PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities

Li Xiong and Ling Liu, *Member*, *IEEE Computer Society*

Abstract—Peer-to-peer (P2P) online communities are commonly perceived as an environment offering both opportunities and threats. One way to minimize threats in such communities is to use community-based reputations to help estimate the trustworthiness of peers. This paper presents PeerTrust—a reputation-based trust supporting framework, which includes a coherent adaptive trust model for quantifying and comparing the trustworthiness of peers based on a transaction-based feedback system, and a decentralized implementation of such a model over a structured P2P network. PeerTrust model has two main features. First, we introduce three basic trust parameters and two adaptive factors in computing trustworthiness of peers, namely, feedback a peer receives from other peers, the total number of transactions a peer performs, the credibility of the feedback sources, transaction context factor, and the community context factor. Second, we define a general trust metric to combine these parameters. Other contributions of the paper include strategies used for implementing the trust model in a decentralized P2P environment, evaluation mechanisms to validate the effectiveness and cost of PeerTrust model, and a set of experiments that show the feasibility and benefit of our approach.

Index Terms—Peer-to-peer, trust, reputation mechanisms, data management, security.

1 Introduction

Peer-to-peer (P2P) online communities can be seen as truly distributed computing applications in which peers (members) communicate directly with one another to exchange information, distribute tasks, or execute transactions. They can be implemented either on top of a P2P network [32], [1], [34] or using a conventional client-server platform. Gnutella is an example of P2P communities that are built on top of a P2P platform. Person-to-person online auction sites such as eBay and many business-to-business (B2B) services such as supply-chain-management networks are examples of P2P communities built on top of a client-server architecture. In eCommerce settings, P2P communities are often established dynamically with peers that are unrelated and unknown to each other. Peers have to manage the risk involved with the transactions without prior experience and knowledge about each other's reputation. One way to address this uncertainty problem is to develop strategies for establishing trust and develop systems that can assist peers in assessing the level of trust they should place on an eCommerce transaction. For example, in a buyer-seller market, buyers are vulnerable to risks because of potential incomplete or distorted information provided by sellers. Trust is critical in such electronic markets as it can provide buyers with high expectations of satisfying exchange relationships.

Recognizing the importance of trust in such communities, an immediate question to ask is how to build trust. There is an extensive amount of research focused on building trust for electronic markets through trusted third parties or intermediaries [19], [7]. However, it is not applicable to self-regulating P2P communities where peers are equal in their roles and there are no entities that can serve as trusted third parties or intermediaries. Reputation systems [28] provide a way for building trust through social control by utilizing community-based feedback about past experiences of peers to help making recommendation and judgment on quality and reliability of the transactions. The challenge of building such a reputation-based trust mechanism in a P2P system is how to effectively cope with various malicious behavior of peers such as providing fake or misleading feedback about other peers. Another challenge is how to incorporate various contexts in building trust as they vary in different communities and transactions. Further, the effectiveness of a trust system depends not only on the factors and metrics for building trust, but also on the implementation of the trust model in a P2P system. Most existing reputation mechanisms require a central server for storing and distributing the reputation information. It remains a challenge to build a decentralized P2P trust management system that is efficient, scalable, and secure in both trust computation and trust data storage and dissemination. Last, there is also a need for experimental evaluation methods of a given trust model in terms of the effectiveness and benefits.

With these research problems in mind, we develop PeerTrust, a P2P reputation-based trust supporting framework. The paper has a number of unique contributions. First, by analyzing a variety of common problems encountered in today's online communities (Section 2), we introduce PeerTrust model (Section 3) with five important parameters and a general trust metric combining these parameters for evaluating the trustworthiness of a peer in an evolving P2P community. We also present the trust information dissemination architecture, the usage of the trust model, and the design and implementation considerations of PeerTrust (Section 4). Finally, we describe a series of simulation-based experiments that are carefully designed to evaluate PeerTrust by showing

• *The authors are with the College of Computing, Georgia Institute of Technology, 801 Atlantic Dr., Atlanta, GA 30332.*
E-mail: {lxiong, lingliu}@cc.gatech.edu.

Manuscript received 25 May 2003; revised 10 Oct. 2003; accepted 28 Oct. 2003.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0070-0503.

its accuracy, robustness, cost, and efficiency (Section 5). We conclude the paper with an analysis of PeerTrust in the context of the common problems in P2P systems and online communities (Section 6), a brief overview of the related work (Section 7), a summary, and a description of some future work (Section 8).

2 Application Scenarios and Research Challenges

P2P electronic communities are increasingly gaining acceptance on the Internet as they provide an infrastructure in which the desired information and products can be located and traded while preserving the anonymity of both requestor peers and provider peers. As recent experience with P2P systems such as Gnutella shows, anonymity opens the door to possible misuses and abuses by malicious peers exploiting the overlay network as a way to spread tampered with information, including malicious programs, such as Trojan Horses and viruses. One way to minimize threats in an open community as such is to use community-based reputations, which can be computed through feedback about peers' transaction histories.

Common Problems in Current Electronic Communities. A variety of online community sites have reputation management built in, such as eBay, Amazon, Yahoo!Auction, Edeal, Slashdot, and Entrepreneur.com. From our experience with these sites, and the survey provided in [21], [28], [12], we summarize a list of common problems and risks observed in the current P2P e-commerce communities.

- Most existing reputation systems lack the ability to differentiate dishonest feedback from honest ones and, hence, are vulnerable to malicious manipulations of peers who provide dishonest feedback.
- Most systems provide no support to incorporate various contexts in evaluating the trustworthiness of peers. For example, a peer can develop a good reputation by being honest for numerous small transactions and then tries to make a profit by cheating for large transactions.
- Most systems do not provide incentives for a peer to rate others and suffer from insufficient feedback.
- Most systems cannot deal with strategic dynamic personality of peers. For example, malicious peers can build a reputation and then start cheating or oscillating between building and milking the reputation.

Common Security Threats in P2P Environments. Most of the security threats presented in P2P information sharing environments are due to two main features of the P2P design: *anonymous P2P communication* (for example, Gnutella servants (peers) are anonymous and are only identified by a self-identified servant id) and *variety of the shared information* (e.g., the files authorized to be shared in Gnutella can include all media types, including executable and binary files). The former feature involves a weakness due to the combination of low accountability and low trust of the individual peers. The latter feature combined with the former make the P2P environments more vulnerable to certain security attacks. Below, we list a number of security threats common in distributed systems.

- *Distribution of tampered with information.* The simplest version of this attack is for a peer u to provide a fake resource with the same name as the real resource peer v is looking for. The actual file could be a Trojan Horse program or a virus like the well-known `VBS.Gnutella` worm.
- *Man in the middle attack.* The malicious peer can intercept the message from the provider peer to the requestor and rewrite it with his IP address and port instead of the provider's. Now, the malicious peer can infect the original content from the provider and pass it on to the requestor.
- *Peers are easily compromised.* Peers in an online community with distributed P2P management are more easily compromised. For instance, the well-known `VBS.Gnutella` worm spreads by making a copy of itself in the Gnutella program directory; then it modifies the `Gnutella.ini` file to allow sharing of `.vbs` files in the Gnutella program folder.

In the following sections, we present the design ideas of PeerTrust, a coherent dynamic trust model, the strategies for developing system-level mechanisms to implement the model and a trust-based peer selection scheme, and the risk evaluation with the proposed approach, including how the above-mentioned problems can be avoided or reduced and how potential corrective and preventive methods can be used for recovery and survival.

3 The Trust Model

The main focus of this paper is the design and development of PeerTrust—a dynamic P2P trust model for quantifying and assessing the trustworthiness of peers in P2P e-commerce communities. A unique characteristic of our trust model is the identification of five important factors for evaluating the trustworthiness of a peer in an evolving P2P e-commerce community.

3.1 Trust Parameters

In PeerTrust, a peer's trustworthiness is defined by an evaluation of the peer it receives in providing service to other peers in the past. Such reputation reflects the degree of trust that other peers in the community have on the given peer based on their past experiences. We identify five important factors for such evaluation:

1. the feedback a peer obtains from other peers,
2. the feedback scope, such as the total number of transactions that a peer has with other peers,
3. the credibility factor for the feedback source,
4. the transaction context factor for discriminating mission-critical transactions from less or noncritical ones, and
5. the community context factor for addressing community-related characteristics and vulnerabilities.

We now illustrate the importance of these parameters through a number of example scenarios.

Feedback in Terms of Amount of Satisfaction. Reputation-based systems rely on feedback to evaluate a peer. Feedback in terms of amount of satisfaction a peer receives during a transaction reflects how well this peer has fulfilled

its part of the service agreement. Some existing reputation-based systems use this factor alone and compute a peer u's trust value by a summation of all the feedback u receives through its transactions with other peers in the community. For example, buyers and sellers in eBay can rate each other after each transaction (+1, 0, -1) and the overall reputation is the sum of these ratings over the last six months. We can clearly see that these feedback-only metrics are flawed. A peer who has performed dozens of transactions and cheated one out of every four cases will have a steadily rising reputation in a given time duration whereas a peer who has only performed 10 transactions during the given time duration, but has been completely honest, will be treated as less reputable if the reputation measures of peers are computed by a simple sum of the feedback they receive. Dellarocas [11] concluded that binary reputation mechanisms will not function well and the resulting market outcome will be unfair if judgment is inferred from knowledge of the sum of positive and negative ratings alone.

Number of Transactions. As described above, a peer may increase its trust value by increasing its transaction volume to hide the fact that it frequently misbehaves at a certain rate when a simple summation of feedback is used to model the trustworthiness of peers. The number of transactions is an important scope factor for comparing the feedback in terms of degree of satisfaction among different peers. An updated metric can be defined as the ratio of the total amount of satisfaction peer u receives over the total number of transactions peer u has, i.e., the average amount of satisfaction peer u receives for each transaction. However, this is still not sufficient to measure a peer's trustworthiness. When considering reputation information, we often account for the source of information and context.

