2012 41st International Conference on Parallel Processing

Collusion Detection in Reputation Systems for Peer-to-Peer Networks

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

***Abstract*—In peer-to-peer networks (P2Ps), many autonomous nodes without preexisting trust relationships share resources (e.g., files) between each other. Due to their open environment, P2Ps usually employ reputation systems to provide guidance in selecting trustworthy resource providers for high system reliability and security. A reputation system computes and publishes reputation score for each node based on a collection of opinions from others about the node. However, collusion behaviors impair the effectiveness of reputation systems in trustworthy node selection. Though many reputation calculation methods have been proposed to mitigate collusion's influence, little effort has been devoted to specifically tackling collusion. In this paper, we analyze transaction ratings in the Amazon and Overstock online transaction platforms during one year. The analysis of real trace confirms the existence of collusion as well as its important behavior characteristics and influence on reputation values in real reputation systems. Accordingly, we propose a collusion detection method to specifically thwart collusion behaviors. We further optimize the method by reducing the computing cost. Experimental results show that the proposed method can significantly enhance the capability of existing reputation systems to deter collusion with low cost.**

***Keywords*-Collusion Detection; Reputation Systems; Peer-to-Peer Networks**

I. Introduction

A P2P is a world-wide distributed system, where each node acts as both a client and a server. It interconnects geographically distributed resources (e.g., file, computing and storage resources) to make possible the sharing of the resources. A recent survey shows that more than 50% of the files downloaded and 80% files uploaded on the Internet are through P2Ps [23]. Due to P2Ps' open environment where many autonomous nodes without preexisting trust relationships share resources between each other, they usually employ reputation systems for high reliability and security.

In a reputation system, reputation manager(s) computes and publishes reputation score for each node based on a collection of ratings from others about the node in order to provide guidance in selecting trustworthy resource servers. Despite the effectiveness of the reputation systems, they are generally vulnerable to collusion [12, 29], which impairs their effectiveness in trustworthy node selection. In collusion, two or more malicious peers conspire to give each other high local reputation values and (or) give all other peers low local reputation values in order to gain high global reputation [9]. Colluders usually offer low quality of service (QoS) and receive low ratings from nodes outside of the colluding collective [12, 29].

Current methods that can indirectly deal with collusion focus on how to calculate node reputations to mitigate the influence of collusion. They can be generally classified into three groups: (1) a node calculates others' reputations based on its own experience [8, 13, 17, 18]; (2) a node includes the feedback of pretrusted peers and (or) assigns weights to nodes' feedback according to their global reputation [9, 21, 24, 27]; and (3) a node uses friends' voting to choose trustworthy nodes [6, 7, 28]. These methods can reduce the impact of collusion when determining node trustworthiness, but they do not specifically tackle collusion. SocialTrust [11] and Bazaar [15] are proposed to detect collusion based on social network and interaction networks. However, building such networks may cost high system overhead. Meanwhile, how to ensure the links in the networks are trustable is a new problem. Our proposed methods can be a complement of existing reputation systems to enhance their capabilities to detect and combat collusion, which is only based on user interaction history.

In this paper, we analyze transaction ratings in the Amazon [1] and Overstock [3] transaction platforms during one year. The analysis of real trace confirms the existence of collusion as well as its important behavior characteristics and influence on reputation values in real reputation systems. That is, the suspected colluders (1) gain high global reputations, (2) frequently give each other high reputation values and (or) give other peers low reputation values, (3) receive low ratings from other nodes, and (4) tend to conspire in pairs rather than in a group of more than 2 nodes. The observations of (1) and (2) are consistent with the definition of collusion in [9]. The observation (3) is consistent with the conclusion in [12] that most group collusion are pairs and groups of three of more are rare in a file sharing system. Thus, in this paper, we mainly consider pair collusion and leave collusion of more nodes as our future work.

According to the behavior characteristics and influence, we propose a collusion detection method to directly thwart collusion behaviors by directly monitoring user interaction history with low overhead. In the method, the reputation manager(s) detects collusion based on collected rating values and rating frequency between nodes. If two high-reputed

0190-3918/12 $26.00 © 2012 IEEE
DOI 10.1109/ICPP.2012.20

nodes give high ratings to each other at a high frequency, while they receive low ratings from other nodes, the two nodes are suspected colluders. We discuss how to conduct collusion detection in centralized reputation systems using a single reputation manager, and in decentralized reputation systems using a number of reputation mangers. We further optimize the method by reducing computing cost. We conduct experiments to compare the proposed methods with another reputation system capable of handling collusion. Experimental results show that the proposed method can significantly enhance the capability of existing reputation systems in deterring collusion with low cost.

II. Related Work

Many reputation systems have been proposed that assign reputation scores based on performance measures of peers, and then judge the peers according to the peers' reputation scores. A number of systems [5, 19–21, 26, 30, 31] focus on how to accurately calculate the global reputation value of the nodes. PeerTrust [26] computes peer reputation scores based on three basic trust parameters and two adaptive factors. Trustme [19] offers an approach toward anonymous trust management, which can provide mutual anonymity for both the trust host and the trust querying peer. TrustGuard [21] incorporates historical reputations and behavioral fluctuations of nodes into the estimation of their trustworthiness. FuzzyTrust [20] uses fuzzy logic inferences to better handle uncertainty, fuzziness, and incomplete information in peer trust reports. GossipTrust [30] enables peers to share weighted local trust scores with randomly selected neighbors until reaching global consensus on peer reputations. Hwang *et al.* [31] proposed PowerTrust, which dynamically selects small number of power nodes that are most reputable using a distributed ranking mechanism. By using a look-ahead random walk strategy and leveraging the power nodes, it significantly improves in global reputation accuracy and aggregation speed. Scrubber [5] fights polluted file content by rating both the file provider and file. A number of systems focus on first-hand reputation calculation [8, 13, 17, 18]. In these systems, a node only believes its own observations about other nodes' behaviors, and exchanges of reputation information between nodes are disallowed. Cornelli *et al.* [4] proposed a hybrid approach for P2P security where servants can share with others information about the reputation of their peers. Each client is allowed to compute a personalized, rather than global, performance score for peers in the network.

