PeerTrust: A Trust Mechanism for an Open Peer-to-Peer Information System

Li Xiong and Ling Liu
College of Computing
Georgia Inst. of Technology
{lxiong,lingliu}@cc.gatech.edu

Abstract

In an open peer-to-peer information system, peers often have to interact with unknown or unfamiliar peers and need to manage the risk that is involved with the interactions without any presence of trusted third parties or trust authorities. It is important for peers to be able to reason about trust when interacting with each other to accomplish a task. This paper presents PeerTrust, a simple and yet effective trust mechanism for quantifying and comparing the trustworthiness of peers. We argue that the amount of satisfaction a peer obtains through interactions, the total number of interactions that a peer has with other peers, and the balancing factor of trust all play a crucial role in evaluating the trustworthiness of the peer. This paper also discusses the architecture and the design considerations in implementing this mechanism in a decentralized peer-to-peer system. We report the set of initial experiments, showing the feasibility, the cost, and the benefit of our approach.

1 Introduction

The interest in peer-to-peer (P2P) computing continues to grow since the rise and fall of Napster. Some popular systems that are currently in operation include SETI@home [7] and Gnutella [4]. A number of research projects have engaged in this area, among which representative ones include Freenet [3] and

Free Haven Project [2]. Most research on peer-to-peer computing to date has been focused on efficient resource placement, resource location, and load balancing. Very few have addressed the need of trust in P2P systems or have incorporated trust into their resource placement and allocation algorithms.

Why is trust important in P2P systems?

An open peer-to-peer system is a totally decentralized P2P system where peers can join or leave the system at any time and there is no central directory to maintain global knowledge of peers. In an open P2P information system, peers often have to interact with unknown or unfamiliar peers and need to manage the risk that is involved with the interactions without any presence of trusted third parties or trust authorities. Trust is an important measure that a peer can use to assess the risk involved when interacting with other peers and to accept or reject a peer's participation based on its trustworthiness for a critical task.

More concretely, in a P2P system, a task is often performed by a subset of peers chosen by the underlying service partition and service lookup schemes. In a centralized scheme like Napster, the selection of peers to perform a task (e.g. from which peer to download a song file) is determined based on the global knowledge about which peer is able to perform which types of tasks. In a decentralized scheme like Gnutella, a service request (e.g. the request for downloading a song) is first broadcasted to other peers to check out who can serve the request. The peers who are online and capable of serving the request will respond. Of course, if more than one peer respond to the broadcast message, a mechanism is needed to choose a peer to perform the task.

Many criteria can be used as guidelines for service partition and service lookup, including network proximity, computing capacity and trust of peers, and application-specific characteristics. Among these, trust is an essential factor in an open system where peers have to interact with other unfamiliar peers in accomplishing a task.

How do we assess trust?

Although trust is a long researched topic both in sociology and computer security, trust in P2P systems presents its own challenges due to the unique characteristics of such systems. First, most existing trust models in computer security focus on how to compute indirect trust given the input of direct trust valuations. Establishing and assessing trust is considered subjective and is outside the scope of these models [19]. However, P2P systems cannot rely on subjective trust valuations and need technical mechanisms for computing trust directly and revising the trustworthiness of a peer at runtime based on the feedbacks from other peers. Second, most existing trust production models proposed for agent systems and online communities require either a central trusted server or the presence of a trusted third party or authority. Hence they are not suitable for a decentralized environment. Third, in an open P2P system, peers come and leave freely. It is hard to uniquely identify peers on a permanent basis. Peers can misbehave in a number of ways, such as serving corrupted or low-quality information or providing false feedback on other peers such as filing false complaint on other peers. Therefore, a trust mechanism has to be robust against the misuse behavior of the system. From a risk assessment perspective, measurement of trust can be regarded as obtaining quantitative bounds on risk of adverse consequences. Higher levels of trust correspond to lower levels of risk. In short, there is a need for a computational trust mechanism with a reliable trust metric for supporting trust in a decentralized P2P environment.

How do we implement a trust model?
The effectiveness of a trust model depends not only on the factors for risk assessment but also on the factors that affect the performance of a P2P system that supports trust. A scalable trust management requires efficient storage of trust data (both the data that are used to assess and compute trust and the trust values of peers) in a decentralized P2P network, and fast lookup of trust data when assessing the risk of a peer in performing a task. Furthermore, there is a need of methods for experimental evaluation of a given trust model in terms of benefits, costs, and execution performance. Most traditional trust models only give an analytical model without any experimental validation due to the subjective nature of trust. Few discuss the implementation issues of a trust mechanism in a distributed environment. There is also a lack of general metrics for evaluating the effectiveness of trust mechanisms.

With these questions in mind, we present PeerTrust, a simple and yet effective trust model for quantifying and assessing the trustworthiness of peers. The PeerTrust model has two unique features. First, we identify three important factors for evaluating the trustworthiness of a peer: the amount of satisfaction a peer obtains through interactions, the total number of interactions that a peer has with other peers, and the balancing factor of trust. We argue that all three play a crucial role in computing trust of peers. In contrast, many existing trust models, when applied to evaluating the trust of a peer, only take into account the first factor – the amount of satisfaction that others peers have over the given peer. Second, we introduce a general trust function that computes the trust of a peer by combining these three parameters. A complaint-based trust function is presented as a concrete instance of our general trust function to illustrate the importance of the three trust assessment factors, and demonstrate the effectiveness and robustness of our approach. We also discuss the architecture and design considerations in implementing the PeerTrust mechanism in a decentralized P2P system. We report the results of our initial experiments, showing the accuracy, robustness, cost, and efficiency of our approach.

The rest of the paper is organized as follows. We introduce the trust model including the trust parameters and the trust metric in Section 2. We discuss the trust management issues in implementing our trust model in Section 3. Section 4 reports our initial experimental results. We conclude the paper with a brief overview of related work in Section 5 and a discussion of our future work in Section 6.

