This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2359020, IEEE Transactions on Parallel and Distributed Systems

Social-P2P: An Online Social Network Based P2P File Sharing System

Haiying Shen*, *Senior Member, IEEE*, Ze Li, *Student Member, IEEE*, Kang Chen

Abstract—A peer-to-peer (P2P) file sharing system provides a platform that enables a tremendous number of nodes to share their files. Retrieving desired files efficiently and trustworthily is critical in such a large and jumbled system. However, the issues of efficient searching and trustworthy searching have only been studied separately. Simply combining the methods to achieve the two goals doubles system overhead. In this paper, we first study trace data from Facebook and BitTorrent. Guided by the observations, we propose a system that integrates a social network into a P2P network, named Social-P2P, for simultaneous efficient and trustworthy file sharing. It incorporates three mechanisms: (1) interest/trust-based structure, (2) interest/trust-based file searching, and (3) trust relationship adjustment. By exploiting the social interests and relationships in the social network, the interest/trust-based structure groups common-multi-interest nodes into a cluster and further connects socially close nodes within a cluster. The comparably stable nodes in each cluster form a Distributed Hash Table (DHT) for inter-cluster file searching. In the interest/trust-based file searching mechanism, a file query is forwarded to the cluster of the file by the DHT routing first. Then, it is forwarded along constructed connections within a cluster, which achieves high hit rate and reliable routing. Moreover, sharing files among socially close friends discourages nodes from providing faulty files because people are unlikely to risk their reputation in the real-world. In the trust relationship adjustment mechanism, each node in a routing path adaptively decreases its trust on the node that has forwarded a faulty file in order to avoid routing queries towards misbehaving nodes later on. We conducted extensive trace-driven simulations and implemented a prototype on PlanetLab. Experimental results show that Social-P2P achieves highly efficient and trustworthy file sharing compared to current file sharing systems and trust management systems.

Index Terms—P2P networks, Online social networks, File sharing.

1 Introduction

Peer-to-peer (P2P) systems are widely used in file sharing applications, such as BitTorrent. More than 50% of the files downloaded and 80% of the files uploaded on the Internet are through P2P networks [1]. P2P file sharing systems attract millions of users. Due to the large-scale of the P2P systems, efficiently locating a desired file has been an open problem for many years. Considering the numerous users without preexisting trust relationships in the P2P open platform, providing trustworthy file sharing has become another important issue. Indeed, many users found themselves downloading the wrong files due to misleading file names and descriptions. Peers with malicious intent upload faulty files, defined as tampered files or files with malicious code (e.g., Trojan horses and viruses).

- * *Corresponding Author. Email: shenh@clemson.edu; Phone: (864) 656 5931; Fax: (864) 656 5910.*
- *Haiying Shen, Ze Li and Kang Chen are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, 29634. E-mail: {shenh, zel, kangc}@clemson.edu*

The research on the issues of efficient and trustworthy P2P file searching has been conducted separately. Although many methods have been proposed to enhance the efficiency or trustworthiness, the individual methods are not efficient by themselves. In order to improve the search efficiency, some works cluster the nodes with the same interest to increase file hit rate [2]–[14]. Since these methods cluster nodes with a single interest, a node with multiple interests needs to maintain multiple clusters, which generates a high overhead for cluster maintenance and inter-cluster searching. Also, most of the previous approaches depend on the contents in users' local storage to infer their file interests. They are not only costly but also unable to retrieve the complete interests of a user with insufficient stored contents, such as new users or the users that have deleted shared files. A widely-used solution for trustworthy file sharing is to employ a cyber-trust management system [15]–[20], in which each node rates service providers based on the service quality. However, accumulating sufficient ratings for calculating an accurate trust may take a long time. Also, periodical trust updates produce a high overhead. Currently, the only approach to achieve both efficient and trustworthy P2P file searching is to directly combine a system for high search efficiency and a cyber-trust management system. In this way, all nodes need to maintain structures for two systems, which would double the cost, thus making the high overhead problem even more severe. Therefore, a system that can simultaneously provide both efficient and trustworthy file sharing with low overhead is greatly needed.

In this paper, we propose a system that integrates a social network into a P2P network, namely Social-P2P, to simultaneously achieve efficient and trustworthy P2P file sharing by leveraging social interests and relationships. Two facts lay the foundation for this work. First, people usually share files that they are interested in [11]. Interests indicated by a user himself in his profile can more accurately reflect the complete interests of the user. Second, users are unlikely to provide faulty files to their socially close friends because it will impair their social relationships with others and degrade their reputation in their social communities in the real world. Thus, by mapping the P2P cyber network to the social network and restricting cyber services (e.g., file sharing and message routing) between socially close nodes, misbehaviors (i.e., providing faulty files and rejecting forwarding messages) can be discouraged.

In this paper, we first study trace data from Facebook

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2359020, IEEE Transactions on Parallel and Distributed Systems

and BitTorrent. We gain a number of observations (O):
O1: Some interests are highly correlated. That is, given a pair of correlated interests A and B, if a person has interest A, (s)he is very likely to also have interest B.
O2: An online social network user has different contact frequency with different users.
O3: Friends in an online social network (users with direct social network connections) usually have very close social relationships in their real life.
O4: A P2P file sharing system possesses a certain percent of comparably stable nodes.
O5: The popularity distribution of interests can be modeled as a Zipf distribution; most of the file queries are for a small percent of file interests.

Guided by these observations, we develop the following three components for Social-P2P:

(1) **Interest/trust-based structure construction.** It groups common-multi-interest nodes into an interest cluster (O1), and forms comparably stable nodes into a Distributed Hash Table (DHT) to connect clusters for efficient inter-cluster data sharing (O4). Within each interest cluster, nodes are connected with their socially close nodes as P2P overlay neighbors (O3). Furthermore, Social-P2P uses anonymous routing to prevent malicious nodes from selectively attacking socially distant nodes and protects the privacy of the nodes.

(2) **Interest/trust-based file searching.** The trustworthiness between nodes is weighted and a node tends to forward a file query to trustworthy neighbors (O2). Since higher popularity files have more file copies being shared in the system, random walk is employed for high hit rate in intra-cluster file searching considering higher popular files have more copies in the system (O5). We further propose routing algorithms to enhance the random-walk based routing.

(3) **Trust relationship adjustment.** Each node in a routing path decreases its trust on the next hop when a faulty file is retrieved in order to avoid routing queries towards misbehaving nodes later on (O2).

With these three components, Social-P2P achieves highly efficient and trustworthy file sharing with low overhead. We present the details below.

(1) **High efficiency**: Clustering common-multi-interest nodes enables nodes to quickly find files in its own cluster. The higher-level DHT enables efficient inter-cluster search, which helps nodes to find files outside of their interests quickly.

(2) **High trustworthiness**: Confining services between socially close nodes discourages nodes from providing faulty services. Trust-based random walk can ensure a query message be forwarded among trustworthy nodes. The trust decrease upon misbehavior can quickly isolate misbehaving nodes. The anonymous routing further reinforces the trustworthiness of file sharing.

(3) **Low overhead**:

- *Low overhead in structure maintenance.* Comparatively stable nodes form a DHT and other dynamic nodes constitute an unstructured P2P by only connecting to their socially close nodes, leading to low P2P overlay maintenance without frequent DHT maintenance in node dynamics.
- *Low overhead in efficient file sharing.* Clustering common-multi-interest nodes rather than common-single-interest nodes reduces the cluster maintenance overhead, enhances file search hit rate within clusters, and reduces the inter-cluster searching overhead.
- *Low overhead in trustworthy file sharing.* Social-P2P does not need to periodically accumulate ratings of each node to calculate its trust. Each node only needs to maintain its neighbors' trusts, resulting in low overhead for trust management.

As far as we know, this is the first work that simultaneously considers both efficient and trustworthy file querying with low overhead in P2P networks. Like some previous P2P works [21]–[26], Social-P2P has a central server, which is mainly used to handle the social network functions and assistance work. The remainder of the paper is organized as follows. Section 2 gives an overview on the existing file search systems and reputation systems. Section 4 describes the design of Social-P2P. The performance evaluation is presented in Section 5. Section 6 concludes the paper.

2 Related Work

Efficient file sharing. Numerous methods including locality-aware searches [5], [6] and social network based searches [7]–[12], [14], [27] have been proposed in hopes of increasing the search efficiency in P2P systems. Searching based on social networks can be classified into two categories: unstructured networks and DHTs. In the unstructured network based search, Carchiolo *et al.* [27] and Lei *et al.* [10] proposed to gradually cluster nodes into the same group if they query or reply for the same resources. Fast *et al.* [11] proposed to use hierarchical Dirichlet filtering to extract user preferences to musical styles from their music libraries. They cluster the available files in the network based on user interests. Although these algorithms can improve the basic search algorithm, since the nodes with the same interests can be grouped only after they have interactions, the clustering process takes a long time. By relying on the social interest information, Social-P2P can quickly and accurately cluster nodes with similar interests.

In the category of DHTs, Li *et al.* [12] proposed Cyber, in which the nodes are associated with certain communities based on their interests. Cyber builds a DHT-based index on the keywords of items. When a node queries for an item, the DHT-indexed peer responsible for the queried keyword returns the items that match the interests of the community the requester belongs to. Zhang *et al.* [14] proposed to improve search in unstructured P2P overlay networks by building a partial index of globally unpopular data and non-major interest data based on a DHT. The index can assist peers in finding other peers with similar interests and provide search hints for a data difficult to be located by exploring peer interests. The current DHT-based social network enables fast node clustering but suffers from high system maintenance overhead in churn. Meanwhile, single interest-based node clustering requires each node to maintain several clusters, which leads to high cluster maintenance overhead. In Social-P2P, each node only needs to maintain a single multi-interest cluster. Moreover, only comparatively stable nodes in these clusters form a stable DHT and the dynamic nodes only maintain a number

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2359020, IEEE Transactions on Parallel and Distributed Systems

of friends in the unstructured cluster. Thus, Social-P2P generates a low system maintenance overhead.

