil nodes) and unfairly influences the system. When a large portion of the system is subverted, either in terms of the number of users or the number of files shared in the system, the overheads due to the bad downloads rapidly increase. We propose ELiSyR, a file search protocol that can tolerate such a hostile environment. ELiSyR uses social networks for P2P file search and finds benign files in 71% of searches even when more than half of the users are malicious. Furthermore, ELiSyR provides similar success with less bandwidth than other general efforts against Sybil attacks. We compare our algorithm to SybilGuard, SybilLimit and EigenTrust in terms of bandwidth consumption and the likelihood of bad downloads. Our algorithm shows lower bandwidth consumption, similar chances of bad downloads and fairer distribution of computation loads than these general efforts. In return, our algorithm takes more rounds of search than them. However the time required for search is usually much less than the time required for downloads, so the delay in search is justifiable compared to the cost of bad downloads and subsequent re-search and downloads.

Keywords: P2p file sharing, Sybil-resilient, p2p file search, sybil attack, load balancing

A preliminary version of this paper appeared in OPODIS 2008, December 15-18, Luxor, Egypt. This research was supported by the MKE (The Ministry of Knowledge Economy), the Korean government, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency)" (NIPA-2010-(C1090-1011-0004))

DOI: 10.3837/tiis.2010.12.019

1. Introduction

Peer-to-peer (P2P) networks consume the most bandwidth in the current Internet and file sharing accounts for the majority of the P2P traffic. iPoque reports that 49% to 83% of the Internet traffic in file world-wide regions in August and September 2007 is P2P, and most of them are file sharing applications like eDonkey and BitTorrent [1]. BitTorrent self-claims to have more than 135 million users [2]. These numbers show that a P2P file sharing application must be efficient in bandwidth consumption.

Bandwidth consumption in file downloading is inevitable. However, bandwidth consumption in file search and bad downloads, e.g. wrong, corrupted, or even malware-infested file downloads, are overheads. In May 2008, a fake MP3 file contained Trojan horse and infected 27% of PCs among those under McAfee's monitoring [3]. In the same report, McAfee also notes that it detected more than half a million adware programs disguising as media files in less than a week. Downloading these files causes unnecessary bandwidth consumption. In this paper, we target to reduce these overheads even in the presence of high volume (even more than half in some cases) of malicious users and their bad files. Sybil attacks are the example of such hostile environment.

In many distributed systems, including P2P systems, a user is often not limited to one unique identity in the system but may create multiple identities with little cost. A malicious user may use this large number of identities (nodes) to unfairly influence the system. This is called the Sybil Attack. The "link spamming" attack to PageRank [4] is an example of Sybil attack, where a single user boosts his reputation by creating a large number of fake (Sybil) identities and giving himself good feedback [5]. Reputation systems based on peer reviews, such as the user rating in eBay, have been a major target of Sybil attacks. Fake (Sybil) accounts were created to boost the seller rating of the attacker or to damage the ratings of other sellers. A recent attack even uses bots to create Sybil identities for faster and larger-scale attacks [6]. There have been efforts to prevent Sybil attacks based on social networks, e.g. SybilGuard [7] and SybilLimit [8], but some of these efforts assume the knowledge of global topology [9], i.e. every user needs to know every other user's friends, some require a substantial work in the setup process [10], and some put unfairly large loads on high-degree nodes, i.e. users with many friends [7]. In P2P networks, it is unlikely that any user will make such sacrifice to serve other users.

In this paper, we present a new file search algorithm, ELiSyR that is Efficient, Lightweight, and Sybil-Resilient. 1) ELiSyR uses social networks for P2P file search and consumes less bandwidth (thus more efficient) than using the state-of-the-art anti-Sybil mechanisms for file search, SybilGuard and SybilLimit. In our simulation, ELiSyR consumes less than half of the total network traffic consumed by SybilGuard or SybilLimit. 2) ELiSyR is lightweight as only requires each user to maintain a local topology of the social network, namely the degrees of their immediate friends. This local topology may be obtained from other social networks, such as graph.facebook.com.. The local topology may also be obtained by piggybacking the degree information on the file downloads. 3) ELiSyR is also Sybil-resilient. In our simulation, users download from honest users more than 71% of the time even when more than half of the users are Sybil.

In the rest of this paper, we assume that the bad downloads happen due to malicious users' responding to search query with bad files. These bad files may be a different file from what

search query specified, or a malware-infested file. The bad downloads may also happen due to difficulties in search. A user may choose to download a file based on matching keywords when in fact the file is not what the user wanted. Search effectiveness in P2P downloads is out of this paper's scope and we focus on bad downloads with malicious intentions.

We compare ELiSyR to SybilGuard, SybilLimit and EigenTrust. EigenTrust [11] is a distributed reputation management system which provides each peer a unique global trust value based on the peer's history of uploads. We use two metrics, bandwidth consumption and the ratio of successful downloads. In both metrics, the ELiSyR shows lower bandwidth consumption, similar ratio of successful downloads and fairer distribution of computation loads in comparison. . In return, the ELiSyR algorithm takes more time in forwarding search queries. However, the time required for search is usually much less than the time required for downloads, so the delay in search is justifiable compared to the cost of bad downloads and subsequent re-search and downloads.

This paper is organized as follows. Section 2 discusses related work in bandwidth saving in P2P file searches and defense mechanisms against Sybil attacks. Section 3 explains the intuition behind our ELiSyR file search algorithm and shows the pseudo-code of the algorithm. Section 4 explains how the experiments were set up and shows the efficiency, lightweightness and robustness in hostile environment (Sybil-resilience) of ELiSyR. Finally, Section 5 proposes future work and concludes this paper.