Credibility of Feedback. The feedback peer u receives from another peer v during a transaction is simply a statement from v regarding how satisfied v feels about the quality of the information or service provided by u. A peer may make false statements about another peer's service due to jealousy or other types of malicious motives. Consequently, a trustworthy peer may end up getting a large number of false statements and may be evaluated incorrectly because of them even though it provides satisfactory service in every transaction. In PeerTrust, we introduce the credibility of feedback as a basic trust building parameter, which is equally important as the number of transactions and the feedback. The feedback from those peers with higher credibility should be weighted more than those with lower credibility. We have developed two mechanisms for measuring the credibility of a peer in providing feedback. The concrete formulas will be discussed in Section 3.3.

Transaction Context Factor. Transaction context is another important factor when aggregating the feedback from each transaction as transactions may differ from one another. For example, if a community is business savvy, the size of a transaction is an important context that should be incorporated to weight the feedback for that transaction. It can act as a defense against some of the subtle malicious attacks, such as the example we mentioned earlier where a seller develops a good reputation by being honest for small transactions and tries to make a profit by being dishonest for large transactions. It can be seen as a simplified mechanism for more sophisticated risk management in e-Commerce [22]. In addition to using the value of the transaction, the functionality of the transactions is another important transaction context as one might trust another to supply books but not supply medical advice.

Community Context Factor. Community contexts can be used to address some of the community-specific issues and vulnerabilities. One example is to add a reward as a community context for peers who submit feedback. This may, to some extent, alleviate the feedback incentive problem. As another example, if a trust authority or pretrusted peers (e.g., with digital certificate from the community) are available, then incorporating these community-specific context factors into the trust computation can make the trust metric more robust against certain manipulation of malicious peers.

3.2 General Trust Metric

We have discussed the importance of each of the five trust parameters used in PeerTrust. In this section, we formalize these parameters, present a general trust metric that combines these parameters in a coherent scheme, and describe the formula we use to compute the values for each of the parameters given a peer and the community it belongs to.

Given a recent time window, let $I(u, v)$ denote the total number of transactions performed by peer u with v, $I(u)$ denote the total number of transactions performed by peer u with all other peers, $p(u, i)$ denote the other participating peer in peer u's ith transaction, $S(u, i)$ denote the normalized amount of satisfaction peer u receives from $p(u, i)$ in its ith transaction, $Cr(v)$ denote the credibility of the feedback submitted by v, $TF(u, i)$ denote the adaptive transaction context factor for peer u's ith transaction, and $CF(u)$ denote the adaptive community context factor for peer u. The trust value of peer u denoted by $T(u)$, is defined in (1).

$$T(u) = \alpha * \sum_{i=1}^{I(u)} S(u, i) * Cr(p(u, i)) * TF(u, i) + \beta * CF(u), \tag{1}$$

where α and β denote the normalized weight factors for the collective evaluation and the community context factor.

The metric consists of two parts. The first part is a weighted average of amount of satisfaction a peer receives for each transaction. The weight takes into account the credibility of feedback source to counter dishonest feedback, and transaction context to capture the transaction-dependent characteristics. This history-based evaluation can be seen as a prediction for peer u's likelihood of a successful transaction in the future. A confidence value can be computed and associated with the trust metric that may reflect the number of transactions, the standard deviation of the ratings depending on different communities. The second part of the metric adjusts the first part by an increase or decrease of the trust value based on community-specific characteristics and situations. The α and β parameters can be used to assign different weights to the feedback-based evaluation and community context according to different situations. For instance, the α and β values can be assigned properly so the

trust value is set to be either the feedback-based evaluation when the peer has enough transactions and feedback, or a default value otherwise. Important to note is that this general trust metric may have different appearances depending on which of the parameters are turned on and how the parameters and weight factors are set. The design choices depend on characteristics of online communities. We argue that the first three parameters—the feedback, the number of transactions, and the credibility of feedback source are important basic trust parameters that should be considered in computation of a peer's trustworthiness in any P2P communities.

3.3 The Basic Metric

We first consider the basic form of the general metric as shown in (2) by turning off the transaction context factor ($TF(u,i)=1$) and the community context factor ($\alpha=1$ and $\beta=0$). It computes the trust value of a peer u by a weighted average of the amount of satisfaction peer u receives for each transaction.

$$T(u)=\sum_{i=1}^{I(u)} S(u,i)*Cr(p(u,i)). \tag{2}$$

The feedback in terms of amount of satisfaction is collected by a feedback system. PeerTrust uses a transaction-based feedback system, where the feedback is bound to each transaction. The system solicits feedback after each transaction and the two participating peers give feedback about each other based on the transaction. Feedback systems differ with each other in their feedback format. They can use a positive format, a negative format, a numeric rating, or a mixed format. $S(u,i)$ is a normalized amount of satisfaction between 0 and 1 that can be computed based on the feedback.

Both the feedback and the number of transactions are quantitative measures and can be collected automatically. Different from these two, the third parameter—credibility of feedback—is a qualitative measure and needs to be computed based on past behavior of peers who file feedback. Different approaches can be used to determine the credibility factor and compute the credible amount of satisfaction. One way is to solicit separate feedback for feedback themselves. This makes the problem of reputation-based trust management more complex. A simpler approach is to infer or compute the credibility value of a peer implicitly. We propose two such credibility measures in this paper. The first one is to use a function of the trust value of a peer as its credibility factor so feedback from trustworthy peers are considered more credible and, thus, weighted more than those from untrustworthy peers. This solution is based on two assumptions. First, untrustworthy peers are more likely to submit false or misleading feedback in order to hide their own malicious behavior. Second, trustworthy peers are more likely to be honest on the feedback they provide. It is widely recognized that the first assumption is generally true, but the second assumption may not be true at all time. For example, it is possible (though not common) that a peer may maintain a good reputation by performing high quality services, but send malicious feedback to its competitors. In this extreme case, using a function of trust to approximate the credibility of feedback will generate errors. This is because the reputation-based trust in PeerTrust model is established in terms of the quality of service provided by peers, rather than the quality of the feedback filed by peers. We call the basic metric that uses the trust value of a peer recursively as its credibility measure PeerTrust TVM metric and it is defined in (3).

$$T_{TVM}(u)=\sum_{i=1}^{I(u)} S(u,i)*\frac{T(p(u,i))}{\sum_{j=1}^{I(u)} T(p(u,j))}. \tag{3}$$

The second credibility measure is for a peer w to use a personalized similarity measure to rate the credibility of another peer v through w's personalized experience. Concretely, peer w will use a personalized similarity between itself and another peer v to weight the feedback by v on any other peers. Let $IS(v)$ denote the set of peers that have interacted with peer v, the common set of peers that have interacted with both peer v and w, denoted by $IJS(v,w)$, is $IS(v)\cap IS(w)$. To measure the feedback credibility of peer v, peer w computes the feedback similarity between w and v over the common set $IJS(v,w)$ of peers they have interacted with in the past. If we model the feedback by v and the feedback by w over $IJS(v,w)$ as two vectors, the credibility can be defined as the similarity between the two feedback vectors. Particularly, we use the root-mean-square or standard deviation (dissimilarity) of the two feedback vectors to compute the feedback similarity. This notion of local or personalized credibility measure provides great deal of flexibility and stronger predictive value as the feedback from similar raters are given more weight. It may also act as an effective defense against potential malicious collusions. Given the observation that peers in a collusive group give good ratings within the group and bad ratings outside the group, the feedback similarity between a peer v in the collusive group and a peer w outside the group will be low, which will effectively filter out the dishonest feedback by peer v for peer w. We call the basic metric that uses the personalized similarity as the credibility measure PeerTrust PSM metric and it is defined in (4).

$$T_{PSM}(u,w)=\sum_{i=1}^{I(u)} S(u,i)*\frac{Sim(p(u,i),w)}{\sum_{j=1}^{I(u)} Sim(p(u,j),w)}, \tag{4}$$

where

$$Sim(v,w)=1-\sqrt{\frac{\sum_{x\in IJS(v,w)}\left(\frac{\sum_{i=1}^{I(x,v)} S(x,i)}{I(x,v)}-\frac{\sum_{i=1}^{I(x,w)} S(x,i)}{I(x,w)}\right)^2}{|IJS(v,w)|}}. \tag{5}$$

Given that one of the design goals of the PeerTrust model is to emphasize on the roles of different trust parameters in computing trustworthiness of peers, in the rest of the paper, we will use the above two measures as examples and study their effectiveness, benefit, and cost. We believe that the study of what determines the precision of credibility of feedback is by itself an interesting and hard research problem that deserves attention of its own.

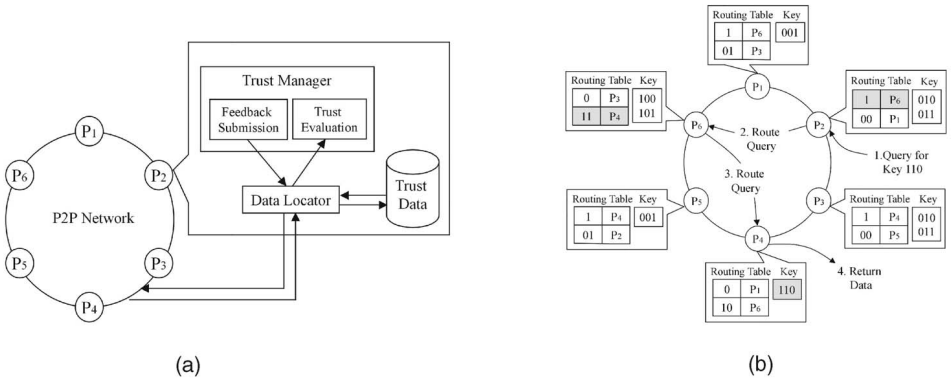


Fig. 1. PeerTrust system architecture. (a) System architecture. (b) Data location.