Trust management. Numerous reputation systems [13], [15]–[20] have been proposed to increase the trustworthiness of the P2P systems. The basic idea is to let peers rate their service providers after each receiving the service. The accumulated rating of a node is used to represent its trustworthiness. However, accumulating sufficient ratings to calculate an accurate reputation value takes a long time. Also, managing the ratings between nodes and calculating the reputation value for each node generate high overhead. Social-P2P takes advantages of social trust relationships between people to increase the trustworthiness of file sharing at lower overhead.

Marti *et al.* [13] investigated how existing social networks could benefit P2P data networks by leveraging the inherent trust associated with social links for DHT routing. This work only deals with misrouting problems, while Social-P2P targets more general file trustworthiness problem. Galuba *et al.* [28] leveraged social networks to avoid free-riders in BitTorrent. Frey [29] addresses the need for trust in user-centric applications by proposing two distributed protocols that combine interest-based connections between users with explicit links obtained from social networks.

3 Trace Data Analysis

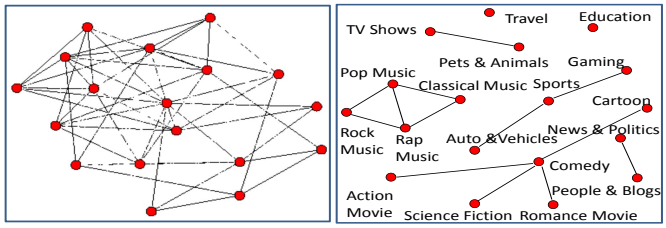
In this section, we analyze Facebook trace data crawled by us and BitTorrent trace data from the Graffiti Network Project [30]. The Facebook trace data covers the interests of 32,344 users in the South Carolina Region in June, 2010. To crawl the data, we selected two users with no social relationship in Clemson University as seed nodes and built a friend graph using breadth first search through each node's friend list. We skipped the users whose personal information cannot be accessed. Finally, we drew a social network graph, where a vertex is a user and a link means these two users are friends. The average number of friends per node is 32.51 and the average path length of the graph is 3.78. The BitTorrent user traffic was collected during a three week period (Oct 28, 2008-Nov 21, 2008) involving 3,570,588 nodes.

Interest clustering. We parsed the interest information from the users' profiles in Facebook. We removed the interests irrelevant to file sharing (e.g., "sleep" and "shopping"). We classified the remaining interests (e.g., "action movie", "classic music" and "sports") into 18 categories. We plotted a graph $G(V, E)$ to show the relationship among the 18 interests. The vertices V are the interests. A link E between V_1 and V_2 indicates the co-existences of both V_1 and V_2 in all profiles of t persons, where t is the threshold of the number of persons.

Figure 1 plots the graphs with threshold t=100 and $t = 500$, respectively. When $t = 100$, the interests are densely connected. When $t = 500$, several interests are still clustered, while two interests are isolated. Also, the number of interests in one interest cluster varies.

Observation(O)1: Some interests are highly correlated. That is, given a pair of correlated interests A and B, if a person has interest A, (s)he is very likely to also have interest B.

Inference(I)1: Instead of clustering the nodes based on each interest, which leads to high overhead, clustering common-multi-interest nodes can improve file retrieval efficiency and reduce cluster maintenance overhead.



(a) Threshold = 100 (b) Threshold = 500

Fig. 1. The clustering feature of interests.

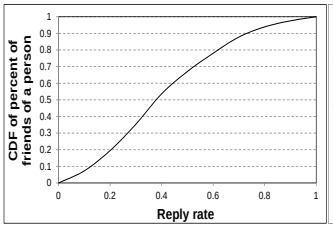


Fig. 2. Closeness distribution.

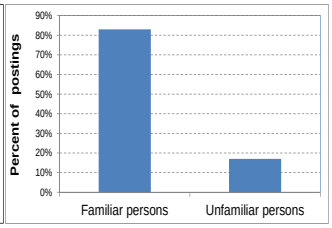


Fig. 3. Posting distribution.

Closeness between online users. We analyzed the reply rate of the posts on user comment walls and pictures in Facebook. If user B posted M comments to user A, and A replied m comments, we define m/M as the reply rate of user A to his friend B. We then calculated the average value of each user's reply rates to his/her friends and used it to represent the closeness among users. Figure 2 shows the distribution of the average reply rates of all users to their friends. It shows that a user has a reply rate of less than 0.7 to almost 90% of its friends on average. That is, for only 10% of friends, a person replies more than 70% of their comments. The results indicate that users treat different persons in an online social network differently. In order to show that the behavior of replying comments is driven by social closeness of nodes rather than the contents of the comments (e.g. interesting comments), we further investigate the comment posting behaviors between people. Given a posting from user A to user B, we call it posting for a familiar person if B has posted comments on A's wall/picture and A replied B's comment before. Otherwise, we call it posting for an unfamiliar person. We calculated the percent of postings for (un)familiar persons for each user, and plotted the average values in Figure 3. We see that 83% of a person's postings are for familiar persons. Combing these results, we observe:

O2: A user in the online social network has different contact frequency with other different users.

We reasonably assume the user contact frequency indicates the trust between them, we can infer that:

I2: The trust relationship between nodes should be weighed. Retrieving files from trustable nodes can increase the trustworthiness of the retrieved files.

Figure 4 further shows the social relationship between the users. We observe that:

O3: Friends in an online social network usually have very close social relationships in their real life.

I3: Requesting services (e.g., providing files and query

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2359020, IEEE Transactions on Parallel and Distributed Systems

routing) from socially close nodes can enhance the trustworthiness of received services, since people do not want to ruin their reputation in real life.

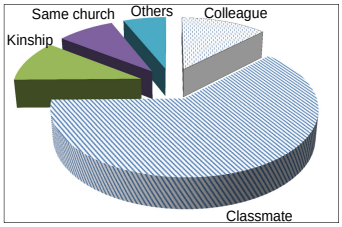


Fig. 4. Social relationship.

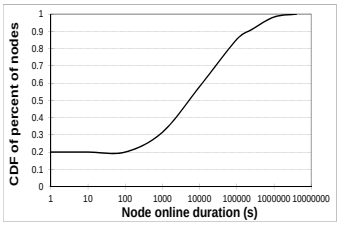


Fig. 5. Node churn rate.

Node stability. We now analyze the distribution of node stability in the BitTorrent file sharing system. Figure 5 shows the cumulative distribution function (CDF) of the length of time that nodes remain online.

O4: A P2P file sharing system possesses certain percentages of comparably stable nodes (15%) and highly dynamic nodes (20%).

I4: Building a DHT using all nodes in the system is not suitable for P2P file sharing due to high churn. Forming the comparatively stable nodes into a DHT to assist other nodes in file retrieval can enhance file sharing efficiency.

File interest popularity. The number of torrents of a file category (i.e., interest) represents its popularity. We ranked 505 interests based on the number of torrents. The interest with rank 1 has the largest number of torrents. Figure 6 shows the number of torrents of an interest versus its rank in the log-log scale. It also includes a line for the best fit Zipf distribution.

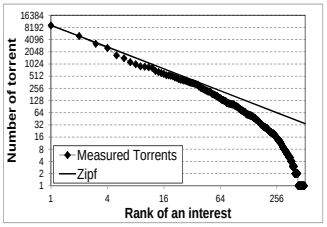


Fig. 6. File popularity distribution.

O5: The popularity distribution of interests can be modeled as a Zipf distribution.

In the random walk file searching algorithm, a node randomly selects one or several neighbor nodes (except the node which forwards the message) as the next hops for message forwarding.

I5: In the case that higher popularity files have more file copies being shared in the system, the random walk file searching algorithm can achieve high hit rate in retrieving popular files.

4 The Design of Social-P2P

The above observations and inferences motivate us to integrate a social network into a P2P network for efficient and trustworthy file retrieval. Thus, we propose Social-P2P, which leverages the *social closeness* and *interest* information in the social network to enable nodes with a social relationship and multiple common interests share files between each other.

Figure 7 shows the system structure of Social-P2P. Based on I1, we group common-multi-interest nodes together into an interest cluster. Based on I2 and I3, within each interest cluster, nodes are connected based on their social network links. The trustworthiness between nodes is weighed and a node tends to forward a file query to higher trustworthy neighbors in file searching. Based on I4, we select a comparably stable node as an ambassador

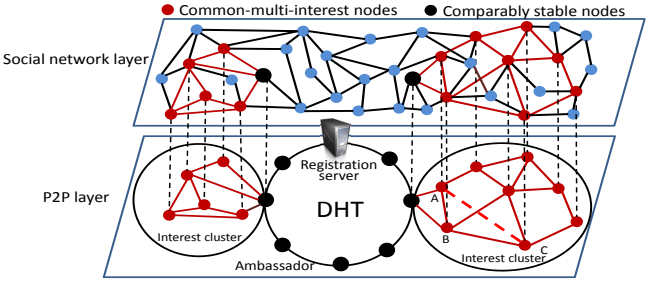


Fig. 7. An overview of the Social-P2P system structure.

for its own cluster, and form all ambassadors to a DHT for efficient inter-cluster data sharing. Like BitTorrent, Social-P2P enables nodes to share their downloaded files with others. Thus, based on I5, a node uses random walk in intra-cluster searching.

4.1 Interest/Trust-based Structure Construction

Social-P2P numerically represents interests of a node based on the Vector Space Model (VSM) [31]. The interests are predefined based on the particular application. For example, for a general-purpose file sharing system such as BitTorrent, the interest pre-determination can be like that in Yahoo! Answers and YouTube. A file sharing system on a campus can use the major names as the interests. It provides an interest dictionary vector, which consists of all the interests (m). Each node compares its own interests with the interest dictionary vector as shown in Figure 8. If it has an interest in the vector, the corresponding position of the vector is set to 1. Otherwise, the position is set to 0. Finally, each node i has an interest vector v_i, which is a binary vector with m dimensions.

Interest Item	Health	Pets	Sports	Science	Arts	⋯	Education	Society	⋯
Peer ID 4123	1	1	0	1	0	⋯	0	1	⋯

Fig. 8. Example of an interest vector.