2. Related Work

EigenTrust [11] is a reputation system for file-sharing P2P networks wherein each peer is assigned a unique global trust value that reflects the experiences of all peers in the network uploaded to or downloaded from this peer. All peers in the network participate in computing these values in a distributed and node-symmetric manner. Among various download source selection methods, we choose the probabilistic algorithm in which not only the upload requests are distributed among high-trusted nodes, but also favors the uploader with high trust value. However, theses reputation systems are vulnerable against Sybil attack. One of the hardest challenges in defending against Sybil attacks is that it is more or less impossible to limit the number of Sybil identities in the system. To protect against arbitrary failures, traditional Byzantine fault tolerance protocols rely on that the Byzantine users do not exceed one-third of the total number of users in the system [12]. However, typical peer-to-peer systems cannot impose such limitation on the number of Sybil nodes. Sybil attackers may exploit IP harvesting and even use botnets outside the system to launch an attack to inflate their numbers in the system.

As mentioned in Section 1, Sybil attack is already pervasive [4][5][6]. A wide variety of countermeasures have been proposed against Sybil attacks[13]. Systematic and economic approaches have been applied in order to detect Sybil nodes [14][15]. Though these detection mechanisms are promising, limiting the effect of Sybil attacks is being accepted as a practical solution to this attack [7][8][9][10][16].

SybilGuard [7] and SybilLimit [8] take advantage of social relationships between nodes in the system to limit the damage Sybil nodes can cause, no matter how many Sybil nodes are in the system. They have the same defense mechanisms. They use random routes in order to decide whether the peer in question has enough social relationships with other peers. Each node stores multiple random routes from itself to other nodes as "witnesses" in the verification process. The witness is the intersection node of two random routes, one from the subject in

question and the other from the verifier. If there are more intersecting nodes than the preset threshold, then the peer is accepted. In our simulation, we use this feature to help the downloader select the most trustworthy uploader among those who responded to the query. We assume that if the number of intersecting nodes of random routes from the downloader and from the responder is over a given threshold, then the responder is trustworthy.

However, they tend to put high load on high-degree nodes as shown in Section 4. When a verifier wants to check if a subject is honest or not, the intersection node of two random routes needs to respond to the verifier to ensure the intersection is not forged by the subject. As a social network tends to have a non-uniform degree distribution, high-degree nodes tend to serve as intersection nodes much more often than other users, and thus have higher overhead in computation and communication. Also, when high-degree nodes are unavailable due to any transient faults, the system's acceptance ratio decreases rapidly.

SybilLimit also requires an elaborate setup process. Each node creates multiple routing tables, goes on multiple random routes and records which random routes from which nodes passed itself, and the keys associated all these random routes. This setup process may discourage users from using this service as it causes delay before they start searching and downloading. Also, users without enough number of pre-established social relationships (edges) may not be able to use this service altogether, depending on how the threshold is set.

LIP [17] is a technique to identify fake files by mining history logs. They collect the hashes of contents and compare a new file's content with existing hashes. If there is a match in hash and yet the filename is different, then check the lifetime of the file. The shorter its lifetime is, the more likely this file is to be fake. They check the contents and the file's lifetime to see whether the file has been compromised. With proper DB of existing file hashes, LIP can effectively verify the compromised files. However, LIP requires a comprehensive DB and the storage requirement is not negligible.

There have been extensive researches on analysis and improvement of the Gnutella-like flooding search protocols. Saroiu et al. gives a detailed measurement study of the two most popular peer-to-peer file sharing systems, Napster and Gnutella [18]. A variety of protocols has been proposed based on random walk models [19][20][21][22][23] and other techniques [24][25] as alternatives to Gnutella's query algorithm. Sarshar et al. propose a percolation search algorithm for locating and retrieving content in random networks with Power-Law and heavy-tailed degree distributions [19]. They reduce the network bandwidth by utilizing the high-degree nodes and random walk model. But, they do not consider the unfair load distribution and Sybil attack.

3. ELiSyR File Search Algorithm

Our intuition for the ELiSyR algorithm is straightforward from our goals: safe search while being efficient in bandwidth consumption, lightweight in maintenance, and Sybil-resilient against high volume of malicious users and their bad files. For safe file search and Sybil resilience, we use social network as our search network: every search query is forwarded to selected neighbors on social network. We already explained that social networks are widely used to protect against Sybil attacks. It is well known that the Internet topology and the social network follow the power-law distribution [26][27]. In such a network, a few nodes have a high degree, i.e. connected to a large number of nodes. Therefore, if a user A initiates a query forwarding sequence, the query inherently gravitates towards the better linked members. In most social networks, the high degree nodes are more connected and therefore have a higher

probability to be attacked [28]. For example, the risk of infection of individuals in social networks can be identified by measuring the degree of each node where high degree nodes are more vulnerable to infection [29]. Traffic on a network also induces high loads on high degree nodes, which in turn makes them more vulnerable to failures [28]. The activity between the uploader and the downloader of a file in the P2P file sharing network results in the similar vulnerability to the high degree nodes. This brings obvious concerns on the security of the P2P file sharing systems.

ELiSyR uses the degree information to achieve our goals. First, we reduce the effect of malicious users by limiting the chances of them responding to search queries. Under this network, the vulnerability of each node depends on its degree [28]. We design our file search algorithm based on this study. ELiSyR favors lower-degree nodes to higher-degree nodes and limit the number of hops each request may be forwarded (TTL). It may also increase an average distance from a node to any other nodes. This increase delays our search, but it also delays the search queries from reaching the Sybil nodes. As shown by simulation and experiments in Section 4, we show comparable Sybil resilience to the state-of-the art anti-Sybil mechanism in avoiding bad downloads.

Second, ELiSyR takes a further step towards saving bandwidth in file search by selectively forwarding the search queries to neighbors. Instead of forwarding a search query to all its neighbors, each node picks a random neighbor and forwards until the sum of the degrees of the chosen neighbors exceeds the threshold. As we discussed in Section 1, forwarding on social networks may impose unbalanced loads on high-degree nodes. In a typical P2P network, the degree of peers forms a power-law network [19][21][30] in which most of the nodes have low connections and few nodes have a large number of edges. This social nature of P2P networks aggravates the load imbalance, as the degree distribution is highly skewed among peers. The random neighbor selection favors low-degree nodes over high-degree nodes. This non-uniform selection helps reducing the load imbalance, as shown in Section 4.