3.4 Adapting the Trust Metric with Context Factors

We have discussed the motivations and scenarios for incorporating the adaptive context factors into our general trust metric. In this section, we gave two examples of adapting the metric using the transaction and community context factor, respectively.

Incorporating Transaction Contexts by Transaction Context Factor. Various transaction contexts, such as the size, category, or time stamp of the transaction, can be incorporated in the metric so that the feedback for larger, more important, and more recent transactions can be assigned more weight than those for other transactions. For example, an adapted metric that incorporates the size of a transaction i, denoted by $D(u,i)$, is defined in (6).

$$T(u) = \sum_{i=1}^{I(u)} S(u, p(u,i)) * Cr(p(u,i)) * D(u,i) \qquad (6)$$

Providing Incentives to Rate by Community Context Factor. Several remedies have been suggested for the incentive problem of reputation systems [28] such as market-based approaches and policy-based approaches in which users will not receive rating information without paying or providing ratings. Implementing these approaches might stifle the growth of online communities and fledgling electronic markets. In PeerTrust, the incentive problem of reputation systems can be addressed by building incentives or rewards into the metric through community context factor for peers who provide feedback to others. For example, an adapted metric is defined in (7) with a reward as a function of a ratio of total number of feedback peer u give others, denoted as $F(u)$, over the total number of transactions peer u has during the recent time window. The weight factors can be tuned to control the amount of reputation that can be gained by rating others.

$$T(u) = \alpha * \sum_{i=1}^{I(u)} S(u,i) * Cr(p(u,i)) + \beta * \frac{F(u)}{I(u)}. \qquad (7)$$

4 Implementation Strategies

Although the trust model for a P2P community is independent of its implementation, the effectiveness of supporting trust in the community depends not only on the factors and metric for computing trust values, but also on the implementation and usage of the trust model in a P2P system. Typical issues in implementing a P2P trust model such as PeerTrust in a decentralized P2P network include decentralized and secure trust data management, i.e., how to efficiently and securely store and look up trust data that are needed to compute the trust value of a peer, trust metric computation execution, and trust-based peer selection scheme. This section discusses the architecture, algorithm, and design considerations in implementing PeerTrust model in a decentralized P2P system.

4.1 Managing Trust Data: System Architecture

Fig. 1a gives a sketch of the system architecture of PeerTrust. There is no central database. Trust data that are needed to compute the trust measure for peers are stored across the network in a distributed manner. The callout shows that each peer has a trust manager that is responsible for feedback submission and trust evaluation, a small database that stores a portion of the global trust data, and a data locator for placement and location of trust data over the network.

The trust manager performs two main functions. First, it submits feedback to the network through the data locator, which will route the data to appropriate peers for storage. Second, it is responsible for evaluating the trustworthiness of a particular peer. This task is performed in two steps. It first collects trust data about the target peer from the network through the data locator and then computes the trust value. We can see that the trust evaluation is executed in a dynamic and decentralized fashion at each peer. Instead of having a central server that computes each peer's trust value, a peer obtains another peer's trust data from the rest of the peers and computes the trust value of this peer on the fly. This allows peers to get an up-to-date evaluation of the peer by other peers.

Different applications may use different data placement schemes, which determine how and where the data can be inserted, updated, and accessed. A number of P2P file sharing systems have emerged and each has its own data location scheme. Examples include Gnutella [1], which uses broadcast-based schemes and do not guarantee reliable content location, and CAN [27], Chord [32], Pastry [29], as well as P-Grid [5], which use a distributed hash table (DHT) to deterministically map keys into points in a logical coordinate

space and guarantee a definite answer to a query in a bounded number of network hops, typically in the order of $logN$. Depending on the choice of a data location scheme, the implementation of the trust model may be somewhat different. Different schemes may also affect the overhead of the trust data management, but should not affect the effectiveness of the trust metric. In the first prototype of PeerTrust, we use P-Grid primarily because we obtained the P-Grid source code. The trust data about a peer u, i.e., feedback u receives for each transaction are stored at designated peers that are located by hashing a unique ID of peer u to a data key. Each piece of feedback includes the following information: ID of peer u as the data key, timestamp or counter of the transaction, feedback for that transaction, ID of the peer who provides feedback, and other applicable transaction contexts. Each peer is responsible for multiple keys and maintains a routing table for other keys. When a peer receives a search or update request with a data key that it is not responsible for, it forwards the request according to its routing table. So, the storage cost at each peer is proportional to the degree of replication and the amount of history to store. Fig. 1b shows a simple example of a PeerTrust network of six peers constructed using P-Grid.

For such a data location scheme, there is a trust issue associated with it, namely, peers may misbehave by providing false data or random data when responding to a search request. Either majority voting or encryption can be used to address this issue. The data locator can be configured to have multiple replicas responsible for the same key. When a peer u is searching for trust data about another peer v, it finds all the replicas responsible for the key and combines the data using a majority voting scheme. An alternative is to incorporate encryption techniques to enhance the security of the data location scheme, so that peers cannot tamper with the data on the routing path (see Section 4.5 for more detail).

4.2 Trust Computation

The trust evaluation component is responsible for computing the trust measure based on the reputation data that are collected about a peer. We propose two strategies for implementing each of the two basic trust metrics—PeerTrust TVM (3) and PeerTrust PSM (4). One is called *dynamic trust computation* (DTC), which uses fresh trust data collected at runtime to compute the trust value. The other is called *approximate trust computation* (ATC), which uses cache to speed up the trust computation process. We refer to the dynamic and approximate implementations of PeerTrust TVM as TVM/DTC and TVM/ATC and the two implementations for PeerTrust PSM as PSM/DTC and PSM/ATC, respectively. We explain below how each of them is implemented.

Dynamic Computation of PeerTrust TVM—TVM/DTC. Recall PeerTrust TVM metric defined in (3), it is clear that the metric defines a recursive function that uses the trust value of a peer as its feedback credibility measure. Peer w needs to recursively compute other peers' trust values as the credibility factor in order to compute peer u's trust value. We implement the dynamic computation with iterative style. Given a column vector of trust values for N peers, N being the size of the community, the algorithm can simply start with a default trust value vector. As peer w obtains feedback for each peer in the recent time window, denoted as win, it repeatedly computes the trust vector until it converges. When this is done, all trust values of the peers in the network will be available. A sketch of the algorithm is given in Algorithm 1.

Algorithm 1 ComputeTrust_TVM/DTC(u)
Input: u, **Output:** $T(u)$
for $v = 1$ to N **do**
  RetrieveFeedback(v, win)
  $T^0(v) \Leftarrow T_{Default}$
end for
repeat
  for $v = 1$ to N **do**
    $T^{i+1}(v) \Leftarrow$ Compute (3) using T^i
  end for
  $\delta \Leftarrow ||T^{i+1} - T^i||$
until $\delta < \epsilon$

Approximate Computation of PeerTrust TVM—TVM/ATC. It is obvious that dynamic computation is expensive as a peer needs to retrieve the trust data of all peers in the network even when it is only interested in evaluating the trustworthiness of a particular peer or a small subset of peers. Approximate computation provides a more cost-effective algorithm by using a trust cache at each peer. Each peer maintains a trust cache, denoted as $Cache_T$, which keeps the most recent trust values of other peers it has interacted with in the past. Peer w only needs to compute the trust value of peer u when it cannot find u's value in the cache.

When computing the trust value of peer u, instead of dynamically computing the trust value of the peers who have filed feedback about u as the credibility factor, peer w looks for their trust values in the cache. It then uses the cache value in the case of a cache hit and simply uses a default value in a cache miss. Thus, it eliminates the recursive or iterative computation. Once peer w computes the trust value for u, it adds the value to the cache. A sketch of the TVM/ATC algorithm is provided in Algorithm 2.

Algorithm 2 ComputeTrust_TVM/ATC(u)
Input: u, **Output:** $T(u)$
$Feedback \Leftarrow$ RetrieveFeedback(u, win)
for $i = 1$ to Length($Feedback$) **do**
  $p(u,i) \Leftarrow$ feedback source of $Feedback(i)$
  if $Cache_T(p(u,i)) \neq Null$ **then**
    $Cr(p(u,i)) \Leftarrow Cache_T(p(u,i))$
  else
    $Cr(p(u,i)) \Leftarrow T_{default}$
  end if
end for
$T(u) \Leftarrow$ Compute (3)
$Cache_T(u) \Leftarrow T(u)$

Dynamic Computation of PeerTrust PSM—PSM/DTC. Now, we consider PeerTrust PSM metric defined in (4) that uses a personalized similarity measure as the credibility to weight feedback from peers. When a peer w needs to evaluate the trustworthiness of another peer u, peer w needs to compute the personalized feedback similarity between w and every other peer. Thus, it needs to retrieve not only the

feedback about peer u, but also all the feedback that are given by the peers who have had transactions with peer u. In algorithms that implement PeerTrust PSM metric, we have each peer also keep a local copy of the latest feedback given by itself in addition to the feedback about other peers it is responsible for, so the storage cost is slightly more expensive than PeerTrust TVM implementations. The PSM computation is straightforward once the data are collected on demand. The algorithm proceeds as in Algorithm 3.

