Social-P2P: Social Network-based P2P File Sharing System

Ze Li and Haiying Shen
Department of Electrical and Computer Engineering
Clemson University, Clemson, SC 29631
Email: {zel, shenh}@clemson.edu

***Abstract*—A peer-to-peer (P2P) file sharing system provides a platform that enables users to share their files. Retrieving files efficiently and trustworthily in such a large and jumbled system is critically important. However, the issues of efficient searching and trustworthy searching have only been studied separately. Simply combining two separate strategies dealing with each issue doubles system overhead. In this paper, we first study trace data from Facebook and BitTorrent. Guided by the study observations, we propose a P2P system based on social networks for simultaneous efficient and trustworthy file sharing, namely Social-P2P. Social-P2P 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 DHT for inter-cluster file searching. A file query is forwarded to the cluster of the file by the DHT routing and then is forwarded along constructed connections within a cluster, which achieves high hit rate and reliable routing. Sharing files among socially close friends discourages nodes from providing faulty files since people are unlikely to risk their reputation in the real world. Experimental results show that by leveraging a social network, Social-P2P achieves highly efficient and trustworthy file sharing.**

I. Introduction

Peer-to-peer (P2P) file sharing is a very popular application that are widely used in our daily life (e.g. BitTorrent [1]). Currently, more than 50% of the files downloaded and 80% of the files uploaded on the Internet are through P2P networks [2]. P2P file sharing systems attract millions of users [2]. Due to the large-scale of the P2P systems, how to efficiently locate a desired file has been an open problem for many years. Also, considering that ubiquitous users without preexisting trust relationships share files in the P2P open platform, how to provide trustworthy file (i.e., authentic file without virus) sharing has become another important issue. Indeed, many users find themselves downloading the wrong files due to misleading file names and descriptions [3]. Peers with malicious intent upload faulty files, such as tampered files or files with malicious code (e.g., Trojan horses and viruses), as legitimate files. A study in 2010 reported that among the surveyed people who acquired music from a P2P network, 41% downloaded spyware, 39% downloaded a virus or trojan and 32% downloaded unwanted explicit content [4]. These problems may scare many users away from the P2P file sharing application.

978-1-4673-2447-2/12/$31.00 ©2012 IEEE

However, the research on the issues of efficient and trustworthy P2P file searching has been conducted separately. Also, though 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 [5]–[17]. Since the methods cluster nodes based on 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 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 users that have deleted the shared files. A widely-used solution for trustworthy file sharing is to employ a trust management system [18]–[21], in which each node rates the service quality of service providers. However, accumulating sufficient ratings for calculating an accurate trust may need 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 trust management system. In this way, all nodes need to maintain structures for two systems, which doubles system cost, 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.

Recently, online social networks such as Facebook have gained significant popularity. Users can publish their personal profiles, which include information such as personal interests, career and education. Users can also contact others with close social relationships such as kinship and friendship. In this paper, we propose a peer to peer file sharing system that based on online social network, namely Social-P2P, to simultaneously achieve efficient and trustworthy P2P file sharing by leveraging social interests and relationships. Three facts lay the foundation for this work. First, it is reported that Americans spend almost a quarter of their time online on social networking sites [22]. Therefore, a large number of active and stable nodes will be available in the system to provide resources to each other. Second, people usually share files that they are interested in [14]. Interests indicated by a user himself in his profile can more accurately reflect the complete interests of the user. Third, 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 human 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 the paper, we first study trace data from Facebook and BitTorrent. We are trying to find the possible gains of deploying P2P file sharing applications in online social networks. We gain a number of interesting 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, he is very likely to also have interest B.
O2: A user in the online social network has different contact frequency with different persons.
O3: Friends in the online social network usually have certain social relationship(s) in their real life.
O4: A P2P file sharing system possesses a certain percent of comparably stable nodes.
O5: Most file queries are for popular file categories.

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 socially close nodes as P2P overlay neighbors (O3). The component aims to increase the file sharing efficiency.

(2) **Interest/trust-based file searching.** The trustworthiness between nodes is weighed and a node tends to forward a file query to higher 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 (O5). The component aims to increase the file sharing efficiency and trustworthiness at the same time.

(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). Furthermore, Social-P2P uses anonymous routing to prevent malicious nodes from selectively attacking socially distant nodes and protects the privacy of the nodes. The component aims to further enhance the file sharing trustworthiness.