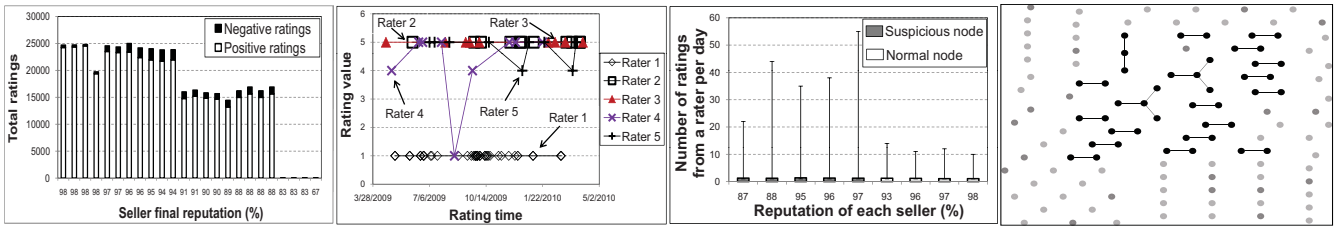
However, as Lian *et al.* [12] indicated, the existing reputation systems are vulnerable to collusion. To understand more about collusion behaviors, they conducted an empirical study of such behaviors in the Maze P2P file sharing system, and detected different types of collusion patterns. Moreton *et al.* [14] proposed the stamp algorithm to combine reputation and payment protocols, which effectively inhibits the collusion behaviors. In the algorithm, peers issue stamps to use as virtual currency for each interaction, and the value of each peer's stamps is maintained by exchange rates that act as reputation values. EigenTrust [9] depends on the friend network to break collusion collectives by using the feedback of pretrusted peers. Yang *et al.* [27] proposed to leverage trust relationships among peers in the social network and assumed that the pretrusted peers only trust their friends other than the other peers in the network. Srivatsa *et al.* [21] proposed the notion of personalized (local) credibility measurement in which the feedbacks from similar raters are given more weight. SocialTrust [11] adaptively adjusts the weight of ratings based on the social distance and interest relationship between peers to combat collusion. XRep [7] and X^2Rep [6] extended the work in [4] by additionally computing object reputations based on weighted peer voting. Sorcery [28] introduces a social network to the P2P content sharing system, in which each client establishes friend-relationships with either acquaintances in reality or those reliable online friends. Sorcery clients utilize the overlapping voting histories of both their friends and the content providers to challenge the latter's actively, and judge whether a content provider is a colluder based on the correctness of its response.

III. Analysis of Reputation Data in Amazon

Amazon [1] and Overstock [3] are by far two of the most successful transaction platforms using a centralized reputation system. We analyze the public domain of Amazon and Overstock reputation systems to verify our conjecture that suspicious collusion exists in a typical reputation system, and to better understand user collusion behaviors in a real reputation system and its influence on the reputation values. Although we cannot confirm whether these observed suspicious behavior are real collusion or not, by preventing colluders from gaining profits (e.g., reputations value) through these suspicious collusion after we detect them, the colluders underlying business model will be destroyed. Then nodes do not have incentives to collude with each other.

In Amazon, the feedback score is within $[1, 5]$. The scores 1 and 2 are classified as negative rating (-1), 3 as neutral rating (0) and 4 and 5 as positive rating (1). A seller's reputation is simply calculated by dividing the number of positive ratings by the sum of all ratings.

We have crawled a total of more than 2.1 million ratings for 97 book sellers during the time period from April 15, 2009 to April 1, 2010. We randomly picked a number of sellers in each reputation level and show their total numbers of ratings (i.e., transactions) consisting of positive and negative ratings in Figure 1. The figure illustrates that high-reputed sellers [0.94-0.98] have more transactions than median-reputed sellers [0.88-0.91]. An exceptional case is that a seller with reputation 0.98 has fewer transactions than median-reputed nodes. This is because the seller joined in



(a) Rating vs. reputation. (b) Ratings on a suspicious seller (c) # of ratings of a rater per day (d) Graph of interaction in Overstock

Figure 1: Analysis of the reputation data from Amazon and Overstock.

Amazon later than others. The figure also shows that low-reputed sellers [0.67-0.79] receive few transactions. This is because most of the customers do not trust the services of the low-reputed sellers and select high-reputed sellers for purchases. Generally, the data implies that a higher reputed seller can attract more transactions, and lower reputation nodes have fewer transactions. This is the reason that colluders try to increase their reputations by collusion. Since high reputation is the objective of colluders, colluders are very likely to have high reputations.

It is very intriguing to find that although the sellers with reputation in [0.94,0.97] have lower reputations than the sellers with reputation 0.98, the former still received approximately the same number of ratings as the latter. We suspect some of these nodes are colluders that receive high ratings from other colluders. Since colluders always repeatedly submit ratings in order to boost the reputation value of each other, in order to verify our conjecture, we further find the sellers that have received a large number of ratings from the same buyer from April 15, 2009 to April 1, 2010. The collected data shows that the average number of transactions of a seller-buyer pair is 1 per year. In order to shed to light to a small number of subspinous users for detailed analysis, we set the suspicious behavior filtering threshold as 20 ratings, which gives us 18 suspicious sellers and 139 suspicious raters. These suspicious sellers include the sellers with reputation in [0.94,0.97] in the Figure 1. It confirms our conjecture that these high-reputed nodes are very likely to be colluders. From this result, we can conclude that:

Characteristic 1 (C1). *Collusion leads to high reputation of the colluders [9].*

C2. *Among the high-reputed nodes, colluders receive more low reputations than non-colluders [12, 29].*

Because our observations are consistent with the conclusions from other papers, we indicate the papers along with the characteristics.

The received rating patterns from the suspicious sellers are similar. Therefore, we randomly picked one suspicious seller to investigate its received ratings as an example. The reputation of the selected suspicious seller is 0.95 and it received 2,037 negative ratings and 21,958 positive ratings. We found that two buyers rated the seller more than 20 times from April 15, 2009 to April 1, 2010. One repeatedly gave score 5 while the other repeatedly gave score 1. This means one node is possibly a partner colluder that tried to increase the seller's reputation, while the other node is possibly the seller's rival that tried to decrease its reputation. We also observed that 26 buyers rated the seller more than 15 times during that time and their behaviors exhibit three patterns. We chose 5 raters with the 3 typical behavior patterns and plot their ratings over time in Figure 1(b). The figure shows that raters 2 and 3 continuously rated the seller with the highest score 5. They are very likely to be colluders since a buyer has a very low probability of repeatedly choosing a median-reputed seller among many high-reputed nodes in Amazon. We also can see that rater 1 continuously rated with the lowest score. Such behavior is also not reasonable. For a normal node, if it receives poor service from a seller, it will no longer choose that seller. Thus, rater 1 is possibly a rival colluder that tried to decrease the seller's reputation. Since raters 4 and 5 sometimes gave high and sometimes gave low ratings to the seller, they should be not colluders. From this result, we can conclude that:

