A Social Network Integrated Reputation System for Cooperative P2P File Sharing

Kang Chen, Haiying Shen, Karan Sapra and Guoxin Liu
Department of Electrical and Computer Engineering
Clemson University
Clemson, SC, USA 29631
Email: {kangc, shenh, ksapra, guoxinl}@clemson.edu

***Abstract*—In current reputation systems for peer-to-peer (P2P) file sharing networks, reputation querying generates high overhead in file service. Also, a high-reputed node may be reluctant to further increase its reputation in these reputation systems. Recently, social network-based P2P systems that favor file sharing among friends have been proposed for reliable file sharing. However, this approach contradicts the objective of widely file sharing in P2P systems. To overcome these drawbacks, we propose a social network integrated reputation system, namely SocialTrust, that synergistically leverages the concept of "friendship fosters cooperation". In SocialTrust, each node maintains trusted relationships for the purpose of file sharing, which are resulted from both real life acquaintance and online partnership established between high-reputed and frequently-interacted nodes. Given a number of server options, a client first chooses a friend or a partner, if available, without querying their reputations. Otherwise, it chooses the server with the highest reputation. The benefits of friendship and partnership on file sharing and cost saving encourage nodes to be continuously cooperative. Further, in order to realize accurate reputation evaluation and strong cooperation incentives, SocialTrust considers the number of friends/partners and reputation of a node in the process of reputation rewarding and punishment. Extensive trace-driven simulation demonstrates the effectiveness of SocialTrust.**

I. Introduction

Due to the open nature of the peer-to-peer (P2P) environment, P2P file sharing systems are prone to have selfish and misbehaving nodes. Selfish nodes are not cooperative in providing files, but still would like other nodes to comply to their requests [1], [2]. Misbehaving nodes can distribute tampered files, corrupted files or files with malicious code into the system, which could be further spread by unsuspecting users. For example, 85% of Gnutella users are selfish nodes sharing no files, and 45% of files downloaded through the Kazaa file sharing application contained malicious code. Therefore, incentives are needed to encourage cooperation in P2P networks. Reputation system, as a cooperation incentive method, has been widely studied in recent years [3]–[7]. In a reputation system, a node's reputation is built based on a collection of feedbacks from other nodes. A pre-defined reputation threshold is used to classify nodes to reputed or selfish nodes. However, a clever node can sustain in the system by maintaining its reputation just above the threshold and take this advantage for uncooperative behaviors. Further, frequent reputation querying can easily overload the reputation center, leading to degraded service quality in P2P systems.

Recently, emerging P2P file sharing systems have been proposed to incorporate online social networks (OSNs) to enhance service cooperation [8]–[11] or malicious node detection [12] by leveraging the social property of "friendship fosters cooperation" [13]. Naturally, such an idea can alleviate the necessity of reputation querying and reduce the load on reputation centers. However, the friendship network of a node usually only consists of a very small part of the entire P2P network, which means that a client may not be able to find a server among its friends. Thus, these OSN-based approaches, if imported into P2P file sharing systems, limit the objective of widely sharing of individuals' files.

In this paper, we propose a credit based reputation system, namely SocialTrust, that synergistically integrates the traditional reputation systems and social networks to overcome their individual shortcomings with three main components:
(1) *Social networks (friend network and partner network).* Each node maintains two lists of trusted nodes, friend-list containing friends from real life (i.e., real-world acquaintance) and partner-list containing frequently-interacted high-reputed nodes (i.e., online acquaintance). They both encourage nodes to be cooperative. Firstly, nodes do not want to damage friendship easily but instead desire to maintain their real-life reputations. Secondly, nodes would like to have more partners in order to receive more requests and gain more benefits.
(2) *Lightweight reliable server selection.* Given a number of server options, a client chooses a friend or partner, if available, without querying their reputations. This reduces the reputation querying cost and service delay, and meanwhile still supports the objective of open and free P2P services. Such overhead saving encourages nodes to be continuously cooperative in order to keep their friends and partners.
(3) *Reputation evaluation for cooperative file serving and honest rating.* We define a node's social degree as the number of its friends and partners. As the social degree of a node represents the degree of trust from others [14], SocialTrust considers both a node's social degree and its reputation as a measurement of its trust and uses it to adjust node reputation reward or punishment in a transaction. It gives lower weight to ratings from lower-trust nodes and vice versa. SocialTrust assigns reputation punishment proportional to a server's trust. Therefore, a node's reputation is built gradually, but drops in proportional to its trust, which helps to prevent nodes with

a high trust from gaining unfair advantages [15]. Further, a node's final reputation value is calculated based on both its file serving and reputation rating behaviors, which encourages nodes to be cooperative in both behaviors.

As a result, SocialTrust has below advantages compared to previous reputation systems [3]–[7].