As far as we know, this is the first work that simultaneously considers both efficient and trustworthy file querying with low overhead of P2P file sharing in online social networks.

II. Trace Data Analysis

In this section, we analyze Facebook trace data crawled by us to study people's social and interests information and BitTorrent trace data from the Graffiti Network Project [23] to study people's P2P file sharing behaviors. 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 direct social relationship 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. 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.

*a) **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") and 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.

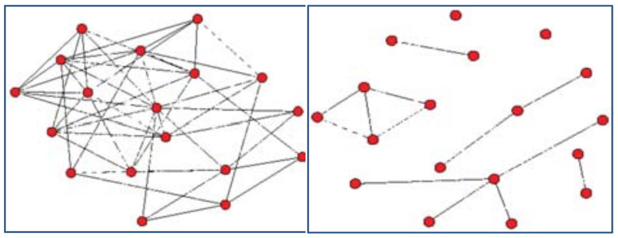

(a) T = 100 (b) T = 500

Fig. 1. The clustering feature of interests.

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 interestes 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 individual interest, which leads to high overhead for cluster maintenance, clustering common-multi-interest nodes can improve file retrieval efficiency and reduce cluster maintenance overhead. For example, suppose a user A has m interests. For single interest based clustering, user A need maintain m interest clusters. Suppose in each interest cluster, the user A need maintain d neighbors. Then, user A need maintain $m * d$ links. If the average churn rate of a user is r, then the overhead for a user to maintain m interest cluster is $m*d*r$. In contrast, for multi-interest based clustering, user A only need maintain d links with cluster maintenance overhead as $d * r$.

*b) **Closeness between online users**:* We analyzed the reply rate of the posts on user comment walls and pictures in Facebook. The reply rate of user A to user B is defined as the

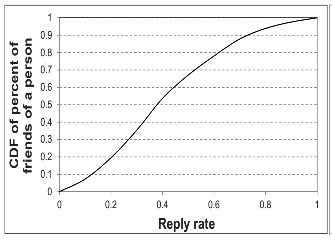


Fig. 2. Distribution of closeness.

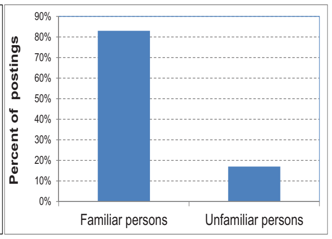


Fig. 3. Distribution of postings.

percent of B's posted comments on A that are replied by A. 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 find whether the behavior of replying comments is driven by social closeness of nodes or the content of the comments (e.g. interesting comments), we further investigate the comments posting behaviors between people. For 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. From the results in both figures, we observe:

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

Current research [24] shows that the users' contact frequency indicates the trust between them. Thus we can infer:

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 certain social relationship(s) in their real life.

I3: Requesting services (e.g., providing files and query routing) from socially close nodes can enhance the trustworthiness of received services, since people do not want to ruin their reputation in real life.

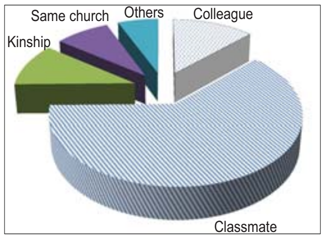


Fig. 4. Social relationship.

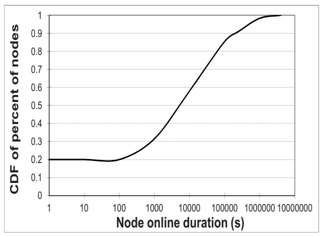


Fig. 5. Churn rate of the nodes.

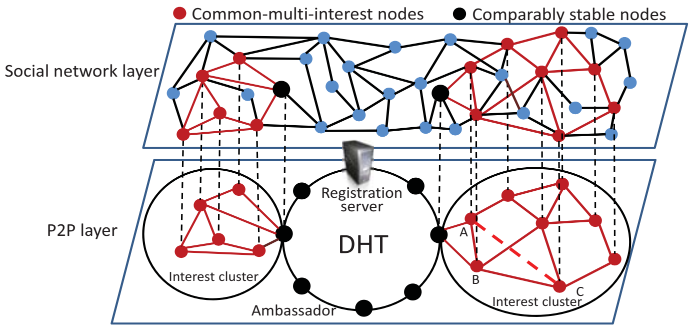


Fig. 7. An overview of the Social-P2P system structure.

