TRUST MODELS FOR P2P E-COMMERCE

Δήμητρης Παναγιώτου, AM: 03001608
dpana@mail.ntua.gr

Αθήνα, Σεπτέμβριος 2006
# Table of Contents

1. **INTRODUCTION** ................................................................. 4

2. **PEERTRUST** ...................................................................... 7

   2.1 General Trust Metric ......................................................... 9
   2.2 The Basic Metric ............................................................. 10
   2.3 Using the Trust Value ...................................................... 11

3. **DECENTRALIZED PUBLIC KEY INFRASTRUCTURE ........ 14

   3.1 P-Grid ........................................................................... 14
   3.2 The P-Grid based PKI ...................................................... 15
   3.3 Enabling P2P E-commerce .............................................. 18

4. **BAYESIAN NETWORK-BASED TRUST MODEL ................ 21

   4.1 Scenario ........................................................................... 22
   4.2 Trust in a file provider’s competence in providing files ...... 22
   4.3 Evaluation of an Interaction ............................................. 25
   4.4 The Procedure .................................................................. 25

**REFERENCES ........................................................................... 29
Introduction
Peer-to-peer (P2P) electronic commerce (eCommerce) 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. P2P eCommerce communities can be implemented either on top of a P2P network [20, 11, 23] or using a conventional client-server platform. Gnutella is an example of a P2P eCommerce community that is built on top of a P2P computing 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 computing architecture.

In eCommerce settings P2P communities are often established dynamically with peers that are unrelated and unknown to each other. Peers of such communities 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 accessing 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. A recent study [7] reported results from both an online experiment and an online auction market, which confirmed that trust can mitigate information asymmetry (the difference between the amounts of information the two transacting parties possess) by reducing transaction-specific risks, therefore generating price premiums for reputable sellers.

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 [14, 19, 6]. However, it is not applicable to P2P eCommerce communities where peers are equal in their roles and there are no entities that can serve as trusted third parties or intermediaries.

Reputation systems [18] provide a way for building trust through social control without trusted third parties. Most research on reputation-based trust utilizes information such as community-based feedbacks about past experiences of peers to help making recommendation and judgment on quality and reliability of the transactions. Community-based feedbacks are often simple aggregations of positive and negative feedbacks that peers have received for the transactions they have performed and cannot accurately capture the trustworthiness of peers. In addition, peers can misbehave in a number of ways, such as providing false feedbacks on other peers. The challenge of building a trust mechanism is how to effectively cope with such malicious behavior of peers. Another challenge is that trust context varies from communities to communities and from transactions to transactions. It is
important to build a reputation-based system that is able to adapt to different communities and different situations.

In this study we present three trust models for P2P eCommerce. Firstly, we present PeerTrust (chapter 2), a peer-to-peer trust model for quantifying and assessing the trustworthiness of peers in P2P eCommerce communities. In chapter 3, we present a decentralized public key infrastructure (PKI) based on a statistical (quorum-based) approach. Finally, chapter 4 presents a Bayesian Network-Based Trust Model which is general enough for all kind of P2P networks.
Chapter 2

PeerTrust
2. PeerTrust

The goal of PeerTrust is to build a general trust metric that provides an effective measure for capturing the trustworthiness of peers, addresses the fake or misleading feedbacks, and has the capability to adapt to different communities and situations.

PeerTrust identifies five important factors for evaluating the trustworthiness of a peer in a P2P eCommerce community:

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

In PeerTrust, a peer’s trustworthiness is defined by an evaluation of the peer in terms of the level of reputation 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 in interacting with the peer. The factors depicted above are used for such an evaluation. These are described below in detail.

Feedback in Terms of Amount of Satisfaction

Reputation-based systems rely on feedbacks to evaluate a peer. In a P2P eCommerce community, the feedbacks in terms of amount of satisfaction a peer receives regarding its service comes primarily from the transactions other peers have had with this peer and reflects how well this peer has fulfilled its part of the service agreement. Most existing reputation based systems use this factor alone and compute a peer $x$’s trust value by a summarization of all the feedbacks $x$ receives through its transactions with other peers in the community. For example, in eBay, buyers and sellers can rate each other after each transaction (+1, 0, -1) and the overall reputation is the sum of these ratings over the last 6 months.

It is obvious that these feedback-only metrics are flawed. A peer who has performed dozens of transactions and cheats on 1 out of every 4 cases will have a steadily rising reputation in a given time duration whereas a peer who has only done 10 transactions during the given time duration but is completely honest will be treated as less reputable if the reputation measures of peers are computed by a simple aggregation of the feedbacks they receive.
Trust Models for P2P eCommerce

Number of Transactions

With a skewed transaction distribution, i.e. some peers have a higher transaction frequency than other peers; the trustworthiness of a peer is not captured fairly when a simple aggregation of feedbacks is used to model the trustworthiness of peers without taking into account the number of transactions. A peer may increase its trust value by increasing its transaction volume to hide the fact that it frequently misbehaves at a certain rate. So the number of transactions is an important scope factor for comparing the feedbacks in terms of amount of satisfaction among different peers. An updated metric can be defined as the ratio of the total amount of satisfaction peer $x$ receives over the total number of transactions peer $x$ has, i.e. the average amount of satisfaction peer $x$ 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 $x$ receives from another peer $y$ during a transaction is simply a statement from $y$ regarding how satisfied $y$ feels about the quality of the information or service provided by $x$. The trust model should consider potential threats. For example, 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. Without a credibility factor built in, this peer will be evaluated incorrectly because of false statements even though it provides satisfactory service in every transaction. Therefore, the feedback from those with better credibility should be weighted more heavily in the trust metric. Intuitively incorporating credibility factor for feedbacks represents the need to differentiate the credible amounts of satisfaction from the less credible ones in computing the reputation of peers. If we consider reputation-based trust as an important mechanism to address threats of untrustworthy peers and their malicious behaviors in the P2P community, then we can see credibility of feedbacks as a mechanism to address the risk of using potentially false feedbacks to rate peers’ reputation.