2 The Trust Model

In this section, we first present the trust definitions we use in the context of PeerTrust, followed by a discussion on the three trust parameters we introduce and their roles in evaluating the trustworthiness of a peer in a decentralized P2P system. We describe the trust metric including the general form and a concrete form. The concrete trust function is intended to illustrate the usefulness and the benefits of the general trust function.

2.1 Trust Definitions

Trust is a fundamental concept in computer security yet remains ambiguous. There are numerous notions of trust and different kinds of trust often satisfy different properties and can be established differently. In general, trust is defined in terms of trusting belief and trusting behavior [15]. Trusting belief between two peers is the extent to which a peer believes that another peer is trustworthy in this situation. Trustworthy means one is willing and able to act in the other entity's best interests. Trusting behavior between two peers is the extent to which a peer depends on another peer in a given situation with a feeling of relative security, even though negative consequences are possible. If a trusting belief means "a peer A trusts that a peer B is trustworthy", then the corresponding trusting behavior means "A trusts B". Trusting belief usually leads to trusting behavior.

In the PeerTrust model, the trust relationship we consider is the trusting belief between two peers. This definition simply says that if a peer is consistently (predictably) proven to be willing (benevolent) and able (competent) to service the trustor's interest in an (honest) manner, then this peer is worthy of trust. More concretely, when a peer A trusts that a peer B is trustworthy, it means that based on the past experiences A trusts that B will do what the P2P interaction protocol says that each participant should do without lying.

An immediate question one would ask now is how to model the consistent proof that a peer is willing and able to service the other peers in an honest manner. We address this question in the following subsections.

2.2 Trust Parameters

In a P2P system, peers are clients and servers at the same time. The peer relationship are established in pair wise interactions during which certain information or service is exchanged. A peer's trust in other peers is the peer's degree of belief that other peers are willing and able to fulfill their part of the service agreement during the interactions. To help peers reason about the trustworthiness of a peer, the problem is essentially to determine a trust metric, i.e. a function that computes a trust value from information that is relevant to the trust decision a source peer is going to make. Such a trust value reflects to what extent a target peer is willing and able to fulfill the service. Once the trust metric yields a trust value for a target peer, the source peer can use the trust value to make a trust decision, such as to decide whether the trust value is sufficient for it to trust the target peer, or to compare the peer with other peers.

In order to determine a trust metric, we need to determine the trust parameters that can be used to assess the trust first. In PeerTrust, a peer's trustworthiness is defined by an evaluation of the peer in terms of the degree of satisfaction it receives in providing service to other peers in the past. Such degree of satisfaction reflects to what extent the peer acts in other peers' best interests. We identify three important factors for such evaluation. They are the amount of satisfaction a peer obtains through interactions, the number of interactions the peer has with other peers, and a balance factor of trust. We argue that all these three factors play an equally important role in evaluating the trustworthiness of a peer.

Amount of Satisfaction
In a P2P network, the amount of satisfaction a peer receives regarding its service is usually resulting from the interactions other peers have had with this peer. Intuitively, during an interaction, the better a peer fulfills its part of the service agreement, the more satisfaction it will receive from the other peer. The larger the number of peers who are satisfied with a peer's service, the more it should be worthy of trust.

Number of Interactions
The total number of interactions a peer has with other peers in the P2P network is another important factor that affects the accuracy of feedback-based trust computation. It reflects over how many services the satisfaction is obtained. Put differently, the amount of satisfaction alone presents the feedbacks without a specific context. The number of interactions from which the feedbacks were derived is an important fairness measure with respect to the amount of satisfaction.

Scenario 1: Consider a simple scenario where two peers obtain the same amount of satisfaction over the same time period but through different number of interactions. A peer A performs 10 interactions (services) with other peers up to time t and obtains satisfaction for each service it performs. Another peer B performs 100 interactions but only obtains satisfaction one out of ten services it performs. The two peers receive the same amount of satisfaction through the services they perform, but it is obvious that peer A is more worthy of trust than peer B in terms of the overall service quality. This example simply shows that the amount of satisfaction will not be a fair measure without using the total number of interactions as the context.

Scenario 2: In a P2P system that relies on a positive measure of feedback, a peer can continue increasing its trust value by increasing its interaction volume to hide the fact that it frequently misbehaves at a certain rate. In contrast, in a system that relies on a negative measure of feedback such as complaints, a peer can reduce the number of complaints it receives by decreasing its interaction volume even though it frequently misbehaves at a certain rate. Consider a simple example in a system that relies on a positive measure of feedback. A peer A performs 50 interactions with other peers up to time t and obtains satisfaction one out of two services it performs. Another peer B performs 100 interactions with other peers up to time t and obtains satisfaction also one out of two services it performs. By increasing the number of interactions, peer B obtains more satisfaction than peer A even though peer B may not be more trustworthy than peer A as they both misbehave one out of two services equally. Such misbehaviors are not captured fairly when only the amount of satisfaction is considered for computing trust.

Balance factor of trust
The amount of satisfaction is simply a statement from a peer about another peer as to how much it is satisfied about the other peer's service and is often collected from peers through a feedback system. Given that a peer may make false statements about another peer's service, the balance factor of trust is used to offset the potential of false feedback of peers and thus can be seen as a way to differentiate the credible amounts of satisfaction from the less credible ones.

Scenario 3: A peer happens to interact with only malicious peers, such as a peer who falsely claims that another peer provides poor service to badmouth the other peer or falsely claims another peer provides good service to boost the other peer's rating. In this scenario, a trustworthy peer may end up getting a large number of false statements. Without a balance factor of trust built in, this peer will be evaluated incorrectly because of false statements even though it provides satisfactory service in every interaction.