We use the Hilbert curve technology [32] to cluster common-multi-interest nodes with similar interest vectors. The Hilbert curve converts a multi-dimensional interest vector to a one-dimensional Hilbert value, so that the closeness of the Hilbert values indicates the closeness of the interest vectors, i.e., the similarity between nodes' interests. We use H_{max} to represent the theoretical largest Hilbert value, which depends on the vector dimension. Assume we build n clusters with ID$\in[0, n-1]$. In the case that the Hilbert values are uniformly distributed in the space of $[0, H_{max}-1]$, then $[0, H_{max}-1]$ is uniformly divided to n intervals as shown in Figure 9. A node with Hilbert number $\in[\frac{(b-1)\cdot Max}{n}, \frac{b\cdot Max}{n})$ should be in cluster $(b-1)$. In the case that the Hilbert values are not uniformly distributed in the space of $[0, H_{max}-1]$, we can divide the space $[0, H_{max}-1]$ to intervals based on the density of the distribution of Hilbert values. A node can identify its cluster according to its generated Hilbert value. Other clustering methods [33], [34] can be used to improve the clustering accuracy but at the cost of higher overhead.

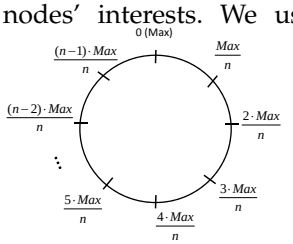


Fig. 9. Hilbert clustering vector.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2359020, IEEE Transactions on Parallel and Distributed Systems

In our future work, we will study the appropriate length of the vector and the granularity of an interest for effective common-multi-interest clustering.

Next, we study the distribution of the number of nodes in a cluster. Using the Hilbert clustering mechanism, we clustered the users in the Facebook trace data based on their interests for comparison. We also used the mechanism to cluster nodes with randomly distributed interests. Recall that the Facebook trace has 18 interests. To generate the random distribution, we assigned each node a certain number of interests randomly selected from the 18 interests (the number is randomly picked from [0,18)), and clustered the nodes based on their Hilbert values. We ranked the clusters by the number of nodes in each cluster. The cluster with the largest number of nodes has the highest rank. Figure 10 shows the distribution of the number of nodes in each cluster versus the cluster rank. It shows that the number of nodes in each cluster conforms to a power law distribution, while the number of nodes in random interest distribution exhibits a small variance. The highly skewed distribution indicates that the interests of people are not randomly distributed but have certain correlation, which is consistent with Figure 1.

Figure 11 shows the average, maximum and minimum number of social friends of each person (degree) in its interest cluster. The average degree ranges from [2.1,18.2], the maximum degree ranges from [6,223] and the minimum degree ranges from [0,1]. Therefore, in most cases, a node has friends in its own common-multi-interest cluster. The average number of friends of each node in our data set is 32.51. Therefore, most nodes have friends in other clusters. Thus, we observe:

O6: In most cases, a node has friends not only in its own common-multi-interest cluster but also in other clusters.

I6: In most cases, each node can establish links with its friends in its own interest cluster, and can ask a friend in another interest cluster to forward a query to that cluster.

Each cluster of common-multi-interest nodes has an ambassador, which is a comparably stable node and is responsible for the inter-cluster file searching. Like current online social networks, Social-P2P has a server managing node registration and ambassadors. The principle of stable node selection is that the longer time a node is online daily in a P2P network, the higher probability it will stay in the network [35]. Initially, the server is the ambassador for each cluster. When a node's online time exceeds a pre-defined threshold, it reports to the server for the promotion to an ambassador. The server designates the node as the ambassador for the cluster if the server is the ambassador. Otherwise, the node becomes a backup ambassador. When an ambassador departs voluntarily, it notifies the server. In the overlay stabilization, when another ambassador notices the abrupt departure or failure of an ambassador, it notifies the server, which selects an alive backup ambassador to replace the leaving ambassador.

Each user is required to submit his *interest information* and *social information* in registration. Interest information, such as leisure preferences and religious beliefs, is used for common-multi-interest node clustering. Social information, such as residence, education and employment, is used to build social links between the nodes within

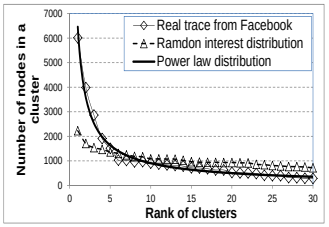


Fig. 10. Number of nodes in an interest cluster.

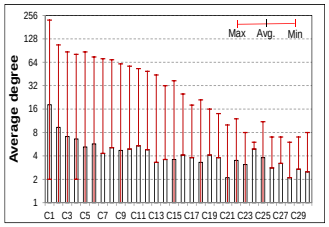


Fig. 11. Average # of social friends of a node in an interest cluster.

a cluster. After a node registers, it calculates its Hilbert number, and then gets the ambassador(s) of its interest clusters from the server. Based on the social information of the node, servers recommend friends to the node, such as classmates in the same college, colleagues in the same company and etc. The node selects familiar friends from the recommendation and adds them into its *social friend list.* It builds overlay links to the friends belonging to its interest clusters. These connected friends become its multi-interest neighbors. If the number of friend nodes is less than a threshold Th_d, the node requests its friends to recommend their trustworthy friends in the same interest clusters as itself. The node connects to the recommended nodes as overlay neighbors in the P2P layer based on the friends-of-friends (FOF) relationship. For example, in Figure 7, node A connects to node C recommended by its friend B as P2P overlay neighbor. As a result, a node's overlay neighbors in Social-P2P include its 1-hop friends and 2-hop FoFs. Querying files from these neighbors ensures trustworthy file sharing, since users possessing a social network primarily interact with 2 to 3 hop partners in real life [36]. If a node has already registered in the system before, when it logs in, it directly connects to its previous overlay neighbors. When a node leaves the system, it needs to notify its neighbors and the server. After being registered, users can add or delete interests in their profiles later on. Then, their interest clusters are updated accordingly.

4.2 Interest/Trust-based File Searching

4.2.1 *Intra-cluster routing algorithm*

A queried file has one or multiple interests. If at least one query interest belongs to the file requester's interests, it uses the intra-cluster routing algorithm that forwards a query to trustworthy nodes. We define the *social distance* between two nodes as the number of hops in the shortest path between them in the social network. As indicated in [37], a reduction in social distance significantly increases trust between nodes. Thus, we use an exponential model to reflect the relationship between trust and social distance. Specifically, the trust weight of node i on j denoted by $w(i,j)$ is calculated by:

$$w(i,j) = e^{(-l_{i,j}-1)}, \quad (1)$$

where $l_{i,j}$ is the social distance between nodes i and j in the social network and "-1" is used for normalization so that the weight of the closest nodes is 1.

Each node employs random walk search, in which a query message is forwarded to one or several randomly chosen P2P network neighbors at each hop until the desired file is found. To avoid repetitive searching paths, a node does not further forward a query if it has received this query previously. A node's P2P network neighbors

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2359020, IEEE Transactions on Parallel and Distributed Systems

are trustworthy since they are friends or FoFs of the nodes in their social networks. Friends should have higher probability than FoFs to be chosen as forwarders because they are relatively more trustworthy. Thus, for a node i with d P2P network neighbors, the probability of a neighbor j being selected from the neighbor set $\mathbb{N}_i$ of i as the message forwarding node is

$$p(i,j) = w(i,j) / \sum_{j \in \mathbb{N}_i} w(i,j). \quad (2)$$

The message is randomly forwarded within the cluster with a time to live (TTL). After a forwarding, the TTL of the message is reduced by 1. The query process is terminated when TTL=0 or when the desired file is found.

4.2.2 Inter-cluster routing algorithm

Inter-cluster querying is needed when users need to query files outside their interests or when the query in the current cluster cannot be satisfied. In this case, using the way described in Section 4.1, based on the multiple interests of the queried file, the requester generates a query vector as generating its interest vector: calculate its Hilbert value and get the ID of the cluster mapped to the Hilbert value. Since the cluster ID represents the common-multi-interests of the nodes in the cluster, the mapped cluster is the destination cluster that holds the requested file. According to I6, the requester first asks its friends in its friend list whether they belong to the destination cluster. If yes, the file request is sent to the cluster through the friend. Otherwise, the requester relies on the ambassador in its current cluster to forward the file request. In this way, the traffic from ambassador can be greatly reduced. Using DHT routing, the request is forwarded to the ambassador in the destination cluster. After the file request arrives at the destination cluster, it is forwarded by the intra-cluster routing algorithm.

Two similar vectors may be divided into two neighboring clusters. For example, a vector with $\frac{Max}{n}$ has cluster ID=1, while a vector with $\frac{Max}{n} - 1$ has cluster ID=0. Therefore, if a query cannot be satisfied within the destination cluster, it is forwarded to the neighboring clusters in both clockwise and counter-clockwise direction with a Cluster TTL (CTTL). Similar to inter-cluster routing, a node tries to send the query via its friends in the social network rather than ambassadors in the DHT. The nodes holding the query with TTL$= 0$ and CTTL$\neq 0$ further forward the request to their neighboring clusters. If the CTTL expires, the query message is sent to the server to locate the file holder. Each node in the system reports files that are seldom queried by others to the server in order to guarantee the file availability. Such files are the files with visit rates lower than a predefined threshold.

4.2.3 Enhanced intra-cluster routing algorithm

In order to further improve the file search speed and efficiency, especially in a large-scale network, we enhance the random-walk routing algorithm using content based routing tables (CRTs). Table 1 shows an example for the CRT. In the table, the "Content Index" refers to the hash value of a file explained in Section 4.2.1, the "Next Hop Node" represents the ID of the neighbor node that can lead to the requested file with the "Content Index", and the "# Hops" denotes the number of remaining hops to reach the requested file.

TABLE 1
Content Based Routing Table

Content Index	Next Hop Node	# Hops
1001001	24	3
1110110	12	5
1010101	23	1
...	...	...

In the enhanced routing algorithm, when a node receives a file request, it first checks whether there is an entry for the requested file in its CRT. If yes, the node further checks whether the trust weight of the next hop node is larger than a predefined threshold, which is high enough to determine that the node is trustable. If yes, the request is forwarded to the node. If there is no entry for the requested file or the next hop node is not trustable, the original random-walk routing algorithm is used.