Transaction Context Factor

Transaction context is another important factor when aggregating the feedbacks from each transaction as transactions may differ from one another even within the same eCommerce community. For example, if a community is business savvy, the size of a transaction is an important context that should be incorporated in the trust metric to weight the feedback for that transaction. It can act as a defense against some of the subtle malicious attacks, such as a seller develops a good reputation by being honest for small transactions and tries to make a profit by being dishonest for large transactions.
Community Context Factor

Various community contexts can be taken into account to address some of the common problems such as lack of the temporal adaptivity. In a pop music sharing community, it may be desirable to only consider the recent transaction histories of a peer to reflect the current trend. However, in a business community, one may wish to use the recent transaction history of a peer and at the same time consider the historical ratings a peer receives in the past but with a lower weight than the recent history in order to evaluate the peer based on its consistent behavior. This historical behavior of a peer is one type of community context that is important to be incorporated into the trust model to give the trust system a temporal adaptivity.

The feedback incentive problem can be also alleviated by adding a reward as a community context for peers who submit feedbacks.

The community context can be also used to adapt the trust system to different communities and address problems that are specific to the community. For instance, free riding is a common challenge with online file sharing communities [5, 17]. The total number of files a peer shares can be seen as a type of community context and be taken into account when evaluating the trustworthiness of a peer. With such a community context factor, a peer that shares a large number of files with the rest of the peers in the community will have a higher trust value than the free riders and alleviate the free riding problem.

2.1 General Trust Metric

In this section the presented parameters are formalized and a general trust metric that combines them in a coherent manner is introduced.

Let $I(x)$ denote the total number of transactions performed by peer $x$ during the given period, $p(x,i)$ denote the other participating peer in peer $x$’s $i$th transaction, $S(x,i)$ denote the normalized amount of satisfaction peer $x$ receives from $p(x,i)$ in its $i$th transaction, $Cr(p(x,i))$ denote the credibility of the feedback submitted by $p(x,i)$, $TF(x,i)$ denote the adaptive transaction context factor for peer $x$’s $i$th transaction, and $CF(x)$ denote the adaptive community context factor for peer $x$ during the given period. The trust value of peer $x$ during the period, denoted by $T(x)$, is defined as:

$$T(x) = \alpha \frac{\sum_{i=1}^{I(x)} S(x,i) \cdot Cr(p(x,i)) \cdot TF(x,i)}{I(x)} + \beta \cdot CF(x)$$

The metric consists of two parts. The first part is the average amount of credible satisfaction a peer receives for each transaction. It may take into account transaction context factor to capture the transaction-dependent characteristics. This history-based evaluation can be seen as a prediction for peer $x$’s likelihood of a successful transaction in the future. A confidence value $\Delta$.Παναγιώτου – «Δίκτυα Προστιθέμενης Αξίας EDI και Εφαρμογές Ηλεκτρονικού Εμπορίου» Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
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. $\alpha$ and $\beta$ denote the normalized weight factors for the two parts.

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 communities. The first three parameters - the feedback, the number of transactions, and the credibility of feedback source - are the important basic trust parameters that should be considered in computation of a peer’s trustworthiness in any P2P eCommerce community.

### 2.2 The Basic Metric

The basic form of the general metric by turning off the transaction context factor ($TF(x,i)=1$) and the community context factor ($\alpha=1$ and $\beta=0$) is given below:

$$T(x) = \frac{\sum_{i=1}^{I(x)} S(x,i) * Cr(p(x,i))}{I(x)}$$

This metric computes the trust value of a peer $x$ by an average of the credible amount of satisfaction peer $x$ receives for each transaction performed during the given period.

The feedbacks in terms of amount of satisfaction are collected by a feedback system. PeerTrust model 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 current 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. $S(x,i)$ is a normalized amount of satisfaction between 0 and 1 that can be computed based on the feedback.

Both the feedbacks and the number of transactions are quantitative measures and can be collected automatically. Different from these two basic parameters, the third trust parameter - credibility of feedback is a qualitative measure and needs to be computed based on past behavior of peers who file feedbacks. Different approaches can be used to determine the credibility factor and compute the credible amount of satisfaction. One way is to solicit separate feedbacks for feedbacks 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. For example, one may use a
function of the trust value of a peer as its credibility factor so feedbacks 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 feedbacks in order to hide their own malicious behavior. Second, trustworthy peers are more likely to be honest on the feedbacks 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 feedbacks to its competitors. In this extreme case, using a function of trust to approximate the credibility of feedbacks 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 feedbacks filed by peers. Therefore it cannot handle the situation of inconsistent behavior, such as peers offering good services but providing false feedbacks to jeopardize its competitors.