C3. *Colluders frequently submit very high ratings for their conspirators [9].*

We randomly chose 5 suspicious and 4 unsuspicious sellers and show each rater's average number of ratings per day, and the maximum and minimum number of all of its ratings during April 15, 2009 to April 1, 2010 in Figure 1(c). The figure shows that for the nodes with similar reputations, the average number of ratings of a rater per day and the maximum number of a rater during all the time received by suspicious sellers are much larger than those of unsuspicious nodes. The suspicious sellers may be colluding with raters to increase their reputations and counter the negative ratings from other raters. We also can see from the figure that the suspicious sellers exhibit much larger rating variance than the unsuspicious sellers. It means that some raters (i.e., suspected colluders) give ratings much more frequently than other raters, and the high maximum ratings of the suspicious sellers are from the colluders. Thus, we can conclude

C4. *The rating frequency between colluders is much higher than the rating frequency between normal nodes. In Amazon, the maximum frequency is 55/year for the former, and is 15/year for the latter.*

In Amazon, buyers rate the sellers but sellers do not

rate buyers. In order to study group collusion patterns, we crawled rating data from the Overstock Auction system where each user can be both seller and buyer. We crawled the ratings among approximately 100,000 users with over 450,000 transactions during Oct., 2009 to Sept., 2010. To make the graph looks more clear, we randomly sample 500 users and represent them as nodes in a graph. If the number of ratings between node i to node j exceeds 20, we drew an edge between the two nodes. Figure 1(d) shows the final graph. We removed many other nodes without edges in order to show the suspected colluders clearly. The black nodes on the graph are suspected colluders since they rate each other with high rating frequency, and grey nodes are buyers and (or) sellers only. We can see that the suspected colluders rate each other in pairs. There is no closed structure with 3 or more nodes, which means there is no suspected collusion involving more than 2 nodes. The figure has three nodes connecting together, but they are still in a pair-wise manner, which means a node may have multiple colluders but only in a pair rather than a group with $>$2 nodes. The graph confirms the conclusion in [12] that:

C5. *Most collusion behaviors are in pairs, and the collusion of multiple colluders in one group rating each other is very rare [12].*

IV. Collusion Detection Methods

A. Background

Our proposed methods can be built on any reputation system to enhance its capacity to combat collusion. In a centralized reputation system, such as the one in Amazon, a resource manager collects the ratings of all nodes and calculates the reputation values of all nodes. The decentralized reputation systems are more complex. We use EigenTrust [16] as an example to explain how a decentralized reputation system works. EigenTrust forms a number of high-reputed power nodes into a Distributed Hash Table (DHT) for reputation aggregation and calculation. These power nodes are reputation managers. We use ID_i to represent the DHT ID of node n_i, which is the consistent hash value [10] of node n_i's IP address. The reputation manager of reputation ratings on node n_i is the DHT owner of ID_i. A node uses DHT function `Insert(`ID_i`,`r_i`)` to send the rating of node n_i to its reputation manager, and uses `Lookup(`ID_i`)` to query the reputation value of node n_i from its reputation manager. A reputation manager periodically collects the ratings and computes the global reputation values of its responsible nodes.

Figure 2 presents a 4-node reputation system built on top of the Chord DHT [22] with 4-bit circular hash space. Other nodes report to $n15$ about $n10$'s local reputation by `Insert(10,`r_{10}`)`. Node $n15$ calculates $n10$'s global reputation value. When a node, say $n6$, wants to select a server from several candidates, it queries for the reputation

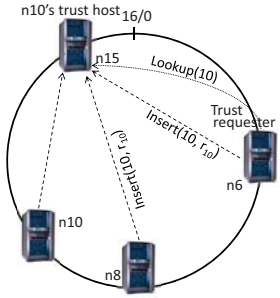


Figure 2: A distributed reputation system.

values of the servers. For example, it uses `Lookup(10)` to query $n10$'s reputation value, denoted by R_{10}.

There are many ways to calculate global reputation values of nodes [9, 25, 26]. We use the local reputation calculation method in eBay [2] and EigenTrust [9] as an example in this paper. That is, the local reputation rating for each interaction for a node is -1, 0 and 1. A node's final reputation is the sum of all its received reputation evaluation values. Reputation systems usually specify a reputation threshold T_R. Nodes whose $R \geq T_R$ are considered as trustworthy while nodes with $R < T_R$ are considered as untrustworthy. To apply our method to the reputation systems that use different reputation calculation methods, we regard local reputation rating with $\geq T_R$ as 1, and local reputation rating with $< T_R$ as -1.

B. Basic Collusion Detection Method

Based on the characteristics of collusion (C1-C5) presented in Section III, we build a collusion model shown in Figure 3 that incorporates all the characteristics. In the collusion, two (C5) nodes frequently (C4) rate high reputation for each other (C3) in order to increase their global reputation values (C1), but offer low-QoS to other nodes and receive low ratings from other nodes (C2).

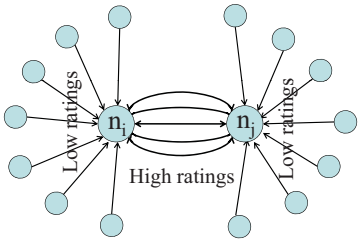


Figure 3: Collusion model.

Consider a pair of nodes n_i and n_j, we define a number of notations shown in Table I. We use a to denote the percent of positive ratings in all ratings from n_j for n_i, and use b to denote the percent of positive ratings in all ratings from all nodes except n_j for n_i. We specify a threshold T_a for a and a threshold T_b for b. T_a and T_b can be determined by the historical data of a and b of pairs of nodes with high interaction frequency. For example, in our crawled data in Section III, for those suspicious colluders found using the threshold=20 for the average number of transactions of a seller-buyer pair per year, the average a=98.37 and average b=1.63. If we want to reduce the false negatives in collusion detection, we can decrease T_a and increase T_b. On the other hand, if we want to reduce the number of false positives in

collusion detection, we can increase T_a and decrease T_b. We use $N_{(i,j)}$ to denote the number of ratings from n_j for n_i in T which is the time period for updating global reputations. We also specify rating frequency threshold T_N for $N_{(i,j)}$ to show how frequently n_j rates n_i. Based on our trace data, T_N=20/year.

Table I: A list of notations in the paper.