Algorithm 3 ComputeTrust_PSM/ATC(u) executed at peer w
Input: u, **Output:** $T(u)$
$Feedback \Leftarrow$ RetrieveFeedback(u, win)
for $i = 1$ to Length($Feedback$) **do**
  $p(u,i) \Leftarrow$ feedback source of $Feedback(i)$
  RetrieveFeedbackBy($p(u,i), win$)
  $Cr(p(u,i)) \Leftarrow$ ComputeSimilarity($w, p(u,i)$)
end for
$T(u) \Leftarrow$ Compute (4)

Approximate Computation of PeerTrust PSM—PSM/ATC. Similar to TVM/ATC, PeerTrust PSM/ATC provides a more cost-effective implementation of PSM metric. The main difference between PSM/DTC and PSM/ATC is how the credibility value is collected, dynamically on the fly or from the credibility cache. In PSM/ATC, each peer maintains a trust value cache ($Cache_T$) and a credibility cache ($Cache_{Cr}$) to keep the trust values and credibility values the peer has computed in the past, respectively. In the case of a miss in its trust cache, peer w retrieves the feedback about peer u, looks up the credibility value of the peers who have provided feedback about peer u in its credibility cache, computes the credibility value in case of a miss, adds the credibility value in the credibility cache and, finally, computes the trust value and adds the trust value in its trust cache. Algorithm 4 gives a sketch of the PSM/ATC implementation.

Algorithm 4 ComputeTrust_PSM/ATC(u) executed at peer w
Input: u, **Output:** $T(u)$
$Feedback \Leftarrow$ RetrieveFeedback(u, win)
for $i = 1$ to Length($Feedback$) **do**
  $p(u,i) \Leftarrow$ feedback source of $Feedback(i)$
  if $Cache_{Cr}(p(u,i)) \neq$ Null **then**
    $Cr(p(u,i)) \Leftarrow Cache_{Cr}(p(u,i))$
  else
    RetrieveFeedbackBy($p(u,i), win$)
    $Cr(p(u,i)) \Leftarrow$ ComputeSimilarity($w, p(u,i)$)
    $Cache_{Cr}(p(u,i)) \Leftarrow Cr(p(u,i))$
  end if
end for
$T(u) \Leftarrow$ Compute (4)
$Cache_T(u) \Leftarrow T(u)$

Note that the extra storage cost that is needed in both ATC implementations for caching trust values and credibility values should be negligible as it only caches a single value for each peer. Assuming caching a trust value for one peer takes 1 unit of storage, say, 1 byte, the extra caching cost in order to cache the trust values for all other peers in the network is $N-1$ bytes. So, the peer should be able to have a cache that can hold the trust and credibility values for all other peers in the network in most cases. Otherwise, it can use an LRU-like cache replacement policy to evict the least recently used data items from the cache when the cache is full. The cache should be refreshed periodically so the values reflect the latest behavior of other peers. A straightforward scheme could be to refresh the value after certain number of transactions and more sophisticated scheme can also be used.

4.3 Dealing with Dynamic Personality of Peers

The trust model we have discussed so far uses recent transactions and feedback to compute the trust value of a peer. This is based on the following justification: When a peer's reputation is based on a cumulative average of his lifetime ratings, once that peer has established a solid reputation, incremental ratings play a little role in changing that reputation and, thus, the peer has diminishing incentives to behave honestly. If, however, older ratings can be discounted, then a peer's recent behavior always matters and the peer has continuing incentives to behave honestly [14]. However, more realistic malicious peers may adaptively decide on a strategy in order to maximize their expected payoff given the rules of the game. There are a number of ways in which such peers can attempt to fool the system and obtain higher payoffs. For example, they can build a good reputation and then start cheating occasionally at a rate that gives them a higher payoff, but still allows them to maintain an acceptable reputation. Or, they could oscillate between periods of building and then milking their reputation.

To address such potential dynamic behaviors of peers, we propose a simple adaptive time window-based algorithm to better react to the above behaviors. The basic idea is to adaptively use a smaller time window to reflect the peer's most recent behavior when the peer is dropping its performance over a threshold. Concretely, when peer w is computing the trust value for peer u, it first collects all feedback about u in the recent time window win, and computes a trust value T using one of the four basic trust computation algorithms (TVM/DTC, TVM/ATC, PSM/DTC, and PSM/ATC). In addition, it computes another trust value T_s using a recent subset of the feedback taken by a time window win_s smaller than win. The second value, T_s, will be returned as the final trust value of u if it is smaller than the first value by a certain threshold, which likely indicates the peer is dropping its performance recently. Otherwise, the first value, T, will be returned. A sketch of the adaptive time-window-based computation method is given in Algorithm 5. By choosing a proper regular time window and adaptive window, this method makes the reputation of a peer hard to build, easy to drop, namely, the reputation cannot be quickly increased by a small number of good transactions, but it will quickly drop if the peer starts cheating. Note that the adaptive algorithms are built on top of the four basic algorithms, so dishonest feedback is handled by the respective basic algorithms. Regarding the computation cost, the only additional cost is to compute the second trust value $T_s(u)$ and to run a condition test to determine if the adaptive trust value should be applied, which is minimal because it is using a subset of the trust data and credibility values that are already retrieved and computed in the computation for the first value $T(u)$. In the rest of the paper, we refer to the four basic algorithms as PeerTrust basic

and the adaptive time window-based implementation of the four basic algorithms as PeerTrust adaptive.

Algorithm 5 ComputeTrustAdaptive(u)
Input: u, **Output:** $T(u)$
$Feedback \Leftarrow$ RetrieveFeedback(u, win)
$T(u) \Leftarrow$ Compute trust value using $Feedback$
$T_s(u) \Leftarrow$ Compute trust value using a subset of $Feedback$ taken by win_s
if $T_(u) - T_s(u) > \epsilon$ **then**
$T(u) \Leftarrow T_s(u)$
end if

4.4 Secure Processing and Transmission of Trust Data

There are a number of known security threats due to P2P communication (recall Section 2). We discuss in this section how to guarantee secrecy and integrity of the trust data obtained from other peers and the accountability of the peers providing such trust data.

The unauthorized manipulation of data can happen either in storage or during transmission. We use two layers of techniques, namely, PKI-based scheme and data replication, to increase the security and reliability of the trust data management. The first layer is the PKI-based scheme. We require each peer to have a public and private key pair. Therefore, a peer ID will be either a digest of its public key, obtained using a secure hash function, or the public key itself. For feedback submission, a peer v submits the feedback about peer u, signed with its private key $SK(v)$, along with its public key $PK(v)$. The fact that each piece of feedback is signed with the feedback source's private key guarantees the integrity of the feedback and the authenticity of the feedback origin. Even though peers may tamper with the data that are stored in its local database and provide false or random data when processing a search request later, other peers are able to detect whether the data is corrupted by verifying the signature and discard the data if it is corrupted. When a peer w wishes to evaluate the trustworthiness of peer u, it issues a search request for peer u's trust data, including in the request its public key $PK(w)$. The peer responsible for the data, encrypts its response with w's public key $PK(w)$, signs it with its own private key, and sends the signed encrypted response, together with its public key, to the polling peer. Upon receiving the response, peer w verifies the signature using the received public key and decrypts the message using its own private key $Sk(w)$. It then verifies the signature of each piece of feedback by the public key of the feedback source. The fact that the data are encrypted with the public key of the polling peer w protects their confidentiality. The fact that data are signed with the responding peer's private key allows the detection of integrity violations of the data and the authenticity of their origin. Note that peers will not be able to selectively discard data during routing, as their content, being encrypted with the polling peer's public key, is not visible to them.

With the above scheme, peers can still cause data loss by corrupting the data or selectively discard data that are stored in its local database. To combat this and data loss caused by other issues such as routing anomaly, data replication can be used at the second layer to improve the data availability and reliability. A secure trust computation algorithm proceeds as in Algorithm 6.

Algorithm 6 ComputeTrustSecure(u) executed at peer w
Input: u, **Output:** $T(u)$
for $j = 1$ to r **do**
$response \Leftarrow$ RetrieveFeedbackecure($u, PK(w), win$)
Verify the signature of $response$ using the attached public key
$Feedback \Leftarrow$ Decrypt the data with its private key $SK(w)$
for $i = 1$ to Length($Feedback$) **do**
Verify the signature of $Feedback(i)$
end for
$T_j(u) \Leftarrow$ ComputeTrust($u, metric, computation$)
end for
$T(u) \Leftarrow Median(T_j(u))$

4.5 Trust-Based Peer Selection Scheme

A key objective of the trust-based peer selection scheme is to select one peer or a subset of peers that are most qualified to perform a job in terms of their reputation-based trustworthiness. The trust value produced by the trust metric gives a reputation-based trust measure. It can help peers to form a trust belief or action on other peers and to compare the trustworthiness of a set of peers. A higher value of $T(u)$ indicates that peer u is more trustworthy in terms of the collective evaluation of u by the peers who have had transactions with u and other community context factors.

There are several usages of the trust value in P2P communities. First, a peer w can derive trust relationship with another peer u to determine whether to perform the next transaction or determining its pricing or negotiation strategies with peer u. A decision rule is needed to derive a trust relationship based on the trust value and the situation. Each peer must consider to which degree the value of $T(u)$ with the associated confidence value will make it trust u given a specific situation. A simple rule for peer w to form a trust action on peer u can be $T(u) > T_{threshold}(w)$, where $T_{threshold}(w)$ is the threshold trust value for peer w to trust another peer. The factors that determine the threshold $T_{threshold}(w)$ include how much peer w is willing to trust others, a manifest of dispositional trust [23], the extent to which an entity has a consistent tendency to trust across a broad spectrum of situations and entities. Other factors include the context of the potential transaction. For example, a more expensive transaction may require a higher threshold. More complex decision rules can be applied and are not our focus in this paper. Interested readers may refer to [22] for a number of models that derive a trust relationship from different parameters in an e-Commerce environment.