(1) By integrating the social networks into the reputation system, SocialTrust saves the reputation querying cost greatly without constraining server options, while the large amount of reputation queries in previous methods may easily overload the reputation center.
(2) SocialTrust considers both a node's social degree and reputation to measure its trust, which is used to determine the weight of its reputation rating for others and the degree of punishment on its misbehavior. Thus, node reputation is more accurately evaluated and even high-reputed nodes are motivated to be constantly cooperative, thereby preventing high-reputed nodes from taking advantage of its reputation for misbehavior.
(3) SocialTrust considers both service providing and rating in evaluating a node's reputation, which encourage nodes to become cooperative in both behavior.

The remainder of this paper is arranged as follows. Section II presents the related work. Section III introduces the design of SocialTrust. In Section IV, the performance of SocialTrust is evaluated through real-trace based simulation. Section V concludes this paper with remarks on future work.

II. Related Work

In recent years, numerous research works have been conducted on reputation systems [3]–[7] in P2P networks. These works focus on how to aggregate reputation ratings and calculate the reputation efficiently and accurately. Though these systems are effective, they fail to utilize the social network properties in P2P file sharing to reduce reputation querying cost and encourage continuous cooperation, as mentioned in the introduction. Zhou and Hwang [3] observed a power-law distribution in user feedbacks in eBay and proposed the PowerTrust reputation system that selects few most reputable nodes to aggregate reputation feedbacks in order to improve the global reputation accuracy and aggregation speed. Zhang *et. al* [4] presented a reputation system built upon the multivariate Bayesian inference theory. It offers a theoretical basis for clients to predict the reliability of candidate servers based on self-experiences and feedbacks from peers. GossipTrust [5] uses randomized gossiping and power nodes to enable fast aggregation and fast dissemination of global reputation. PeerTrust [6] includes a coherent adaptive trust model for quantifying and comparing the trust of peers based on a transaction-based feedback system, which combines multiple parameters such as feedback a peer receives from other peers and the total number of transactions a peer performs. EigenTrust [7] computes a global reputation value for a peer by calculating the left principal eigenvector of a matrix of normalized local reputation values, thus taking into consideration the entire system's history with each peer and increasing the accuracy of global reputation.

There are also a number of works that leverage OSNs for reliable services in P2P networks [8]–[12], [16] based on the property of "friendship foster cooperation" [13]. Since a user's friends are usually trustworthy and share similar interest with it, Chen *et al.* [16] exploited the friend relationships to perform reputation estimation, i.e., a node selects a file based on its friends' evaluation on the file. TRIBLER [8] is a social-based P2P file sharing system, which enables fast, trusted content discovery and recommendation by allowing nodes to retrieve files from taste groups, friends and friends-of-friends. Turtle [9] builds its overlay on top of pre-existent trust relationships among its users in order to withstand most of the denial of service attacks and allow both data sender and receiver anonymity. MyNet [10] is a P2P middleware platform and user interaction tool that allows everyday users to easily and securely access and share their devices, services, and content in real time. In the F2F system [11], a node chooses its neighbors (the nodes with which it shares resources) based on existing social relationships. This approach provides incentives for nodes to cooperate. SybilGuard [12] exploits the property that social connections usually represent trust relationships to detect sybil nodes. It is based on the property that sybil nodes usually have disproportionately small amount of connections with honest nodes. In these algorithms, though the social network is exploited to enhance service quality, they limit the operation (i.e., file sharing and data service) only among friends. Such a limitation contradicts with the goal of widely file sharing in the P2P file sharing systems.

III. The Design of SocialTrust

SocialTrust is a credit based reputation system for P2P file sharing systems. Below, we present the three main components of SocialTrust in detail.

A. Social Networks

In a general OSN, a user's friends include offline friends with certain social connections (e.g., friends, classmates, colleagues, etc.) in real life and online friends. Similarly, the social relationships of a node in SocialTrust include both offline friends and trustable online partners. Both friends and partners in SocialTrust represent certain trust. Since friends are connected by certain social relationships in a social community, they would offer high QoS to each other with the intention of building high real-life reputations. Thus, the friendship network motivates nodes to be cooperative continuously. A node's partners also have high probabilities to offer high QoS to the node according to their previous interaction (collaboration) records.

1) Friendship Maintenance: The friendship update (i.e., addition and deletion) in SocialTrust is user-dependent behavior, and users are responsible for the consequence of adding a new friend. In other words, rational users would be cautious in accepting friend invitations. Consequently, the friendship network can reflect trustable and stable social relationship in real lives. Since the friendship is user centric, each user

maintains its own friend-list. When a node, say N_i, wants to add another node, say N_j, into its friend-list, it sends a friend invitation to N_j. If N_j accepts the invitation, they become friends of each other. If a user deletes a friend, they remove each other from their friend-lists.