T	the time period for updating global reputations
N_i	the number of all ratings for n_i in T
$N_{(i,j)}$	the number of ratings from n_j for n_i in T
$N_{(i,-j)}$	the number of ratings from all nodes except n_j for n_i in T
$N^+_{(i,j)}$	the number of positive ratings from n_j for n_i in T
$N^+_{(i,-j)}$	the number of positive ratings from all nodes except n_j for n_i in T
$N^-_{(i,j)}$	the number of negative ratings from n_j for n_i in T
$N^-_{(i,-j)}$	the number of negative ratings from all nodes except n_j for n_i in T
a	percent of positive ratings in all ratings from n_j for n_i
b	percent of positive ratings in all ratings from all nodes except n_j for n_i

In the proposed collusion detection method, resource manager(s) relies on the reputation values and frequencies of ratings between a pair of nodes for collusion detection according to the constructed collusion model in Figure 3. We first describe the method in a centralized reputation system, and then describe how the method works in a decentralized reputation system.

Since colluders are usually high-reputed nodes, and reputation systems regard nodes whose $R > T_R$ as trustworthy nodes, we only check these nodes in collusion detection. For each node in the system, the centralized reputation manager keeps track of the frequency of ratings and frequency of positive ratings of every other node to the node. The reputation manager builds an $n \times n$ matrix, where n is the number of nodes in the network. The matrix records the reputation ratings for nodes whose $R \geq T_R$. If node n_i's reputation value $R_i \geq T_R$, matrix element a_{ij}=$< ID_i, R_i, N^+_{(i,j)}, N_{(i,j)} > (1 \leq j \leq n)$. Otherwise, a_{ij}=empty.

The manager periodically updates the matrix with its collected information and detects collusion according to the characteristics in the collusion model. In collusion detection, the manager scans each row in the matrix in the top-down manner, and scans elements in each row from the left to the right. For high-reputed node n_i (C1) in a row (non-empty line), the manager checks a_{ij} from every other node n_j (each column) for n_i. If R_j>T_R and $N_{(i,j)}$≥T_N, which means n_j also has a high reputation (C1) and n_j rates n_i frequently (C4), then $N^+_{(i,j)}$ is further checked. If $N^+_{(i,j)}/N_{(i,j)}$≥T_a, which means a large portion of n_j's ratings are positive (C3), then the ratings from all other nodes except n_j (all other columns) are checked. The manager scans each element in the line of n_i except a_{ij} and calculate the sum of all positive ratings, $N^+_{(i,-j)}$, and the sum of all ratings, $N_{(i,-j)}$. If $N^+_{(i,-j)}/N_{(i,-j)}$<T_b, which means a large portion of ratings from other nodes except n_j are negative (C2), then we can conclude that n_i's high reputation is mainly caused by n_j's frequent ratings that are deviated from most others' rating values. In this case, the manager finds the line of n_j in the matrix and repeats the same process for n_j to check if n_j's high reputation is also mainly caused by n_i's ratings. Then, if $N_{(j,i)}$≥T_N, $N^+_{(j,i)}/N_{(j,i)}$≥T_a and $N^+_{(j,-i)}/N_{(j,-i)}$<T_b, n_i and n_j are very likely to be colluding together. During the checking process, after an a_{ij} is checked, the manager marks a_{ij} and a_{ji} to indicate that the two elements no longer need checking.

Unlike centralized reputation systems, decentralized reputation systems distribute the role of the centralized resource manager to a number of trustworthy nodes. As mentioned, a reputation manager M_i of node n_i keeps track of all ratings of other nodes for n_i. Thus, using the same way as the centralized reputation systems, each reputation manager builds a $\tilde{n} \times n$ matrix, where $\tilde{n}$ is the number of its responsible nodes. For reputation manager M_i, for each of its responsible node n_i with R_i≥T_R, if $N_{(i,j)}$>T_N, $N^+_{(i,j)}/N_{(i,j)}$≥T_a and $N^+_{(i,-j)}/N_{(i,-j)}$<T_b, n_i is suspected to collude with n_j. Then, if M_i is the reputation manager of node n_j, it uses the same method in the centralized reputation system for the collusion detection. Otherwise, M_i contacts n_j's reputation manager M_j by the DHT function `Insert(j,msg)`. Then, M_j checks R_j and ratings from n_i for n_j. If n_j has high reputation and its reputation is mainly caused by n_i's frequent ratings that deviate from most others' ratings, i.e., R_j>T_R, $N_{(j,i)}$≥T_N, $N^+_{(j,i)}/N_{(j,i)}$≥T_a and $N^+_{(j,-i)}/N_{(j,-i)}$<T_b, M_j sends a positive response to M_i indicating that n_i and n_j are likely to be colluders.

We use m to denote the number of high-reputed nodes and n to denote the total number of nodes in the system.

Proposition 4.1: In the collusion detection method, the computation complexity to identify colluders in the P2P system is O(mn^2).

Proof: For each high-reputed node n_i $(1 \leq i \leq m)$, at most n elements should be checked. For each checking, at most n elements are scanned. Thus, the computation complexity to identify colluders in the system is O(mn^2). ■

The collusion detection method can effectively identify the colluders based on the characteristics of collusion behaviors. However, in order to calculate $N^+_{(i,-j)}$ and $N_{(i,-j)}$, for each rater n_j, each element in matrix line i should be scanned, generating a high computing cost. Therefore, we propose an optimized collusion detection method that produces much lower computation cost without compromising the collusion detection performance.

C. Optimized Collusion Detection Method

In the optimized collusion detection method, a resource manager does not need to scan each element in matrix line i for each $N_{(i,j)}$ $(1 \leq j \leq n)$. Each manager detects collusion only based on the global reputation value of each of their responsible nodes n_i and the frequency of ratings of each of

other nodes for n_i. According to the global reputation value calculation method, for a given pair of nodes n_i and n_j, we can get:

$$\begin{cases} b \cdot (N_i - N_{(i,j)}) = N^+_{(i,-j)} \\ (1-b) \cdot (N_i - N_{(i,j)}) = N^-_{(i,-j)} \\ a \cdot N_{(i,j)} = N^+_{(i,j)} \\ (1-a) \cdot N_{(i,j)} = N^-_{(i,j)} \\ N^+_{(i,-j)} + N^+_{(i,j)} - N^-_{(i,-j)} - N^-_{(i,j)} = R_i \end{cases}$$

$$\Rightarrow R_i = 2b(N_i - N_{(i,j)}) + 2aN_{(i,j)} - N_i. \quad (1)$$

Figure 4 visualizes Formula (1) when $1 \geq a \geq T_a$ (1) and $T_b \geq b \geq 0$ (2), which means that a large portion of ratings from n_j are positive, whereas a large portion of ratings from other nodes are negative. In this case, n_i's high reputation is mainly due to n_j's ratings. Thus, we suspect that nodes n_i and n_j collude with each other. The surface in the figure shows the range of the reputation values of a suspected colluder corresponding to $N_{(i,j)}$ and N_i, given different set of T_a and T_b. Since $(N_i - N_{(i,j)}) > 0$ and $N_{(i,j)} > 0$, based on Equation 1 and the conditions (1) and (2), we can derive

$$2T_b(N_i - N_{(i,j)}) + 2N_{(i,j)} - N_i \geq R_i \geq 2T_a N_{(i,j)} - N_i. \quad (2)$$

If the values of $N_{(i,j)}$, N_i and R_i conform Formula (2), node n_i and n_j are very likely to collude with each other.