A second usage is to compare the trustworthiness of a list of peers. For example, in a file sharing community like Gnutella, a peer who issues a file download request can first choose a set of potential peers from those that respond to its request based on their connection speed, etc. Then, it can compare the trustworthiness of the potential peers based on their trust value and choose the peer with the highest trust value to download the file. By doing this, it reduces the risk of downloading inauthentic or corrupted files from untrustworthy peers.

TABLE 1
Simulation Parameters

	Parameter	Description	Default
Community Setting	N	# of peers in the community	128
	k	% of malicious peers in the community	25%
	$mrate$	% of transactions a malicious peer acts malicious	100%
	res	% of peers who respond to a transaction request	5%
Trust Computation	I	# of transactions of peer u in recent time window	100
	I_s	# of transactions of peer u in adaptive window	20
	n_{cache}	# of cache units in ATC implementations	127
	r	# of replicas of underlying DHT structure	4
	$nExp$	# of experiments over which results are averaged	5

5 Experimental Evaluation

We performed initial experiments to evaluate the PeerTrust approach and show its feasibility, effectiveness, and benefits. The first one evaluates effectiveness of PeerTrust in terms of its computation error against malicious behaviors of peers in two settings. The second one demonstrates the benefit of PeerTrust peer selection scheme in the above two settings. The third one evaluates the effectiveness of PeerTrust adaptive algorithm against dynamic personality of peers. Last, we evaluate the runtime overhead of different implementation strategies of the PeerTrust approach.

5.1 Simulation Setup

We implemented a simulator in Mathematica 4.0 and this section describes the general simulation setup, including the community setting, peer behavior pattern, and trust computation.

Community Setting and Behavior Patterns. Our initial simulated community consists of N peers. We have one experiment with varying N to show the scalability of the PeerTrust approach and, otherwise, N is set to be 128. We also ran experiments with different number of peers and it did not show an effect on the effectiveness of trust computation. The game theory research on reputation introduced two types of players [12]. One is commitment type or a long-run player who would always cooperate because cooperation is the action that maximizes the player's lifetime payoffs if the player could credibly commit to an action for the entire duration. In contrast, a strategic type corresponds to an opportunistic player who cheats whenever it is advantageous for him to do. We split peers into these two types in our simulation, namely, good peers and strategic or malicious peers. The percentage of malicious peers is denoted by k. We have one experiment with varying k to show its effect and, otherwise, k is set to be 25 percent.

The behavior pattern for good peers is to always cooperate in transactions and provide honest feedback afterwards. While it is a challenging task to model peers' malicious behavior realistically, we start with two malicious behavior patterns to study the robustness of PeerTrust approach, namely, noncollusive setting and collusive setting. In a noncollusive setting, malicious peers cheat during transactions and give dishonest ratings to other peers, i.e., give bad rating to a peer who cooperates and give good rating to a peer who cheats. A malicious peer may choose to occasionally cooperate in order to confuse other peers and fool the system. We use $mrate$ to model the rate that a malicious peer acts malicious. We have one experiment varying $mrate$ to show its effect on trust computation effectiveness and, otherwise, $mrate$ is set to 100 percent. In a collusive setting, malicious peers act similarly to the noncollusive one and, in addition, they form a collusive group and deterministically help each other by performing numerous fake transactions and give good ratings to each other.

Two transaction settings are simulated, namely, random setting and trusted setting. In a random setting, peers randomly perform transactions with each other. In a trusted setting, peers initiate transactions by issuing a transaction request. For each request, a certain percentage of peers respond. The response percentage is denoted by res and is set to 5 percent in the experiments. The initiating peer then uses the trust-based peer selection algorithm to select the peer with highest trust value to perform the transaction.

Trust Computation. We use a binary feedback system where a peer rates the other peer either 0 or 1 according to whether the transaction is satisfactory. The number of transactions each peer has during the latest time window win, denoted by I, is set to be 100 for all peers. We evaluate the four algorithms—PeerTrust TVM/DTC, TVM/ATC, PSM/DTC, and PSM/ATC, as described in Algorithms 1, 2, 3, and 4, respectively. In the two ATC algorithms, the number of cache units (assuming storing a trust value for one peer takes one unit), denoted by n_{cache}, is set to be $N-1$. We also evaluate PeerTrust adaptive algorithm as described in Algorithm 5. The number of transactions each peer has during the smaller time window win_s, denoted by I_s, is set to be 20. We use P-Grid as the data location scheme to store and retrieve feedback data about peers. The degree of replication, denoted by r, is set to be 4 in the experiments. For comparison purpose, we compare PeerTrust approaches to the conventional approach, referred as Conventional, in which an average of the ratings is used to measure the trustworthiness of a peer without taking into account the credibility factor. All experiment results are averaged over five runs of the experiments. Table 1 summarizes the main parameters related to the community setting and trust computation. The default values for most experiments are listed.

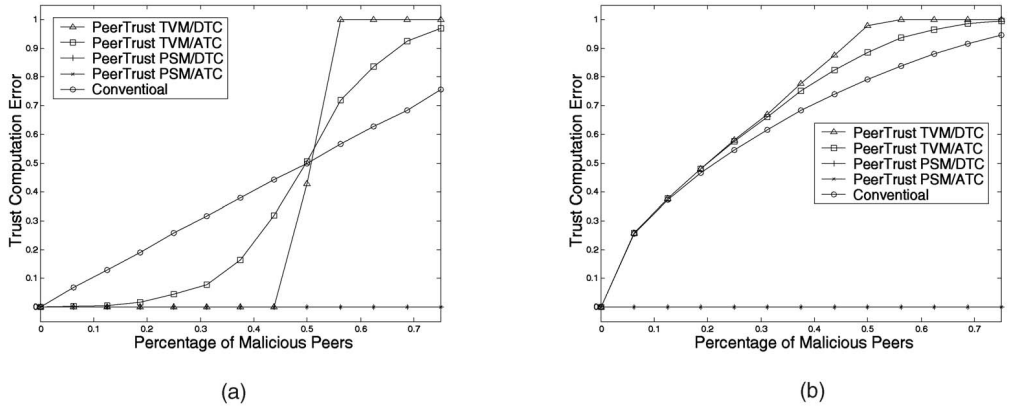


Fig. 2. Trust computation error with respect to percentage of malicious peers. (a) Noncollusive setting. (b) Collusive setting.

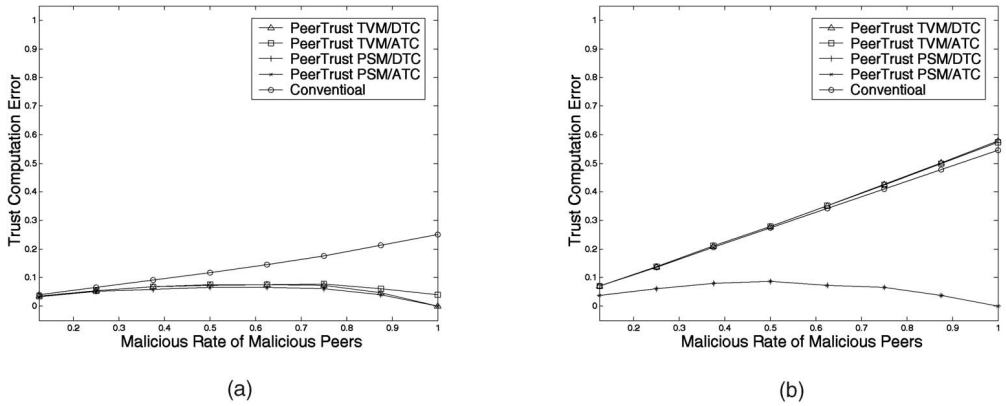


Fig. 3. Trust computation error with respect to malicious rate of malicious peers. (a) Noncollusive setting. (b) Collusive setting.

5.2 Effectiveness against Malicious Behaviors of Peers

The objective of this set of experiments is to evaluate the effectiveness and robustness of the trust model against different malicious behaviors of peers. The experiments start as peers perform random transactions with each other. After 6,400 transactions in the community, i.e., an average of 100 transactions for each peer, a good peer is selected to evaluate the trustworthiness of all other peers. Each experiment is performed under both noncollusive and collusive settings described earlier. We compute the trust computation error as the root-mean-square (RMS) of the computed trust value of all peers and the actual likelihood of peers performing a satisfactory transaction, which is 1 for good peers and $(1 - mrate)$ for malicious peers. A lower RMS indicates better performance.

For the first experiment, we vary the percentage of malicious peers (k) and set the malicious rate to 1 ($mrate = 1$). Fig. 2 represents the trust computation error of different PeerTrust algorithms and the conventional approach with respect to k in the two settings. Consider the noncollusive setting in Fig. 2a first. We can make a number of interesting observations. First, we can see that the performance of the conventional approach drops almost linearly when k increases. Without taking into account the credibility of feedback source, it is very sensitive to malicious peers who provide dishonest feedback. Second, both PeerTrust TVM/DTC and TVM/ATC stay effective when k is less than 50 percent. Using trust values of peers recursively as the weight for their feedback, they are able to filter out dishonest feedback and make correct trust computations. However, the error becomes 100 percent when k is greater than 50 percent, which indicates they completely make wrong evaluations by mistaking good peers as untrustworthy and malicious peers as trustworthy. This is particularly interesting because it shows that malicious peers are able to fool the system by overriding the honest feedback provided by good peers when they are the majority. We can also see that the performance of TVM/ATC and TVM/DTC are reasonably close. Last, both PeerTrust PSM/DTC and PSM/ATC stay effective, even with a large percentage of malicious peers. This confirms that the personalized similarity-based credibility acts as a very effective measure to filter out dishonest feedback. Also note that PSM/DTC and PSM/ATC give the same result when the system is stable. In the collusive setting in Fig. 2b, the interesting observations are both the conventional approach and the PeerTrust TVM approach are extremely sensitive to collusive attempts that dishonestly provide feedback, even when the number of malicious peers is very small. On the other hand, the PeerTrust PSM approach, as we have expected, acts as a very effective defense against collusion by filtering out dishonest feedback from the collusive group.