We then introduce the creation and maintenance of the CRTs. In the random-walk routing, each request records the nodes it has traversed in routing. When a request successfully locates the requested file, the recorded information is forwarded back along the original routing path. Each node in the path then updates its CRT. Specifically, it first checks whether there is an entry for the content index of the requested file. If not, the entry is added directly with the associated next hop and the number of remaining hops. Otherwise, it checks whether the new route has fewer remaining hops to reach the file. If yes, the entry is updated with the new "Next Hop Node" and "# Hops". Figure 12 demonstrates an example of the CRT update with the path information of three successful requests. When a better routing path is discovered, the corresponding entry is updated. Note that such a CRT update will not break the trust property of the routing protocol due to two reasons. First, the new routing path is along trustable nodes as indicated in Section 4.2.1, so the new path is trustable. Second, when using the next hop in a CRT, a node needs to make sure the next hop is trustable. The path information of a successful request is still forwarded back for nodes to verify the activeness of the entries in its CRT. To continually discover better routes, each requester can periodically use the random walk algorithm.

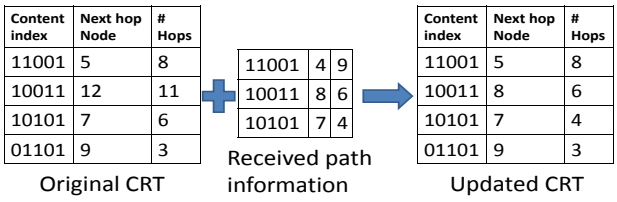


Fig. 12. Update of the content based routing table.

When a requester fails to find its requested content following the CRT, which means the entries are outdated, it sends a notification to all nodes on the path to delete the corresponding entries. Since each content maintains at most one entry in a CRT, the size of a CRT is bounded by the number of contents. In the case of limited memory resource, we can limit the size of the CRT by deleting the entry with the oldest update time.

As random walk sends a query without direction, it may lead to a long path. Letting all nodes along a long routing path record the next hop would generate a high overhead. The small world nature of social networks indicates that two people can be connected through social relationships by six steps on average. Also, nodes

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2359020, IEEE Transactions on Parallel and Distributed Systems

in an interest cluster (i.e., common-interest-nodes) should have more tight connections. Thus, we expect that a node in a cluster can find a file in another node in the same cluster within no more than 6 hops. Therefore, to reduce the CRT maintenance cost without compromising the search efficiency, we propose an advanced algorithm. It only lets nodes that are within N_h ($N_h \leq 6$) hops to the file holder update their CRTs for a successful request. This algorithm eliminates the unnecessary overhead for updating CRTs for long paths, and also enables each node to provide efficient routing guidance for most files. As a result, CRTs can lead the request to the file holder quickly.

4.3 Trust Relationship Adjustment

Based on I3, Social-P2P confines the query traffic to the socially close nodes in order to make sure that the query can be successfully forwarded and the retrieved file is trustworthy. The trust relationship adjustment algorithm enables nodes to avoid forwarding messages to malicious nodes in order to reinforce the trustworthiness of the services in the system. Below, we use file provision as an example for the service. Specifically, when a file requester receives its queried file, if it finds that it receives a faulty file, the node propagates a misbehaving node notification back along the previous query path. Each node i in the routing path adjusts the weight of its link to the next hop on the path, so that it has lower probability of forwarding a message to the misbehaving node. Since a node located closer to a misbehaving node is more likely to forward a query to it, it needs to reduce more weight on the link to the misbehaving node, and vice versa. Also, the number of hops between the querying node and the misbehaving node in the path affects the likelihood of the querying node to send a query to the misbehaving node. With these considerations, we designed Equation (3) for node i in a path to adjust the weight of its link to the next hop j in the path:

$$w(i,j) = w(i,j) - \alpha(\frac{b}{h})^{\frac{h}{\theta}}, \quad (3)$$

where b is the number of hops from the querying node to node i, h is the number of hops between the querying node and the misbehaving node in the path, θ is a scaling parameter and α is a weight parameter. Thus, the nodes that are distant from the misbehaving nodes (small b) reduce less link weights, and the nodes that are closer to the misbehaving nodes reduce more link weights. Like previous reputation systems [13], [15]–[20], we set a threshold Th_w which is a low weight value to identify untrustworthy nodes. If $w(i,j)$ is less than threshold Th_w, node i puts node j into the blacklist and removes the P2P overlay link to j. Since it is possible that some faulty files are sent out by some careless benign peers who did not delete received faulty files from their sharing folder, Social-P2P periodically forgives the occasional misbehavior of nodes in the system in every T_u time interval by increasing every node's weight periodically:

$$w(i,j) = \text{Minimum}\{(w(i,j) + \beta), 1\}, \quad (4)$$

where $\beta > 0$ is the weight increase value at every T_u. T_u is relatively very long and β is very small. Therefore, it needs a very long time for a node to get a maximal reputation 1 through the refreshing. Also, the weight decreasing speed of malicious nodes is much faster than this refreshing speed. Therefore, during each time interval, the variable $w(i,j)$ still functions well.

TABLE 2
Parameter table

Social network topology	Facebook trace
Number of interests	18
Number of clusters	30
Churn rate	Figure 5
CTTL and TTL	3 and 100
Link weight threshold Th_w	0.1
P2P node degree threshold Th_d	3
Link weight update interval T_w	1000s
α, β, θ, T_u	0.05, 0.1, 3, 100s

5 Performance Evaluation

We have conducted trace-driven experiments using the trace data from Facebook and BitTorrent on PlanetSim [38] and PlanetLab testbed [39]. We evaluated the efficiency and trustworthiness of the Social-P2P system in comparison with Partial Indexed Search (PIS) [14] and PROSA [27]. PIS is a hybrid system that clusters the nodes based on their major interests, and also forms the nodes into a DHT to index the non-major interests and globally unpopular files for file retrieval. PROSA is an unstructured P2P system in which nodes that share the same interests are virtually clustered together if they have interacted before. The nodes use random walk to locate interest clusters and to search for files.

We also compare the file searching trustworthiness performance of Social-P2P with Pure-P2P and EigenTrust [40]. Pure-P2P does not have any mechanisms to guarantee file trustworthiness. EigenTrust is a trust management system, in which every peer has a trust manager to calculate its trust value based on others' feedback. A node's trust manager is the DHT owner of the node's ID. Each node sends the rating of the file supplier to its trust manager after receiving a file.

Table 2 lists partial parameters used in the experiments. Other parameters are derived from the trace data from Facebook and BitTorrent. We generated 300,000 synthetic files according to the popularity distribution shown in Figure 6, i.e., the numbers of files in each interest follow the distribution in Figure 6. Each file is represented by an ID and an interest. A file's ID is unique when it is generated, and its interest is selected following the probability distribution derived from Figure 6, i.e., a file's probability to belong to interest l is determined as the portion of torrent for interest l in Figure 6. Each file is assigned to a node randomly selected from nodes whose interests match the file's interest. The interests of the nodes and the number of nodes were determined based on our Facebook trace. The file querying rate is derived from the BitTorrent trace. Figure 13 shows the average querying rate of the nodes in the BitTorrent trace data along with a line for power-law distribution. We rank the nodes in terms of the number of queries issued by the nodes. The node generating the most queries is ranked first. We see that the querying rate of nodes follows a power-law distribution. Thus, we used a power-law distribution generator with scaling exponent parameter k=-1.2 to generate a querying rate within the range of [0.01,100] messages per simulation cycle (i.e., simulated second), and randomly assigned the rate to each node in the system. Since a node is more likely to query files in its interests [11], for each

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2359020, IEEE Transactions on Parallel and Distributed Systems

node, 90% of its initiated queries are for files in its own interests and 10% are not. The churn rate distribution of nodes follows that of Figure 5. After a node leaves the system, it waits for t_w and joins the system again. t_w is randomly selected from [1,10] simulation cycles.

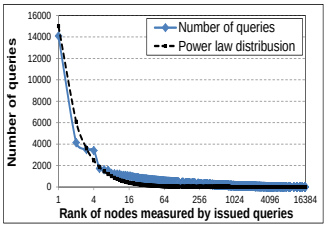


Fig. 13. Querying rate.

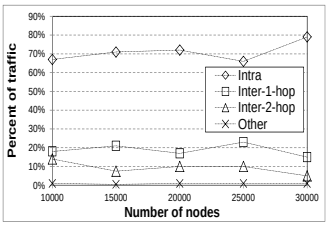


Fig. 14. Traffic distribution.

We primarily use the following six metrics for performance evaluations:
(1) *Percent of traffic:* the percent of query messages forwarded by different kinds of nodes (friends, ambassadors and server) in a searching stage.
(2) *Average query delay:* the average delay of all file queries.
(3) *Query overhead:* the total number of query forwarding hops in file querying of all file queries.
(4) *Maintenance overhead:* the number of messages in maintaining the system structure in churn.
(5) *Overall overhead:* the total number of messages issued for file searching, system maintenance and trust management.
(6) *Victimized probability:* the percentage of nodes receiving faulty files.

We first conducted simulation on PlanetSim (Section 5.1) and then used Planetlab for performance evaluation in real scenario (Section 5.2). In both experiments, we disabled the advanced CRT-based routing algorithm proposed in Section 4.2.3. We evaluate its performance separately in Section 5.3.

5.1 Performance Evaluation on PlanetSim

The number of nodes in the PlanetSim simulation was set to 32,344, which is equal to the number of nodes in our crawled Facebook trace. The duration of each experiment is 50,000 simulation cycles. In each simulation cycle, every node in the system sends out one query message. All experiments have been conducted 10 times, and the average values of the results are reported. In the figures, "Social-P2P" denotes Social-P2P in which a node sends 1 message for a query, and "Social-P2P-c" denotes Social-P2P in which a node sends c messages for a file query in the random walk file searching in Social-P2P.