Recent research shows that users in online social network (OSNs) (i.e., Facebook) can be tricked into accepting friend requests from malicious parties [17]. We claim that this does not affect the practicality of SocialTrust. Firstly, the friendship establish requirement in SocialTrust is different from that in common OSNs in which a user would like to add friends with no constraint. Secondly, SocialTrust is designed for P2P file sharing system, where node interactions (i.e., file exchange) can detect malicious nodes more easily. These nodes are them removed from friend list. However, as we will see later, a node would not delete friends arbitrarily since when it has more friends, it can save more querying overhead.

2) Partnership Maintenance: A bi-directional partnership is established when both below conditions are satisfied.

(1) The interaction frequency between the two nodes is larger than a threshold, denoted by T_f.
(2) Each node's reputation value is larger than the partnership threshold, denoted by T_r.

The above two requirements match how people build partners in their businesses, i.e., entities with good and frequent past transaction records are more reliable.

The interaction frequency of a node, say N_i, with another node, say N_j, is calculated by

$$F(N_i, N_j) = \gamma F_{old}(N_i, N_j) + (1-\gamma)F_{new}(N_i, N_j)$$

where F_{new} and F_{old} denote the frequency measured in the current and previous period, respectively, and $\gamma \in [0, 1]$ is an adjusting factor. The update period can be decided as the average time period between two interactions among nodes in the system multiplied by a factor. In order to avoid periodical querying of partners' reputation values and ensure timely update of partnerships, the partnership of each node is maintained on the reputation system. After each reputation update, the reputation system notifies each node the change of partnerships, if necessary. Similar to the friendship, a node would not delete partners arbitrarily in order to save reputation querying cost and receive reliable services.

3) Further Discussion: A friend or a partner may also be uncooperative. Therefore, when a node receives uncooperative service from a friend or partner, it may remove the friend or partner based on aforementioned criteria. This strategy, in turn, provides cooperation incentives to friends and partners since fewer friends and partners would lead to fewer opportunities to provide service and earn reputation. A node would be regarded as a selfish node if its reputation drops below a pre-defined threshold. Then, its services will be rejected by other nodes in the system. The mechanism of friendship and partnership network also motivates nodes to keep increasing their reputations to establish more friendships/partnerships (i.e., making its reputation value pass the threshold of more nodes), which helps to save more reputation querying cost, as explained in the next section.

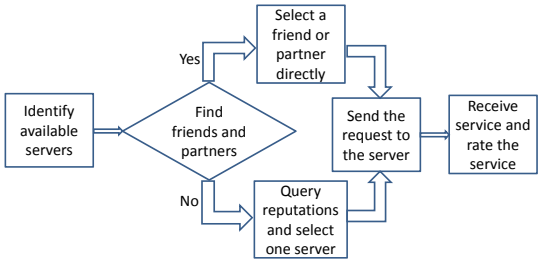


Fig. 1: The process of a client's server selection.

B. Lightweight Reliable Server Selection

Since the friendship and partnership represent trust, we exploit this property to alleviate the reputation querying cost. That is, a node directly selects friends or partners when available for service without querying their reputation values. Each node also maintains local ranks for friends/partners based on previous service records to handle the case when when several friends or partners appear in the available server list.

Figure 1 shows the process of a client's operation in server selection. When a client needs a service, it first identifies the available servers (i.e., server candidates) for the requested service. For example, a node uses the P2P file lookup function to identify all file owners for its requested file. The client node then checks whether any of its friend(s) or partner(s) are in the list. If yes, it skips the step of querying the available servers' reputations and selects the friend or partner with the highest local ranking as the server directly. If there is no friend or partner in the server list, it queries the reputation value of each server candidate from the reputation system, and chooses the one with the highest reputation. After the transaction is completed, the client sends the service rating to the reputation system. If the service is from a friend or partner, the node also records the rating for the purpose of local ranking.

We then deduce the percentage of reputation queries (P_{sc}) that can be avoided in SocialTrust. We assume that the servers that can satisfy a request are evenly distributed among all nodes. We use M to denote the total number of nodes in the system. Let m_i be the number of service requests generated by node i, N_s be the average number of available servers for a request, and n_{fp} be the number of friends and partners node N_i has. Then, the probability that none of the available servers is a friend or partner (P_ϕ) can be calculated by $P_\phi = (1 - n_{fp}/N)^{n_s}$. Then,

$$P_{sc} = \sum_{i=1}^{M}((1-(1-n_{fp}/M)^{N_s})N_s m_i)/\sum_{i=1}^{M} N_s m_i \quad (1)$$

Note that a node is more likely to find requested services from its friends or partners. This means actual P_{sc} should be larger.