*c) **Node stability**:* We now analyze the distribution of node stability in BitTorrent. Figure 5 shows the cumulative distribution function (CDF) of the online time duration of nodes in the system. It shows that 20% of the nodes leave the system within 1s after they finish file downloading, while 1.5% of the nodes stay in the system for more than 10^5s.

O4: A P2P file sharing system possesses a certain percent of comparably stable nodes (1.5%) and a large percent of 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.

*d) **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 Zipf distribution. We see that the popularity distribution of interests can be modeled as a Zipf distribution.

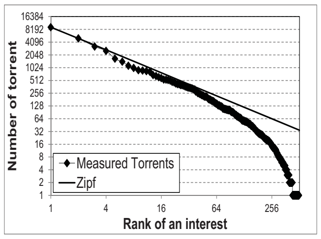


Fig. 6. File popularity distribution.

O5: According to the Zipf distribution, most of the file queries (80%) are for a small percent (20%) of file categories (interests), which are the popular categories.

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

I5: In the case that higher popularity files have more file copies being held by different users in the system, the random walk file searching algorithm can achieve high hit rate in retrieving popular files as it is likely to meet a user that hold the popular file nearby.

III. SOCIAL-P2P: SOCIAL NETWORK-BASED P2P SYSTEM

Based on the above observations and inferences, we propose Social-P2P, which leverages the *social closeness* and *interest* information in the social network to enable nodes with close social relationship and common multiple 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 for its own cluster, and form all ambassadors to a DHT for efficient inter-cluster file sharing [5]–[7]. 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. Social-P2P can build a social network connecting the users. Social-P2P can also be used as a plugin in current on-line social online networks, such as Facebook, MySpace to provide anonymous file sharing services.

A. *Interest/Trust-based Structure Construction*

Social-P2P numerically represents interests of a node based on the Vector Space Model (VSM) [25]. It provides an interest dictionary vector, which consists of all interests of the nodes in the system. 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 and m is the number of all interests.

Interest Item	Piano	Violin	Sci-Fi	Action	Horror	···	Football	Soccer	···
Peer ID 4123	1	1	0	1	0	···	0	1	···

Fig. 8. An example of an interest vector.

We use the Hilbert curve [26] to converts a multi-dimensional interest vector to an one-dimensional Hilbert value. The closeness of the Hilbert values indicates the closeness of the interest vectors, i.e., the similarity between nodes' interests. Then we use hash table to cluster the nodes based on their Hilbert values. The nodes with similar Hilbert values are located in the same cluster or adjacent cluster. By adjusting the number of clusters, we can change the resolution of the clustering. Specifically, we use H_{max} to represent the theoretically largest Hilbert value, which depends on the vector dimension. Assume we build n clusters with ID$\in[0, n-1]$, then $[0, H_{max}-1]$ is uniformly divided to n intervals. A node with Hilbert value $\in[\frac{(a-1)\cdot H_{max}}{n}, \frac{a\cdot H_{max}}{n})$ $(1 \leq a \leq n)$ is assigned into cluster $(a-1)$. Thus, a node can identify its cluster according to its generated Hilbert value. Comparing to some commonly used clustering method such as K-Means [27], which is computationally intensive, the Hilbert curve based clustering need much less computation costs.

Next, we study the distribution of the number of nodes in a cluster. Using the Hilbert clustering mechanism, we clustered the persons in the Facebook trace data based on their interests. We also used the mechanism to cluster nodes with randomly distributed interests for comparison in order to further justify

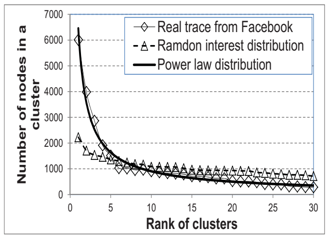


Fig. 9. Number of nodes in an interest cluster.

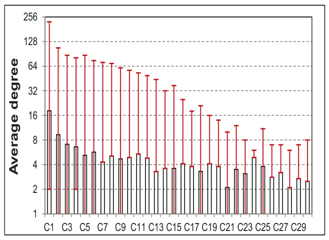


Fig. 10. Ave. # of social friends of a node in an interest cluster.

our clustering method. To generate the random distribution, we assigned each node a random number of interests randomly selected from the 18 interests, 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 9 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 in the Facebook trace conforms to a power law distribution, while the number of nodes in random interest distribution exhibits a small variance. The highly skewed distribution in the Facebook trace indicates that the interests of people are not randomly distributed but has certain correlation, which is consistent with O1.