Traditional security approaches such as digital certificate and encryption can be used to assure the information is originated from the right party and not tampered during transportation. But they cannot answer the question of whether the party can be trusted. The role of a balance factor is critical in computing trust based on feedbacks in the situations where false statements are not impossible.

Surprisingly, most existing trust mechanisms only takes into account the first factor, the amount of satisfaction that other peers have had over a given peer, when computing the trust value of this peer. Furthermore few have even considered the role of a balancing factor of trust in their trust computation models.

2.3 Trust Metric

In this section we define a general trust metric that computes a trust measure by combining the three trust factors identified in the previous section. The goal of such a metric is to produce a fair measure of trust belief in the presence of unknown or unfamiliar peers and possible misbehaviors of such peers.

Let P denote the set of N peers in the peer network, and $u, v \in P$ be peers in the network P. Note that at any given time, the number of active peers may be different, and not decidable in advance. Let $I(u, v, t)$ denote the number of interactions that peer u has with v up to time t. Let $S(u, v, t)$ denote the amount of satisfaction peer u has with v up to time t. Let $Cr(v, t)$ be the balance factor of trust that offsets the risk of non-credible feedbacks from v. The evaluation of peer u's service up to time t by the rest of peers in the peer network P, denoted by $T(u, t)$, can be defined as a function of $I(u, v, t)$, $S(u, v, t)$, and $Cr(v, t)$. A simple and straightforward definition of $T(u, t)$ could be the ratio of the summary of a balanced amount of satisfaction other peers have over u and the total number of interactions u has with other peers in the network P. That is:

$$T(u,t) = \frac{\sum_{v \in P, v \neq u} S(u,v,t) * Cr(v,t)}{\sum_{v \in P, v \neq u} I(u,v,t)} \qquad (1)$$

In the equation 1, a higher value of $T(u, t)$ indicates that the peer u is more trustworthy in terms of the collective evaluation of u by the peers who have had interactions with u. It is important to note that the value of $T(u, t)$ alone neither represents any peer's trust in u nor implies any trusting behavior between any other peer and u, but rather gives a measure that helps other peers to form a trust belief (opinion) about u. Each peer must consider to which degree the value of $T(u, t)$ will make it trust u. Different peers may have different perception over the same value of $T(u, t)$.

A simple decision rule can be conducted at peer w as follows: If $T(u, t) > c_w$, then u is trustworthy. This simply says that w believes u is trustworthy if the collective evaluation of u is above a specific threshold defined by w. The threshold indicates how much the observing peer is willing to trust others. A more tolerant peer may have a lower threshold. It is a manifest of dispositional trust [15], which is the extent to which an entity has a consistent tendency to trust across a broad spectrum of situations and entities. A multi-level trust decision rule can be also defined by introducing multiple thresholds.

In addition to view the value of $T(u,t)$ as a trust measure of u, one can also view the value of $T(u,t)$ as a global reputation measure of the peer u. In real life, reputation is the general estimation that a person is held by the public. It is often established through a good tractable history so that their behavior can be predicted reliably. A person consistently provides good service to others will in turn establish a good reputation. We are usually more willing to trust a person with a higher reputation. Similarly, the trust value can be seen as the peer's reputation, i.e. a general estimation the peer is held by the other peers in the network.

2.4 An Example Metric

Now we have a general trust metric to evaluate the trustworthiness of peers in a P2P network. The next question one may ask immediately is how to measure the amount of satisfaction according to the service-level agreements. Different systems may use different measures and the metric may be adapted into different forms. We present a concrete form of our general metric in this section and will use it throughout the rest of the paper to demonstrate the feasibility, the benefit, and the cost of our metric.

Different feedback systems differ from each other in terms of the feedback mechanism and the feedback measure. In the first implementation of PeerTrust, we choose to use an interaction based complaint system.

With such a feedback system, if a peer receives a complaint from another peer after an interaction, it simply means the amount of satisfaction it gets through this interaction is 0. Otherwise, it gets 1. Let $C(u,v,t)$ denote the number of complaints a peer u receives from v up to time t. The amount of satisfaction $S(u,v,t)$ can be simply measured in terms of the number of interactions for which peer u does not receive a complaint. That is $I(u,v,t) - C(u,v,t)$.

Now let us consider the balancing factor of trust. It is obvious that peers cannot be blindly trusted for their feedbacks. Intuitively, a less trustworthy peer is more likely to file fake complaints to hide their own misbehavior and badmouth other peers. To make this problem tractable, in the first prototype of PeerTrust, we apply a simplifying assumption. We assume peers are rational in a game theoretic sense, i.e. trustworthy peers do not file fake complaints and untrustworthy peers file fake complaints when they misbehave during an interaction. Based on this assumption, complaints from a trustworthy peer can be trusted while complaints from an untrustworthy peer cannot be trusted. Therefore, it is reasonable to use the trust measure of a peer obtained from the collective evaluation of this peer in the network as the balance factor to offset the risk of its non-credible complaints. Thus, the balanced amount of satisfaction $S(u,v,t) * Cr(v,t)$ can be measured as $I(u,v,t) - C(u,v,t) * T(v,t)$ where $C(u,v,t) * T(v,t)$ denotes the credible complaints filed by v. The complaint-based trust metric can be defined as follows:

$$T(u,t) = 1 - \frac{\sum_{v \in P, v \neq u} C(u,v,t) * T(v,t)}{\sum_{v \in P, v \neq u} I(u,v,t)} \tag{2}$$

Note that the trust metric given above gives a value between 0 and 1. It determines how many complaints out of the total complaints a peer files against other peers will be counted. For example, complaints from a complete untrustworthy peer, i.e. who misbehaves during each interaction, will not be counted at all.

2.5 Discussions

We have discussed the trust metric in its general form and presented a concrete trust metric based on complaints and interactions. This section answers a number of questions related to the concrete trust metric.