As shown in Equation (1), the more friends/partners a node has, the more reputation queries (i.e., cost) it can avoid. As stated in Section III-A, the friendship is usually stable while the partnerships are built dynamically. If a node has a higher reputation, it can pass the reputation thresholds of more nodes and be accepted as the partner of more nodes. As a result, a

node would not maintain its reputation value barely above the reputation threshold but would like to accumulate its reputation as high as possible to save more cost on reputation querying.

C. Reputation Evaluation

The reputation system updates node reputations based on received reputation feedbacks, which determines the accuracy and the effectiveness of the incentive system in encouraging cooperation and discouraging non-cooperation.

In order to deter reputed nodes from conducting occasional misbehavior while still being regarded as reputed, we follow the principle that reputation is built gradually, but drops in proportional to its ability to bring about harm when a node misbehaves [15]. The probability that a misbehaving node is chosen as a server determines its potential harm to the system performance. Recall a node either selects its friend/partner or the available server with the highest reputation as the server for its request. As a result, the probability is determined by the node's reputation and social degree, which is the sum of the degrees in the friendship and partnership network. Therefore, we propose a metric called *Trust* (denoted by $T \in [0, 1]$) that integrates both reputation and social degree:

$$T(i) = \beta \frac{R(N_i)}{R_{max}} + (1 - \beta) \frac{D(N_i)}{D_{max}}, \tag{2}$$

where $R(N_i)$ is the reputation of node N_i, R_{max} is the maximal reputation value allowed in the system, $D(N_i)$ is the social degree of N_i, D_{max} is the maximal number of friends and partners a node can have in the system, and β is an adjusting factor. Then, the trust of a node represents its harm when it is non-cooperation and the credibility of its service rating when it is cooperative.

Unlike the previous reputation systems that only consider the QoS of servers and assume that clients always provide honest ratings, SocialTrust considers both file serving and service rating of a node to determine its reputation, which motivates a node to be cooperative in both actions. We name the file sharing process between two nodes as an interaction between them. Each interaction is a game, in which each node selects a strategy for the sharing of a file: cooperative and non-cooperative. We discuss the reputation evaluation in different scenarios below and use T_c and T_s to represent the trust of the client node and the server node, respectively.

1) Cooperative Server and Cooperative Client: In this case, the server provides requested service to the client, which then rates the quality of the received service honestly. Then, the rating is always larger than 0: $Y_c \in (0, 10]$. The reputation system decides whether the rating is honest by asking the server whether it accepts the rating. If the server accepts the rating, the reputation credits given to the server node ($R_{s1} \in [0, 1]$) and the client node ($R_{c1} \in [0, 1]$) are

$$\begin{cases} R_{s1} = (1 + T_c * Y_c/10) * \alpha \\ R_{c1} = \alpha \end{cases} \tag{3}$$

In Equation (3), $\alpha \in [0, 0.05]$ is the unit reputation credit for file serving and service rating. Thus, the amount of reputation credits earned by the server depends on the rating and the trust of the client. The rationale behind such a design is that the higher trust the client has, the more trustable its rating is, and more reward should be assigned to the server. Equation (3) also shows that a server prefers to serve clients with high trust in order to earn more reputation, which motivates nodes to always maintain high trust.

2) Cooperative Server and Non-cooperative Client: In this situation, the server is cooperative in providing service but the client gives a bad feedback or does not provide any feedbacks. Then the server refuses to accept the rating (or no rating) given by the client and files a claim. When the non-cooperative behavior of the client is confirmed, the reputation credits assigned to the server ($R_{s2} \in [0, 1]$) and the client ($R_{c2} \in [-1, 0]$) are as below. We explain how the reputation system investigates whether the client rates honestly later.

$$\begin{cases} R_{s2} = \alpha \\ R_{c2} = -(1 + T_c) * \alpha \end{cases} \tag{4}$$

We only assign the unit credit to the server since the rating provided by the client is dishonest and thus cannot reflect the QoS of the service provided by the server. The non-cooperative client is punished based on its trust: the higher trust it has, the more punishment it receives. Such a design reduces the reputation of nodes with higher trust more quickly when they are non-cooperative, preventing them from taking advantage of the high trusts for non-cooperative behavior. Comparing Equation (4) with Equation (3), we see that the server earns $T_c * \alpha$ less reputation credits, which is the cost of choosing a dishonest client. Thus, servers are further motivated to provide service to high-reputed nodes, which in turn, motivates nodes to be cooperative in serving and rating.

3) Non-cooperative Server and Cooperative Client: In this situation, the server provides false or malicious service and the client node rates the service honestly (i.e., giving bad feedback). Therefore, the rating is less than 0: $Y_c \in [-10, 0)$. Although the server can reject the bad feedback, the reputation system can check the QoS correctly using the method introduced later. Then, the reputation credits assigned to the server ($R_{s3} \in [-1, 0]$) and the client node ($R_{c2} \in [0, 1]$) are

$$\begin{cases} R_{s3} = -(1 + T_c * |Y_c|/10) * (1 + T_s) * \alpha \\ R_{c3} = \alpha \end{cases} \tag{5}$$

Above design follows the rationale that a non-cooperative server with high trust receives more punishment to prevent it from exploiting high reputation for misbehavior. An unit credit (α) is assigned to the client to encourage honest rating.