5.1.1 Evaluation of File Sharing Efficiency

Traffic distribution. Figure 14 shows traffic distribution of the queries in Social-P2P versus network size. In this figure, "Intra" denotes the percentage of traffic in intra-clustering searching. "Inter-1-hop" and "Inter-2-hop" denote the percentages of traffic in inter-cluster searching when the destination cluster is 1 and 2 cluster hops away from the source cluster, respectively. The number of cluster hops describes the distance between two clusters measured by the number of clusters. A cluster is 1 cluster hop away from its neighboring cluster. The inter-cluster searching traffic when the destination is > 2 cluster hops away and the traffic through the server are included in "Other". The figure shows that about 70% of the queries can be satisfied by the nodes within the same cluster. This implies that common-multi-interests clustering can accurately cluster nodes with similar multi-interests. We also see that 99% of the queries can be satisfied within 2 cluster hops. This is because the nodes in neighboring clusters also have similar multi-interests. Therefore, these nodes are very likely to satisfy the queries. This is the reason that there is more traffic within clusters 1 hop away than clusters 2 hops away. The experimental results also show that only 0.1% of the traffic is through the server, which demonstrates the effectiveness of interest/trust-based random walk and P2P file sharing in Social-P2P.

Figure 15 illustrates the distribution of the inter-cluster traffic through friends, ambassadors and the server, respectively. It shows that approximately 80% of the inter-cluster queries are sent to the destination cluster through friends, about 18% of the queries are forwarded through ambassadors, and only 1% are through the server. This result is consistent with O6 that a node has friends in other clusters, which can help the node to forward its query to the other clusters. Since a requester sometimes cannot find a friend in the destination cluster, it resorts to the ambassador for file searching. Due to the TTL, sometimes an unpopular file cannot be discovered. This is the reason that the server contributes to a slight querying traffic.

Query delay. Figures 16 (a), (b) and (c) show the query delay versus network size for queried files with three different popularities, respectively. Popularity of a file is reflected by the percentage of the nodes in the system holding the file. Comparing the three figures, we see that as the popularity of the queried file decreases, the delay in Social-P2P and PROSA increases. Higher popularity files have more copies in the system, hence the probability that the files can be retrieved from its neighbors is high, which results in a low query delay. The query delay in PIS does not increase significantly when file popularity decreases because for querying unpopular files, PIS relies on the DHT, where the IDs of all nodes holding a file are stored together in one node. Thus, an unpopular file can always be located within a limited number of hops. However, the additional DHT structure generates high overhead for structure maintenance. In contrast, for both Social-P2P and PROSA which use random walk for file retrieval, as file popularity decreases, the probability that a file is located near the query node decreases, thus the query delay increases.

The figures also show that the query delay in PIS and PROSA increase significantly with network size while the query delay in Social-P2P increases slightly. In PIS, the nodes inferred their interests from files in their current folders and only main interests are used. For new types of files and non-major interest files, it uses the DHT for file retrieval. Since the average transmission hops in the DHT increases as network size increases, query delay also increases. In PROSA, the clusters are formed based on the interactions between the nodes. In a larger network, it takes longer time before the nodes can be clustered. Also, nodes with the same interests may be grouped to different clusters because of the limited in-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2359020, IEEE Transactions on Parallel and Distributed Systems

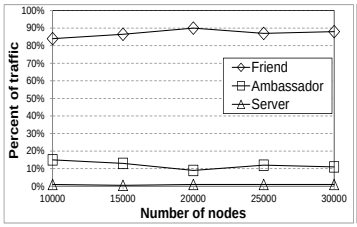
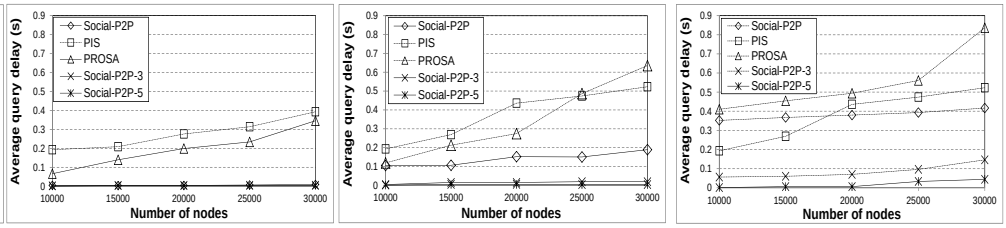
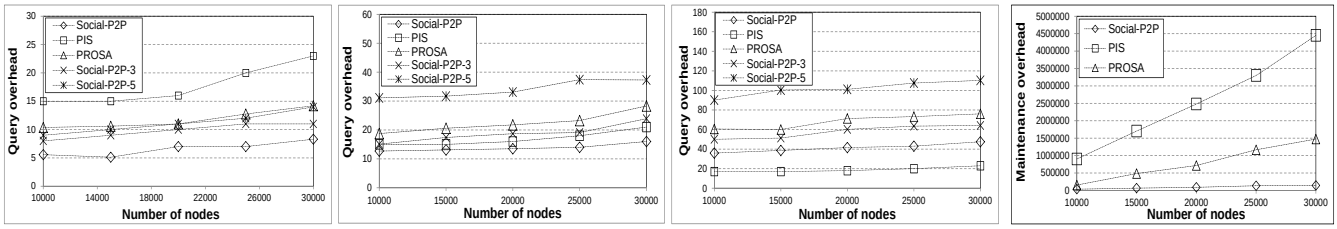


Fig. 15. Inter-cluster traffic distribution.



(a) popularity=50% (b) popularity=5% (c) popularity=0.5%

Fig. 16. Delay versus network size.



(a) popularity=50% (b) popularity=5% (c) popularity=0.5%

Fig. 17. Query overhead versus network size.

Fig. 18. System maintenance overhead.

teraction range of nodes. Thus, the insufficiently accurate clustering in PROSA leads to longer query delay.

Figure 16 (a) shows that for highly popular queried files, the query delay exhibits PIS>PROSA>Social-P2P. PIS only clusters the nodes based on their major interests. However, popular files do not necessarily match the major interests of the nodes. Therefore, PIS needs to refer to the DHT for the file query. The $O(\log n)$ query hops lead to high query delay. Two factors contribute to the higher delay of PROSA than Social-P2P. First, the clustering in Social-P2P is much more accurate than PROSA. In Social-P2P, the nodes are globally clustered based on their multi-interest information in their personal profiles. The query can always be satisfied within the cluster with high probability without searching other clusters. In PROSA, the clustering is based on the interaction history between the nodes. Nodes with the same interests may form several clusters because of the limited interaction range between them. The inaccurate clustering leads to long query delay. Second, Social-P2P has shorter inter-cluster query time. Social-P2P uses a stable DHT to locate a destination cluster. In contrast, the cluster localization in PROSA is based on random walk, which needs more query time.

In Figure 16 (b), we see that for queried files with median popularity, the query delay of PROSA increases rapidly and exceeds DIS when the number of nodes is 30,000. This is because the lower popularity of a file leads to a longer time for node clustering and intra-cluster search in PROSA, and large network size exacerbates the delay due to random walk. Social-P2P can accurately cluster the nodes with similar interests. Therefore its overall delay is lower than PIS. However, as shown in Figure 16 (c), when file popularity is very low, the file search delay of random walk in Social-P2P is long. PIS has a short delay in a small-size network due to the small size of DHT. As a result, Social-P2P generates higher delay than PIS. PROSA leads to higher delay than others because low popularity of the queried file leads to a longer time for node clustering and intra-cluster search.

Figure 16 shows that Social-P2P-3 and Social-P2P-5 have the smallest transmission delay with different popularities of the queried files. This is because sending out more copies of the query messages can increase the hit probability in Social-P2P. Therefore, for unpopular files in the system, Social-P2P can reduce the query delay by sending more query copies.

Query overhead. Figures 17(a), (b) and (c) show the query overhead versus the network size for querying files with three popularities. The figures show that as the network size increases, the amount of system overhead increases, which is the outcome of the increased average query hops in the network.

Figure 17 (a) shows that the query overhead of PIS is larger than all other systems for querying files with high popularity. Due to the high popularity of the queried files, other systems can find the files in the neighbor nodes with high probability. However, DHT routing in PIS leads to high routing overhead. In PROSA, since the nodes are not well clustered initially, it takes more hops to find a file than Social-P2P which are well clustered. Social-P2P-3 and Social-P2P-5 produce higher query overhead than Social-P2P because of more messages. Because every copy of the query message can be satisfied within a small number of hops, the overall overheads of Social-P2P-3 and Social-P2P-5 are less than PIS. As shown in Figures 17 (b) and (c), for the files with lower popularity, the average query overhead in Social-P2P and PROSA increases sharply, because the random walk algorithm takes more hops to meet lower popularity files. For Social-P2P-c, as there are c individual copies sent out for file retrieval, the overhead increases extremely fast. The query overhead in PIS exhibits a very slight increase as popularity decreases because it largely depends on the DHT. The routing overhead in DHT increases over the file popularity due to the same reason as in Figure 16.

We also note that the query overhead of Social-P2P-c is not c times of that of Social-P2P. This is because Social-P2P-c only has multiple messages during the random walk process, while the query overhead refers to the total number of query forwarding hops. Since the random walk only take few hops, the overall query overhead does not increase linearly in Social-P2P-c. All these ex-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2359020, IEEE Transactions on Parallel and Distributed Systems

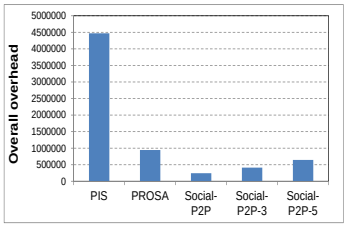


Fig. 19. System overall overhead.

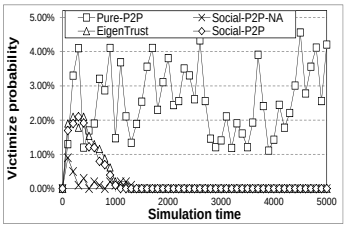


Fig. 20. Victimized probability with malicious nodes.

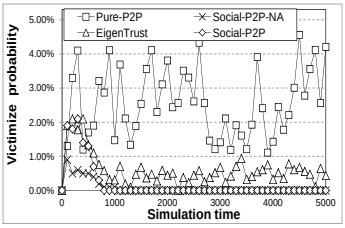


Fig. 21. Victimized probability under collusion.