2.3 Using the Trust Value

The value given by the trust metric gives a measure that helps peers to form a trust belief or action on other peers or to compare the trustworthiness of other peers. A higher value of $T(x)$ indicates that peer $x$ is more trustworthy in terms of the collective evaluation of $x$ by the peers who have had transactions with $x$ and other community context factors.

There are several usages of the trust value in P2P eCommerce communities. First, a peer $y$ can derive trust relationship with another peer $x$ to determine whether to perform the next transaction with peer $x$. 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 $T(x)$ with the associated confidence value will make it trust $x$ given a specific situation. Different peers may have different perception over the same value. A simple rule for peer $y$ to form a trust action on peer $x$ can be conducted as:

If $T(x) > T_{\text{threshold}}(y)$, then trust $x$

where $T_{\text{threshold}}(y)$ is the threshold trust value for peer $y$ to trust another peer.

The factors that determine the threshold $T_{\text{threshold}}(y)$ include how much peer $y$ is willing to trust others. A more tolerant peer may have a lower threshold. It is a manifest of what is called dispositional trust [15], 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.

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 compare the trustworthiness of the peers that respond.
Trust Models for P2P eCommerce

to its request based on their trust value and choose the peer with the highest trust value to download the file.

Furthermore, the trust values of peers can be used to compute the aggregate trust values of a peer group in order to derive a trust relationship for a task that requires a group of peers.
Decentralized Public Key Infrastructure
3 Decentralized Public Key Infrastructure

Managing a public key infrastructure (PKI) in a P2P way needs an efficient and reliable distributed information access structure, and also effective functionalities like updates of replicas even in the presence of frequent disconnections and possibly uncooperative peers. This is not possible in unstructured P2P systems like Gnutella [8], that is why the "web of trust" model, which essentially depends on random walks (basically flooding) for exploring the trust graph of a P2P network has gained more popularity. But with the recent development of efficient access structures like CAN, Chord, Freenet and P-Grid among others, it is indeed possible to realize more systematic models, rather than relying on the ad-hoc web of trust model, for which no probabilistic guarantees have been provided so far. In section 3.1, a brief introduction of P-Grid [4] is given. Section 3.2 presents the P-Grid based PKI, while section 3.3 explains how P2P eCommerce is enabled using P-Grid based PKI.

3.1 P-Grid

P-Grid [1, 4] is a peer-to-peer lookup system based on a virtual distributed search tree: Each peer only holds part of the overall tree, which comes into existence only through the cooperation of individual peers. Searching in P-Grid is efficient and fast even for unbalanced trees [2] (O(log(n)), where n is the number of leaves). Unlike many other peer-to-peer systems P-Grid is a truly decentralized system which does not require central coordination or knowledge. It is based purely on randomized algorithms and interactions. It is assumed that peers fail frequently and are online with a very low probability. Figure 1 shows a simple P-Grid.

Every participating peer’s position is determined by its path, that is, the binary bit string representing the subset of the tree’s overall information that the peer is responsible for. For example, the path of Peer 4 in Figure 1 is 10, so it stores all data items whose keys begin with 10. For fault-tolerance multiple peers can be responsible for the same path, for example, Peer 1 and Peer 6. P-Grid’s query routing approach is simple but efficient: For each bit in its path, a peer stores a reference to at least one other peer that is responsible for the other side of the binary tree at that level. Thus, if a peer receives a binary query string it cannot satisfy, it must forward the query to a peer that is “closer” to the result. In Figure 1, Peer 1 forwards queries starting with 1 to Peer 3, which is in Peer 1’s routing table and whose path starts with 1. Peer 3 can either satisfy the query or forward it to another peer, depending on the next bits of the query. If Peer 1 gets a query starting with 0, and the next bit of the query is also 0, it is responsible for the query. If the next bit is 1, however, Peer 1 will check its routing table and forward the query to Peer 2, whose path starts with 01.
The P-Grid construction algorithm [4] guarantees that peer routing tables always provide at least one path from any peer receiving a request to one of the peers holding a replica so that any query can be satisfied regardless of the peer queried. Additionally it guarantees that a sufficient number of replicas exist for any path and that the peers representing a certain path also know their replicas. Thus the routing tables will hold also multiple references for each level which the routing algorithm selects randomly [4].

Also, P-Grid, unlike most contemporary P2P systems, supports updates of the stored, replicated data via a push/pull strategy with probabilistic success and consistency guarantees in an unreliable environment [10].

### 3.2 The P-Grid based PKI

Each peer $p$ is uniquely identified by a universally unique identifier (UUID) $ld_p$. This identifier is generated once when a peer joins the P-Grid community, by applying a cryptographically secure hash function to the concatenated values of the current date and time, the current IP address $addr_p$ and a large random number. At bootstrap, each peer $p$ also generates a private/public key pair $D_p / E_p$.

In P-Grid, routing tables and the index hold only these identifiers. Each peer $p$ additionally has a cache of “identity to physical address” mappings $(ld_i, addr_i, TS_i)$ ($TS_i$ denotes a time-stamp) that it already knows, in order to be able to communicate with other peers. Since disconnections of peers may lead to changing IP address, peers must update their latest “identity to physical address” mapping in P-Grid. The update functionality is provided in P-Grid as described in [10].
To correctly identify a peer it is essential to detect old mappings and retrieve and cache up-to-date ones.