Finally, for efficient bandwidth consumption and lightweight maintenance, we only require each node to remember the degrees of its neighbors. This information may be periodically updated, or piggybacked as part of search queries. Whenever a node sends a search query to its neighbor, it may include its own degree so that the neighbor has up-to-date information.

3.1. ELiSyR Algorithm Description

Algorithm 1. describes our ELiSyR file search algorithm. Each peer i maintains d_{ij}, the degree of neighbor j in peer i's local repository. Note that d_{ij} and d_{kj} may be different for two peers i and k depending on how the degree information is updated. ELiSyR function decides to which peers a file search query should be forwarded, and this function is used for both queries initiated by peer i or the queries received by peer i.

Lines 17-19 compute p_{ij}, the probability of peer i forwarding the query q to its neighbor j for every neighbor. As mentioned above, ELiSyR favors the low degree nodes. The simplest way to favor low degree nodes would be to set the forwarding probability p_{ij} to the inverse of the degree of neighbor j, $1/d_{ij}$. However, this probability distribution would favor low degree nodes to the point that ELiSyR avoids high degree nodes altogether, due to the skewed degree distribution in social networks. As Adamic et al. showed in [26], the high-degree nodes appear on most shortest paths to other peers and avoiding them at all often results in network partition, i.e. not being able to find files at all.

```
 1: function issue_query(i, q) {peer i issues a query q}
 2: TTL(q) := max {max is 20 in the evaluation results}
 3: ELiSyR(i, q)
 4: process query responses
 5: end function

 6: function received_query(i, q)
 7: if i has the requested file then
 8:       send a response message to the query issuer
 9: else
10:       ELiSyR(i, q)
11: end if
12: end function

13: function ELiSyR(i, q) {peer i sends (or forwards) the query q}
14: N_i: a set of peer i's neighbors
15: d_ij : degree of neighbor j of peer i where j ∈ N_i
16: T_fwd: forwarding threshold
17: for all j ∈ N_i do
18:       p_ij = 1 / d_ij^α
19: end for
20: sum := 0
21: while sum < T_fwd do
22:       if TTL(q) = 0 then
23:             return
24:       end if
25:       randomly select node j from N_i based on p_ij
26:       TTL(q) := TTL(q) - 1
27:       send q to peer j
28:       sum := sum + d_ij
29: end while
30: end function
```

Algorithm 1. ELiSyR file search algorithm

We introduce a discount factor, α, and now p_{ij} is set to $1/d_{ij}^{\alpha}$. α is a parameter that balances the queries between low-degree nodes and high-degree nodes for load balancing and search efficiency. When $\alpha = 1$, $p_{ij} = 1/d_{ij}$ and the algorithm works as mentioned in the previous paragraph. This suffers from low file search success rate, but achieves better load balancing among nodes with different degrees. On the other hand, when $\alpha = 0$, then p_{ij} becomes the same for every neighbor j of peer i. The default setting in our experimental results in Section 4 is $\alpha = 0.5$. This probability computation may be cached and reused if the social network is relatively stable and the degrees do not change often.

In lines 21-29, peer i selects a random neighbor j to forward q to based on the probability p_{ij}. For each neighbor j the request q is forwarded to, peer i adds d_{ij}, the degree of neighbor j, to the sum, and repeat forwarding as long as the sum is less than T_{fwd}, the *forwarding threshold*. The reasoning behind this is that by forwarding to a high-degree node we increase the chances of finding the requested file, but also increase the chances of load imbalance between peers and

of reaching a Sybil attacker by taking a short cut in the social network. So we limit peer *i* from forwarding to high-degree nodes too many times. On the other hand, forwarding to a low-degree node may suffer from low chances of finding the requested file, so we repeat forwarding to many low-degree nodes to increase the chances. We set $T_{fwd} = 500$ in our experiment.

4. Experimental Results

We implemented the ELiSyR algorithm based on the Query-Cycle Simulator (QCSim) [26], which is a P2P file-sharing network simulator written by the authors of EigenTrust. We made extensive modifications in QCSim so that the simulator might behave more like real trace. We conducted experiments with synthetic and trace-based dataset and network.

We study the behavior of ELiSyR in simple network settings to compare with basic query forwarding algorithms as well as EigenTrust and SybilGuard. In each cycle of the Query-Cycle Simulator [31], a peer issues queries for a file, other peers may respond to queries, and the file is transferred from one of the responders to the peer who sent out the query to conclude a search process. For synthetic dataset and network, we used the network and file distribution of Query-Cycle Simulator. QCSim uses the Barabási-Albert (BA) model [32] to generate the power-law network, where high degree nodes tend to have high chances to connect to the newly-joining node (preferential attachment). Query-Cycle Simulator also distributes files among the nodes according to its popularity. It first assigns popularity to files according to Zipf distribution, and distributes (and replicates) files according to Zipf distribution again. As a result, more popular files are replicated in more nodes. Details of the simulator are described in [26]. We first set the simulation parameters the same as EigenTrust where the network consists of 62 good peers and 40 malicious peers. This setting is already intended to test the system in a heavily-attacked network environment, but we increase the percentage of malicious peers to simulate Sybil attacks in subsequent experiments. In our simulation, malicious peers respond to a query even when they do not have the requested file by the query and induce download of fake files (bad download). When unspecified, the forwarding threshold is set to 50. Each simulation consists of 150 cycles. We ran each setting for five times and averaged the results. In each setting, we recorded the network bandwidth consumption and the download performance. We compared the performance of our algorithms with EigenTrust and SybilGuard. In our simulation the global trusts in EigenTrust are computed in every 30 query cycles, which is the same with the value used in [11].

We used the trace as well collected by Fast et al. [33] on the OpenNap [34] network, an open-source descendant of Napster. OpenNap is a centralized P2P network in which users log on to a central server that tracks all search requests and file downloads. The trace was collected from a campus network sharing mp3 files during an 81-day period between February 28, 2003 and May 21, 2003 [33]. The trace includes 1) the file distribution, 2) the queries issued by peers, and 3) the file transfer between peers. We constructed the peer network by generating a data-sharing graph with the trace obtained above. Iamnitchi et al. defined the data-sharing graph as a graph in which nodes are users and an edge connects two users with similar interests in data [35]. They present the characteristics of the three different types of data-sharing graphs that correspond to three file-sharing communities: high-energy physics collaboration, the Web as seen from the Boeing traces, and the Kazaa peer-to-peer file-sharing system seen from a large ISP in Israel. According to their research, the Kazaa data-sharing graph is the closest to a power-law, and data-sharing graphs for the three systems all display small-world properties. The data-sharing graphs we built from OpenNap trace would have such properties as well.