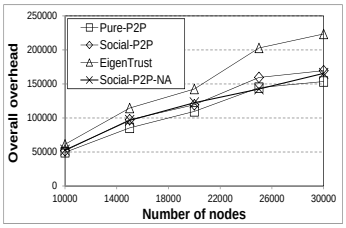


Fig. 22. System overall overhead.

perimental results in Figure 17 verify the low overhead of Social-P2P in file querying.

Maintenance overhead. Figure 18 shows the system maintenance overhead of PIS, PROSA and Social-P2P versus the network size. It shows that PIS has the highest system maintenance overhead and it increases sharply as network size increases. Its overhead is mainly caused by the DHT structure maintenance, which leads to a high overhead especially in churn. Social-P2P constructs ambassadors into a DHT for inter-cluster communication. Since the size of the DHT is small, the ambassadors are relatively stable, and the other nodes only need to maintain their connection with their friends, Social-P2P only produces a slight maintenance overhead. PROSA is also an unstructured P2P network. Since each node must maintain several interest clusters, PROSA generates higher maintenance overhead than Social-P2P and its maintenance overhead increases rapidly as the network size grows.

Overall system overhead. Figure 19 shows the overall system overhead of PIS, PROSA and Social-P2P in a network with 32,344 nodes. The figure shows that PIS has the highest overall system overhead. Although the query overhead of PIS for unpopular files is small, the DHT maintenance overhead in PIS is very large, which leads to an extremely high overhead. Since Social-P2P has a lower query and system maintenance overhead, its overall overhead is the lowest. Although the increasing number of query copies lead to a higher overall system overhead in Social-P2P, it is still less than PIS. PROSA consumes a high overhead for cluster formation and multi-cluster maintenance. Therefore, PROSA has the second highest overall system overhead.

5.1.2 Evaluation of File Sharing Trustworthiness

In this section, we evaluated the querying trustworthiness of Social-P2P in comparison with Pure-P2P and EigenTrust. Social-P2P is evaluated with two mechanisms: (1) The routing is not anonymous. Malicious nodes do not send faulty files to their socially close nodes, but they send faulty files to the requesters 3 hops away in the social network. This mechanism is denoted as "Social-P2P-NA". (2) The routing is anonymous. Malicious nodes are unconstrained and reply to every query with a faulty file. This mechanism is denoted as "Social-P2P". In order to make the methods comparable, all systems use the common-multi-interest clustering mechanism for file sharing. In the experiments, 500 nodes out of the 32,344 nodes were randomly selected to act as malicious nodes. A file requester that has received a faulty file will not forward the file to other nodes.

Performance under malicious nodes. Figure 20 shows the victimized probability over the simulation time. It shows that in Pure-P2P, without any protection, a large percentage of nodes constantly receive faulty files. In EigenTrust, initially the victimized probability is very large and then gradually decreases to 0. This is because the nodes in EigenTrust do not have any reputation initially, which leads to a high victimized probability. However, as EigenTrust decreases the reputation of the malicious nodes, other nodes no longer query files from malicious nodes due to their low reputations. Therefore, the victimized probability of EigenTrust decreases. In Social-P2P-NA, the probability of the nodes receiving faulty files is the lowest initially, because a very small number of queries from socially distant nodes can be received by the malicious nodes. Since trust relationship adjustment can further reduce the probability of forwarding query messages to the malicious nodes, its victimized probability decreases. In Social-P2P, because malicious nodes send faulty files to all requesters, the victimized probability is initially high. Since the neighbors of the malicious nodes can quickly stop forwarding messages to the malicious nodes, the victimized probability decreases sharply. The results imply that if the malicious nodes do not send faulty files to their friends in order to avoid degrading their reputations in real life, Social-P2P can provide higher file sharing trustworthiness than EigenTrust. Even if all malicious nodes are unconstrained and send faulty files to all nodes in the system, the performance of Social-P2P is still comparable to EigenTrust.

Performance under malicious and colluding nodes. Malicious nodes may collude to enhance the reputations of each other by rating each other highly. In this experiment, we let 100 out of the previous 500 malicious nodes to act as colluding nodes. Figure 21 shows that Pure-P2P still suffers from high victimized probability over time due to the same reason in Figure 20. For EigenTrust, since the malicious nodes collude with each other to enhance their reputations, the queries are still forwarded to the colluders. Since nodes generate different amounts of queries over time, the victimized probability of nodes fluctuates over time. We can see that EigenTrust produces much higher victimized probability than Social-P2P and Social-P2P-NA. For both Social-P2P and Social-P2P-NA system, the colluding group can be identified within 1,000s. The reason is that although the weights between the links among the colluders are not reduced, the weights of the links to the colluders are reduced. Therefore, in a short period of time, the entire colluding group is isolated. Since in Social-P2P-NA, only the socially distant nodes from malicious nodes receive faulty files, the victimized probability of Social-P2P-NA is much less than Social-P2P .

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2359020, IEEE Transactions on Parallel and Distributed Systems

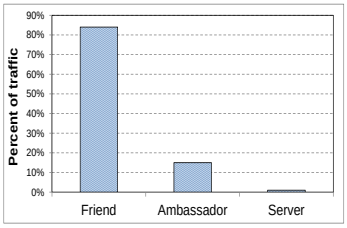


Fig. 25. Inter-cluster traffic distribution.

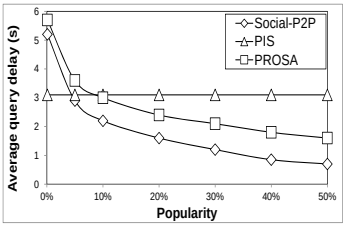


Fig. 26. Query delay versus popularity.

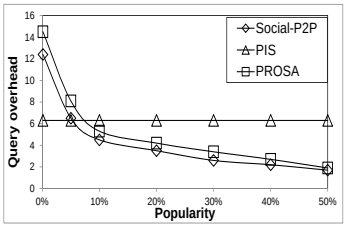


Fig. 27. Query overhead versus popularity.

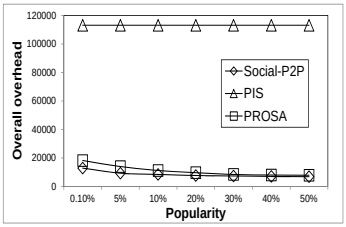


Fig. 28. System overall overhead.

Overall overhead. Figure 22 compares the overall overhead of the Pure-P2P, Social-P2P, Social-P2P-NA and EigenTrust. The figure shows that the overheads of all systems increase as the network size increases since they need to maintain more nodes and queries are routed in a larger scale. The figure shows that Pure-P2P has the lowest overhead because it has no reputation management. EigenTrust incurs more overhead than Social-P2P and Social-NA because it has doubled overhead due to file sharing and trust management. The DHT maintenance and reputation management system leads to a high overhead. The nodes in Social-P2P and Social-NA only need to locally adjust their link trust weights to their neighbor nodes when receiving misbehavior notification messages. Therefore, the overhead in Social-P2P is extremely small and is close to Pure-P2P. Since Social-P2P and Social-P2P-NA have the same trust management and routing mechanisms, their overall overheads are the same. The experimental result verify the advantages of dealing with efficient and trustworthy file sharing simultaneously, and the low overhead of link trust weight adjustment.

Server overhead. We measured the overhead for a server as the number of requests it handles plus the number of recommendations it makes. Querying frequency means the probability that a node generates a query in the round of simulation. We varied the querying frequency from 0.4 to 0.8 in the experiments. Figure 24 shows the server overload versus the number of nodes and the querying frequency. We see that the server overhead increases almost proportionally with the querying frequency. This is because when the querying frequency increases, more queries are generated, leading to more overhead to the server. We also see that the server overhead increases as the number of nodes increases, since more nodes lead to more recommendations from the server and more queries that need to be handled by the server.

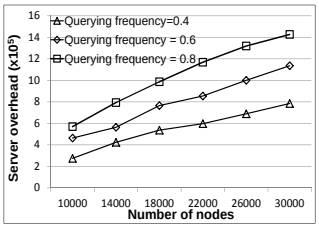


Fig. 24. Server overhead.

5.2 Performance Evaluation on PlanetLab

We implemented a prototype of SocialP2P on PlanetLab in order to show the performance of SocialP2P in the real world environment. We chose 300 online nodes in PlanetLab and chose the computer with the IP address 128.112.139.26 at Princeton University as the server. Since the connection and bandwidth between the nodes vary over time, we ran the client program on each PlanetLab node twice at different times. The average results of each experiment are presented in the paper. We randomly selected 300 nodes from the trace data and mapped the nodes to the PlanetLab nodes. We generated 3000 synthetic files according to the popularity distribution shown in Figure 6 and randomly distributed these files to the nodes whose interests match the file contents. The duration of each experiment was set to 5000s.

5.2.1 Evaluation of File Sharing Efficiency

Figure 25 shows the percent of traffic through friends, ambassadors and servers. We see that almost 80% of the traffic goes through friends, 19% of the traffic goes through ambassadors and only 1% through the server. The result is consistent with Figure 15 due to the same reasons.

Figure 26 plots the average query delays of nodes versus file popularity. The figure shows that as the file popularity increases, the average query delay of PROSA and Social-P2P decreases while the average query delay of PIS remains almost the same. The results of relative performance between different systems are consistent with Figure 16 due to the same reasons. We also find that the absolute results on PlanetLab are larger than those in simulation in Figure 16 since a message needs a longer latency to travel between two nodes in the PlanetLab real-world testbed.

Figure 27 shows the query overhead of Social-P2P, PIS and PROSA versus file popularity. We see that the query overhead of PIS is almost constant and those of PIS and PROSA decrease as the file popularity increases. These results are consistent with those in Figure 17 due to the same reasons. We also find that the absolute results on PlanetLab are smaller than those in simulation in Figure 17 because the PlanetLab experiment has a much smaller network scale.

Figure 28 plots the overall overhead of the systems versus node popularity. We see that PIS has much higher overall overhead than Social-P2P and PROSA. These results are consistent with the simulation results in Figure 19 due to the same reasons. The lower absolute overall overhead in the PlanetLab experiment is caused by the smaller scale network in the test.

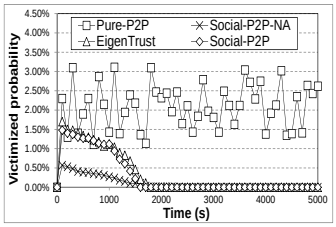


Fig. 29. Victimized probability with malicious nodes.