The algorithm for building a decentralized PKI on top of P-Grid is given below.

**Bootstrap**

Bootstrap is the phase when a new peer $p$ joins the P-Grid.

1. $p$ determines its current IP address. The IP address must be routable and reachable, i.e., not behind a firewall. The IP address is inserted in the P-Grid in order to handle possible changes of physical address of peers reconnecting after staying offline, and has been described and analyzed in [12].

2. $p$ generates $Id_p$, $D_p$, $E_p$.

3. $(Id_p, addr_p, E_p, TS_p, D_p(Id_p, addr_p, E_p, TS_p))$ (for brevity denoted as *tuple* in the following) is inserted into P-Grid by $p$ using $Id_p$ as the key ($TS_p$ prevents replay attacks). Inserting in P-Grid means that the request is routed to a peer $R_i \in R_p$. $R_p$ is the set of replicas responsible for the path using $Id_p$ as the key value ($path(Id_p)$). If $Id_p$ already exists in the P-Grid (though this is very unlikely) $p$ is notified. If so, $p$ generates a new $Id_p$ and repeats this step.

4. $p$ repeats the insertion operation at $R_{min1}$ random and distinct P-Grid peers, so that the insertion request reaches an expected $R_{min2}$ distinct replicas.

5. All $R_i$ that receive the *insert(tuple)* message initiate *update(tuple, $R_i$)* among their replicas $R_p$. All replicas, including the ones that originated such updates locally store the tuple only if it receives and forms a quorum of $R_{min3} \leq R_{min2}$ distinct such update messages within a $T_{out1}$ time.

Peers who received the original insert then send a confirmation to $p$. This of course holds for the peers/replicas that are online during the update operation. Those peers that come online later use a quorum based pull (anti-entropy) to get a current view as described in [10]. If after $T_{out1}$ since receiving the first update message, $R_{min3}$ distinct messages have not been received, the peers discard the information.

In the absence of Byzantine/malicious peers it would have been sufficient to make a single insert in P-Grid, since the update mechanism would have updated all replicas. However, malicious peers may initiate updates with false information. Since search and insert requests are routed to random replicas, multiple requests are used and then a quorum to address this properly.
6. As a result of the previous steps the mapping will be physically stored at peers in $R_p$. Based on the randomized algorithms that P-Grid uses we can assume that the individual replicas $R_i \in R_p$ are independent and they collude or behave Byzantine only to a degree that can be handled by existing algorithms.

7. If $p$ receives $R_{min4} \leq R_{min3}$ confirmations (within some $T_{out2} > T_{out1}$), it is convinced (probabilistically) that its public key has been replicated amply for fault tolerance. Otherwise $p$ generates a new $ld_p$ and repeats the previous steps.

Peer startup

Whenever a peer $p$ rejoins the P-Grid it performs the following step.

1. $p$ starts up and checks whether its $addr_p$ has changed. If yes, it initiates an update of its new physical address (signed with its private key). The complete algorithm for update along with the cost incurred and its reliability can be found in [12]. This step is necessary in order to make sure that the routing tables are correct.

Operation phase

This phase denotes the standard operation, i.e., $p$ is up and running, has registered an up-to-date mapping of its identity/physical ($ld_p$, $addr_p$, $TS_p$) and is ready to process queries and update requests. Both queries and updates need to be routed to at least one replica peer responsible for the concerned key space. The following steps are to route it successfully despite frequent peer disconnections and changes in peers’ physical address resulting in temporarily inconsistent routing tables.

By establishing the correct mapping, we ensure that operations (query/insert/update) may be successfully carried out. Then, such operations pertaining to either peers’ public key, current physical address, reputation or any other kind of information may be conducted in a reliable manner. The steps incurred are given below.

1. $p$ receives a request $Q$ from a peer $q$.

2. In case $p$ can satisfy $Q$ the result is returned to $q$. Otherwise $p$ finds out which peer $p_f$ to forward the query to. It checks its routing table and retrieves ($ld_{pf}$, $addr_{pf}$, $E_{pf}$, $TS_{pf}$) which had been entered during the construction of P-Grid.

3. $p$ generates a random number $r$, contacts $p_f$ and sends $E_{pf}(r)$. As an answer $p_f$ must send $(D_{pf}(E_{pf}(r)))$ and $q$ can check whether $D_{pf}(E_{pf}(r))=r$. If yes, $p_f$ is correctly identified, i.e., $p$ really talks to the peer it intends to, and $Q$ is forwarded to $p_f$. 

Δ.Παναγιώτου – «Δίκτυα Προστιθέμενης Αξίας EDI και Εφαρμογές Ηλεκτρονικού Εμπορίου» 17
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Trust Models for P2P eCommerce

4. If not, then \( p_f \) has a new IP address (the case that somebody tries to impersonate \( p_f \) is covered implicitly by the signature check above) and \( p \) sends a query to P-Grid to retrieve the current \( addr_{pf} \) using \( Id_{pf} \) as the key.

Since \( p_f \) may be offline multiple routing entries for each level are maintained to offer alternative peers to route to.

5. \( p \) collects all answers \( t_i = (Id_{pf}, E_{pf}, addr_{pf}, TS_{pf}, D_{pf}(Id_{pf}, E_{pf}, addr_{pf}, TS_{pf})) \) it receives from the \( R_j \in R_{pf} \).