Figure 10 shows the average, maximum and minimum number of social friends of each person (i.e., degree) in its interest cluster. The average degree, maximum degree and minimum degree range from [2.1,18.2], [6,223] and [0,1] respectively. Therefore, a node has a large number of friends in its own common-multi-interest cluster. Recall that the average number of friends of each node in our data set is 32.51. Therefore, most of nodes should have friends in other clusters. Thus, we observe:

O6: A node has a large number of friends in its own common-multi-interest cluster, and also has friends in other clusters.

I6: Each node is able to establish links with its friends in its own interest cluster, and it 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 that is responsible for the inter-cluster searching between its responsible cluster and other clusters. 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 [28]. Initially, the server is the ambassador in each cluster. When a node's online time exceeds a pre-defined threshold (i.e., 10% of the most stable nodes in each cluster), it reports to the server for the promotion to an ambassador. The server assigns it as the ambassador in its cluster if the server is the ambassador of that cluster. Otherwise, the node becomes a backup ambassador, which becomes the ambassador of the cluster after the current ambassador leaves.

Each user is required to submit his *interest information* and *social information* in registration. Interest information, such as preference of file resources, 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 a cluster. After a node registers, it calculates its Hilbert value, 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 friend recommendation can accelerate the process of building a social network. 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 the same interest clusters as itself. If the number of friend nodes is less than a threshold Th_d, the node requests its friends to recommend their trustable 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 Friend of Friend (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 [29]. 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.

B. Interest/Trust-based File Searching

1) Intra-cluster routing protocol: Using this protocol, a node forwards a query to a trustworthy node in the intra-cluster routing. Recall a node has overlay links with its friends and FoFs in a cluster. We define the social distance between two nodes as the number of hops in the shortest path between them in the social network. We use an exponential model to reflect the relationship between trust and social distance. Specifically, the trust weight of node i on node j denoted by $w(i,j)$ is calculated by:

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

where $l_{(i,j)}$ is the social distance between node i and node j. This relationship has been confirmed by other studies [29], [30]. Binzel et al. [30] discovered that a reduction in social distance between two persons significantly increases the trust between them. Swamynathan et al. [29] found that people normally conduct e-commerce business with people within 2-3 hops in their social network.

Each node employs random walk search, in which a query message is forwarded to several randomly chosen P2P network neighbors at each hop until the desired file is found. A node's P2P network neighbors are trustable since they are friends and FoFs of the nodes in their social networks. We hope that friends have higher probability than FoFs to be chosen as forwarders because they are relatively more trustable. Thus, the probability of a neighbor j being selected from the P2P neighbor set $\mathbb{N}_i$ of i as the message forwarding node is

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

Specifically, a node sequentially maps its neighbors to interval [0,1], and a neighbor with $p(i,j)$ owns a segment with length=$p(i,j)$. The node randomly generates a value within [0,1], and the neighbor who is the owner of the segment covering the value is selected as the next hop. The message is randomly forwarded within the cluster with a time to live (TTL) time stamp. For every forwarding hop, the TTL is reduced by 1. The query process is terminated when TTL=0 or the desired file is found. A file request indicates the file name or keywords. Upon receiving a file request, a node checks whether it has the requested file using the Bloom filter method [31].

2) Inter-cluster routing protocol: Inter-cluster querying is needed when users need to query files not within their interests or the query in the current cluster cannot be satisfied. In this case, using the same way as described in Section III-A, the requester generates a query vector as generating its interest vector, calculates its Hilbert value, and identifies 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, we know a node has some friends in other clusters. Thus the requester first asks its friends in its friend list whether they belong to the destination cluster. If yes, the query is sent to the cluster through the friend. Otherwise, the requester relies on the ambassador in its current cluster to forward the query. Using DHT routing, the ambassador forwards the query to the ambassador in the destination cluster, and then the query is forwarded by the intra-cluster routing protocol.

Two similar vectors may be divided into two neighboring clusters. 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 server to locate the file holder. Each node in the system reports the files that are seldom queried by other nodes to the server in order to guarantee the file availability.