Our data-sharing graph consists of total 6,464 nodes and 99,680 edges. The average degree of peers is 30. In our experiments, we regard all these peers as honest nodes and inject Sybil nodes into the network artificially. We model the Sybil attack by creating g collective Sybil groups and inserting random edges between good and Sybil peer, called attack edges. Sybil nodes in each Sybil group are all connected with each other and respond to every file search query with fake file.

In our experiment, the number of Sybil groups, g, is 5, and the number of Sybil nodes per Sybil group is 1,000, so 5,000 Sybil nodes in total. This setting of 5,000 Sybil nodes and 6,464 honest nodes is to test the system in a heavily attacked network environment. In ELiSyR, the only data that Sybil peers can manipulate is the degree of Sybil peers. A Sybil node j lies to its neighbor node *i* that its degree is 1, which maximizes p_{ij}. In EigenTrust, the Sybil peers boost the reputation of the fellow Sybil peers in the same Sybil group while undermining the reputation of honest nodes. In SybilLimit's behavior, the random routes passed Sybil nodes. In every experiment, 200,000 queries were issued. Every query is repeated up to 3 times with pre-determined delay between retries. This delay between retires mimics the user's behavior of waiting until the search completes and retrying. Since our simulator does not have a clock, we use the number of cycles as the delay measure, and waits for 3,000 queries to pass before each retry. We have experimented with two other algorithms that operate on social networks for comparison. The first algorithm "proportional" forwards requests to neighbor nodes with a probability proportional to their degrees. This algorithm finds the target file fast, but has more load imbalance and less Sybil-resilience than our other algorithms. The second algorithm "random" has a peer forward a search query to its neighbors with the equal chances, in other words according to probability from the uniform random distribution. The proportional algorithm shows how utilizing high-degree peers give faster answers but less load balancing and Sybil resilience. The random algorithm achieves better load balancing than the proportional but also suffers from higher chances of inauthentic downloads.

4.1. Safe File Search

We measure our Sybil resilience in the ratio of bad downloads per requested file and also in the average query repetition until successful download in the presence of a large number of malicious peers. **Fig. 1** and **Fig. 2** show the results with the synthetic dataset and network. **Fig. 1** depicts the fraction of good downloads with varying malicious peer ratio. As expected, since EigenTrust and SybilGuard broadcast the query into the network, peers can download the good files for most of the queries. On the other hand, the ELiSyR algorithm forwards the query message for a limited number of times. Yet the graph shows that more than 90% of the queries are successfully responded with good files. Even if malicious nodes occupy 70% of the total nodes, the fraction of successful query is sustained over 90%. This result is more interesting compared to the bandwidth consumption in the next subsection, where the ELiSyR algorithm consumes only 10 to 20% of EigenTrust or SybilGuard.

In **Fig. 2**, we show the average number of query repetition until successful download. In Query-Cycle Simulator, after receiving the query responses from those who have the specified file or those who pretend to have the file, a query issuer first sends download request one by one to those who responded to the query until it downloads a good file. Therefore, the less the peer downloads inauthentic files, the better the effectiveness of the algorithm becomes. The computation of trust in EigenTrust effectively reduces the download of inauthentic files. With the ELiSyR algorithm, each query on average is repeated two times. Note that this is the case when the malicious peers are more than 40% of the total number of peers in the network, and

even with SybilGuard every query is repeated 3 to 4 times, which is 1.5 to 2 times more than the ELiSyR algorithm.