4) Non-cooperative Server and Non-cooperative Client: This situation occurs in collusion, where the server provides service with low quality or does not provide service at all, but the client still gives good feedback in order to boost the server's reputation. We first assume that collusion is not detected. In this case, the reputation credits assigned to the server node ($R_{s4} \in [0, 1]$) and the client ($R_{c4} \in [0, 1]$) are the same as those in Equation (3). But in reality, collusion generally is detected with certain probability under different detection algorithms [18]. When the collusion is detected,

involved nodes are punished. Then the actual credits assigned to collusion nodes can be expressed as

$$\begin{cases} R_{s4} = D_d * R_{s1} = D_d * (1 + T_c * Y_c/10) * \alpha \\ R_{c4} = D_d * R_{c1} = D_d * \alpha \end{cases} \quad (6)$$

where $D_d \in [-1, 1)$ is the adjusting factor considering the possible punishment for detected collusion behaviors. It is determined based on the probability of successful detection and the amount of punishment for collusion. We leave how to detect non-cooperative behaviors in the system as future work.

IV. Performance Evaluation

We used the trace from LiveJournal [19], which contains over 4 million users, to construct the friendship links in social network. We randomly selected a medium social network with 10,500 users from the trace for our test. Among these nodes, we randomly selected 520 nodes as non-cooperative nodes. To simulate different node willingness on cooperation, we assigned each node (both cooperative and non-cooperative) a level (L) randomly in range [1, 10]. Then, the probability of a cooperative node to behave cooperatively (P_g) and the probability for a non-cooperative node to behave non-cooperatively (P_b) are defined as $P = 0.55 + 0.05 * (L - 1)$. Such a setting means that the high level a node is, the more likely it behaves cooperatively or non-cooperatively.

The reputation value of a node lies in the range of [-1,1]. Initially, all nodes are regarded as cooperative nodes by setting their initial reputation value in range [0,1]. Nodes with negative reputation values are identified as non-cooperative nodes, and their services requests are always rejected by others. The friendship connections in the social network do not change. The partner list of each node is empty initially and is gradually built or deleted during experiments. The reputation threshold of partnership (T_r) of a node was randomly chosen from a medium range of [0.3, 0.8]. Based on the average interaction frequency in the trace, the contact frequency threshold (T_f) was set to 1 transaction every 50 rounds. Also, based on analysis of the trace, the reward unit (α) was set to 0.04.

In the test, each experiment consists of 200 rounds. In each round, we reasonably assume each node has 60% probability to generate a request. We assume that there are m nodes containing the requested file for each request. Considering the total number of nodes is not huge, m is randomly selected from [3, 6]. We let the file quality fall in range [-10,10]. For the server, the QoS of its service is randomly selected from (0,10] and [-10, 0) when it is cooperative and non-cooperative, respectively. If the client is cooperative, it honestly gives rating $QoS+r$, where r represents the rating bias and is chosen from [-1,1] according to the client node's trust. The final rating value is in the range [-10, 10]. If the client is non-cooperative, it gives rating randomly selected from [-10,0) for cooperative service and (0,10] for non-cooperative service.

We compared SocialTrust with an OSN-based method [16] (denoted by OSNTrust) and EigenTrust [7]. OSNTrust provides a social reputation model to guide users to browse desirable content using the co-relation between different users by considering the trust and similarity of interest between friends. We modified it to adapt to a general reputation system model, in which each node queries the reputation values of available servers from its friends. If none of a node's friends knows the reputation of available servers, the node randomly selects a server. In OSNTrust, a node considers a server as non-cooperative when at least four of its friends report negative reputation value for the server. EigenTrust [3] calculates the reputation of a node based on the feedbacks from other nodes weighted by the rater's reputation as opposed to our method that weights the feedback based on trust. We set the confidence interval to 95% in the experiment part.