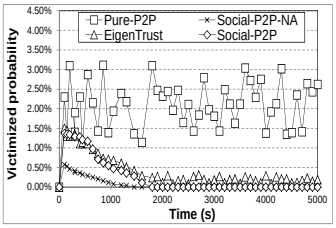


Fig. 30. Victimized probability under collusion.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2359020, IEEE Transactions on Parallel and Distributed Systems

5.2.2 Evaluation of File Trustworthiness

We evaluated the querying trustworthiness of Social-P2P in comparison with Pure-P2P and EigenTrust on PlanetLab. We randomly selected 15 nodes out of 300 nodes as malicious nodes. We evaluated Social-P2P in two scenarios: (1) malicious nodes are not colluders, and (2) malicious nodes are colluders. A node that has received a faulty file will not forward the file to others.

Figure 29 shows the victimized probability of nodes in the system over time. We see that Pure-P2P leads to a high victimized probability over time. In contrast, the victimized probability of the nodes in Social-P2P-NA, Social-P2P and EigenTrust decreases over time. Social-P2P-NA has a much smaller victimized probability than EigenTrust and Social-P2P. Social-P2P can still reach the comparable performance as EigenTrust. These relative results are consistent with Figure 20 due to the same reasons. Figure 30 shows the victimized probability of nodes in different systems over time when the malicious nodes are colluders. Similar to Figure 29, the victimized probability of nodes in Pure-P2P is much higher than all other systems. Social-P2P-NA, Social-P2P and EigenTrust can gradually detect the malicious nodes by trust management. The figure also shows that the victimized probability of EigenTrust is always larger than 0. For both Social-P2P and Social-P2P-NA, the colluding malicious nodes can be effectively detected. The victimized probability of Social-P2P-NA is much less than Social-P2P. These results are consistent with Figure 21 due to the same reasons. We notice that the absolute results of the victimized probability in the PlanetLab experiments are lower than those in simulation because the PlanetLab experiments have fewer malicious nodes than the simulation.

Figure 31 plots the system overall overhead of different systems. The overall overhead in EigenTrust is the highest. The overall overhead in Pure-P2P is the smallest. The system overall overhead in Social-P2P and Social-P2P-NA are comparable to Pure-P2P. We also see that Social-P2P-NA produces smaller overhead than Social-P2P. These relative results are consistent with Figure 22 due to the same reasons. We find that the PlanetLab experiment has higher absolute results than simulation because of its smaller scale of the network.

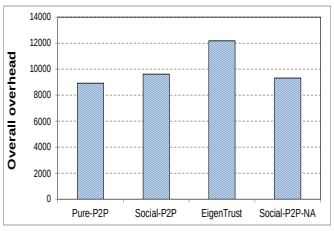


Fig. 31. System overhead.

5.3 Evaluation of the CRT-based Enhanced Routing Algorithm

We further evaluate the CRT-based enhanced routing algorithm proposed in Section 4.2.3. Recall that we proposed both a basic method and an advanced method. For the basic method, we tested two variances, namely CRT1 and CRT2. They are different on the start time when the content based routing table can guide file searching. In detail, the content based routing table in CRT1 and CRT2 starts to guide file searching since the beginning of the experiment and since half of the experiment, respectively. We also tested the performance of Social-P2P without the CRT-based enhanced routing algorithm and with the advanced method, which are denoted as RandomWalk and Advanced-CRT, respectively.

In this test, since memory is not a bottleneck on modern computers, we did not limit the size of the CRT on each node. Considering the social network used in our experiment is more tightly connected (i.e., from Facebook), we set $N_h = 3$ in the Advanced-CRT, which guides the file searching since the beginning of the experiment. In order to maximally and directly show the influence of different variances, we disabled the function of periodically using random walk to discover better routes mentioned in Section 4.2.3.

We focused on the simulation on PlanetSim for performance evaluation but also provided some results from the PlanetLab experiment. In the simulation, we followed the same default setup of cluster structure and file distribution as in the setting in Section 5.1. In the PlanetLab experiment, we followed the settings in Section 5.2. The results of simulation and PlanetLab experiment are shown in Section 5.3.1 and Section 5.3.2, respectively. In both experiments, since the experimental results become stable after tens of seconds of running, we only show the results of 100 cycles in the simulation and 80s in the PlanetLab experiment below.

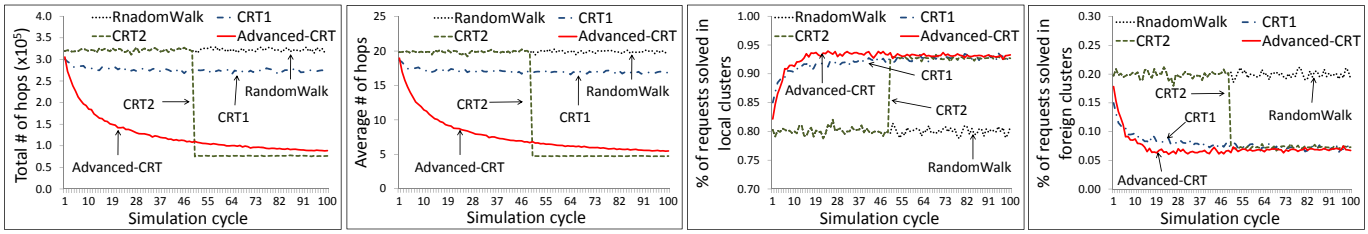
5.3.1 Results from PlanetSim Simulation

Routing Hops: We measured the number of routing hops from a file requester to the file holder for each successful request. Figure 32(a) and 32(b) show the total number and average number of routing hops per request of successful requests in each cycle in the experiments, respectively. We see that RandomWalk has the most query hops in every second. CRT1 reduces the query hops and produces the second most query hops. CRT2 generates the same number of query hops as RandomWalk in the first half of experiment and the lowest number of query hops in the second half of experiment. Advanced-CRT reduces the number of query hops gradually to a very low level.

In RandomWalk, requests are forwarded blindly, leading to the most query hops. CRT1 reduces the hops of RandomWalk due to guidance of CRTs. However, since it uses the first sets of routes discovered through random walk to guide subsequent requests, it cannot discover better routes. Thus, CRT1 only slightly decreases the number of query hops of RandomWalk. In CRT2, the CRTs are well built in the first 50 seconds, so they can guide route requests to file holders effectively, leading to the lowest number of hops. In Advanced-CRT, routes are only built on nearby nodes of file holders. Therefore, nodes gradually learn the nearby contents to guide requests, and the number of query hops reduces gradually to a very low level. This result indicates that a small N_h can ensure the file search efficiency.

Percentage of Routing Hops in Local/Foreign Clusters: Recall that if a request cannot find the requested file in a local cluster, it searches in the foreign clusters. We measured the percentage of routing hops in local clusters and foreign clusters of successful requests. The results are shown in Figure 32(c) and Figure 32(d). Figure 32(c) shows that RandomWalk has the lowest percentage of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2359020, IEEE Transactions on Parallel and Distributed Systems



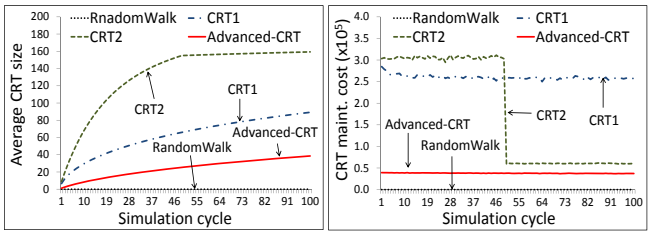
(a) Total number of routing hops. (b) Average number of routing hops. (c) Percentage of hops in local clusters. (d) Percentage of hops in foreign clusters.

Fig. 32. Performance of different CRT-based enhanced routing algorithms in PlanetSim simulation.

routing hops in local clusters and CRT1 has the second highest percentage. CRT2 has the same percentage as RandomWalk in the first half of experiment and high percentage in the second half of experiment. Advanced-CRT increases the percentage gradually to the highest level. Figure 32(d) exhibits the opposite trends since the sum of the two percentages equals 1. RandomWalk has a low percentage of hops in local clusters because requests are always forwarded through the random walk, which has a high probability to fail in the local cluster and resort to the neighboring clusters. For CRT2, it is the same as RandomWalk in the first half of experiment as it does not use CRTs in file searching. In the second half of experiment, the well-built CRTs lead requests to file holders in the same cluster quickly, leading to a higher percentage in local clusters. In CRT1 and Advanced-CRT, their CRTs provide file locations within local clusters, leading to a high percentage of hops in local clusters.

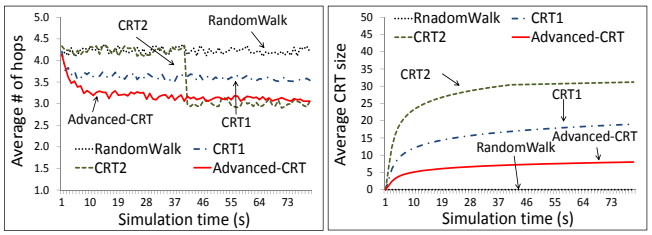
Costs of the Content Based Routing Tables: Figure 33(a) and 33(b) demonstrate the average size (i.e., total number of entries of all CRTs divided by the total number of nodes) and the maintenance cost (i.e., the number of routing path information forwards) of the CRTs in each cycle, respectively. We also include RandomWalk in the figures for reference though it has no cost on CRTs. Figure 33(a) shows that the average size of CRTs follow CRT2>CRT1>Advanced-CRT. CRT2 produces the largest average CRT size due to two reasons. First, it builds CRTs in all nodes along the full path of each successful request. Second, it builds CRTs during the first half of experiment. In the second half of test, requests just follow the established CRTs and do not discover new routes for existing routes, leading to nearly stable average size of CRTs. In CRT1, once an entire route from a node to a file holder is built along nodes in the path, no new routes for the queried file will be discovered. Thus, CRT1 produces smaller average CRT size than CRT2. CRT1' average size increases gradually as routes for new files are added. For Advanced-CRT, each node only needs to record the neighbors for the nearby contents and no new routes for queried files will be discovered, leading to the smallest CRT size.