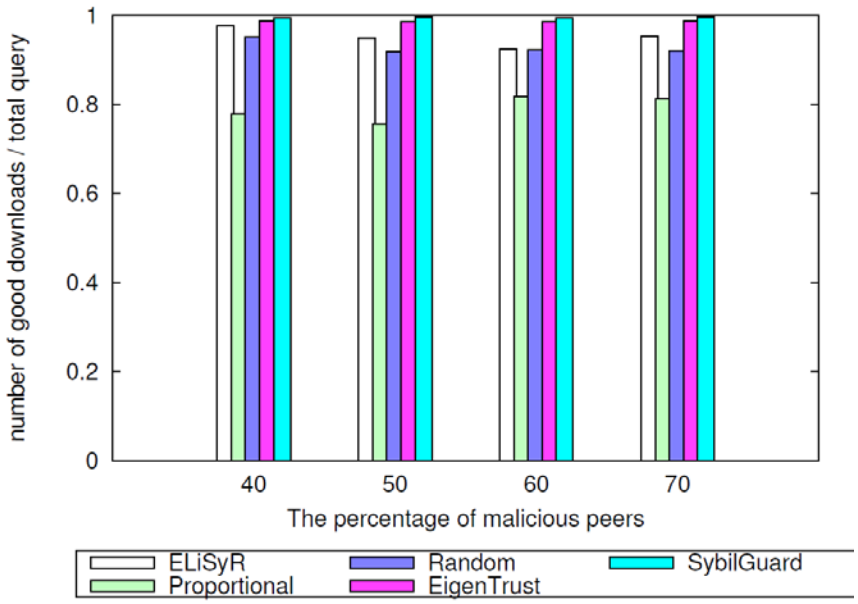


Fig. 1. Ratio of good downloads for each requested file

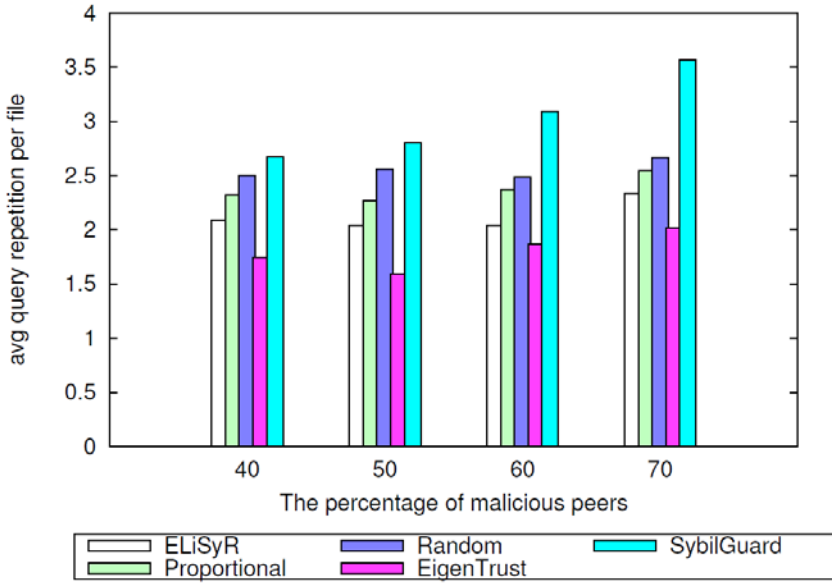


Fig. 2. Average query repetition until good download

The following shows the result with the trace-based dataset and network. We evaluate our Sybil resilience in two metrics, the number of bad downloads for each successful download and the ratio of successful query to all queries, and compare it to SybilLimit. In **Table 1**, we show the average number of bad downloads for each successful download and the ratio of successful queries. When there are multiple responders for the query, each algorithm chooses the uploader differently. In SybilLimit, the responder with the highest number of intersections

of the random routes is chosen. In ELiSyR, the responder with the shortest distance from the query issuer is chosen. As shown in **Table 1**, we achieve very low overhead even in the presence of 5,000 Sybil nodes attacking honest users. SybilLimit accepts very low percentage of Sybil nodes, but it suffered more inauthentic downloads than ELiSyR as once it reaches the Sybil region they showed very high intersection numbers. We expect this value to go down if the selection process changes to favor closer uploaders as ELiSyR does. As we mentioned earlier, each peer gives its query three chances of forwarding. The duplicate query is issued when a query did not succeed for the first two times. If the query finally failed to find the requested file, it is dropped and marked as failed query. ELiSyR succeeds in finding safe file to download 71.5% of the times even in the presence of 233 attack edges from 5,000 Sybil nodes to 6,464 honest nodes. SybilLimit suffers again due to high intersection numbers of Sybil nodes, but we expect that this value would go up if the selection method changes as well.

Table 1. The average number of bad downloads per successful download

Attack edges	Avg. number of inauthentic downloads per successful download		Percentage of successful queries	
	ELiSyR	SybilLimit	ELiSyR	SybilLimit
13	.1657	.9849	74.2%	28.9%
79	.1767	1.8887	74.0%	15.6%
233	.2145	1.8542	71.5%	16.5%

4.2. Efficiency and Light-Weight Maintenance

We now compare the network usage of algorithms. First, we collected the network traffic used for query-forwarding and other reputation-related messages. In addition to the query forwarding, EigenTrust uses *RequestTrust* and *Trust* messages in order to request and transfer the trust values computed so far. A peer *i* sends a *RequestTrust* message to those who responded the query issued by peer *i*. Then peer *i* can select a node based on the trust value obtained. Trust message which holds *i*'s trust on *j* is transferred from peer *i* to peer *j* during the trust computation. In SybilGuard, a peer *i* requests peer *j*'s random routes from peer *j* by sending *RequestTrust* message. Then peer *i* queries the intersecting node of random routes of peer *i* and peer *j* if peer *j*'s random route actually passes the intersecting node by sending *RequestVerify* message. With the same simulation setting, we obtained the following results.

Fig. 3 plots the network traffic used in five algorithms with synthetic workload. We extracted the network usage every thirty cycles and displayed the result with varying simulation time (cycle). ELiSyR, Proportional, and Random only use the query-forwarding messages. The amount of network traffic EigenTrust consumes to maintain the trust values is larger than the amount of network traffic of ELiSyR's total messages. Although Trust messages are interchanged not so frequently (in this simulation, every thirty cycles), it is not ignorable that amount of ELiSyR's total network usage is smaller than the network traffic necessary to compute the global trust. SybilGuard consumes larger amount of network traffic due to its verification protocol which includes the transfer of random routes.

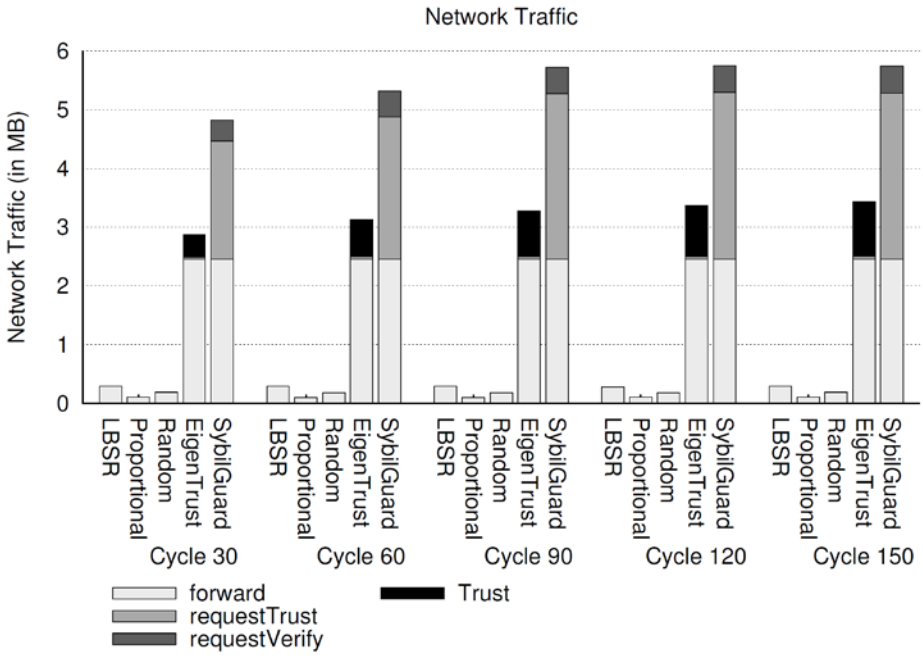


Fig. 3. Bandwidth consumption for each cycle

We compare the bandwidth consumption of ELiSyR and SybilLimit with trace-based dataset and network in **Table 2**. We collected the network traffic used to forward query messages and to transmit protocol-specific information. SybilLimit consumes larger amount of network traffic due to its verification protocol which includes the transfer of random routes. The query forwarding bandwidth of EigenTrust and SybilLimit may be reduced by using more efficient forwarding method than flooding. However, the bandwidth spent on exchanging trust information is an overhead that ELiSyR does not have. This result is consistent with the simulation based on the synthetic queries.