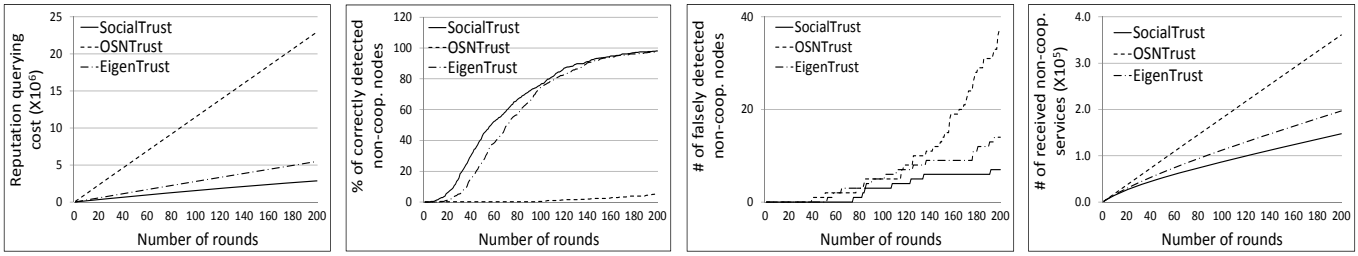
A. Efficiency of Reputation Systems

Figure 2(a) shows the reputation querying cost, measured by the total number of generated reputation queries, in the three reputation systems. We see that the results follow OSNTrust>EigenTrust>SocialTrust, and OSNTrust generates much higher cost than SocialTrust and EigenTrust. For OSNTrust, each node needs to query all its friends for each available server, leading to a large amount of reputation queries. In SocialTrust, friendship and partnership are exploited to alleviate the necessity of querying the reputation system for the reputation of available servers, while in EigenTrust, the service requester needs to query the reputation of all available servers. Therefore, SocialTrust produces less cost than EigenTrust. This result justifies that SocialTrust realizes the goal of reducing reputation querying cost by leveraging friendship and partnership. Though the cost saving is resulted from selecting friends/partners directly, we can see later that such a design does not sacrifice the effectiveness of SocialTrust in guiding trustworthy server selection.

B. Effectiveness of Reputation Systems

Figure 2(b) illustrates the percentage of correctly detected non-cooperative nodes throughout the test. We find that the results follow SocialTrust>EigenTrust>>OSNTrust in the first 180 rounds. Also, both SocialTrust and EigenTrust can detect almost all non-cooperative nodes in the end while OSNTrust can only identify a small portion of non-cooperative nodes. SocialTrust decides the punishment for a non-cooperative node based on its trust. Thus, it can decrease the reputation of non-cooperative nodes with high trust quickly. Moreover, SocialTrust adjusts the rating based on the reliability (i.e., trust) of the rater. As a result, its reputation evaluation can more accurately reflect the behavior of nodes, leading to better detection of non-cooperative nodes. EigenTrust only considers the rater's reputation for the credibility of its reputation feedback. Therefore, the reputation evaluation in EigenTrust is not as accurate as that in SocialTrust, especially for high reputation nodes, leading to a slow detection. For OSNTrust, each node usually has limited interaction records and thereby a node's friends can provide very limited and updated reputation information of available servers. As a result, OSNTrust can only detect a small amount of non-cooperative nodes.

Figure 2(c) presents the number of falsely identified non-cooperative nodes. We see that the results follow OSNTrust



(a) Trust querying cost. (b) % of correct detections. (c) # of false detections. (d) # of received non-cooperative services.

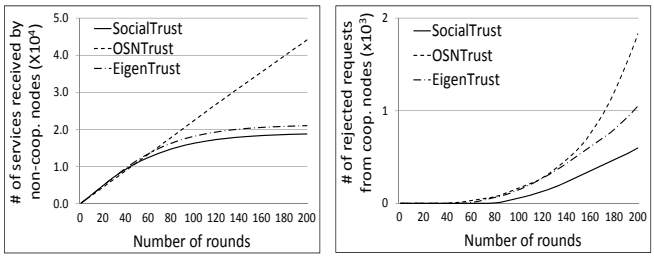
Fig. 2: Efficiency and effectiveness of the three reputation systems.

>EigenTrust>SocialTrust. OSNTrust has the most false detections due to its local reputation calculation with limited interaction records, which fails to provide the true trustworthiness of nodes. For SocialTrust, it generates less false detections than EigenTrust since it adjusts the reputation punishment by trust. Therefore, when a node has low trust, the punishment for its non-cooperative behavior is relatively low, which prevents it from falling into non-cooperation category occasionally. Also, the consideration of the rater's trust leads to more accurate reputation evaluation in SocialTrust.

Figure 2(d) plots the number of non-cooperative services received by all nodes. We see that the result follows OSNTrust >EigenTrust>SocialTrust. This is because the abilities of the three methods to exclude non-cooperative nodes follow OSNTrust>EigenTrust> SocialTrust, as shown in Figure 2(b). The more unidentified non-cooperative nodes in the system, the more non-cooperative services would be provided. This result further verifies the efficiency of SocialTrust in excluding non-cooperative nodes and preventing non-cooperative services in P2P system. The result also shows that the scheme in SocialTrust that selects friends directly when available doesn't compromise the performance since friends are more likely to be cooperative with each other [8]–[11], [13].