Figure 33(b) shows that in each cycle, CRT1 always generates a high maintenance cost, CRT2 has the highest maintenance cost in the first half of the experiment and a low maintenance cost thereafter, and Advanced-CRT produces the lowest maintenance cost. CRT1 has high maintenance cost because the path information of each successful request is sent to all nodes on the path to update the routes. In CRT2, the whole path information is forwarded along all nodes in a path in the first half of the test, leading to a high maintenance cost. In the



(a) Average size of the CRT tables. (b) Maintenance cost of the CRT tables.

Fig. 33. Costs of different CRT-based enhanced routing algorithms in PlanetSim simulation.



(a) Average number of routing hops. (b) Average size of the CRT tables

Fig. 34. Performance of different CRT-based enhanced routing algorithms in PlanetLab experiment.

second half of the test, the CRTs are used in file searching and path information is forwarded along nodes in a shorter path, reducing the maintenance cost to a low level. Advanced-CRT limits the number of nodes to receive the path information to $N_h = 3$, generating the lowest maintenance cost. With the above results, we conclude that Advanced-CRT leads to search performance improvement comparable to CRT but at the lowest cost.

5.3.2 Results from Planetlab Experiment

Figures 34(a) and 34(b) show the average number of routing hops and average size of the CRT tables in the PlanetLab experiment. Note that results on other metrics are not shown here due to page limit and the fact that they show similar trends as the two metrics .We see that the relationship between the performance of four methods is consistent with those in PlanetSim Simulation, i.e., Figures 32(b) and 33(a). The reasons are also the same. Such results demonstrate the superiority of the advanced-CRT algorithm.

We also see that the four methods have smaller average routing hops and average CRT table size in the PlanetLab experiment than in the PlanetSim simulation. This is because the PlanetLab experiment is on a small scale with only 300 nodes. This further shows the scalability of the proposed CRT based routing algorithm.

6 Conclusion

In this paper, driven by the observations from the trace data of Facebook and Bittorrent file sharing system, we

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2359020, IEEE Transactions on Parallel and Distributed Systems

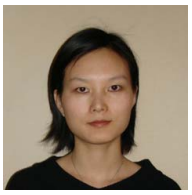
propose Social-P2P that synergistically integrates a social network into a P2P network for efficient and trustworthy file sharing. Taking advantage of the interest information in the social network, the socially close nodes with similar multi-interests are clustered together. Nodes are connected with their friends within a cluster. Within each cluster, trust-based routing algorithms are proposed to forward a query message along trustworthy links, enhancing file searching efficiency and trustworthiness. Comparatively stable nodes from clusters form a DHT for inter-cluster communication. Nodes also decrease the trust weights of links to their neighbors which have high probability to forward messages to misbehaving nodes. The experimental results from trace driven simulations and the prototype on PlanetLab demonstrate the efficiency and trustworthiness of file sharing in Social-P2P in comparison with other file sharing systems and trust management systems. In our future work, we will exploit the Information Centric Networking (ICN) routing to improve the file searching performance.

Acknowledgements

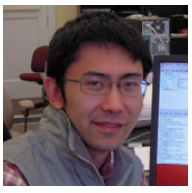
This research was supported in part by U.S. NSF grants NSF-1404981, IIS-1354123, CNS-1254006, CNS-1249603, CNS-0917056, Microsoft Research Faculty Fellowship 8300751.

References

[1] P2P Statistics. http://www.freemusictodownload.eu/p2p-statistics.html [accessed in Aug. 2014].
[2] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *TON*, 2003.
[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of SIGCOMM*, 2001.
[4] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware*, 2001.
[5] Y. Liu, X. Liu, L. Xiao, L.M. Ni, and X. Zhang. Location-aware topology matching in P2P systems. In *Proc. of Infocom*, 2004.
[6] Y. Liu, L. Xiao, and L. M. Ni. Building a scalable bipartite P2P overlay network. *TPDS*, 2007.
[7] H. Xie and Y. R. Yang. P4P: Provider portal for applications. In *Proc. of ACM Sigcomm*, 2008.
[8] D. R. Choffnes and F. E. Bustamante. Taming the torrent: A practical approach to reducing cross-isp traffic in P2P systems. In *Proc. of Sigcomm*, 2008.
[9] A. Iamnitchi, M. Ripeanu, and I. Foster. Small-world file-sharing communities. In *Proc. of Infocom*, 2004.
[10] J. Lei and X. Fu. Interest-based peer-to-peer group management. *Lecture notes in computer science*, 2009.
[11] A. Fast, D. Jensen, and B. N. Levine. Creating social networks to improve peer to peer networking. In *Proc. of KDD*, 2005.
[12] Y. Li, L. Shou, and K. L. Tan. Cyber: A community-based search engine. In *Proc. of P2P*, 2008.
[13] S. Marti, P. Ganesan, and H. G. Molina. DHT routing using social links. In *Proc. of IPTPS*, 2004.
[14] R. Zhang and Y. C. Hu. Assisted peer-to-peer search with partial indexing. *TPDS*, 2007.
[15] L. Xiong and L. Liu. PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities. *TKDE*, 2004.
[16] R. Zhou and K. Hwang. PowerTrust: A robust and scalable reputation system for trusted peer-to-peer computing. *TPDS*. 2007.
[17] R. Zhou, K. Huang, and M. Cai. GossipTrust for fast reputation aggregation in peer-to-peer networks. *TKDE*, 2008.
[18] S. Song, K. Hwang, R. Zhou, and K. Y. Kwok. Trusted P2P transactions with fuzzy reputation aggregation. *Internet Computing*, 2005.
[19] Z. Liang and W. Shi. PET: a personalized trust model with reputation and risk evaluation for P2P resource sharing. In *Proc. of HICSS*, 2005.
[20] A. Selcuk, E. Uzun, and M. R. Pariente. A reputation-based trust management system for P2P networks. *IJNS*, 2008.
[21] H. Shen, Z. Li, H. Wang, and J. Li. Leveraging Social Network Concepts for Efficient Peer-to-Peer Live Streaming Systems. In *Proc. of ACM Multimedia*, 2012.
[22] H. Shen, Z. Li, and J. Li. A DHT-Aided Chunk-Driven Overlay for Scalable and Efficient Peer-to-Peer Live Streaming. *TPDS*, 2010.
[23] X. Cheng and J. Liu. Nettube: Exploring social networks for peer to peer short video sharing. In *Proc. of Infocom*, 2009.
[24] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. van Steen, and H. J. Sips. Tribler: A social-based peer-to-peer system. In *Proc. of IPTPS*, 2006.
[25] S. Marti, P. Ganesan, and H. Garcia-Molina. SPROUT: P2P Routing With Social Networks. In *Proc. of P2P&DB*, 2004.
[26] S. Marti, P. Ganesan, and H. Garcia-Molina. DHT routing using social links. In *Proc. of IPTPS*, 2004.
[27] V. Carchiolo, M. Malgeri, G. Mangioni, and V. Nicosia. An adaptive overlay network inspired by social behavior. *JPDC*, 2010.
[28] W. Galuba, K. Aberer, Z. Despotovic, and W. Kellerer. Leveraging social networks for increased bittorrent robustness. In *Proc. of CCNC*, 2010.
[29] D. Frey, A. Jégou, A.-M. Kermarrec, M. Raynal, and J. Stainer. Trust-aware peer sampling: Performance and privacy tradeoffs. *Theoretical Computer Science*, 2013.
[30] Bittorrent user activity traces. http://www.cs.brown.edu/∼pavlo/torrent/ [accessed in Aug. 2014].
[31] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 1975.
[32] D. Bayer and M. Stillman. Computation of Hilbert functions. *JSC*, 1992.
[33] H. Shen, Y. Lin, and T. Li. Combining efficiency, fidelity and flexibility in resource information services. *TC*, 2013.
[34] H. Shen and K. Hwang. Locality-preserving clustering and discovery of resources in wide-area distributed computational grids. *TC*, 2012.
[35] P. Godfrey, S. Shenker, and I. Stoica. Minimizing churn in distributed systems. In *Proc. of Sigcomm*, 2006.
[36] G. Swamynathan, C. Wilson, B. Boe, K. C. Almeroth, and B. Y. Zhao. Can Social Networks Improve e-Commerce: a Study on Social Marketplaces. In *Proc. of WOSN*, 2008.
[37] C. Binzel and D. Fehr. How Social Distance Affects Trust and Cooperation: Experimental Evidence from A Slum. In *Proc. of ERF*, 2009.
[38] P. Garca, C. Pairot, R. Mondéjar, J. Pujol, H. Tejedor, and R. Rallo. PlanetSim: a new overlay network simulation framework. In *Proc. of SEM*, 2004.
[39] L. Peterson, A. Bavier, M. Fiuczynski, and S. Muir. Experiences implementing PlanetLab. In *Proc. of OSDI*, 2006.
[40] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proc. of WWW*, 2003.



Haiying Shen received the BS degree in Computer Science and Engineering from Tongji University, China in 2000, and the MS and Ph.D. degrees in Computer Engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Associate Professor in the Department of Electrical and Computer Engineering at Clemson University. Her research interests include distributed computer systems and computer networks, with an emphasis on P2P and content delivery networks, mobile computing, wireless sensor networks, and cloud computing. She is a Microsoft Faculty Fellow of 2010, a senior member of the IEEE and a member of the ACM.



Ze Li received the BS degree in Electronics and Information Engineering from Huazhong University of Science and Technology, China in 2007, and the Ph.D. degree in Computer Engineering from Clemson University. His research interests include distributed networks, with an emphasis on peer-to-peer and content delivery networks. He is currently a data scientist in the MicroStrategy Incorporation.

Kang Chen received the BS degree in Electronics and Information Engineering from Huazhong University of Science and Technology, China in 2005, the MS in Communication and Information Systems from the Graduate University of Chinese Academy of Sciences, China in 2008, and the PhD in Computer Engineering from Clemson University. He is currently a Postdoctoral Fellow in the Department of ECE at Clemson University. His research interests include vehicular networks, software defined networks, mobile ad hoc networks and delay tolerant networks.