We show the load balancing comparison in **Fig. 4**. With each algorithm, we collected the network bandwidth for each peer and computed the share of its load over all the participants. Peers with spikes in other algorithms are high-degree nodes, while ELiSyR shows not as high spikes. The load distribution of EigenTrust and SybilGuard is quite unfair to highly-connected nodes. First of all, it is due to the flooding search protocol. Under flooding, highly-connected node cannot avoid a great number of forwarding messages coming from its incoming edges. Furthermore, the algorithm-specific protocols of EigenTrust and SybilGuard aggravate the

Table 2. Bandwidth consumption per query in MB

Class	ELiSyR	SybilLimit	EigenTrust
Query forward	.13	3.57	3.57
Random route transfer	N/A	.159	N/A
Verification	N/A	0.000322	N/A
Trust computation	N/A	N/A	0.0237

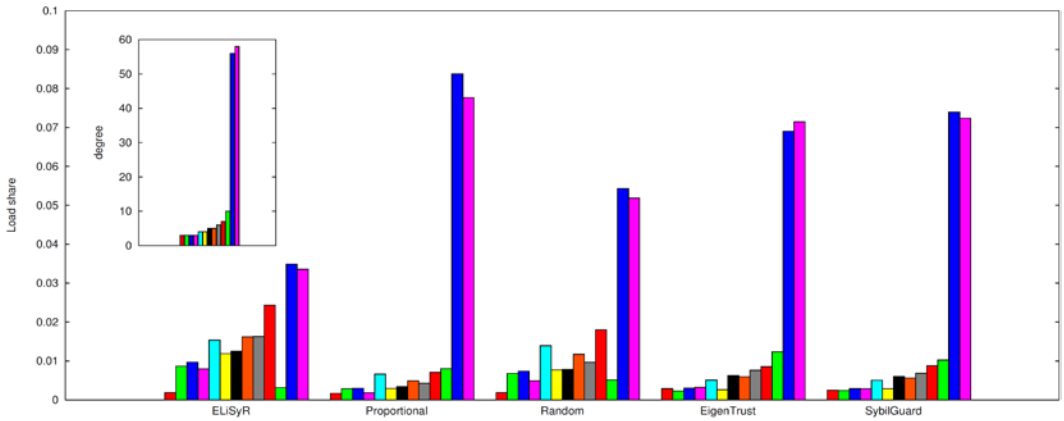


Fig. 4. Load balancing in bandwidth (We sampled every 5th node in the increasing order of degree. The degree of each node is displayed in the inner figure.)

load imbalance. For SybilGuard, since random routes are highly likely to intersect on high-degree nodes, the verification requests tend to be driven to them. But ELiSyR fairly distributed the load among peers. One thing to note is that distributing the load evenly is not recommended since it can be a serious burden to several low-degree nodes such as newcomers. ELiSyR is also scattering the load fairly well in that not so much load is imposed on low-degree nodes.

We now measure the setup cost of each algorithm in terms of bandwidth consumption. The setup cost of each algorithm is determined as follows. For ELiSyR, each peer should advertise its degree to its neighbors. For SybilLimit, every peer should generate its random routes enough to prove its authenticity and test others'. EigenTrust does not have any setup cost. In our experiment, ELiSyR cost about 91MB of bandwidth and SybilLimit about 2048MB. This setup cost is also relevant to the maintenance cost. When the social network changes, for example there are new nodes or edges added to the network or nodes and edges removed from the network, then the ELiSyR requires to update the degree information and the SybilLimit requires to update the random route information. ELiSyR requires more maintenance cost than EigenTrust, but less than SybilLimit.

Even though our main goal is to reduce bandwidth consumption, it is notable to show the storage cost of three algorithms in comparison. The comparison of storage cost for each peer *i* is shown in **Table 3**. The ELiSyR algorithm incurs storage cost only in maintaining the degree of neighbor nodes. It requires $SIZE_{nodeid} \times d(i)$ to remember degrees of its neighbors. In EigenTrust, each peer *i* keeps the history of file transfer for upload and download: the set of

Table 3. Comparison of storage cost for each peer *i*. SIZE_xxx denotes the size of xxx. The storage cost to store the neighbor of peer *i* is not considered in this table. *d(i)*, *dl(i)*, and *ul(i)* denote the degree of peer *i*, the set of peers from which peer *i* has downloaded files, and the set of peers which have downloaded file from peer *i*, respectively.

Algorithm	**Storage cost for each peer *i***
ELiSyR	$SIZE_{nodeid} \times d(i)$
EigenTrust	$SIZE_{trust} \times dl(i) + SIZE_{nodeid} \{dl(i) + ul(i)\}$
Sybil*	$SIZE_{nodeid} \times d(i) + 2 \times SIZE_{key} \times w \times d(i)$

peers which have downloaded file from peer *i* (*ul(i)*) and the set of peers from which peer *i* has downloaded files (*dl(i)*). In addition, peer *i* stores evaluation from download experience for each peer in the download set. This adds $SIZE_{trust} \times dl(i)$ to store the local trust values. SybilGuard and SybilLimit require each peer to remember random routes it took in the initial setup, and also the random routes that have passed this peer. This takes $2 \times SIZE_{key} \times w \times d(i)$. SybilGuard and SybilLimit also require a symmetric key between each peer and its neighbor, which takes $SIZE_{nodeid} \times d(i)$ storage.