The first question is related to setting up a default trust value. More concretely, what is the trust value we can assign to a peer that does not have any interaction history, e.g. a new peer just joins the network? One solution is to assign the default trust measure to be 1 for all new peers upon their entrance to the system, assuming that most of the peers are trustworthy. An obvious drawback of this setting is the possibility for peers to raise their trust measure by frequently leaving the system and re-entering it as a new peer. An alternative solution is to give the peers a default trust value that is fairly low, ideally lower than all the peers in the network, so the peers will have an incentive to keep their trust measure and keep on improving it instead of leaving the system and registering again to get a higher default trust value. Therefore, it might be desirable in some situations to set the default trust (reputation) measure of a new peer to be 0. This also conforms to real world experiences where a new manufacturer or store does not have any reputation and it has to build its reputation through continuously good services.

The second question is related to binding of complaints (feedbacks) with interactions in the feedback system. It has a number of important features. First, the system solicits feedback after each interaction and the two participating peers give feedback about each other based on the current interaction. So a peer only gets a chance to give feedback about another peer after an interaction with the other peer. This binding of feedback to interactions reduces vulnerabilities in the system by making it difficult to spam feedbacks. Second, a peer only gives feedback based on current interaction so it does not have to keep track of any previous interactions with the same peer in order to give an overall rating or update the rating.

The third question is related to the benefits of a complaint-based feedback system. The feedback is collected in a form of complaint in the system. A complaint filed by u against v after an interaction is simply a statement from u that it is not satisfied over v's performance during that interaction. Different feedback systems may use different feedback measures such as a positive measure, a negative measure or a hybrid measure such as a numeric rating. Most of the existing feedback systems tend to use a numeric rating. There are several reasons why we chose a negative measure. First, it is not desirable to store complex feedback data as peers only volunteer a portion of their resources to a P2P system. Since it is reasonable to assume peers provide satisfactory services most of the time, the feedback data that need to be stored are relatively reduced compared to positive or hybrid feedbacks. Second, it is unreasonable to expect peers to take the initiative in providing feedbacks. The complaint-based feedback alleviates this problem. A peer will have the incentive to file a complaint against the other peer if the other peer misbehaves because it wants the misbehaving party to be penalized by the society in the future.

3 Trust Management

The effectiveness of a trust model depends not only on the factors and the metric for computing trust measure, but also on implementation of the trust model in a P2P system. Typical issues in implementing a trust model include decentralized trust data management and trust computation execution. A decentralized trust management not only reduces bottleneck but also avoids a single point of failure. This section discusses the architecture and the design considerations in implementing the PeerTrust model in a decentralized P2P information system.

3.1 System Architecture

Figure 1 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. If individual database fails due to negligence or intentional corruption by some peers, it will be corrected by valid information in rest of the community. In addition, the trust computation 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.

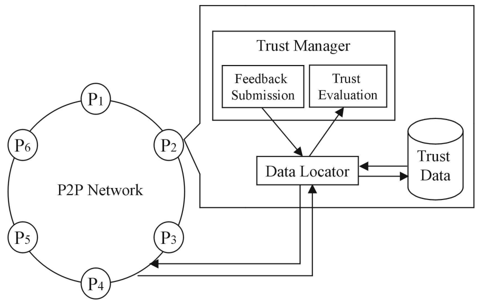

Figure 1: PeerTrust System Architecture

The callout of the peer shows that each peer maintains a small database that stores a portion of the global trust data such as complaints filed. Each peer has a trust manager for submitting feedbacks, collecting feedbacks and computing trust measures. Each peer also has a data locator, a data placement and data location component for placement of trust data over multiple peers and managing the access to the trust data.

The data locator provides a P2P data location scheme for accessing and updating data in the network. Different applications may use different data placement and location schemes, which determine how and where the data can be inserted, updated, and accessed.

The trust manager has two main functions. First, it submits feedbacks to the network through the data locator, which will route the data to the appropriate peer 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 on the fly.

We discuss the details of the trust data location and trust computation in the following subsections.

3.2 Trust Data Location

A number of P2P file sharing systems have emerged and each has its own data location scheme. Examples include Gnutella [4], Freenet [12], CAN [16], Chord [18], Pastry [17], and P-Grid [10]. Most of them are decentralized systems where there is no central server and data retrieval is self-organized by peers.

Depending on the choice of a data location scheme, the implementation of the trust model may be a little 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 version of the PeerTrust, we decide to use P-Grid primarily because we obtained the P-Grid source code.

P-Grid uses the approach of scalable replication of tree structures [13, 20]. Randomized algorithms based on local interactions among peers are used to partition the peers into a virtual binary search tree. By local interactions, peers successively partition the key space and retain routing information to other peers. Readers who are interested in the detail of the construction process of P-Grid may refer to [10]. Once the P-Grid is constructed, each peer maintains a small fragment of the data and maintains a routing table to other peers for the data it doesn't store locally.

Figure 2 shows a simple example of a PeerTrust network of 6 peers with constructed P-Grid. The callout at each peer shows the data keys each peer is responsible for and the routing table for those keys stored at other peers. The search keys are binary strings encoded from peer IDs. If $K_1K_2...K_n$ denotes the common prefix of the keys that a peer is responsible for, the peer keeps n rows in its routing table and the ith row indicates to which peer the query with a data key of a prefix $K_1K_2...\overline{K_i}$ will be routed. For example, P_4 is responsible for key 110, which means P_4 stores the trust data for P_6, including the number of complaints P_6 receives from other peers and

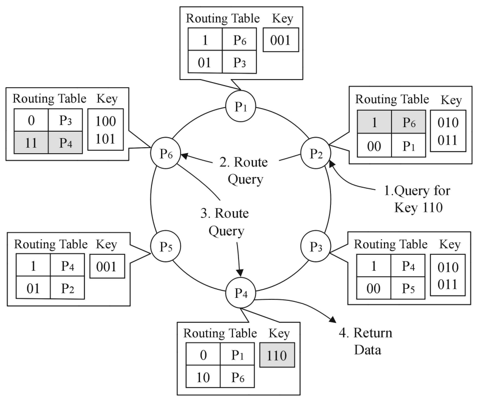

Figure 2: PeerTrust Data Location

the number of interactions P_6 has with other peers. It keeps routing entries for all the other data keys, i.e. the data keys that have a prefix 0 or 10.