C. 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 protocol enables nodes to avoid forwarding messages to malicious nodes in order to reinforce the trustworthiness of the services in the system. When a node receives a faulty file from a malicious node, the node propagates a misbehaving node notification back along the previous query path. Each node i in the routing path decreases the weight of its link to the previous 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 link weight. Specifically, node i adjusts its link to the previous hop j by:

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

where b is the number of hops from the requester to node i, h is the number of hops between the requester 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 vice versa. If $w(i,j)$ is less than a threshold Th_w, node i puts node j into the blacklist and removes the P2P overlay link to j. Social-P2P periodically forgives the occasional misbehavior of nodes every T_u time interval by increasing every node's weight periodically:

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

where $\beta > 0$ is the weight increase value at every T_u.

Anonymity. A big concern of P2P file sharing users is privacy. Some users do not wish to be identified as a file provider or file receiver by their friends. Also, to counter Social-P2P's strategy for trustworthy file sharing based on I3, a malicious node may selectively provide faulty files to socially distant nodes or falsely accuse a normally behaving node of misbehaviors. Anonymity routing can protect node privacy and prevent such misbehaviors by preventing a node from finding out the initiator and receiver of a query. As most of the files are shared among socially close node, a malicious node dares not to arbitrary attack the initiator or receiver of a query. Therefore, Social-P2P uses a lightweight anonymous routing protocol. In the protocol, the source of a query is removed from the query message. After a file is discovered, the file is sent back along the previous query path. Thus, the forwarding nodes in the routing path do not know who provided the file. As a node in the path only knows its predecessor and successor which are its socially close friends, and the two end points could be anywhere among the network's hundreds of thousands of nodes, the file sharing achieves anonymity. Authentication and encryption techniques can be further used to encrypt b and h to hide them from users [32].

IV. Performance Evaluations

We have conducted trace-driven experiments using the trace data from Facebook and BitTorrent on PlanetSim [33]. We evaluated the efficiency and trustworthiness of the Social-P2P system in comparison with Partial Indexed Search (PIS) [17] and PROSA [34]. 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 the nodes share the same interests are virtually clustered together if they have interactions before. The nodes use random walk to locate interest clusters and to search for files.

We also compare the searching trustworthiness of Social-P2P with Pure-P2P and EigenTrust [35]. Pure-P2P does not have any mechanisms to guarantee file trustworthiness. Eigen-Trust is a trust management system, in which every peer has a trust manager to calculate its trust value based on others' feedbacks. A node's trust manager is the DHT owner of the node's ID. Each file requester sends the rating of the file supplier to the supplier's trust manager.

TABLE I
Parameter table

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_u	1000s
α, β ,θ, T_u	0.05, 0.1, 3, 100s

Table I lists parameters used in the experiments. The reason why we set α, β ,θ, T_u as the values shown in the table is because these parameters can ensure SocialP2P to achieve a reasonable performances based on the trace data. We generated 300,000 synthetic files according to the popularity distribution shown in Figure 6. The files are randomly distributed to the nodes whose interests match the file contents. Figure 11 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 most queries is ranked first. We see that the querying rate of nodes follows the power-law distribution. Thus, we used a power-law distribution generator with scaling exponent parameter k=-1.2 to generate querying rate within the range of [0.01,100] message/s, and randomly assigned the rate to each node in the system. Since a node is more likely to query files in its interests [14], for each 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 in the system again and t_w is randomly selected from [1-10]s.

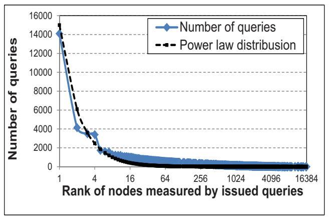


Fig. 11. Node querying rate.

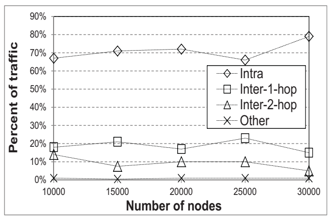


Fig. 12. Traffic distribution.

We use the following metrics: (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 hops in file searching 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.

A. Performance Evaluation on PlanetSim

We set the number of nodes to 32344, which is the number of nodes in the Facebook trace. The duration of each experiment was 50,000 simulation seconds. In each simulation second, every node has one chance to send query message. All experiments were conducted 10 times, and the average values of the results are reported. In the figures of the experimental results below, "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.