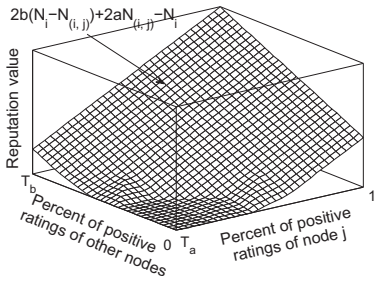


Figure 4: Reputations of colluders.

Therefore, to detect collusion, each manager M_i first identifies nodes whose $R \geq T_R$. For each of such node n_i, the manager then checks the rating frequency of each rater of n_i, $N_{(i,j)}$. If $N_{(i,j)} \geq T_N$, the manager then uses Formula (2) to check whether n_i's high reputation is possibly due to its collusion with n_j. In the case that Formula (2) is satisfied, if M_i is n_j's reputation manager, it checks whether n_j's high reputation is possibly due to its collusion with n_i in the same manner. Otherwise, M_i uses `Insert(j,msg)` to contact the reputation manager of n_j. Reputation manager M_j conducts the same process to check whether n_j's high reputation is possibly due to its collusion with n_i. If $R_j \geq T_R$, $N_{(j,i)} \geq T_N$ and Formula (2) are also satisfied, n_i and n_j are very likely to be colluding with each other.

Proposition 4.2: In the optimized collusion detection method, the computation complexity to identify colluders in the P2P system is O(mn).

Proof: For each high-reputed node n_i ($1 \leq i \leq m$), at most n elements should be checked for collusion detection. Thus, the computation complexity to identify colluders in the system is O(mn). ■

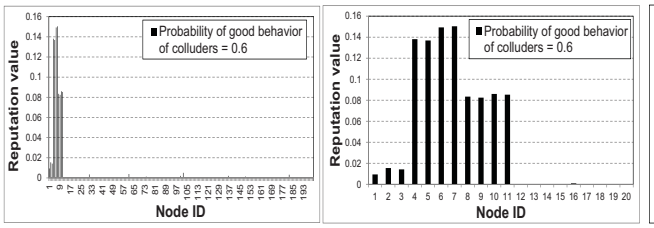
V. Performance Evaluation

In the simulation, we compare our proposed methods with EigenTrust [9], which is a representative of reputation management systems. EigenTrust employs pretrusted nodes to combat collusion. A node's reputation $R = \sum_j w_{1_j} r_j + \sum_p w_2 r_p$, where w_{1_j} and r_j are the rating weight and rating from node n_j, and w_2 and r_p are the rating weight and rating from the pretrusted node n_p. A node with higher reputation has higher w_{1_j} and $w_2 > w_{1_j}$. In our experiment, $w_{1_j} = 0.2$ and $w_2 = 0.5$, which is the honey spot parameters of the system.

Network model. In order to fully evaluate how the behaviors of people affect the collusion detection performance of our proposed algorithm and EigenTrust, we consider a generic P2P resource (e.g., file) sharing network in which the peers are able to issue resources queries and resources offers with different qualities directly between each other. We built an unstructured P2P network with 200 nodes. The ratio of the number of individual node's interests to the number of all interest categories is similar to the actual ratio in Overstock [11]. Specifically, we assume there are 20 interest categories in the system. The number of interests a node has is randomly chosen from [1,5], and the interests are randomly chosen from the 20 interests. In the P2P network, nodes with the same interest are connected with each other in a cluster. A node with m interests is in m clusters. Each node in the system has maximum 50 units of capacity (i.e., it can handle 50 requests simultaneously per query cycle). For a request of a file in an interest, a node queries all of its neighbors in the cluster of the interest, and chooses its highest-reputed neighbor with available capacity greater than 0. If a number of options have an identical reputation value, then the client randomly selects a node as a server.

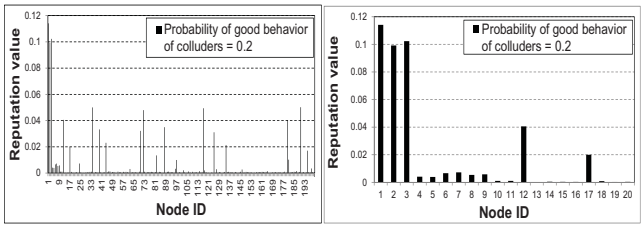
Node model. We consider three types of nodes: pretrusted nodes, colluders and normal nodes. The pretrusted nodes always provide authentic files to the requesters. Normal nodes provide inauthentic files with a default probability of 20% unless otherwise specified. We use B to denote the probability that a node offers an authentic file (i.e., good behavior). We randomly chose 3 pretrusted nodes and 8 colluders. In order to show the results more clearly, the IDs of the pretrusted nodes were set to 1, 2 and 3, and the IDs of the colluders were set to 4-11. The weight of the ratings from pretrusted nodes was set to 0.5 in EigenTrust.

Simulation execution. In the simulation process, a node can only send out a file request when it is active. The probability that a node is active is randomly chosen from [0.3, 0.8]. The simulation proceeds in simulation cycles. Each simulation cycle is subdivided into 20 query cycles.



(a) All nodes. (b) First 20 nodes.

Figure 5: Reputation distribution in EigenTrust when B=0.6 (Pretrusted node IDs 1-3, colluder IDs: 4-11).



(a) All nodes. (b) First 20 nodes

Figure 6: Reputation distribution in EigenTrust when B=0.2 (Pretrusted node IDs 1-3, colluder IDs: 4-11).

In each query cycle, each peer issues a query if it is active. Each experiment has 20 simulation cycles. Each experiment is run 5 times and the average of the results is the final result.