If extended security is required then the \( R_i \) should sign their answers, i.e., send \( (t_i, D_{R_j}(t_i)) \). \( p \) has to collect at least \( R_{min3} \) answers to detect misinformed or malicious peers, i.e., checks whether a certain quorum of the answers is identical \( (R_{min3} \) is defined by each individual \( p \) according to its local requirements for trustworthiness of the reply). Otherwise the query is repeated a certain number of times before aborting.

a. As an optimization the quorum can be avoided under certain circumstances. If \( p \) already knows \( E_{pf} \), e.g., from the construction of the P-Grid or because it has already done a certain number of (quorum-based) queries for \( E_{pf} \) that have resulted in identical answers, so that it can assume that its \( E_{pf} \), then it can immediately check the validity of the answer by \( E_{pf}(D_{pf}(Id_{pf}, E_{pf}, addr_{pf}, TS_{pf})). Id_{pf} = t_iD_{pf}. \)

b. The scheme can be further optimized (and made more robust and secure) by having all peers store the \( E_{pf} \)’s that they receive.

6. Now \( p \) can proceed with step 3. In case this is successful \( p \) enters \( (Id_{pf}, addr_{pf}, E_{pf}, TS_{pf}) \) in its local cache.

Following the above steps, a peer \( p \) can obtain the latest physical address of other peers in a recursive manner and thus successfully handle the basic P-Grid operations of query, insert and update. Any information, including public key and reputation related information may then be accessed and maintained similarly in an efficient and reliable manner. A P-Grid based PKI is efficient because the basic operations such as search or insert in P-Grid take \( O(log(n)) \) messages to discover one random replica responsible for the relevant key. Since the routing process is randomized, reliability of results is then obtained by using quorum based techniques.

3.3 Enabling P2P E-commerce

In order to enable P2P e-commerce it is essential to provide security functionalities, many of which can only be realized if a PKI is available, for example: authentication, confidentiality and trust. Figure 2 shows the necessary functionalities which all rely on a PKI in the context of the envisioned model for P2P e-commerce.
**Authentication:** Verification of the identity of a participant. Authentication of an entity’s identity is typically done using digital signature, which uses a public key infrastructure (PKI). While other means for authentication exist, for example username/password schemes, such approaches are excluded because of their centralized architecture and consequent incompatibility with the P2P approach.

**Non-repudiation:** To provide undeniable proof of any operation conducted by an entity, it is again necessary to apply digital signatures.

**Accountability:** Past actions of participants need to be taken into account in the present, thus penalizing for past misbehaviour or rewarding for past compliance with the (possibly implicit) rules. This is done using reputation management of peers. A P2P trust model is provided in [3]. For this, it is essential to have reliable authentication and non-repudiation schemes, which themselves rely on a reliable PKI.

**Authorization:** Participants may decide to authorize other participants to use certain resources after having authenticated their identity, and possibly after making a judgement on their trustworthiness.

**Confidentiality and data integrity:** Participants in an activity may require confidentiality out of privacy concerns or for preserving digital rights. Message digesting and digital signatures can be employed to prevent data corruption. These in turn again rely on the existence of a PKI.

These basic security services are needed to implement any eCommerce platform (on top of a peer-to-peer network). Distributed access structures such as P-Grid not only require authentication of peers for reliability, but also can serve as a new means for authentication, and provide maintenance of other resources and services like trust (reputation) information, and thus be used as platforms for C2C commerce in P2P. Additionally this fits well with the P2P design principle of avoiding any kind of centralization or specialized roles, and thus the P-Grid based PKI is an important step towards enabling eCommerce in P2P.
Bayesian Network-Based Trust Model
4 Bayesian Network-Based Trust Model

Agents are often used to manage and reason about trust and reputation on behalf of users. In this situation, trust is defined as an agent’s belief in attributes such as reliability, honesty and competence of the trusted agent. The reputation of an agent defines an expectation about its behavior, which is based on other agents’ observations or information about the agent’s past behavior within a specific context at a given time. An agent broadly builds two kinds of trust in another agent. One is the trust in another agent’s competence in providing services. The other is the trust in another agent's reliability in providing recommendations about other agents. Here the reliability includes two meanings. One meaning is whether the agent is truthful in telling its information. The other is whether the agent is trustworthy or not. Since agents are heterogeneous, they judge issues by different criteria. If their criteria are similar, one agent can trust another agent. If their criteria are different, they cannot trust each other even if both of them tell the truth. In the implementation of such a system based on trust and reputation, some issues have to be considered.

1) How does an agent model its user? Each user has different preferences and ways of judging the quality of interaction. In order to behave as its user wants, an agent has to keep learning its user’s preferences and behaviors. If an agent fails to do as what its user expects, it will be useless.

2) How is an interaction to be evaluated? Trust is built on an agent’s direct interactions with other agents. For each interaction, an agent’s degree of satisfaction of the interaction will directly influence its trust in the other agent involved in the interaction. Usually, an interaction has multiple aspects and can be judged from different points of view. Since an agent behaves on behalf of its user, it has to know how its user judges an interaction so that it can evaluate it in the same way.

3) How does an agent update its trust in another agent?

4) When will an agent ask for recommendations about another agent that it is going to interact with?