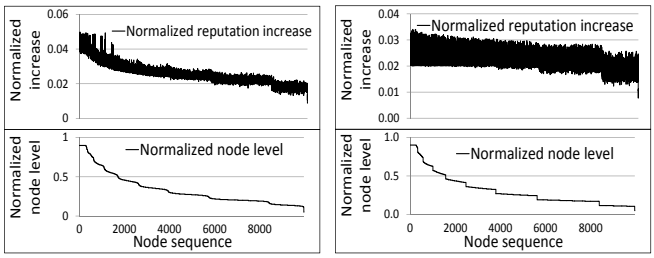
Figure 3(a) shows the number of cooperative services received by non-cooperative nodes. As the number of rounds increases, OSNTrust cannot prevent non-cooperative nodes from receiving services, while EigenTrust and SocialTrust successfully exclude non-cooperative nodes from receiving services. Also, SocialTrust can prevent more services for non-cooperative nodes than EigenTrust. This is because the number of cooperative services provided to non-cooperative nodes is proportional to the number of non-cooperative nodes that are not identified, and as demonstrated in Figure 2(b), SocialTrust has higher ability in excluding non-cooperative nodes than EigenTrust, which has higher ability than OSNTrust.

Figure 3(b) plots the number of rejected service requests from cooperative nodes. We see that the number follows the same relationship as in Figure 2(c) (i.e. OSNTrust> EigenTrust>SocialTrust). This is because the number of rejected service requests from cooperative nodes is proportional to the number of falsely detected non-cooperative nodes. When cooperative nodes are falsely detected as non-cooperative nodes, their subsequent requests are rejected. Therefore, SocialTrust generates the least rejected service requests, and EigenTrust produces fewer rejected service requests than OSNTrust. This further justifies that SocialTrust is more reliable than EigenTrust and OSNTrust on reputation evaluation.



(a) # of services received by non-cooperative nodes. (b) # of rejected requests from cooperative nodes.

Fig. 3: Effectiveness of reputation systems.

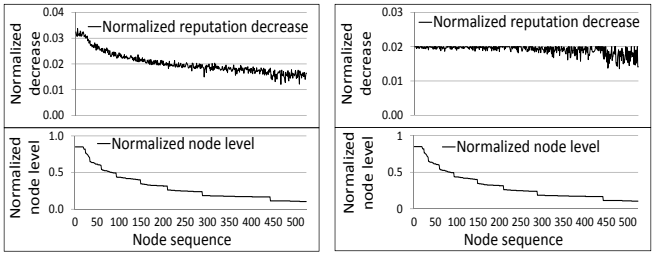


(a) SocialTrust. (b) EigenTrust.

Fig. 4: Accuracy in reputation evaluation of cooperative nodes.

C. Accuracy of Reputation Evaluation

In this experiment, we evaluate the accuracy of the reputation systems in measuring how cooperative or how uncooperative a node is. The degree of a server's QoS and a rater's rating honesty are decided by their trust. If the selected server is cooperative in serving, the QoS of its service is calculated as $r + 10 * T_i/T_{Max}$, where T_i is the node's trust, and T_{Max} is the maximal trust. If the calculated QoS is less than 1 or larger than 10, it is set to 1 or 10, respectively. If the client is cooperative, it gave rating by: $QoS+r*(1-D_i/D_{Max})$, where its social degree D_i controls how large the deviation from the honest rating is, and r determines the deviation direction. If the client is non-cooperative, it's rating is randomly selected from range [-10,-1]. Here, we do not set the rating fluctuation associated with the rater's social degree because we only want to test the effect of punishing nodes based on their trust. If the selected server is non-cooperative, the QoS of its service is calculated as $-(r+T_i/T_{Max})$, which is also confined to the range of [-10, -1]. Then, if the client is cooperative, it gives rating by: $QoS+r*(1-T_i/T_{Max})$. Otherwise, it gives rating randomly in range [1, 10].



(a) SocialTrust. (b) EigenTrust.

Fig. 5: Accuracy in reputation evaluation of non-cooperative nodes.

The normalized reputation increase refers to the average reputation increase per interaction, and it is calculated as the final reputation value increase divided by the total number of interactions. The normalized node level represents how actually cooperative a node is. Recall that the probability of a node to behave cooperatively is decided by its node level, and its QoS or rating honesty is decided by its social degree. Therefore, a node i's normalized node level is defined as $N_L = 0.5 * L/L_{max} + 0.5 * T/T_{max}$, in which L and T are the level and the average trust, while L_{max} and T_{max} are their maximal values, respectively.

Since OSNTrust does not provide global reputation, we only measured that of SocialTrust and EigenTrust. We rank all cooperative nodes by their normalized node levels, and show each node's normalized node level and normalized reputation increase in SocialTrust and EigenTrust in Figure 4(a) and Figure 4(b), respectively. We see that when normalized node level decreases, SocialTrust leads to more similar trend between the normalized reputation increase and the normalized node level. The reason is that SocialTrust considers both a node's social degree and reputation in evaluating a node's trust, rather than only considering reputation as in EigenTrust.