When a peer receives a search or update request with a particular data key, it first determines if it is able to process this request locally. If yes, it will process the request with the data key; otherwise it will look up its routing table to find the matching prefix of the data key and route the request to the corresponding reference. Figure 2 shows an example of how a search request is processed in the PeerTrust network. Suppose that peer P_2 receives a query with key 110. P_2 cannot process this query. So it looks up its routing table and finds the row that has the common prefix with the query key 110, which is 1 in this case. According to the row 1, the query with key 110 is now routed to peer P_6. P_6 cannot serve this query locally and again looks up its routing table and finds the row that has the common prefix with the key 110, which is 11. P_6 now routes the query to P_4. P_4 can serve the query with key 110 locally and returns the trust data.

For any given P2P data location scheme, there is a trust issue associated with it. Namely peers may misbehave by providing false data or random data when processing a search or update request for a data item it is responsible for. To address this issue, P-Grid can be configured to have multiple peers responsible for the same key. When a peer is searching for trust data about a particular peer, it simply issues the search request multiple times and finds multiple peers that are responsible for the same key, referred as witnesses. The peer then combines the data from all witnesses and gets a consensus value by using a voting scheme [11].

3.3 Trust Computation

The trust computation component is responsible for computing the trust measure based on the complaints a peer has received from other peers and the number of interactions. We propose two implementations of the trust value computation in this section. One is called dynamic computation, which uses the fresh trust data collected at runtime to compute the trust value. The other is called approximate computation, which uses cache to speed up the trust computation process.

Dynamic Computation
Let $I(u,v,t)$ denote the number of interactions that peer u has with v up to time t, and $C(u,v,t)$ denote the number of complaints a peer u received from v up to time t. Let $I(u,t)$ denote the total number of interactions peer u has with other peers in the network, i.e. $\sum_{v \in P, v \neq u} I(u,v,t)$. Let $\mathbf{A} = (a_{uv})$ be a square matrix of order N (recall N is the number of peers in the network P) with rows and columns corresponding to the peers and a_{uv} is computed as follows:

$$a_{uv} = \begin{cases} \frac{C(u,v,t)}{I(u,t)} & \text{if } I(u,t) \neq 0 \\ 0 & \text{if } I(u,t) = 0 \end{cases} \tag{3}$$

Let $\mathbf{T} = (t_u)$ be a column vector of the trust values of all peers up to time t where $t_u = T(u,t)$. Based on the metric in equation 2, we have:

$$\mathbf{T} = 1 - \mathbf{AT} \tag{4}$$

We can simply start with a default trust value vector. As we obtain new trust data (such as complaints) for each peer, we repeatedly compute the trust vector until it converges. When this is done, all trust values of the peers in the network will be available.

We can easily see that this computation is very expensive as it retrieves the trust data of all peers in the network even when a peer is only interested in evaluating the trustworthiness of a particular peer or a small subset of peers. To address the high communication cost involved in dynamic computation, we propose approximate computation, a more cost efficient computation at each peer using a trust cache.

Approximate Computation
Each peer maintains a trust cache that keeps the trust values it has computed for other peers in the past. Cache size can be determined based on different factors such as size of the overlay network, available resources at each peer, and so on. When a peer w needs to evaluate the trustworthiness of another peer u, it retrieves the trust data of u. Instead of computing the trust value for the peers who filed complaints against u as the balance factor, w looks for their trust values in its cache. It then uses the value if it finds one and uses a default value if it does not. Once the peer w computes the trust value for u, it can again add the trust value of u to the cache or replace an old value in the cache. When the cache is full, it uses an LRU-like cache replacement policy to evict the least recently used data items from the cache.

Let $T_{cache}(v, t')$ denote an old trust value of v up to time t' in peer w's cache and $T_{default}$ denote a default trust value. The approximate computation conducted at peer w computes the trust value of u up to time t as follows:

$$T(u,t) = 1 - \frac{\sum_{v \in P, v \neq u} C(u,v,t) * T'(v,t)}{\sum_{v \in P, v \neq u} I(u,v,t)} \tag{5}$$

Where

$$T'(v,t) = \begin{cases} T_{cache}(v,t') & \text{cache hit} \\ T_{default} & \text{cache miss} \end{cases} \tag{6}$$

Our simulation results later show this approximate computation still produces effective trust evaluation results and significantly reduces the communication cost for the trust computation.

4 Experiments and Results

This section proposes a few evaluation metrics and reports the set of initial experiments to show the feasibility, cost, and benefit of our approach.

4.1 Evaluation Metrics

We first define trust evaluation accuracy as a metric to evaluate how well the trust model helps peers in making trust decisions. A trust evaluation is considered correct when a trustworthy peer is evaluated as trustworthy or an untrustworthy peer is evaluated as untrustworthy. In contrast, a trust evaluation is considered incorrect if a trustworthy peer is evaluated as untrustworthy or an untrustworthy peer is evaluated as trustworthy. For the first type of incorrect evaluations, the peer may miss an opportunity to interact with a trustworthy peer. For the second type of incorrect evaluations, the peer may end up interacting with an untrustworthy peer and coping with the risk of misbehavior from the other peer.