5) How does an agent combine together the recommendations for a given agent coming from different references? Since the recommendations might come from trusted agents, nontrusted agents or strangers, an agent has to decide how to deal with them.

6) How does an agent decide if another agent is trustworthy to interact with or not, according to its direct experiences or reputation, or both?

7) How does an agent develop and update its trust in a reference that makes recommendations?
8) How many kinds of trust does an agent need to develop with another agent in a single context? In most situations, agents need to develop multiple trust relationships with each other in order to evaluate each other from different perspectives. For example, agent A might trust agent B in providing music files with good quality. But agent A might not trust agent B in offering movie files with the same quality as music files.

The proposed approach will deal with all the issues above except the first one, which is beyond the scope of this study, although it is extremely important. We will focus on the file sharing system in peer-to-peer networks. But the idea can be applied to P2P eCommerce also.

### 4.1 Scenario

In the area of file sharing in peer-to-peer networks, all the peers are both providers and users of shared files. So each peer plays two roles, the role of file provider offering files to other peers and the role of user using files provided by other peers. In order to distinguish the two roles of each peer, when a peer acts as a file provider, we call it file provider; otherwise, we call it simply agent. Agents will develop two kinds of trust, the trust in file providers’ competence in providing files and the trust in other agents’ reliability in making recommendations. We assume all the agents are truthful in telling their local information. Thus, only the situation where agents have different ways of judging issues, which reflects different user types, is taken into consideration.

### 4.2 Trust in a file provider’s competence in providing files

In a peer-to-peer network, file providers’ capabilities are not uniform. For example, some file providers may be connecting through a high-speed network, so they are able to send files to other agents at a fast speed. Some file providers might like music, so they share a lot of music files. Some may be interested in movies and share some movies. Some may be very picky in file quality, so they only keep and share files with high quality. Therefore, the file provider’s capability can be presented in various aspects, such as the download speed, file quality and file type (see figure 3). The agent’s needs are also different in different situations. Sometimes, it might want to know the file provider’s overall capability. Sometimes it might only be interested in the file provider’s capability in some particular aspect. For instance, an agent wants to download a music file from a file provider. At this time, knowing the file provider’s capability in providing music files is more valuable for the agent than knowing the file provider’s capability in other aspects. Agents also need to develop differentiated trust in file providers’ capabilities. For example, the agent who wants to download a music file from the file provider cares about whether the file provider is able to provide the music file with good quality at a fast speed, which involves the file provider’s capabilities in two aspects, quality and speed. How does the agent combine its two separated trusts together, the trust in the file provider’s capability in providing music files with good quality...
and the trust in the file provider's capability in providing a fast download speed, in order to decide if the file provider is trustworthy or not?

A Bayesian network provides a flexible method to solve the problem. A Bayesian network is a relationship network that uses statistic methods to represent probability relationships between different agents. Its theoretical foundation is the Bayes rule [13].

\[ p(h \mid e) = \frac{p(e \mid h) \cdot p(h)}{p(e)} \]

\( p(h) \) is the prior probability of hypothesis \( h \); \( p(e) \) is the prior probability of evidence \( e \); \( p(h \mid e) \) is the probability of \( h \) given \( e \); \( p(e \mid h) \) is the probability of \( e \) given \( h \).

A naive Bayesian network is a simple Bayesian network. It is composed of a root node and several leaf nodes. We will use a naive Bayesian network to represent the trust between an agent and a file provider.

Every agent develops a naive Bayesian network for each file provider that it has interacted with. Each Bayesian network has a root node \( T \), which has two values, “satisfying” and “unsatisfying”, denoted by 1 and 0, respectively. \( p(T = 1) \) represents the value of agent’s overall trust in the file provider’s competence in providing files. It is the percentage of interactions that are satisfying and measured by the number of satisfying interactions \( m \) divided by the total number of interactions \( n \). \( p(T = 0) \) is the percentage of not satisfying interactions.

\[ p(T = 1) = \frac{m}{n} \quad \text{(1)} \]

\[ p(T = 1) + p(T = 0) = 1 \]
The leaf nodes under the root node represent the file provider’s capability in different aspects. Each leaf node is associated with a conditional probability table (CPT). The node, denoted by FT, represents the set of file types. Suppose it includes five values, “Music”, “Movie”, “Document”, “Image” and “Software”. Its CPT is showed in table 1. It includes two columns of values. Each column follows one constraint, which corresponds to one value of the root node. The sum of values of each column is equal to 1.

<table>
<thead>
<tr>
<th></th>
<th>T = 1</th>
<th>T = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Music</td>
<td>$p(FT = &quot;Music&quot;</td>
<td>$T = 1)$</td>
</tr>
<tr>
<td>Movie</td>
<td>$p(FT = &quot;Movie&quot;</td>
<td>$T = 1)$</td>
</tr>
<tr>
<td>Document</td>
<td>$p(FT = &quot;Document&quot;</td>
<td>$T = 1)$</td>
</tr>
<tr>
<td>Image</td>
<td>$p(FT = &quot;Image&quot;</td>
<td>$T = 1)$</td>
</tr>
<tr>
<td>Software</td>
<td>$p(FT = &quot;Software&quot;</td>
<td>$T = 1)$</td>
</tr>
</tbody>
</table>

Table 1. The CPT of Node FT

$p(FT= "Music" | T=1)$ is the conditional probability with the condition that an interaction is satisfying. It measures the probability that the file involved in an interaction is a music file, given the interaction is satisfying. It can be computed according to the following formula:

$$p(FT = "Music" | T = 1) = \frac{p(FT = "Music", T = 1)}{p(T = 1)}$$

$p(FT= "Music", T=1)$ is the probability that interactions are satisfying and files involved are music files.

$$p(FT = "Music", T = 1) = \frac{m_1}{n}$$

$m_1$ is the number of satisfying interactions when files involved are music files.