Collusion model. We simulate the pair-wise collusion, which is the most common collusion (C5), in the experiments. In addition to functioning as normal nodes, colluders also mutually rate each other with positive value in order to boost the reputation of each other. We paired up two colluders and let them rate each other 10 times per query cycle.

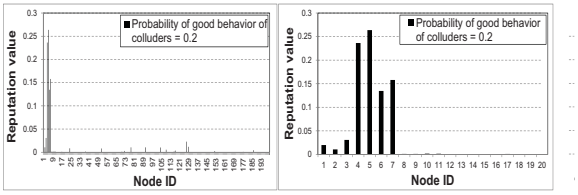
Reputation model. The initial reputation value of each node is 0. Similar to the rating mechanism used in Amazon and Overstock, a client gives a server rating 1 when it receives an authentic file and rating -1 when it receives an inauthentic file. A node's local reputation value equals to the sum of all ratings. Each node updates reputations once after each simulation cycle based on the EigenTrust reputation calculation method. The reputation threshold was set to 0.05. With a collusion-resilient reputation system, we expect to see that the nodes with ID 4-11 (i.e., colluders) have extremely low reputation values and the normal nodes have comparably higher reputation values. We also conducted experiments with different numbers of nodes and colluders. The relative performance differences between the different systems remain almost the same as those we will report. Though we determine the experimental setting parameters randomly in their reasonable ranges, changing the parameter values will not change the relative performance differences in a given experiment setup.

A. The Effectiveness of EigenTrust in Thwarting Collusion

Figure 5 shows the reputation distribution of all the nodes in the system when $B = 60\%$ (i.e., a colluder offers authentic files with 60% probability). Figure 5(a) shows that the high-reputed nodes are skewed at pretrusted nodes with IDs in $1-3$ and colluders with IDs in $4-11$. We also can see that some normal nodes have relatively higher reputations than others. In order to make the results clearer, we show the reputations of 20 nodes with IDs in $1-20$ in Figure 5(b). We can see that colluders have much higher reputation values than others. The reputations of pretrusted nodes are higher than normal nodes, but are significantly lower than colluders. Since the colluders behave well with probability of 60%, they gain a certain number of high ratings though they have 40% probability to receive negative ratings. Furthermore, because of the collusion, colluders increase the reputations of each other greatly, which helps them to attract many file requests to further increase their reputations. Though the normal nodes and pretrusted nodes offer good services with probabilities of 80% and 100%, respectively, they still receive lower reputation values than colluders. The results show that EigenTrust cannot detect the collusion behavior and its generated reputations cannot accurately reflect the trustworthiness of nodes.

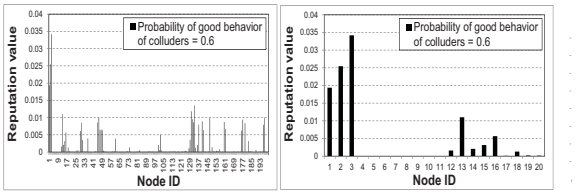
Figure 6(a) shows the reputation distribution of the nodes in the system when the colluders offer authentic files at a probability of 20%. From this figure, we can see that some normal nodes have higher reputations while others have lower reputations. This is because at first when all reputation values are 0, nodes randomly choose servers. Since the chosen servers earn reputation, they will have higher probability to be chosen and to further increase their reputations later on. As a result, the nodes first chosen have higher reputations than others. Figure 6(b) further shows the reputation values of nodes with IDs in $1-20$. Comparing this figure with Figure 5(b), we can see that EigenTrust is able to reduce the reputation values of the colluders when $B = 20\%$. Though colluders can try to increase the reputation of each other, the reputations they receive from many other nodes are very low with 20% good behavior probability. Therefore, they are unable to greatly boost the reputation of each other due to the low weight of their ratings. Consequently, due to their low reputations, other nodes will not ask for files from them. Also, the reputation values of some normal nodes are accumulated constantly and finally reach high values. Since pretrusted nodes with IDs in $1-3$ always behave well, they can continuously accumulate high reputation values, which finally leads to a much high reputation. Therefore, EigenTrust can reduce the influence of collusion behaviors in the system when the colluders offer low QoS services at most of the time.

In the above experiment, we assume that the pretrusted nodes are trustable and are not involved in the collusion. In this experiment, we assume that pretrusted node n_1 colludes with node n_4 and pretrusted node n_2 colludes



(a) All nodes. (b) First 20 nodes.

Figure 7: Reputation distribution in EigenTrust with compromised pretrusted nodes when B=0.2 (Pretrusted node IDs 1-3, colluder IDs: 4-11).



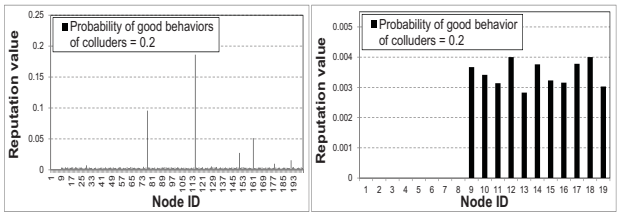
(a) All nodes. (b) First 20 nodes.

Figure 9: Reputation distribution in EigenTrust employing our proposed methods when B=0.6 (Pretrusted node IDs 1-3, colluder IDs: 4-11).

with n_6. Nodes $n_4 - n_{11}$ are still paired up and collude with each other. Colluders offer authenticate files with probability of 0.2. Figure 7(a) shows that the high-reputed nodes are skewed among pretrusted nodes and colluders. Figure 7(b) further shows the reputation distribution of the first 20 nodes. The figure shows that the reputations of nodes $n_4 - n_7$ are boosted while the reputation value of nodes $n_8 - n_{11}$ are much lower compared to Figure 6(b). The reason is that because pretrusted nodes n_1 and n_2 rate highly on node n_4 and node n_6, since EigenTrust assigns more weight to the ratings of pretrusted nodes, n_4 and n_6's reputations are significantly increased. Since nodes n_4 and node n_6 rate highly on their colluding partners n_5 and n_7, respectively, the reputations of n_5 and n_7 also increase greatly. Because nodes always choose the highest-reputed nodes with available capacity, the reputations of these colluders continually increase and ultimately even exceed the pretrusted nodes' reputations. Since they attract more file requests, colluders $n_8 - n_{11}$ with lower reputations cannot receive more requests to increase their reputations. The result implies that compromising pretrusted nodes will exacerbate the negative impact of collusion on the reputation system, and EigenTrust cannot effectively deal with such malicious behaviors.