1) Evaluation of File Sharing Efficiency:

a) Traffic distribution: Figure 12 shows traffic distribution of the queries in Social-P2P versus network size. In the figure, "Intra" denotes the percent of traffic in intra-clustering searching. "Inter-1-hop" and "Inter-2-hop" denote the percents 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 means 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 13 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. Because a requester sometimes cannot find a friend in the destination cluster, it then 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 a slight querying traffic.

b) Query delay: Figure 14 shows the query delay versus network size for queried files with three different popularities. Popularity of a file is reflected by the percentage of the nodes in the system holding the file. From the figures, we see that as the popularity of the queried files decreases, the delay in all tested systems increases. Higher popularity files have more copies in the system, hence the probability that the files can be retrieved from its neighbors is higher, which results in a lower query delay. The figure also indicates that query delay in PIS does not increase significantly when file popularity changes. This is 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 querying node decreases, thus the query delay increases.

The figure also shows that the query delay in PIS and PROSA increase significantly with network size while the query delay in Social-P2P increases marginally. In PIS, the nodes inferred their interests from files in their current folders and only major 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, and 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 interaction range of nodes. Thus, the inaccurate clustering in PROSA leads to longer query delay.

Figure 14 (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 $O(\log n)$ DHT querying 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 delay since it uses a stable DHT to

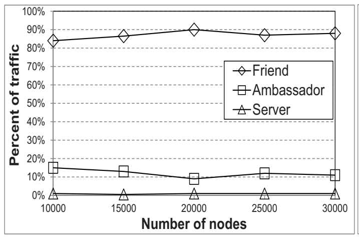
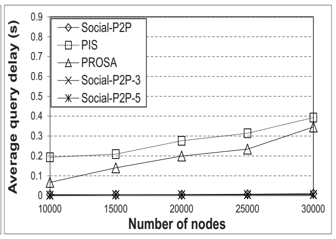
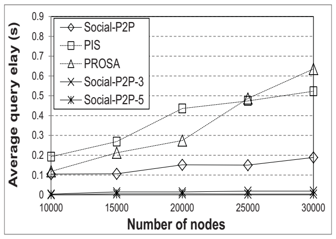


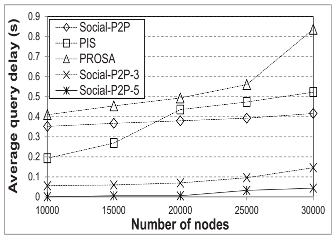
Fig. 13. Inter-cluster traffic distribution.



(a) popularity of queried files: 50%

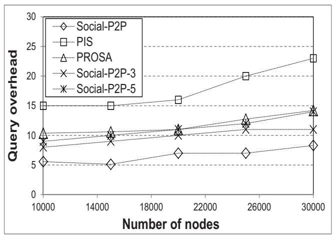


(b) popularity of queried files: 5%

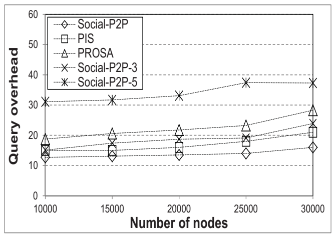


(c) popularity of queried files: 0.5%

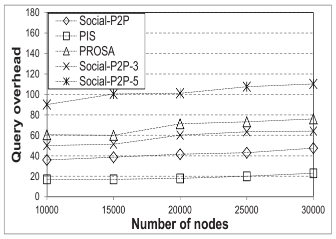
Fig. 14. Delay vs. network size.



(a) popularity of queried files: 50%



(b) popularity of queried files: 5%



(c) popularity of queried files: 0.5%

Fig. 15. Query overhead vs. network size.

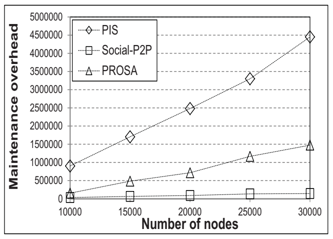


Fig. 16. System maintenance overhead.

locate a destination cluster. The cluster localization in PROSA is based on random walk, which needs more query time.

In Figure 14 (b), we see that for queried files with median popularity, the query delay of PROSA increases rapidly and exceeds DIS when the network size is 30,000. The reason is that the lower popularity of the file leads to a longer time for node clustering and intra-cluster search in PROSA, and large size of the network 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 14 (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. At 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 14 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 rate in Social-P2P. Therefore, for unpopular files in the system, Social-P2P can reduce the query delay by sending more query copies.

c) Query overhead: Figure 15 shows the query overhead versus 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 15 (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 that is 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 Figure 15 (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 does not change quickly with popularity 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 14. The experimental results in Figure 15 verify the low overhead of Social-P2P in file querying.