$p(FT= "Music" | T=0)$ denotes the probability that files are music files, given interactions are not satisfying. The probabilities for other file types in Table 1 are computed in a similar way.

Node DS denotes the set of download speeds. It has three items, “Fast”, “Medium” and “Slow”, each of which covers a range of download speed.

Node FQ denotes the set of file qualities. It also has three items, “High”, “Medium” and “Low”. Its CPT is similar to the one in table 1.
Here we only take three aspects of trust into account. More relevant aspects can be added in the Bayesian network later to account for user preferences with respect to service.

Once getting nodes’ CPTs in a Bayesian network, an agent can compute the probabilities that the corresponding file provider is trustworthy in different aspects by using Bayes rules, such as $p(T=1 \mid FT=\text{"Music"})$ – the probability that the file provider is trustworthy in providing music files, $p(T=1 \mid FQ=\text{"High"})$ – the probability that the file provider is trustworthy in providing files with high quality, $p(T=1 \mid FT=\text{"Music"}, FQ=\text{"High"})$ – the probability that the file provider is trustworthy in providing music files with high quality. Agents can set various conditions according to their needs. Each probability represents trust in an aspect of the file provider’s competence. With the Bayesian networks, agents can infer trust in the various aspects that they need from the corresponding probabilities. That will save agents much effort in building each trust separately, or developing new trust when conditions change. After each interaction, agents update their corresponding Bayesian networks.

### 4.3 Evaluation of an Interaction

Agents update their corresponding Bayesian networks after each interaction. If an interaction is satisfying, $m$ and $n$ are both increased by 1 in formula (1). If it is not satisfying, only $n$ is increased by 1. Two main factors are considered when agents judge an interaction, the degree of their satisfaction with the download speed $s_{ds}$ and the degree of their satisfaction with the quality of the downloaded file $s_{fq}$. The overall degree of agents’ satisfaction with an interaction $s$ is computed as the following:

$$s = w_{ds} \cdot s_{ds} + w_{fq} \cdot s_{fq}, \text{ where } w_{ds} + w_{fq} = 1 \quad (2)$$

$w_{ds}$ and $w_{fq}$ denote weights, which indicate the importance of download speed and the importance of file quality to a particular agent (depending on the user’s preferences). Each agent has a satisfaction threshold $s_t$. If $s < s_t$, the interaction is unsatisfying; otherwise, it is satisfying.

### 4.4 The Procedure

In current file sharing peer-to-peer applications, users find files by using the search function. In most situations, they get a long list of providers for an identical file. If a user happens to select an unsuitable provider, who provides files with bad quality or slow download speed, the user will waste time and effort. If this situation happens several times, the users will be frustrated. In order to solve the problem, the mechanism of trust and reputation is used. Once an agent receives a list of file providers for a given search, it can arrange the list according to its trust in these file providers. Then the agent chooses the most trusted file providers in the top of the list to download files from. If the agent has no experiences with the file provider, it can ask other agents to make recommendations for it. The agent can send various recommendation
requests according to its needs. For example, if the agent is going to download a movie, it may care about the movie’s quality. Another agent may care about the speed. So the request can be “Does the file provider provide movies with good quality?”. If the agent cares both about the quality and the download speed, the request will be something like “Does the file provider provide files with good quality at a fast download speed?”. When other agents receive these requests, they will check their trust-representations, i.e. their Bayesian networks, to see if they can answer such questions. If an agent has downloaded movies from the file provider before, it will send recommendation that contains the value $p(T=1 \mid FT=\text{Music}, FQ=\text{High})$ to answer the first request or the value $p(T=1 \mid FT=\text{Music}, FQ=\text{High}, DS=\text{Fast})$ to answer the second request. The agent might receive several such recommendations at the same time, which may come from the trustworthy acquaintances, untrustworthy acquaintances, or strangers. If the references are untrustworthy, the agent can discard their recommendations immediately. Then the agent needs to combine the recommendations from trustworthy references and from unknown references together to get the total recommendation for the file provider:

$$r_{ij} = w_t \cdot \sum_{l=1}^{k} tr_{il} \cdot t_{lj} + w_s \cdot \sum_{z=1}^{g} t_{zj}, \text{ where } w_t + w_s = 1 \quad (3)$$

$r_{ij}$ is the total recommendation value for the $j^{th}$ file provider that the $i^{th}$ agent gets. $k$ and $g$ are the number of trustworthy references and the number of unknown references, respectively. $tr_{il}$ is the trust that the $i^{th}$ user has in the $l^{th}$ trustworthy reference. $t_{lj}$ is the trust that the $l^{th}$ trustworthy reference has in $j^{th}$ file provider. $t_{zj}$ is the trust that the $z^{th}$ unknown reference has in $j^{th}$ file provider. $w_t$ and $w_s$ are the weights to indicate how the user values the importance of the recommendation from trustworthy references and from unknown references. Since agents often have different preferences and points of view, the agent’s trustworthy acquaintances are those agents that share similar preferences and viewpoints with the agent most of time. The agent should weight the recommendations from its trustworthy acquaintances higher than those recommendations from strangers. Given a threshold $\theta$, if the total recommendation value is greater than $\theta$, the agent will interact with the file provider; otherwise, not.