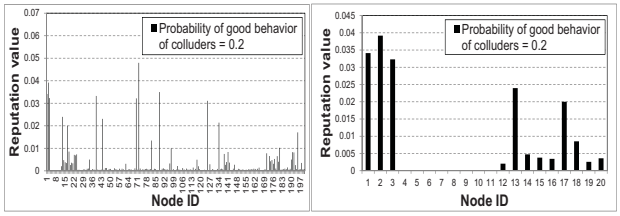
B. The Effectiveness of Our Proposed Methods in Thwarting Collusion

In this section, we test the collusion detection effectiveness of our proposed basic collusion detection method (denoted by *Unoptimized*) and optimized collusion detection method (denoted by *Optimized*). Since our proposed



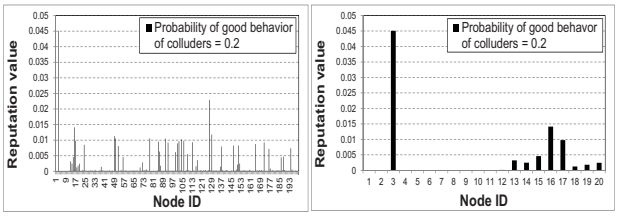
(a) All nodes. (b) First 20 nodes.

Figure 8: Reputation distribution in our proposed collusion detection methods when B=0.2 (Colluder IDs: 1-8).



(a) All nodes. (b) First 20 nodes.

Figure 10: Reputation distribution in EigenTrust employing our proposed methods when B=0.2 (Pretrusted node IDs 1-3, colluder IDs: 4-11).



(a) All nodes. (b) First 20 nodes.

Figure 11: Reputation distribution in EigenTrust employing our proposed methods with compromised pretrusted nodes when B=0.2 (Pretrusted node IDs 1-3, colluder IDs: 4-11).

methods do not use pretrusted nodes, in the experiments, nodes $n_1 - n_8$ are configured as colluders. After the methods detect the colluders, they set their reputations to 0. Since *Unoptimized* and *Optimized* are only used for collusion detection rather than reputation value calculation, their final reputation distributions of the nodes are the same. We show the results of both *Unoptimized* and *Optimized* in Figure 8(a) and (b) with the probability of good behavior of colluders equals to 20%. Figure 8(a) shows that some normal nodes have very high reputation values. As explained previously, this is because nodes randomly choose servers at first and later choose the highest-reputed node with available capacity for files. Thus, the nodes first chosen have high probability to be chosen later on. Figure 8(b) shows that both *Unoptimized* and *Optimized* can detect all colluders, which indicates that the methods can effectively detect the colluders based on their contact frequency and reputation values.

Next, we test how *Unoptimized* or *Optimized* can help

enhance EigenTrust's capability in detecting collusion. Since *Unoptimized* and *Optimized* generate the same results in collusion detection, we use *Optimized* to represent both and use EigenTrust+*Optimized* to denote EigenTrust employing *Optimized*. Figure 9(a) shows the reputation distribution of the nodes in EigenTrust+*Optimized* when the colluders offer authentic files with probability of 60%. Figure 9(b) further shows the reputation distribution of the nodes with IDs in $1-20$. Comparing to Figure 5, we find that EigenTrust+*Optimized* leads to increased average reputation values for many normal nodes. Also, it reduces the high reputations of colluders with IDs in $4-11$ to 0, and significantly increases the reputations of pretrusted nodes. The results demonstrate the effectiveness of *Optimized* in detecting collusion. By analyzing the rating patterns of the nodes, *Optimized* quickly identifies the colluders and reduces their reputation to 0. Subsequently, the colluders can no longer attract requests. By receiving more requests, the reputation values of other nodes increase, especially the pretrusted nodes due to their better QoS and higher reputations.

Figure 10(a) shows the reputation distribution of the nodes in EigenTrust+*Optimized* when the colluders offer authentic files with probability of 20%. Figure 10(b) further shows the reputation distribution of the first 20 nodes. Comparing to Figure 6, we observe that EigenTrust+*Optimized* increases the reputations of normal nodes by a greater amount than EigenTrust. Also, it reduces the reputation values of the colluders to 0. The results confirm the effectiveness of *Optimized* in collusion detection by their rating patterns. Since the pretrusted nodes can attract more file requests as they always offer authentic files, their reputation values remain high. As the normal nodes receive more opportunities to offer service, their average reputation values are also increased.

Figure 11 (a) and (b) show the reputation distribution of the nodes in Eigentrust+*Optimized* in the same scenario as Figure 7. Comparing Figure 11 and Figure 7, we see that Eigentrust+*Optimized* increases the reputations of normal nodes in Eigentrust. By compromising pretrusted nodes, colluders can receive much higher reputations than pretrusted nodes in Eigentrust. While in Eigentrust+*Optimized*, both colluders and compromised pretrusted nodes receive 0 reputation values. Since the pretrusted node with ID=3 does not involve in the collusion, its reputation value is still high. Because the detected colluders and compromised nodes have 0 reputations, the normal nodes have more opportunities to receive file requests and their reputations subsequently increase. In conclusion, our proposed *Unoptimized* and *Optimized* collusion detection methods can greatly enhance the collusion detection capability of EigenTrust.

C. Performance Comparison

Figure 12 shows the percent of the file requests sent to the colluders in the total number of requests in the system versus the number of colluders in the system. In this experiment, the setting of EigenTrust is identical to Figure 6. If the computed global reputation values can accurately reflect the actual behavior of nodes, the number of requests sent to the colluders should be small. The figure shows that the percentages in *Unoptimized* and *Optimized* are approximately the same. Meanwhile, as the number of colluders in the system increases, the percentages of the requests sent to the colluders increase a little. The results indicate that *Unoptimized* and *Optimized* are effective in Thwarting Collusion. In contrast, in EigenTrust, the percent of the requests sent to the colluders is much higher than the other methods. Moreover, as the number of colluders increases, the percent increases sharply. Since EigenTrust is not effective at collusion detection, the high reputations of the colluders attract many file requests from other nodes. Also, as the number of the colluders in the system increases, the total number of requests received by them increases accordingly, which subsequently leads to a higher percentage. The results in Figure 12 are in line with the results in Figure 6 and Figure 8.

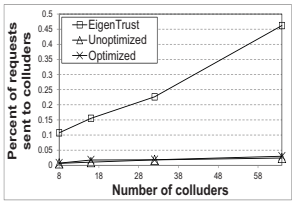


Figure 12: Effectiveness of thwarting collusion.

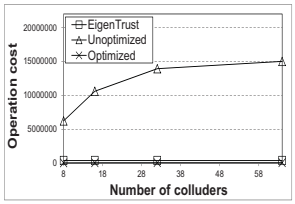


Figure 13: Cost for thwarting collusion.