Let E denote the total number of evaluations peers make in the network over a time period. Among these evaluations, let E_{tt} denote the number of evaluations during which trustworthy peers are evaluated as trustworthy, let E_{tu} denote the number of evaluations during which trustworthy peers are evaluated as untrustworthy, let E_{uu} denote the number of evaluations during which untrustworthy peers are evaluated as untrustworthy, and let E_{ut} denote the number of evaluations during which untrustworthy peers are evaluated as trustworthy. The trust evaluation accuracy can be simply defined as the ratio of the correct evaluations over the total number of evaluations. That is:

$$Accuracy = \frac{E_{tt} + E_{uu}}{E} \tag{7}$$

Other metrics can be defined for different application domains with different requirements. For example, we can define a false negative rate in terms of detecting the untrustworthy peers as $FNR = \frac{E_{ut}}{E}$. This may be important for evaluating the trust metric for application domains where it is critical to avoid untrustworthy peers, such as electronic commerce applications.

As performance is always a concern with additional security layer, we also define a metric that measures the overhead a trust model introduces to the application. A good trust model should be effective and robust against the misbehavior of peers in the P2P network without introducing significant overhead to the application. As the communication cost for trust data lookup is the main overhead, we use the average number of messages among peers that are needed for each evaluation as the metric for this purpose.

4.2 Simulation Setting

We implemented a simulator in Mathematica 4.0 to evaluate our approach and demonstrate the importance of the three trust parameters we have identified as against the conventional approach in which only the first parameter, i.e. the amount of satisfaction, is used to measure the trustworthiness of a peer.

Our simulated P2P network consists of 128 peers, among which some are trustworthy and some are untrustworthy. The experiment proceeds as follows. Peers perform interactions with each other. During the interactions, trustworthy peers always perform satisfactory services and only file complaint against the other peer if the other peer performs poor service; untrustworthy peers perform poor services and file a fake complaint against the other party one out of four times and perform satisfactory services in other times. After 6400 interactions, i.e. an average of 100 interactions for each peer, 4 peers evaluate the trustworthiness of 100 randomly chosen peers from the remaining peers.

For the trust evaluation, P-Grid is used as the data location scheme to store and retrieve trust data about peers. We simulated the misbehavior of peers for processing queries, namely, untrustworthy peers provide random data for a query for which it is responsible one out of four times. During each evaluation, a peer issues a search request for the trust data 15 times and then uses the mean value computed from data returned by different witnesses. Both the dynamic computation and approximate computation are simulated for computing the trust value. A simple decision rule with a threshold 0.8 is used to decide whether a peer is trustworthy or not based on peer's trust value.

In order to see how our mechanism performs in different scenarios, we simulated random interactions and the three special scenarios listed in Section 2. Specifically, in random scenario, peers perform interactions with each other in a random pattern so each peer ends up having about the same number of interactions. In scenario 1, half randomly chosen peers perform 2 times more interactions than the other half. In scenario 2, half randomly chosen untrustworthy peers perform 1/10 fewer interactions than other peers. In scenario 3, half randomly chosen trustworthy peers perform interactions only with untrustworthy peers. The probability of the third case happening in the real world may be very low. We use this setup to test the reliability of our approach in the worse case scenario.

We also simulated different peer communities. In each scenario, we vary the number of untrustworthy peers from 4 to 128.

For comparison purpose, we use the P2P trust method proposed by Aberer et al. [11] as an example of the conventional method, esp. given the fact that we obtained their simulation code of the method. Their method uses a complaint based feedback system and uses the number of complaints for the trust measure. We refer to this method as complaint-only method. They proposed two implementations using P-Grid referred as simple and complex algorithm in their paper. The simple algorithm takes a simple majority decision from the trust data returned by multiple witnesses. The complex algorithm checks the trustworthiness of the witnesses first and removes untrustworthy witnesses before taking a majority decision. We compare the trust evaluation accuracy and the communication cost of dynamic computation and approximate computation of PeerTrust to the simple and complex implementation of the complaint-only method.

4.3 Simulation Results

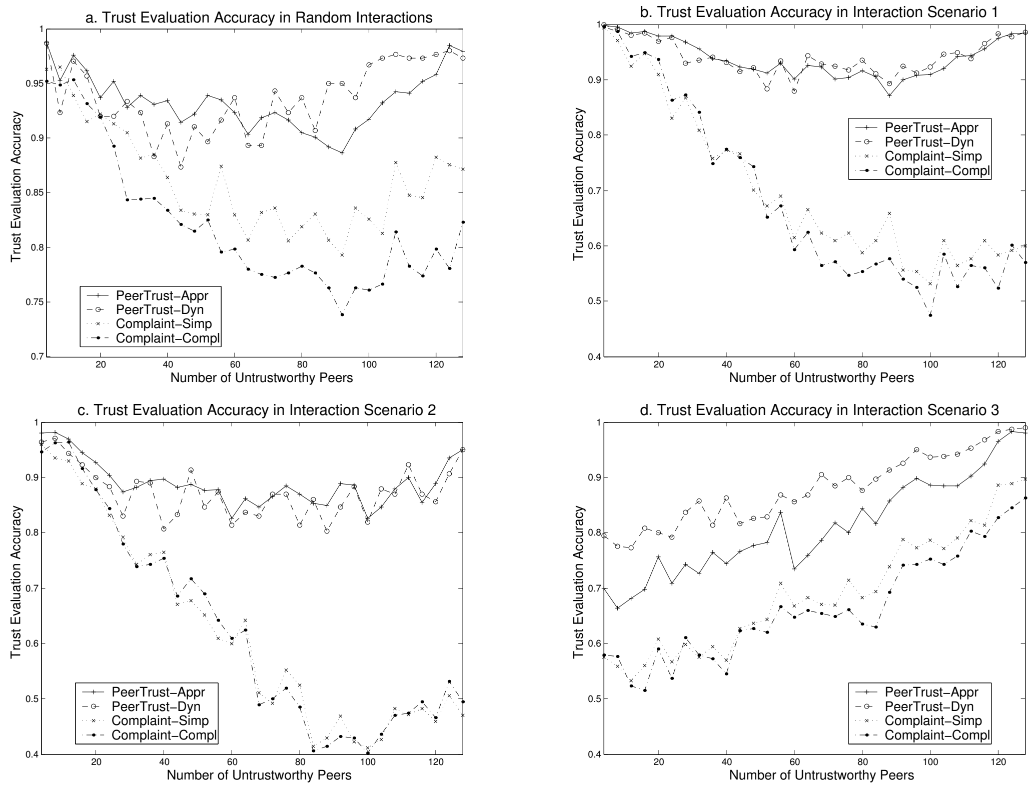

Figure 3: Trust Evaluation Accuracy

First we compare the effectiveness of the two models. Figure 3 represents the trust evaluation accuracy of the two implementations of each model with respect to the number of untrustworthy peers in different interaction scenarios.