If the agent interacts with the file provider, it will not only update its trust in the file provider, i.e. its corresponding Bayesian network, but also update its trust in the agents that provide recommendations by the following reinforcement learning formula:

$$tr_{ij}^\alpha = \alpha \cdot tr_{ij}^\phi + (1 - \alpha) \cdot \phi$$

(4)
\( n^\nu_{ij} \) denotes the new trust value that the \( i^{th} \) agent has in the \( j^{th} \) reference after the update; \( n^\phi_{ij} \) denotes the old trust value. \( A \) is the learning rate – a real number in the interval \([0,1]\). \( e_\alpha \) is the new evidence value, which can be -1 or 1. If the value of recommendation is greater than \( \theta \) and the interaction with the file provider afterwards is satisfying, \( e_\alpha \) is equal to 1; in the other case, since there is a mismatch between the recommendation and the actual experience with the file provider, the evidence is negative, so \( e_\alpha \) is -1.

Another way to find if an agent is trustworthy or not in telling the truth is the comparison between two agents’ Bayesian networks relevant to an identical file provider. When agents are idle, they can “gossip” with each other periodically, exchange and compare their Bayesian networks. This can help them find other agents who share similar preferences more accurately and faster. After each comparison, the agents will update their trusts in each other according the formula:

\[
n^\alpha_{ij} = \beta * n^\phi_{ij} + (1 - \beta) * e_\beta
\]

The result of the comparison \( e_\beta \) is a number in the interval \([-1, 1]\). \( \beta \) is the learning rate – a real number in the interval \([0,1]\) which follows the constraint \( \beta > \alpha \). This is because the Bayesian network collectively reflects an agent’s preferences and viewpoints based on all its past interactions with a specific file provider. Comparing the two agents’ Bayesian networks is tantamount to comparing all the past interactions of the two agents. The evidence \( e_\alpha \) in formula (4) is only based on one interaction. The evidence \( e_\beta \) should affect the agent’s trust in another agent more than \( e_\alpha \).

How do the agents compare their Bayesian networks and how is \( e_\beta \) computed? First, it is assumed that the structures of Bayesian networks of all agents have the same structure. Only the values in their Bayesian networks are compared. Suppose agent 1 will compare its Bayesian network (see figure 3) with the corresponding Bayesian network of agent 2. Agent 1 obtains the degree of similarity between the two Bayesian networks by computing the similarity of each pair of nodes (T, DS, FQ and FT), according to the similarity measure based on Clark’s distance \([16]\), and then combining the similarity results of each pair of nodes together.

\[
e_\beta = 1 - 2^4 \sum_{i=1}^{4} (w_i^1 * c_i), \text{ where } w_1^1 + w_1^2 + w_1^3 + w_1^4 \leq 1
\]

\[
c_1 = \sqrt{\frac{(v1_{11} - v2_{11})^2 + (v1_{12} - v2_{12})^2}{(v1_{11} + v2_{11})^2 + (v1_{12} + v2_{12})^2}}
\]
\[ c_i = \frac{1}{2} \sum_{j=1}^{h_i} \left( \frac{(v1_{ij1} - v2_{ij1})^2}{(v1_{ij1} + v2_{ij1})^2} \right), \quad \text{where } i = 2, 3, 4 \quad (8) \]

\( w1_1, w1_2, w1_3 \) and \( w1_4 \) are the weights of the node T, DS, FQ, and FT, respectively, related to agent 1, which indicate the importance of these nodes in comparing two Bayesian networks. \( c_1, c_2, c_3 \) and \( c_4 \) are the results of comparing agent 1 and agent 2’s CPTs about node T, DS, FQ and FT. Since the node T is the root node and it has only one column in its CPT, while other nodes (DS, FQ, FT) are the leaf nodes and have two columns of values in theirs CPTs, we compute \( c_1 \) differently from \( c_2, c_3, \) and \( c_4. \) \( h_i \) denotes the number of values in the corresponding node. \( h_2 = 3, h_3 = 3; h_3 = 5. \) \( v1_{i11} \) and \( v1_{i12} \) are the values of \( p(T = 1) \) and \( p(T = 0) \) related to agent 1. \( v2_{i11} \) and \( v2_{i12} \) are the values of \( p(T = 1) \) and \( p(T = 0) \) related to agent 2.\( v1_{ij1} \) and \( v2_{ij1} \) are the values in agent 1’s CPTs and agent 2’s CPTs, respectively.

The idea of this metric is that agents compute not only their trust values, their CPTs, but also take into account their preferences (encoded as the weights, \( w1_1, w1_2, w1_3, w1_4 \)). So agents with similar preferences, such as the importance of file type, quality, download speed, will weight each other’s opinions higher.
References


[22] L. Xiong, L. Liu, “A reputation-based trust model for peer-to-peer ecommerce communities”. In IEEE Conference on ECommerce (EC’03), 2003