For the second experiment, we vary the malicious rate ($mrate$) and set the percentage of malicious peers to 25 percent ($k = 25\%$). Fig. 3 represents the trust computation error of different PeerTrust algorithms and the conventional approach with respect to $mrate$ in the two settings. Again, we

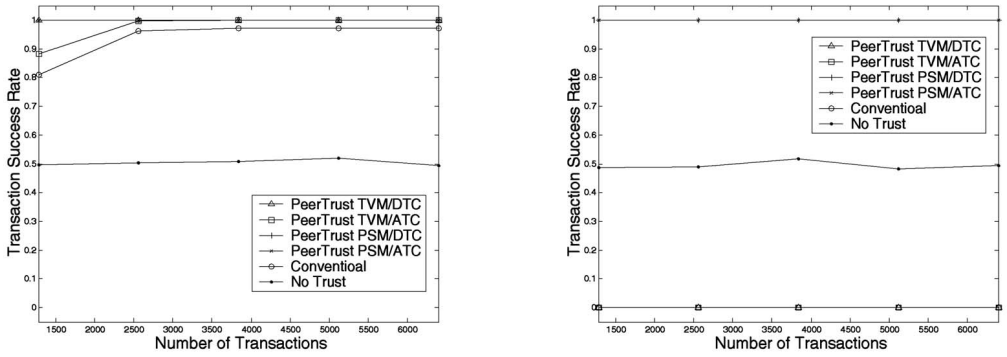


Fig. 4. The benefit of trust-based peer selection scheme. (a) Noncollusive setting. (b) Collusive setting.

can make a number of interesting observations in both settings. First, in the noncollusive setting (Fig. 3a), the performance of the conventional approach drops when $mrate$ increases. Second, both PeerTrust TVM and PSM approaches have a slightly dropped performance when the malicious rate is less than 100 percent. This indicates that peers are able to confuse the system a little when they occasionally cooperate and gives honest feedback. The collusive setting (Fig. 3b) shows similar results, but to a larger extent.

5.3 Benefit of the Trust-Based Peer Selection

This set of experiments demonstrates the benefit of the PeerTrust peer selection scheme in which peers compare the trustworthiness of peers and choose the peer with the highest trust value to interact with. A transaction is considered successful if both of the participating peers cooperate. We define a successful transaction rate as the ratio of the number of successful transactions over the total number of transactions in the community up to a certain time. A community with a higher transaction success rate has a higher productivity and a stronger level of security. The experiment proceeds by repeatedly having randomly selected good peers initiating transactions. In a community that has a trust mechanism, the source peer selects the peer with the highest trust value to perform the transaction. Otherwise, it randomly selects a peer. The two peers then perform the transaction and the transaction succeeds only if the selected peer cooperates. The experiment is performed in both a noncollusive setting and a collusive setting. We show the benefit of the PeerTrust approach and the conventional approach compared to a community without any trust scheme.

Fig. 4 shows the transaction success rate with regard to the number of transactions in the community in the two settings. The graph presents a number of interesting observations. First, in the noncollusive setting (Fig. 4a), we see an obvious gain of the transaction success rate in communities equipped with a trust mechanism. This confirms that supporting trust is an important feature in a P2P community as peers are able to avoid untrustworthy peers. Second, different trust mechanisms have different effectiveness. This shows a similar comparison to the previous experiment. It is worth noting, however, that the conventional approach achieves a transaction success rate close to 100 percent, even though its trust computation error is much higher than 0 shown in Fig. 2a. This is because, even if the computed trust values do not reflect accurately the likelihood of the peers being cooperative, but they do differentiate good peers from bad peers in most cases by the relative ranking. Last, in the collusive setting (Fig. 4b), we can see that the transaction success rate is 0 for the conventional and the PeerTrust TVM approach. This indicates that malicious peers are able to completely fool these trust schemes by collusion and render the system useless, even worse than the system without a trust scheme. However, the system still benefits from PeerTrust PSM approach significantly and shows robustness against the collusion.

5.4 Effectiveness against Dynamic Personality of Peers

So far, we only considered peers with fixed personality in the above two settings. The goal of this experiment is to show how PeerTrust adaptive algorithm works against strategic dynamic personality of peers. Since we showed the effectiveness of the PeerTrust basic algorithms against malicious behaviors of peers in providing dishonest feedback, we focus on the changing behaviors of peers without simulating dishonest feedback in this experiment. We simulated a community with all good peers, but a malicious peer with dynamic personality. We simulated three changing patterns. First, the peer builds a reputation and then starts milking it. Second, the peer is trying to improve its reputation. Third, the peer oscillates between building and milking reputation. The experiment proceeds as peers randomly perform transactions with each other and a good peer is selected to compute the trust value of the malicious peer periodically. We compare PeerTrust adaptive algorithm in Algorithm 5 that uses an adaptive time window to PeerTrust basic approach that uses a fixed time window.

Fig. 5 shows the computed trust value of peer u by PeerTrust adaptive metric and PeerTrust basic metric against different changing patterns. Fig. 5a shows the computed trust value of the peer who is milking its reputation. We can see that, by using a time window-based metric that discounts the old feedback of peers, both PeerTrust basic and PeerTrust adaptive lead to a collapse of the reputation eventually. However, with PeerTrust basic, the peer is able to take

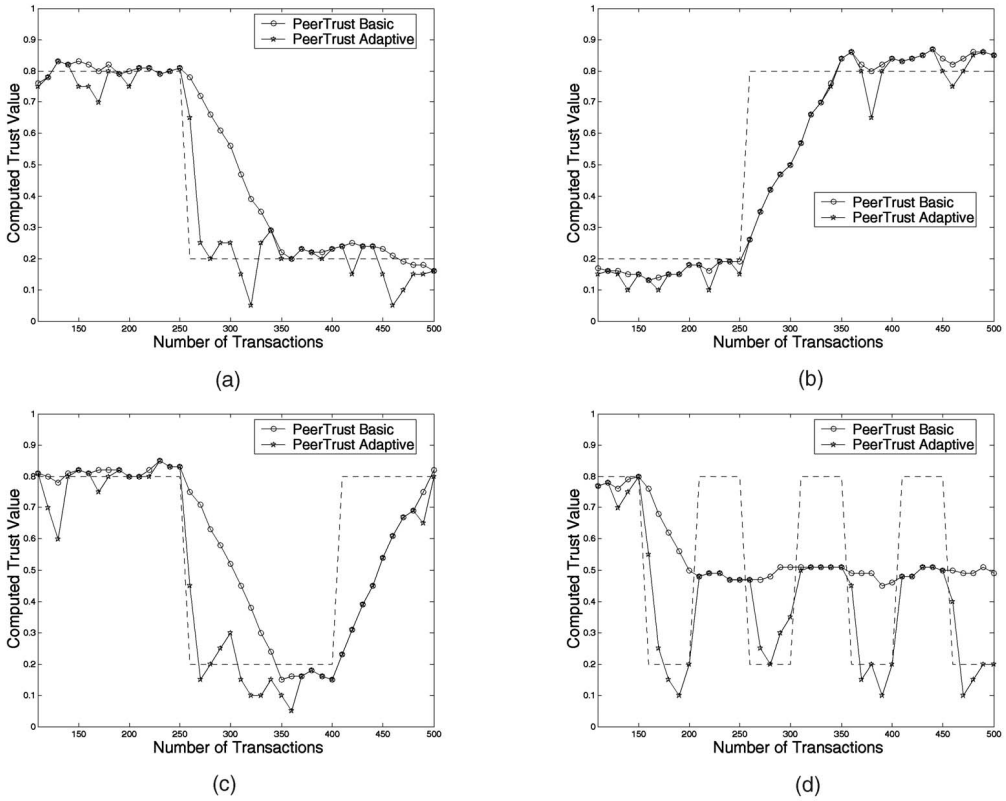


Fig. 5. Effectiveness against dynamic personality and reputation oscillation. (a) Peer milking reputation. (b) Peer building reputation. (c) Peer oscillating reputation. (d) Peer oscillating reputation.

advantage of the transition period by cheating while still maintaining a reasonable reputation before it completely collapses. On the other hand, PeerTrust adaptive is able to correct the trust value much sooner depending on how small the adaptive window is. Fig. 5b shows the computed trust value of the peer who is building its reputation. The peer has to continuously perform good services for a period of time to achieve a good reputation from a bad one in both approaches. Figs. 5c and 5d show the computed trust value of the peer who is oscillating between building and milking its reputation, but with a different frequency. With PeerTrust basic, the peer gains by milking the reputation, but it also has to pay the same amount of cost to build it back. In PeerTrust adaptive, a peer is quickly detected for its bad behavior, but it cannot simply increase its trust value quickly by acting well for a short period so the cost of rebuilding reputation is actually higher than the gain of milking it.

5.5 Trust Evaluation Cost

The objective of this experiment is to understand the runtime overhead of different PeerTrust metrics and implementation strategies and how well it scales. Runtime overhead mainly comes from the cost of retrieving required information for each trust evaluation. It is proportional to two factors—the number of lookups and the cost of each lookup. The number of lookups varies with different trust metrics and implementation strategies. The cost of each lookup is determined by the underlying DHT scheme. As the lookup cost in DHT scheme is usually represented by the number of network hops or number of messages that are required in the routing process, we use the number of network hops for each trust computation as the metric for the runtime cost and compare the cost of different PeerTrust metrics and implementation strategies.