We also tested the relationship between the normalized reputation decrease and normalized node level for non-cooperative nodes. The results are shown in Figure 5(a) and Figure 5(b). The normalized reputation decrease is calculated as the average reputation decrease per interaction. We see that the normalized reputation decrease more closely approximates the normalized node level in SocialTrust than in EigenTrust. This is because SocialTrust utilizes the trust of a node to adjust the punishment for non-cooperative behaviors: high-trust nodes receive more reputation punishment. Above results demonstrate the accuracy of SocialTrust in evaluating the reputation by considering node trust.

V. Conclusions

In this paper, we propose a social network reputation system for P2P networks, namely SocialTrust, which integrates social network properties in order to save reputation querying cost and enhance the reputation evaluation accuracy. Since friends/partners in online social networks usually trust each other, SocialTrust lets each node directly select its friends/partners, if available, as service provider without querying their reputation values, thereby reducing the reputation querying cost. Further, SocialTrust integrates both social degree and reputation value of a node as its trust for reputation clearance. Since nodes with high social degree/reputation are more likely to be selected as servers, they are more harmful when they are non-cooperative. Therefore, the trust is used to adjust the reputation reward and punishment, motivating nodes to be continually cooperative, rather than remaining just above the cooperation threshold. Extensive simulation demonstrates the efficiency and effectiveness of SocialTrust. In the future, we plan to investigate the correlation between the OSN friendship and reputation evaluation to further enhance the accuracy of reputation evaluation.

Acknowledgements

This research was supported in part by U.S. NSF grants CNS-1254006, CNS-1249603, 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] D. Hughes, G. Coulson, and J. Walkerdine, "Free Riding on Gnutella Revisited: The Bell Tolls?" *IEEE Dist. Systems Online*, 2005.
[2] E. Sit and R. Morris, "Security considerations for peer-to-peer distributed hash tables," *Peer-to-Peer Systems*, 2002.
[3] R. Zhou and K. Hwang, "PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing," *IEEE TPDS*, 2007.
[4] Y. Zhang and Y. Fang, "A Fine-Grained Reputation System for Reliable Service Selection in Peer-to-Peer Networks," *IEEE TPDS*, 2007.
[5] R. Zhou, K. Hwang, and M. Cai, "GossipTrust for Fast Reputation Aggregation in Peer-To-Peer Networks," *IEEE TKDE*, 2008.
[6] L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities," *IEEE TKDE*, 2004.
[7] S. Kamvar, M. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," in *Proc. of WWW*, 2003.
[8] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Y. 0015, A. Iosup, D. H. J. Epema, M. J. T. Reinders, M. van Steen, and H. J. Sips, "Tribler: A social-based peer-to-peer system," in *Proc. of IPTPS*, 2006.
[9] B. Popescu, B. Crispo, and A. Tanenbaum, "Safe and Private Data Sharing With Turtle: Friends Team-Up And Beat The System," in *Proc. of SPW*, 2004.
[10] D. N. Kalofonos, Z. Antonious, F. D. Reynolds, M. Van-Kleek, J. Strauss, and P. Wisner, "MyNet: A Platform For Secure P2P Personal And Social Networking Services," in *Proc. of PerCom*, 2008.
[11] J. Li and F. Dabek, "F2F: Reliable Storage in Open Networks," in *Proc. of IPTPS*, 2006.
[12] H. Yu, M. Kaminsky, and A. D. Flaxman, "Sybilguard: Defending against sybil attacks via social networks," in *Proc. of Sigcomm*, 2006.
[13] E. Pennisi, "How did Cooperative Behavior Evolve?" *Science*, 2005.
[14] B. Viswanath, A. Post, K. Gummadi, and A. Mislove, "An Analysis of Social Networkbased Sybil Defenses," in *Proc. of SIGCOMM*, 2010.
[15] M. Srivatsa, L. Xiong, and L. Liu, "TrustGuard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks," in *Proc. of WWW*, 2005.
[16] R. Chen, E. K. Lua, and Z. Cai, "Bring Order to Online Social Networks," in *Proc. of INFOCOM*, 2011.
[17] D. Irani, M. Balduzzi, D. Balzarotti, E. Kirda, and C. Pu, "Reverse social engineering attacks in online social networks." in *DIMVA*, ser. Lecture Notes in Computer Science, 2011.
[18] Z. Li, H. Shen, and K. Sapra, "Leveraging Social Networks to Combat Collusion in Reputation Systems for Peer-to-Peer Networks," in *Proc. of IPDPS*, 2011.
[19] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, "Group Formation in Large Social Networks: Membership, Growth, and Evolution," in *Proc. of KDD*, 2006.