d) Maintenance overhead: Figure 16 shows the system maintenance overhead 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. Although 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, it only produces a slight maintenance overhead. Although PROSA is also an unstructured P2P network, since each node must maintain several interest clusters, it generates higher maintenance overhead than Social-P2P and its maintenance overhead increases rapidly as the network size grows.

e) Overall system overhead: Figure 17 shows the overall system overhead 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,

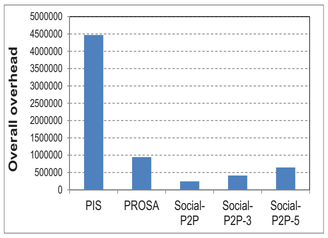


Fig. 17. System overall overhead.

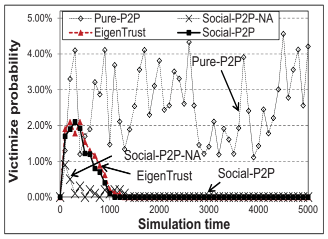


Fig. 18. Victimized probability with malicious nodes.

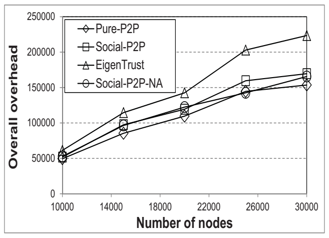


Fig. 19. System overall overhead.

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. PROSA consumes a high overhead for cluster formation and multi-cluster maintenance. Therefore, PROSA has the second highest overall system overhead.

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) 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 by "Social-P2P-NA". (2) Malicious nodes reply to every query with a faulty file. This mechanism is denoted by "Social-P2P". In order to make the methods comparable, all systems use the common-multi-interest clustering protocol for file sharing. In the experiments, 500 nodes out of the 32,344 nodes were randomly selected to act as malicious nodes. A node that has received a faulty file will not forward the file to other nodes.

a) Performance under malicious nodes: Figure 18 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. As EigenTrust decreases the reputation of the malicious nodes, other nodes no longer query files from malicious nodes. Therefore, the victimized probability of EigenTrust decreases. In Social-P2P-NA, the probability of the nodes receiving faulty files is the lowest initially, because 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 the 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.

b) Overall overhead: Figure 19 compares the overall overhead of the four systems. 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. 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 lead 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 results verify the advantages of dealing with efficient and trustworthy file sharing simultaneously, and the low overhead of link trust weight adjustment.

V. Related Works

Efficient file sharing. The most related works to Social-P2P that enhance file sharing efficiency are social network based searching methods in P2P systems [13]–[15], [17], [34]. These methods can be classified into two categories: unstructured networks and DHTs. In the unstructured network based search, Carchiolo *et al.* [34] and Lei *et al.* [13] proposed to gradually cluster nodes into the same group if they query or reply for the same resources. Fast *et al.* [14] proposed to extract user preferences from their music libraries and cluster the users based on user interests. Although these methods can improve searching efficiency, as the nodes with the same interests can be grouped only after they have interactions, the clustering process may take a long time. In the category of DHTs, Cyber [15] builds a DHT-based index on the keywords of items. When a node queries for an item, the items that match the interests of the community the requester belongs to will be returned. Zhang *et al.* [17] 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 data difficult to be located. The current DHT-based social network enables fast node clustering but suffers from high system maintenance overhead in churn. Meanwhile, a node maintain multiple single interest-based node clustering requires each node to maintain several clusters, which leads to high cluster maintenance overhead.

Trust management. Reputation systems [18]–[21] enable peers to rate their service providers after receiving the service and use the accumulated rating of a provider 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. Marti *et al.* [16] 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. Kalofonos *et al.* [36] proposed a platform for secure P2P personal and social networking services. They focused on accesses control rather than file trustworthiness.

Tribler [37] exploit social phenomena as a set of extensions of BitTorrent. However, they simply group all friends of a user for file sharing. SOS further explore the interests pattern and social closeness among friends for more efficient and secure P2P file sharing service.

VI. Conclusion

In this paper, driven by the observations from the trace data from Facebook and Bittorrent, we 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, a trust-based random walk is used to forward a query message along trustworthy links, enhancing file searching efficiency and trustworthiness. Comparably 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.