In the random scenario (Figure 3a), we can see the PeerTrust approximate computation is comparable to the dynamic computation in most cases. Interestingly, the complex implementation of complaint-only approach does not achieve higher trust evaluation accuracy than the simple implementation. Without an effective metric that takes into account all the necessary trust parameters, the complex implementation does not help by injecting trust considerations into the implementation.

We can make a few observations when comparing PeerTrust and the complaint-only approach. First, the two approaches perform almost equally well when the number of untrustworthy peers is small. Second, when the number of untrustworthy peers increase, PeerTrust stays effective while the performance of complaint-only approach deteriorates. Last, when the number of untrustworthy peers is close to the total number of the peers, both approaches gain a little accuracy. These observations can be explained as follows. When the number of untrustworthy peers is small, the complaint-only approach relies on there being a large number of trustworthy peers who offer honest statements to override the effect of the false statement provided by the untrustworthy peers and thus achieves a high accuracy. When the number of untrustworthy peers increases, the chances for trustworthy peers to interact with untrustworthy peers and receive fake complaints increase. The complaint-only approach uses the number of complaints only for assessing the trust of peers without taking into account the credibility of the complaints. Therefore, the trustworthy peers with fake complaints will likely be evaluated as untrustworthy incorrectly in the complaint-only approach. On the contrary, PeerTrust uses a balance factor of trust to offset the risk of fake complaints and thus is less sensitive to the misbehavior of untrustworthy peers. When the number of untrustworthy peers is close to the total number of peers, there are actually few trustworthy peers that can be evaluated incorrectly. Consequently, the trust accuracy of both approaches goes up at the end.

In scenario 1 (Figure 3b) and scenario 2 (Figure 3c), there is also no significant difference between the two implementations of each model. However, PeerTrust performs significantly better than the complaint-only approach. The difference between the two models increases dramatically as the number of untrustworthy peers increases. This can be well explained by the lack of both number of interactions and the balance factor of trust as the trust parameters in the complaint-only approach.

In scenario 3 (Figure 3d), the dynamic computation of PeerTrust achieves a better accuracy than the approximate computation. Both approaches are not performing well when the number of untrustworthy peers is small. This can be explained as follows. As defined in our simulation, half of the trustworthy peers only interact with untrustworthy peers. As a result, when the number of untrustworthy peers is small, more trustworthy peers will interact with untrustworthy peers only and be likely evaluated incorrectly. However, PeerTrust is still consistently better than the complaint-only approach.

In summary, we verified that PeerTrust is much more stable and performs significantly better in various scenarios regardless of the number of untrustworthy peers in the community. In contrast, the complaint-only model is overly sensitive to the misuse behavior in the network and only performs well in limited cases.

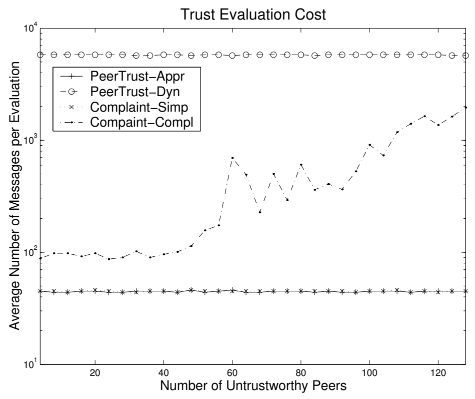

Figure 4: Trust Evaluation Cost

Next we compare the communication cost that the trust evaluations incur in the two models. Figure 4 represents the average number of messages exchanged for each evaluation of each implementation of the two models with respect to the number of untrustworthy peers. As the communication cost does not relate to the interaction scenarios, we only show the random scenario. It can be easily seen that the PeerTrust approximate computation and the complaint-only simple implementation are very efficient and the messages needed do not change with the number of untrustworthy peers. On the other hand, the complaint-only complex method has a significant overhead as the number of untrustworthy peers increases. The reason is that the method evaluates the trustworthiness of the witness recursively. When the number of untrustworthy peers increases, there are more complaints in the system and hence it causes more communications. As we foresee, the cost of the dynamic computation of PeerTrust is very high. It is interesting to note, however, if a peer wishes to evaluate the trustworthiness of a large number of peers at the same time, the dynamic computation cost will stay the same while the cost of all the other methods will be multiplied by the number of peers to evaluate.

5 Related Work

Marsh is one of the first who presented a formal trust model based on social models of interactions and social properties of trust [14]. With a strong sociological foundation, the model is complex and is not suitable as an automatic trust mechanism for a P2P system.

A few other models are proposed that can be seen as an adaptation of Marsh's model to today's online environments and agent systems [22, 9, 21]. However, they are not suitable for P2P environments for a number of reasons. First, the models still tend to be complex with a lot of parameters and operations. In contrast, each peer only donates limited resources to the network in a P2P network and cannot afford to maintain complex information or perform complex computations locally. Second, they all rely on a central server or database or some kind of global knowledge to be maintained at each agent, which is also not suitable for a decentralized P2P environment. Last and most importantly, little regard is given for whether participant can conspire to provide false ratings. Most of them assume the feedback is always given honestly and with no bias and thus cannot adequately deal with malicious behaviors of agents.