From this table, it is obvious that SybilGuard and SybilLimit require additional $2 \times SIZE_{key} \times w \times d(i)$ bytes compared to ELiSyR. To compare EigenTrust and ELiSyR, we formulate the ratio of the storage cost of EigenTrust and that of ELiSyR as follows.

$$
\begin{aligned}
R = \frac{\text{cost}_{\text{EigenTrust}}}{\text{cost}_{\text{ELiSyR}}} &= \frac{SIZE_{key} \times dl(i) + 2 \times SIZE_{nodeid} \{dl(i) + ul(i)\}}{SIZE_{nodeid} \times d(i)} \\
&= \frac{SIZE_{nodeid} \cdot (3dl(i) + 2ul(i))}{SIZE_{nodeid} \times d(i)} \text{ , if } SIZE_{nodeid} \cong SIZE_{key} \\
&= \frac{3dl(i) + 2ul(i)}{d(i)} = \frac{dl(i)(3 + 2a(i))}{d(i)} \text{ , let } \frac{ul(i)}{dl(i)} = a(i)
\end{aligned}
$$

.

ELiSyR consumes less storage than EigenTrust if the following inequality holds.

$$
R = \frac{dl(i)(3 + 2a(i))}{d(i)} \geq 1 \Leftrightarrow \frac{d(i)}{dl(i)} \leq 3 + 2a(i)
$$

According to the above inequality, the per-peer storage cost of ELiSyR becomes smaller than that of EigenTrust when the number of peers that peer *i* have downloaded files from is $3 + 2a(i)$ times greater than the degree of peer *i*. For example, if peer *i* has 9 neighbors and *i* has downloaded files from more than 3 peers, then the storage cost of peer *i* in ELiSyR is smaller than that in EigenTrust assuming that $a(i) = 0$. When $a(i) > 0$, it is sufficient for peer *i* to download files from less than 3 peers in order to satisfy the above inequality. This condition is not hard to satisfy. In the system's viewpoint, since the vast majority of nodes are those with small degree in a power-law network, the total storage cost of ELiSyR is similar to or less than that of EigenTrust.

5. Future Work and Conclusion

P2P networks consume a major part of the Internet bandwidth, and most consumption comes from file sharing applications. Therefore it is important that bad downloads are avoided even before detection. In this paper, we propose ELiSyR, an efficient, light-weight and Sybil-Resilient file search protocol. ELiSyR uses the degree information to achieve our goals. Even though ELiSyR requires a small amount of information - the degree information of neighboring peers, ELiSyR shows relatively low bad download rates even in the severely attacked environment and consumes low bandwidth compared to EigenTrust, SybilGuard and

SybilLimit. As future work, we will incorporate reputation system such as EigenTrust into forwarding probability computation for better bad download rate.

References

[1] H. Schulze and K. Mochalski. "iPoq Internet Study 2007," http://www.ipoque.com/userfiles/file/internet_study_2007.pdf, Dec. 2007.

[2] "BitTorrent Acquires µTorrent," http://www.bittorrent.com/pressreleases/2006/12/06/bittorrent-acquires-%C2%B5torrent, Dec. 2006.

[3] T. Claburn. "Fake mp3 Trojan detected on 27% of PCs," http://www.informationweek.com/story/showArticle.jhtml?articleID=207600502, May 2008.

[4] L. Page, S. Brin, R. Motwani and T. Winograd, "The PageRank citation ranking: Bring order to the Web," in *Proc. of the 7th International World Wide Web Conf.*, Apr. 1998. Article (CrossRef Link)

[5] A. Cheng and E. Friedman, "Sybilproof reputation mechanisms," in *SIGCOMM 05 Workshop*, Aug. 2005. Article (CrossRef Link)

[6] M. Oiaga. "eBay Bot Fraud - positive feedback is used as an incentive to attract victims," http://news.softpedia.com/news/eBay-Bot-Fraud-31711.shtml, Aug. 2006.

[7] H. Yu, M. Kaminsky, B. P. Gibbons and A. Flaxman, "SybilGuard: Defending against sybil attacks via social networks," in *ACM SIGCOMM*, Sept. 2006. Article (CrossRef Link)

[8] H. Yu, P. Gibbons, M. Kaminsky and F. Xiao, "SybilLimit: A near-optimal social network defense against sybil attacks," in *Proc. of the 2008 IEEE Symposium on Security and Privacy*, May 2008. Article (CrossRef Link)

[9] G. Danezis, C. Lesniewski-Laas, F. M. Kaashoek and R. Anderson, "Sybil-resistant DHT routing," in *ESORICS*, Sept. 2005. Article (CrossRef Link)

[10] H. Rowaihy, W. Enck, P. McDaniel and L. T. Porta, "Limiting sybil attacks in structured p2p networks," in *INFOCOM Mini-Symposium*, May 2007. Article (CrossRef Link)

[11] D. S. Kamvar, T. M. Scholosser and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," in *Proc. of the 12th International Conference on World Wide Web*, May 2003. Article (CrossRef Link)

[12] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *The Third Symposium on OPerating Systems Design and Implementations (OSDI)*, Feb. 1999. Article (CrossRef Link)

[13] N. B. Levine, C. Shields and B. N. Margolin, "A survey of solutions to the sybil attack," Univ. of Massachusetts Amherst, Tech report 2006-052, Oct. 2006. Article (CrossRef Link)

[14] B. N. Margolin and N. B. Levine, "Informant: detecting sybils using incentives," in *Proc. of Financial Cryptography (FC)*, Feb. 2007. Article (CrossRef Link)

[15] T. Kohno, A. Broido and K. Claffy, "Remote physical device fingerprinting," in *Proc. of IEEE Symposium on Security and Privacy*, May 2005. Article (CrossRef Link)

[16] C. Hota, J. Lindqvist, K. Karvonen, A. Yla-Jaaski and J. C.K. Mohan, "Safeguarding Against Sybil Attacks via Social Networks and Multipath Routing," in *Proc. of International Conf. on Networking, Architecture, and Storage*, Aug. 2007. Article (CrossRef Link)