Acknowledgements

This research was supported in part by U.S. NSF grants CNS-1249603, OCI-1064230, CNS-1049947, CNS-1156875, CNS-0917056 and CNS-1057530, CNS-1025652, CNS-0938189, CSR-2008826, CSR-2008827, Microsoft Research Faculty Fellowship 8300751, and U.S. Department of Energy's Oak Ridge National Laboratory including the Extreme Scale Systems Center located at ORNL and DoD 4000111689.

References

[1] Bittorrent. http://en.wikipedia.org/wiki/Bittorrent.
[2] P2p statistics. http://www.freemusictodownload.eu.
[3] The risks of p2p. http://www.ifpi.org/.
[4] Bpi study quantfies the risk of p2p file sharing. http://www.peer2peerterminator.com/.
[5] I. Stoica, R. Morris, and et al. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *TON*, 2003.
[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. 2001.
[7] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware*, 2001.
[8] Y. Liu, X. Liu, L. Xiao, L.M. Ni, and X. Zhang. Location-aware topology matching in P2P systems. In *Proc. of Infocom*, 2004.
[9] Y. Liu, L. Xiao, and L. M. Ni. Building a scalable bipartite P2P overlay network. *TPDS*, 2007.
[10] H. Xie and Y. R. Yang. P4P: Provider portal for applications. In *Proc. of ACM Sigcomm*, 2008.
[11] 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.
[12] A. Iamnitchi, M. Ripeanu, and I. Foster. Small-world file-sharing communities. In *Proc. of Infocom*, 2004.
[13] J. Lei and X. Fu. Interest-based peer-to-peer group management. *Lecture notes in computer science*, 2009.
[14] A. Fast, D. Jensen, and B. N. Levine. Creating social networks to improve peer to peer networking. In *Proc. of KDD*, 2005.
[15] Y. Li, L. Shou, and K. L. Tan. Cyber: A community-based search engine. In *Proc. of P2P*, 2008.
[16] S. Marti, P. Ganesan, and H. G. Molina. DHT routing using social links. In *Proc. of IPTPS*, 2004.
[17] R. Zhang and Y. C. Hu. Assisted peer-to-peer search with partial indexing. *TPDS*, 2007.
[18] R. Zhou and K. Hwang. PowerTrust: A robust and scalable reputation system for trusted p2p computing. *TPDS*. 2007.
[19] R. Zhou, K. Huang, and M. Cai. GossipTrust for fast reputation aggregation in peer-to-peer networks. *TKDE*, 2008.
[20] S. Song, K. Hwang, R. Zhou, and K. Y. Kwok. Trusted p2p transactions with fuzzy reputation aggregation. *Internet Computing*, 2005.
[21] A. Selcuk, E. Uzun, and M. R. Pariente. A reputation-based trust management system for p2p networks. *IJNS*, 2008.
[22] Americans spend 23 percent of internet time on social networks. http://mashable.com/2011/09/12/23-percent-online/.
[23] Bittorrent user activity traces. http://www.cs.brown.edu.
[24] R. Xiang, J. Neville, and M. Rogati. Modeling relationship strength in online social networks. In *Proc. of WWW*, 2010.
[25] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 1975.
[26] D. Bayer and M. Stillman. Computation of Hilbert functions. *JSC*, 1992.
[27] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *JRSS.*, 1979.
[28] P. Godfrey, S. Shenker, and I. Stoica. Minimizing churn in distributed systems. In *Proc. of Sigcomm*, 2006.
[29] 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.
[30] C. Binzel and D. Fehr. How Social Distance Affects Trust and Cooperation: Experimental Evidence from A Slum. In *Proc. of ERF*, 2009.
[31] B.H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 1970.
[32] N. Ferguson, B. Schneier, and T. Kohno. *Cryptography Engineering: Design Principles and Practical Applications.* 2010.
[33] Planetsim simulator. http://projects-deim.urv.cat/trac/planetsim/.
[34] V. Carchiolo, M. Malgeri, G. Mangioni, and V. Nicosia. An adaptive overlay network inspired by social behavior. *JPDC*, 2010.
[35] S. Kamvar, M. Schlosser, and H. G. Molina. The EigenTrust algorithm for reputation management in p2p networks. In *Proc. of WWW*, 2003.
[36] D. Kalofonos, Z. Antonious, F. Reynolds, M. Kleek, J. Strauss, and P. Wisner. Mynet: a platform for secure p2p personal and social networking services. In *Proc. of Percom*, 2008.
[37] A. Pouwelse and P. Garbacki. Tribler: a social-based p2p system. *Concurrency and Computation: Practice and Expe.*, 2008.