We first compare the cost for the four PeerTrust algorithms with respect to the number of peers. Fig. 6a represents the cost with respect to the number of peers in the P2P community. Note that the graph uses a log-log scale. We analyze the result in detail. First, the two ATC approaches—PeerTrust TVM/ATC and PSM/ATC have the same cost and scales well. Consider peer w computing the trust value of peer u. In these two algorithms, peer w only needs to retrieve the feedback about u and uses the cached values as the credibility value. As feedback about a peer is stored at multiple designated peers using the DHT structure, peer w issues multiple DHT lookups to get the data from all replicas and each DHT lookup takes $O(lgN)$ network hops. So, the cost is in the order of lgN as well. Second, as expected, PeerTrust TVM/DTC requires $O(N)$ network hops for each computation and does not scale well. Third, PeerTrust PSM/DTC requires higher lookup cost than PSM/ATC, but it still scales well. In this algorithm, in addition to the feedback about peer u, peer w also needs to retrieve the feedback that are given by the peers who have given feedback about peer u in order to compute the similarity-based credibility dynamically. As feedback by a peer is also stored locally by the peer itself, peer w issues a direct lookup to every peer who has interacted with peer u. Specifically, since the number of transactions for each peer during the recent time window is set to be 100, so the difference of the cost between PSM/DTC and PSM/ATC,

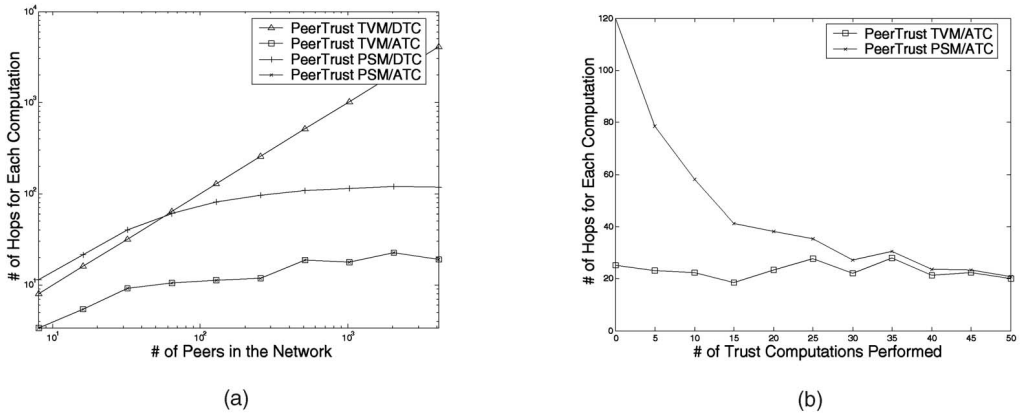


Fig. 6. Trust computation overhead. (a) Trust computation scalability. (b) Cost of cache bootstrapping.

which is the distinctive number of peers who interacted with peer u in recent 100 transaction, is close to 100 in most cases.

We also compared the lookup cost for TVM/ATC and PSM/ATC during the bootstrapping period when the trust cache and the credibility cache are being filled. Fig. 6b represents the cost with respect to the number of trust computations performed. We can see that TVM/ATC does not have extra bootstrapping cost because it uses a default trust value as the credibility factor and the cache is filled as it computes trust values for other peers. On the contrary, PSM/ATC requires extra cost because it has to compute the credibility in the first few trust evaluations to fill the credibility cache. Once the credibility cache is filled, the cost becomes the same as TVM/ATC.

6 Discussion

In this section, we briefly discuss how the proposed approach addresses the common problems of current reputation systems and some of the security threats described in Section 2. We also discuss some risks and threats that cannot be prevented or detected and discuss a set of potential corrective and preventive solutions for recovery and survival.

Risks that are solved by the proposed approach. We presented PeerTrust as a reputation-based trust mechanism for alleviating or resolving some of the security problems in P2P communities by choosing reputable peers and avoid untrustworthy peers. For example, the simplest version of a virus attack or a DOS attack in a file sharing system would be that an adversary responds to a file request with a fake file name and delivers a virus or bogus content to flood the disks. With a reputation-based trust mechanism in place, the peer who receives the malicious content will be able to submit a negative feedback about the malicious peer and help other peers to avoid it in the future.

Not surprisingly, a reputation-based trust mechanism also introduces vulnerabilities and problems by itself. Common attacks are known as shilling attacks where adversaries attack the system by submitting fake or misleading ratings to confuse the system. A shilling attack is often associated with a pseudospoofing attack, where one identity creates multiple pseudonyms to boost each other's ratings, or collusion among peers, where a group of malicious peers collaborate to raise each other's rating and to badmouth other peers. Further, peers can amount attacks on the trust management system by distributing tampered with trust information. PeerTrust tries to minimize such security weaknesses. For example, the use of the credibility factor of the feedback source can be seen as an effective step toward handling fake or misleading ratings in reputation-based feedback. The ability to incorporate various transaction and community contexts can also act against some of the subtle attacks. Furthermore, by combining the proposed trust metric and the secure trust data transmission built on top of mature public key cryptographic algorithms, it prevents *distribution of tampered with trust information* and *man in the middle attack*. An identity is established by a public key that corresponds to a secret private key. Therefore, each identity cannot be spoofed without the knowledge of the corresponding private key. Any content properly signed will not have its integrity or origin compromised.

Risks that will require a deeper investigation. Unfortunately, there is so far no mechanism that can completely prevent the attack of *peers being compromised*. We plan to engage in a study of attacks made via anonymous operations, and develop corrective and preventive methods as a part of trust building and trust management research. Another risk is that peers can easily discard their old identity and adopt a new one through *reentry to get rid of the bad history*. Friedman and Resnick [13] discuss two classes of approaches to this issue: either make it more difficult to change online identities, or structure the community in such a way that exit and reentry with a new identity becomes unprofitable. These are among our future to-do list. Finally, the proposed trust building techniques are based on experiences, therefore, a peer that has been consistently reliable can perform an unavoidable *one-time attack*. Although the proposed trust system utilizes an adaptive time window, it is very hard if not impossible to fully prevent all possible one-time attacks.

7 Related Work

Reputation-based trust research stands at the crossroads of several distinct research communities, most notably computer science, economics, and sociology. We first review general related reputation research in e-Commerce and agent systems, and then review a number of recent works on reputation-based systems in P2P networks.

Reputation and Trust Management in e-Commerce. Dellarocas [12] provides a working survey for research in

game theory and economics on the topic of reputation. Mui et al. [25] also give a review summarizing existing works on reputation across diverse disciplines, including distributed artificial intelligence, economics, and evolutionary biology. The game theory-based research [20], [14], [15] lays the foundation for online reputation systems research and provides interesting insight into the complex behavioral dynamics. Most of the game theoretic models assume that stage game outcomes are publicly observed. Online feedback mechanisms, in contrast, rely on private (pair-wise) and subjective ratings of stage game outcomes. This introduces two important considerations, the incentive for providing feedback and the credibility or the truthfulness of the feedback [12].

A number of reputation systems and mechanisms were proposed for online environments and agent systems. Abdul-Rahman and Hailes [4] proposed a model for supporting trust in virtual communities, based on direct experiences and reputation. They introduced the semantic distance of the ratings. However, there are certain aspects of their model that are ad-hoc, such as the four trust degrees and fixed weightings assigned to the feedback. Pujol et al. [26] applied network flow techniques and proposed a generalized algorithm that extracts the reputation in a general class of social networks. Josang and Ismail [16] and Josang and Tran [17] developed and evaluated the beta reputation system for electronic markets based on β distribution by modeling reputation as posterior probability given a sequence of experiences. Among other things, they showed that a market with limited duration rather than infinite longevity of transaction feedback provides the best condition. Sabater and Sierra [30] proposed Regret system and showed how social network analysis can be used in the reputation system. Sen and Sajja [31] proposed a word-of-mouth reputation algorithm to select service providers. Their focus is on allowing querying agent to select one of the high-performance service providers with a minimum probabilistic guarantee. Yu and Singh [35] developed an approach for social reputation management and their model combines agents' belief ratings using combination schemes similar to certainty factors. The reputation ratings are propagated through neighbors. Zacharia and Maes [36] proposed an approach that is an approximation of game-theoretic models and studied the effects of feedback mechanisms on markets with dynamic pricing using simulation modeling.

A few proposals specifically attempted to address the issue of quality of the feedback. Chen and Singh [8] differentiate the ratings by the reputation of raters that is computed based the majority opinions of the rating. Adversaries who submit dishonest feedback can still gain a good reputation as a rater in their method simply by submitting a large number of feedback and becoming the majority opinion. Dellarocas [10] proposed mechanisms to combat two types of cheating behavior when submitting feedback. The basic idea is to detect and filter out exceptions in certain scenarios using cluster-filtering techniques. The technique can be applied into feedback-based reputation systems to filter out the suspicious ratings before the aggregation. A recent paper by Miller et al. [24] proposes a mechanism, based on budget-balanced payments in exchange for feedback, that provides strict incentives for all agents to tell the truth. This provides yet another approach to the problem of feedback trustworthiness. However, such a mechanism is vulnerable to collusion. The development of effective mechanisms for dealing with collusive manipulations of online reputations systems is currently an active area of research.

Reputation and Trust Management in P2P Systems. There are some recent research on reputation and trust management in P2P systems. Aberer and Despotovic [6] are one of the first in proposing a reputation based management system for P2P systems. However, their trust metric simply summarizes the complaints a peer receives and files and is very sensitive to the skewed distribution of the community and misbehaviors of peers. P2PRep proposed by Cornelli et al. [9] is a P2P protocol where servants can keep track of information about the reputation of other peers and share them with others. Their focus is to provide a protocol complementing existing P2P protocols, as demonstrated on top of Gnutella. However, there are no formalized trust metric and no experimental results in the paper validating their approach. Another work is EigenTrust proposed by Kamvar et al. [18]. Their algorithm again focuses on a Gnutella like P2P file sharing network. They based their approach on the notion of transitive trust and addressed the collusion problem by assuming there are peers in the network that can be pretrusted. While the algorithm showed promising results against a variety of threat models, we argue that the pretrusted peers may not be available in all cases and a more general approach is needed. Another shortcoming of their approach is that the implementation of the algorithm is very complex and requires strong coordination and synchronization of peers.