[17] Qinyuan Feng and Yafei Dai, "LIP: A LIfetime and Popularity Based Ranking Approach to Filter out Fake Files in P2P File Sharing Systems," in *Proc. of International Workshop on Peer-To-Peer Systems*, Feb. 2007. Article (CrossRef Link)

[18] S. Saroiu, P. K. Gummadi and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proc. of Multimedia Computing and Networking (MMCN)*, Jan. 2002. Article (CrossRef Link)

[19] N. Sarshar, O. P. Boykin and P. V. Roychowdhury, "Percolation search in power law networks: Making unstructured peer-to-peer networks scalable," in *Proc. of the 4th International Conf. on Peer-to-Peer Computing (P2P)*, Aug. 2004. Article (CrossRef Link)

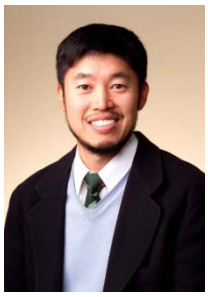
[20] C. Gkantsidis, M. Mihail and A. Saberi, "Random walks in peer-to-peer networks," in *Proc. of Thenty-third Annual Joint Conf. of the INFOCOM*, Mar. 2004. Article (CrossRef Link)

[21] Q. Lv, P. Cao, E. Cohen, K. Li and S. Shenker, "Search and replication in unstructured

peer-to-peer networks," in *Proc. of International Conf. on Supercomputing*, June 2002.
[22] Y. Chawathe, S. Ratnasamy, N. L. B. Lanham and S. Shenker, "Making gnutella-like p2p systems scalable," in *Proc. of the 2003 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Aug. 2003. Article (CrossRef Link)
[23] C. Gkantsidis, M. Mihail and A. Saberi, "Random walks in peer-to-peer networks: Algorithms and evaluation," *Performance Evaluation*, vol. 63, no. 3, pp. 241-263, Mar. 2006. Article (CrossRef Link)
[24] D. Tsoumakos and N. Roussopoulos, "Adaptive probabilistic search for peer-to-peer networks," in *Proc. of the 3rd International Conf. on Peer-to-Peer Computing (P2P)*, Sept. 2003. Article (CrossRef Link)
[25] S. Jin and H. Jiang, "Novel approaches to efficient flooding search in peer-to-peer networks," *The International Journal of Computer and Telecommunications Networking*, vol. 51, no. 10, pp. 2818-2832, July 2007. Article (CrossRef Link)
[26] L. Adamic, R. Lukose and B. Huberman, "Search in power-law networks," *Physical Review*, vol. 64, no. 4, p. 046135, Sept. 2001. Article (CrossRef Link)
[27] Michalis Faloutsos, Petros Faloutsos and Christos Faloutsos, "On power-law relationships of the Internet topology," in *Proc. of ACM SIGCOMM*, Aug. 1999. Article (CrossRef Link)
[28] Lazaros K Gallos, Reuven Cohen, Panos Argyrakis, Armin Bunde and Shlomo Havlin, "Stability and Topology of Scale-Free Networks under Attack and Defense Strategies," *Physical Review Letters*, vol. 94, no. 18, p. 188701, May 2005. Article (CrossRef Link)
[29] M. R. Christley, L. G. Pinchbeck, G. R. Bowers, D. Clancy, P. N. French, R. Bennett and J. Turner, "Infection in Social Networks: Using Network Analysis to Identify High-Risk Individuals," *American Journal of Epidemiology*, vol. 162, no. 10, pp. 1024-1031, Sept. 2005. Article (CrossRef Link)
[30] M. Ripeanu, "Peer-to-peer architecture case study: Gnutella network," in *Proc. of the 1st International Conf. on Peer-to-Peer Computing (P2P)*, Aug. 2001. Article (CrossRef Link)
[31] T. M. Schlosser, E. T. Condie and D. S. KIamvar, "Simulating a file-sharing p2p network," in *Porc. of Workshop in Semantics in Peer-to-Peer and Grid computing*, Dec. 2002. Article (CrossRef Link)
[32] A. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509-512, Oct.1999. Article (CrossRef Link)
[33] A. Fast, Jensen D. and N. B. Levine, "Creating social networks to improve peer-to-peer networking," in *Proc. of the eleventh International Conf. on Knowledge Discovery in Data Mining (SIGKDD)*, Aug. 2005. Article (CrossRef Link)
[34] "OpenNap: Open source napster server," http://opennap.sourceforge.net, September 2001.
[35] A. Iamnitchi, M. Ripeanu and I. Foster, "Small-world file-sharing communities," in *Proc. of The 23rd Conf. of the IEEE Communication Society (INFOCOM)*, Mar. 2004. Article (CrossRef Link)

Hyeong S. Kim received his B.S. degree in Computer Science and Engineering from Seoul National University, Seoul, Korea, in 2003. He received his M.S. degree in 2005 and is currently a Ph.D. Candidate in the School of Computer Science and Engineering at Seoul National University. His research interests are in the areas of Cloud and Grid computing, distributed storage, social networks, and energy efficiency.

EJ Jung is an assistant professor in Computer Science at USF. She has received a Master's degree and a Ph.D. in Computer Science at the University of Texas at Austin. Her research interests are security and fault-tolerance in distributed, peer-to-peer, and grid systems, privacy and anonymity in databases, security policy management, and usable security. Her current projects include anti-malware toolbar, Sybil attacks in social networks, privacy and anonymity support in medical databases, and XACML policy management.



Heon Y. Yeom received the B.S. degree in computer science from Seoul National University, Seoul, Korea, in 1984 and the M.S. and Ph.D. degrees in computer science from Texas A&M University, College Station, in 1986 and 1992, respectively. From 1986 to 1990, he was with Texas Transportation Institute as a Systems Analyst and from 1992 to 1993, he was with Samsung Data Systems as a Research Scientist. He joined the Department of Computer Science, Seoul National University, in 1993, where he currently is a Professor and teaches and conducts research on distributed systems.