There are also a few existing reputations systems. The most well known one is the feedback system of the online auction site eBay [1]. It collects feedback about each participant in the form of ratings and simply sums them up. Another example is the reputation system of a news service site Slashdot [8]. Their reputation system is to help a user figure out which news store is worth reading. All these systems use the single factor of feedbacks as the reputation measure. Plus all these systems use a centralized approach and again are not applicable to P2P environment.

A number of research projects have engaged in P2P computing. As we stated in Section 1, most of these work has been focused on efficient resource location and load balancing and very few have addressed the need of trust in P2P systems or have incorporated trust into their resource placement and allocation algorithms.

The most relevant work is the trust method proposed by Aberer et al [11] for a P2P information system. This is the closest to our work in the sense that the data management and trust management are both in a distributed manner. The important contribution of their work is to emphasize the question of choosing the right model to assess trust and the question of obtaining the data needed to compute trust cannot be investigated in separation. They also utilize a complicated probability based threshold for trust decisions. Yet again their trust metric only takes into account the number of complaints and thus only works for limited scenarios and is very sensitive to the misbehavior of peers, as we have shown through our simulations.

Our approach differs from all these work in two significant ways. First, we identify three important trust parameters and argue that they are equally important for computing trust measure of peers in a decentralized P2P system. We also present the experimental results, verifying our argument. Second, we introduce a general trust metric that incorporate all three trust parameters in the trust computation and describe a complaint-based trust metric as a concrete form to illustrate the usefulness and benefit of our trust metric. We also validated our metric through a set of experiments, demonstrating the significant gain in accuracy when using our trust mechanism, compared with the complain-only methods.

6 Conclusions and Future Work

We presented PeerTrust, a simple and yet effective trust mechanism for an open P2P information system. We identified the three important trust parameters and developed a trust metric that combines these parameters for quantifying and comparing the trustworthiness of peers. We discussed the implementation considerations of the trust model in a decentralized P2P environment. We also simulated two implementations of the metric and demonstrated the effectiveness, cost, and benefit of our approach in comparison with another method that only takes into account one of the three parameters we identified.

Trust in P2P systems is a new research area and there are many open questions and interesting issues to be addressed. Our research on PeerTrust continues along several directions.

First, we are interested in incorporating P2P dynamics into the trust management. For example, a peer may join a network and behave reliably until it reaches a high reputation and then start committing fraud. Specifically, a peer may act completely trustworthy for the first x number of interactions but start behaving completely untrustworthy for the next y number of interactions. An interesting extension to our current mechanism is to allow the trust computation to reflect such dynamic change in time. Second, we are interested in combining trust management with intrusion detection to address concerns of sudden and malicious attacks. Finally, we are working towards incorporating PeerTrust into two P2P applications that are currently under development in our research group, namely PeerCQ [6] and HyperBee [5]. PeerCQ is a P2P system for information monitoring on the web based on continual queries. HyperBee is a peer-to-peer search engine that looks to solve the problems that challenge today's search engines by utilizing peers to crawl the web page faster and cheaper. Both systems are faced with trust issues such as whether the peer can be trusted for performing a particular task, namely executing a continual query in PeerCQ and crawling the web page in HyperBee.

Acknowledgement

We would like to thank Karl Aberer and Zoran Despotovic for providing us the source code of P-Grid and their trust model for our comparison. This research is supported partially by a NSF ITR grant. The second author would like to acknowledge the partial support from a NSF CCR grant, a DOE SciDAC grant, and a DARPA ITO grant.

References

[1] ebay. http://www.ebay.com.

[2] Free haven project. http://www.freehaven.net.

[3] Freenet. http://freenetproject.org.

[4] Gnutella. http://www.gnutella.com.

[5] Hyperbee. http://www.hyperbee.com.

[6] Peercq. http://www.cc.gatech.edu/projects/disl/PeerCQ.

[7] Seti@home. http://setiathome.ssl.berkeley.edu.

[8] Slashdot. http://slashdot.org.

[9] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *33rd Annual Hawaii International Conference on System Sciences (HICSS-33)*, 2000.

[10] K. Aberer. P-grid: A self-organizing access structure for p2p information systems. In *Cooperative Information Systems, 9th International Conference, CoopIS 2001*, 2001.

[11] K. Aberer and Z. Despotovic. Managing trust in a peer-to-peer information system. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management*, 2001.

[12] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[13] T. Johnson and P. Krishna. Lazy updates for distributed search structure. In *1993 ACM SIGMOD International Conference on Management of Data*, 1993.

[14] S. Marsh. *Formalising Trust as a Computational Concept.* PhD thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.

[15] D. H. McKnight and N. L. Chervany. The meanings of trust. Technical Report WP9604, University of Minnesota Management Information Systems Research Center, 1996.

[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *ACM SIGCOMM*, 2001.

[17] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms*, 2001.

[18] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM*, 2001.

[19] V. Swarup and J. T. Fabrega. Trust: Benefits, models, and mechanisms. In *Secure Internet Programming: Security Issues for Mobile and Distributed Objects, Lecture Notes in Computer Science.* New York: Springer-Verlag, 1999.

[20] H. Yokota, Y. Kanemasa, and J. Miyazaki. Fat-btree: An update-conscious parallel directory structure. In *15th International Conference on Data Engineering*, 1999.

[21] B. Yu and M. P. Singh. A social mechanism of reputation management in electronic communities. In *Cooperative Information Agents, 7th International Conference, CoopIS 2000*, 2000.

[22] G. Zacharia, A. Moukas, and P. Maes. Collaborative reputation mechanisms in electronic marketplaces. In *32nd Annual Hawaii International Conference on System Sciences (HICSS-32)*, 1999.