Our work differs from them in a number of ways. First, we take a coherent approach to analyze the trust problems in communities and identify the important trust parameters in addition to the feedback in order to effectively evaluate the trustworthiness of peers and to address various malicious behaviors in a P2P community. We address the problems of dishonest feedback and lack of incentives by building the credibility factor and context factors into the metric. Second, we also emphasize on how to implement the solution in the P2P network in an efficient and secure manner and present detailed algorithms and experimental evaluation of our approach in a distributed P2P environment.

8 Conclusion

We have presented PeerTrust—a reputation-based trust supporting framework, which includes a coherent adaptive trust model for quantifying and comparing the trustworthiness of peers based on a transaction-based feedback system, and a decentralized implementation of such model over a structured P2P overlay network. We reported initial simulation-based experiments, demonstrating the feasibility, effectiveness, and benefits of our approach.

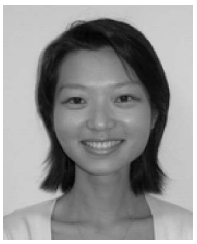
Our research on PeerTrust continues along several directions. First, we are investigating different threat models of P2P online communities and exploring mechanisms to make PeerTrust model more robust against malicious behaviors. Second, we are working toward implementing PeerTrust in a distributed and secure manner and incorporating PeerTrust into two P2P applications that are currently under development in Georgia Institute of Technology, namely, PeerCQ [3] and HyperBee [2].

ACKNOWLEDGMENTS

This research is supported partially by a US National Science Foundation ITR grant. The second author would like to acknowledge the partial support from grants funded by US National Science Foundation CCR grant, DOE SciDAC, and Defense Advanced Research Projects Agency ITO. The authors would like to thank Karl Aberer and Zoran Despotovic for the source code of P-Grid. They would also like to thank the guest editor and the reviewers of this paper for their detailed and insightful comments. A preliminary version of this paper appeared in the Proceedings of 2003 IEEE Conference on E-Commerce (CEC '03) [33].

REFERENCES

[1] Gnutella, http://www.gnutella.com, 2000.
[2] HyperBee, http://www.hyperbee.com, 2001.
[3] PeerCQ, http://disl.cc.gatech.edu/PeerCQ, 2004.
[4] A. Abdul-Rahman and S. Hailes, "Supporting Trust in Virtual Communities," *Proc. 33rd Ann. Hawaii Int'l Conf. System Sciences (HICSS-33),* 2000.
[5] K. Aberer, "P-Grid: A Self-Organizing Access Structure for P2P Information Systems," *Proc. Ninth Int'l Conf. Cooperative Information Systems,* 2001.
[6] K. Aberer and Z. Despotovic, "Managing Trust in a Peer-to-Peer Information System," *Proc. ACM Conf. Information and Knowledge Management (CIKM),* 2001.
[7] Y. Atif, "Building Trust in E-Commerce," *IEEE Internet Computing,* vol. 6, no. 1, 2002.
[8] M. Chen and J.P. Singh, "Computing and Using Reputations for Internet Ratings," *Proc. Third ACM Conf. Electronic Commerce,* 2001.
[9] F. Cornelli, E. Damiani, S.D.C. di Vimercati, S. Paraboschi, and P. Samarati, "Choosing Reputable Servents in a P2P Network," *Proc. 11th Int'l World Wide Web Conf.,* 2002.
[10] C. Dellarocas, "Immunizing Online Reputation Reporting Systems against Unfair Ratings and Discriminatory Behavior," *Proc. Second ACM Conf. Electronic Commerce,* 2000.
[11] C. Dellarocas, "Analyzing the Economic Efficiency of Ebay-Like Online Reputation Reporting Mechanisms," *Proc. Third ACM Conf. Electronic Commerce,* 2001.
[12] C. Dellarocas, "The Digitization of Word-of-Mouth: Promise and Challenges of Online Reputation Mechanism," *Management Science,* vol. 49, no. 10, 2003.
[13] E. Friedman and P. Resnick, "The Social Cost of Cheap Pseudonyms," *J. Economics and Management Strategy,* vol. 10, no. 1, 2001.
[14] B. Holmstrom, "Managerial Incentive Problems: A Dynamic Perspective," *Rev. Economic Studies,* vol. 66, no. 1, 1999.
[15] B.A. Huberman and F. Wu, *The Dynamics of Reputation.* 2002.
[16] A. Josang and R. Ismail, "The Beta Reputation System," *Proc. 15th Bled Conf. Electronic Commerce,* 2002.
[17] A. Josang and N. Tran, "Simulating the Effect of Reputation Systems on E-Markets," *Proc. First Int'l Conf. Trust Management,* 2003.
[18] S. Kamvar, M. Scholsser, and H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks," *Proc. 12th Int'l World Wide Web Conf.,* 2003.
[19] S. Ketchpel and H. Garcia-Molina, "Making Trust Explicit in Distributed Commerce Transactions," *Proc. 16th Int'l Conf. Distributed Computing Systems,* 1996.
[20] D. Kreps and R. Wilson, "Reputation and Imperfect Information," *J. Economic Theory,* vol. 27, 1982.
[21] R.A. Malaga, "Web-Based Reputation Management Systems: Problems and Suggested Solutions," *Electronic Commerce Research,* vol. 1, no. 4, 2001.
[22] D.W. Manchala, "E-Commerce Trust Metrics and Models," *IEEE Internet Computing,* vol. 4, no. 2, 2000.
[23] D.H. McKnight and N.L. Chervany, "The Meanings of Trust," Technical Report WP9604, Univ. of Minnesota Management Information Systems Research Center, 1996.
[24] N.H. Miller, P. Resnick, and R.J. Zeckhauser, "Eliciting Honest Feedback in Electronic Markets," KSG Working Paper Series RWP02-039, 2002.
[25] L. Mui, M. Mohtashemi, and A. Halberstadt, "Notions of Reputation in Multi-Agent Systems: A Review," *Proc. First Int'l Joint Conf. Autonomous Agents and Multiagent Systems,* 2002.
[26] J.M. Pujol, R. Sanguesa, and J. Delgado, "Extracting Reputation in Multi-Agent Systems by Means of Social Network Topology," *Proc. First Int'l Joint Conf. Autonomous Agents and Multiagent Systems,* 2002.
[27] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content Addressable Network," *Proc. ACM SIGCOMM,* 2001.
[28] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara, "Reputation Systems," *Comm. ACM,* vol. 43, no. 12, 2000.
[29] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Middleware,* 2001.
[30] J. Sabater and C. Sierra, "Reputation and Social Network Analysis in Multi-Agent Systems," *Proc. First Int'l Joint Conf. Autonomous Agents and Multiagent Systems,* 2002.
[31] S. Sen and N. Sajja, "Robustness of Reputation-Based Trust: Boolean Case," *Proc. First Int'l Joint Conf. Autonomous Agents and Multiagent Systems,* 2002.
[32] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM,* 2001.
[33] L. Xiong and L. Liu, "A Reputation-Based Trust Model for Peer-to-Peer Ecommerce Communities," *Proc. IEEE Conf. E-Commerce (CEC '03),* 2003.
[34] J.E. Youll, "Peer to Peer Transactions in Agent-Mediated Electronic Commerce," Master's thesis, Massachusetts Inst. of Technology, 2001.
[35] B. Yu and M.P. Singh, "A Social Mechanism of Reputation Management in Electronic Communities," *Proc. Seventh Int'l Conf. Cooperative Information Agents,* 2000.
[36] G. Zacharia and P. Maes, "Trust Management through Reputation Mechanisms," *Applied Artificial Intelligence,* vol. 14, no. 8, 2000.



Li Xiong is a PhD candidate in the College of Computing at the Georgia Institute of Technology. Her research interests are in Internet data management, distributed computing, and Internet security. Previously, she received the BS and MS degrees in computer science from the University of Science and Technology of China and Johns Hopkins University, respectively, and worked as a software engineer after that for several years with companies including Internet Security Systems.

Ling Liu is currently an associate professor at the College of Computing at Georgia Institute of Technology. Her research involves both experimental and theoretical study of distributed systems, in general, and distributed data intensive systems, in particular, including distributed middleware systems, advanced Internet systems, and Internet data management. Her current research interests include performance, scalability, reliability, and security of Internet services, pervasive computing applications, as well as mobile and wireless services. Her research group has produced a number of software systems that are either open sources or directly accessible online, including WebCQ, XWRAPElite, Omini, and PeerCQ. Dr. Liu has published more than 100 articles in international journals and conferences. She is currently on the editorial board of *International Journal of Very Large Database Systems* (VLDBJ), *International Journal of Web Services Research*, *International Journal of Grid and Utility Computing*, editor-in-chief of ACM SIGMOD Record, a vice PC chair of IEEE International Conference on Data Engineering (ICDE 2004), a co-PC chair of 2004 IEEE International Conference on Web Services. She was a PC cochair of the 2001 International Conference on Knowledge and Information Management (CIKM 2001), and a PC cochair of the 2002 International Conference on Ontology's, DataBases, and Applications of Semantics for Large Scale Information Systems (ODBASE). Her current research is partially funded by grants from the US National Science Foundation, Defense Advanced Research Projects Agency, Department of Energy, and IBM. She is a member of the IEEE Computer Society.