We define *operation cost* as the number of computer cycles for thwarting collusion. The operation cost of EigenTrust includes the cost for calculating all global reputations for all nodes, and the operation cost of *Unoptimized* and *Optimized* includes the cost for information analysis and computation. Figure 13 shows the operation costs of *Unoptimized*, *Optimized* and EigenTrust. The figure shows that *Unoptimized* generates significantly higher operation cost than others, and EigenTrust produces higher operation cost than *Optimized*. The reason why *Unoptimized* has the highest operation cost is that for each high-reputed node in the system, *Unoptimized* needs to scan all of its raters for rating values and frequency for each rater. Meanwhile, for all high-reputed nodes, the method needs to check each possible pair combination to identify the nodes rating for each other with high frequency. As the number of colluders in the system increases, more high-reputed nodes need to be checked, leading to higher operation cost. The operation cost in EigenTrust is caused by the recursive matrix calculation, which is determined by the number of the nodes in the system rather than the number of colluders in the system. This is the reason that the operation cost of EigenTrust is constant as the number of colluders in the system increases. Since the matrix normally can converge within several iterations, EigenTrust generates less operation cost than *Unoptimized*.

The operation cost of *Optimized* is very low because there is no need to scan the ratings from a node's raters. The only operation cost of *Optimized* is caused by checking the high contacting frequency between high-reputed nodes and collusion inference.

VI. Conclusion

Reputation systems calculate reputation values based on performance measures of peers, and then find the deceptive peers according to the reputation values. Despite the effectiveness of these systems, they are generally vulnerable to collusion behaviors. Though many reputation systems try to reduce the influence of collusion on calculating reputation values, there are few works that specifically tackle collusion currently. After analyzing the Amazon and Overstock transaction traces during a year, we confirm the previously claimed collusion characteristics in a real reputation system. According to the collusion characteristics, we propose a collusion detection method to thwart collusion behavior and further optimize the method by reducing the computing cost. Experiment results show the methods achieve higher capacity in combating collusion in comparison with the EigenTrust reputation system. The methods can detect colluders even when they compromise pretrusted high-reputed nodes to increase their reputations. In our future work, we will study how to determine the threshold values used in this paper effectively and efficiently according to the given system parameters. We will also investigate how to detect a collusion collective having more than two nodes such as Sybil attack.

Acknowledgements

This research was supported in part by U.S. NSF grants 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 Oak Ridge Award 4000111689.

References

[1] Amazon. http://www.amazon.com/.
[2] ebay, the online marketplace. http://www.ebay.com.
[3] Overstock. http://www.overstock.com/.
[4] F. Cornelli, E. Damiani, S. Vimercati, S. Paraboschi, and P. Samarati. Choosing reputable servents in a P2P network. In *Proc. of WWW*, 2002.
[5] C. P. Costa and J. M. Almeida. Reputation systems for fighting pollution in peer-to-peer file sharing systems. *In Peer-to-Peer Computing*, pages 53–60, 2007.
[6] N. Curtis, R. Safavi-Naini, and W. Susilo. X^2Rep: Enhanced Trust Semantics for the XRep Protocol. In *Proc. of ACNS*, 2004.
[7] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proc. of CCS*, 2002.
[8] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *Proc. of EC*, 2004.
[9] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *Proc. of WWW*, 2003.
[10] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and P. R. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In *Proc. of STOC*, 1997.
[11] 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.
[12] Q. Lian, Z. Zhang, M. Yang, B. Y. Zhao, Y. Dai, and X. Li. An empirical study of collusion behavior in the maze P2P file-sharing system. In *Proc. ICDCS*, 2007.
[13] Z. Liang and W. Shi. Pet: a personalized trust model with reputation and risk evaluation for P2P resource sharing. In *Proc. of HICSS*, 2005.
[14] T. Moreton and A. Twigg. Trading in trust, tokens, and stamps. In *Proc. of P2PECON*, 2003.
[15] A. Post, V. Shah, and A. Mislove. Bazaar: Strengthening user reputations in online marketplaces. In *Proc. of the NSDI*, 2011.
[16] M. S. S. Kamvar and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proc. of WWW*, 2003.
[17] R. S. S. Lee and B. Bhattacharjee. Cooperative peer groups in nice. In *Proc. of INFOCOM*, 2003.
[18] A. A. Selcuk, E. Uzun, and M. R. Pariente. A reputation based trust management system for P2P networks. *International Journal of Network Security*, 6(3):235245, 2008.
[19] A. Singh and L. Liu. TrustMe: anonymous management for trust relationships in decentralized P2P systems. In *Proc. of P2P*, 2003.
[20] S. Song, K. Hwang, R. Zhou, and Y.-K. Kwok. Trusted P2P transactions with fuzzy reputation aggregation. *IEEE Internet Computing*, 2005.
[21] M. Srivatsa, L. Xiong, and L. Liu. Trustguard: Countering vulnerabilities in reputation management for decentralized overlay networks. In *Proc. of WWW*, 2005.
[22] I. Stoica, R. Morris, D. Liben-Nowellz, D. R. Kargerz, M. F. Kaashoekz, F. Dabekz, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE TON*, 2003.
[23] Topic: p2p statistics. http://www.bloglines.com/ref/p2p-statistics.html.
[24] K. Walsh and E. G. Sirer. Experience with a distributed object reputation system for peer-to-peer filesharing. In *Proc. of the NSDI*, 2006.
[25] Y. Wang and J. Vassileva. Trust and reputation model in peer-to-peer networks. In *Proc. of P2P*, 2003.
[26] L. Xiong and L. Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *TKDE*, 16(7), 2004.
[27] M. Yang, Y. Dai, and X. Li. Bring reputation system to social network in the maze P2P file-sharing system. In *Proc. of CTS*, 2006.
[28] E. Zhai, R. Chen, Z. Cai, L. Zhang, E. K. Lua, H. Sun, S. Qing, L. Tang, and Z. Chen. Sorcery: Could we make P2P content sharing systems robust to deceivers? In *Proc. of P2P*, 2009.
[29] S. Zhao and V. Lo. Result verification and trust-based scheduling in open peer-to-peer cycle sharing systems. In *Proc. of P2P*, 2005.
[30] R. Zhou and K. Hwang. Gossip-based reputation management for unstructured peer-to-peer networks. *IEEE TKDE*, 2007.
[31] R. Zhou and K. Hwang. Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE TPDS*, 2007.