A Survey Study on Trust Management in P2P Systems

Chen Ding, Chen Yueguo, Cheng Weiwei
Department of Computer Science, School of Computing
National University of Singapore

ABSTRACT

Trust management in P2P system is used to detect malicious behaviors and to promote honest and cooperative interactions. In this report, we first present a computational model that shows the relationships based on trust and reputation. This model can be implemented in a real system to consistently calculate trust and reputation scores for each peer. Furthermore, we introduce two reputation-based trust management solutions: DMRep and EigenRep. DMRep implements the trust management at both data management and semantic level. It is based on analyzing earlier transactions of peers and deriving reputation values from those transactions. EigenRep tries to minimize the impact of malicious peers by using global reputation values. Finally, we discuss the security concerns involved in P2P trust management through two protocols, P2PRep and XRep, which try to provide support on authenticity of the votes, and protect the integrity of messages exchanged. In conclusion, we point out the potential future trend or works to be done in P2P trust management.

1. INTRODUCTION

1.1 Malicious Usages and Attacks in P2P

Ever since Napster [1] that allowed individuals to trade in the music commodity, P2P applications of many different kinds have become a popular medium to share huge amounts of data. Some allow distributed file sharing, while others facilitate real-time communications. Many large vendors are beginning to integrate P2P applications with their core business programs. Their ability to build an extremely resourceful system by aggregating the resources of a large number of independent nodes enables P2P systems to compete the capabilities of many centralized systems for relatively little cost. P2P systems are believed to remain an important approach and continue to gain popularity and impact in the future due to the anonymity, cost sharing, dynamism and scalability that P2P systems possess.

However, there are important challenges to overcome before the full potential of P2P systems can be realized. The main goal of the earlier P2P systems is the capability of aggregating resources, which assumes certain honesty level of peers. However, as P2P systems grow tremendously in size, there will be a considerable number of malicious peers who bring security attacks and threats to the whole network. In a distributed infrastructure without centralized server for authority, providing security mechanism is more complicated than in server-centric solutions, as

the existence of multiple sites increases the vulnerability and security efforts must be replicated at multiple sites. Like traditional client-server systems, there are malicious peers trying to attack against integrity, confidentiality and authenticity, which are the main issues in the area of system securities. Therefore security issues are one of the major challenges that need to be carefully analyzed and addressed, especially for fully decentralized unstructured P2P systems, like Gnutella [2].

1.2 Traditional Security Solutions

However traditional solution (public key encryption) for such attacks requires a trusted third party to act as certificate authority. In particular P2P environments, lack of centralized trustworthy party, and the existence of multiple anonymous peers have handled the public key solution. On the other hand, some proposed public key mechanism suited for P2P system requires the declaration of the real IP and port. However, this approach compromises the anonymity by revealing IP and port to other peers.

Moreover, there are seemingly less serious problems in P2P applications, which also downgrade the credibility and trustworthiness of the system. For example, in a P2P environment the collaboration of all peers is very important for the correct functioning of the system. Every peer is soaking up network bandwidth. If too many users access the same network resource, the network bandwidth may be used up, resulting in a denial of service (DoS). P2P network is prone to DoS attack, which takes advantage of the flooding algorithm that is used for querying the P2P network. A malicious node continuously issues queries with high TTL values on the network, which generate huge amount of network traffic rendering the network unusable by other honest peers. Other examples are as follows. The peer who offers a resource may go offline while other fellow peers are downloading from it; a malicious peer may just simply route a query to a non-existent peer or an unreliable peer with long latency. Malicious or improper usages like these cannot be addressed by security solutions, as these problems relate more with trustworthiness rather than security. Therefore there is demand for mechanisms to maintain the trust of P2P systems to promote peers for correct and positive usage so as to keep the trust level of the whole system.

1.3 Trust Management Approach

These issues have motivated substantial research on trust management in P2P networks. Trust management is a successful approach that helps to maintain overall credibility level of the system as well as to encourage honest and cooperative behavior. The intuitive motivation of trust management is as follows. Since in P2P system there is no central authority that can authenticate and guard against the actions of malicious peers, it is up to the peer to protect itself and to be responsible for its own actions. Consequently, each peer in the system needs to somehow evaluate information received from another peer in order to determine the trustworthiness of both the information as well as the sender. This can be achieved in several ways such

as relying on direct experiences or acquiring reputation information from other peers [3].

Trust management is shown to be nicely suited to those problems briefly discussed in section 1.2. Particularly trust management systems are classified into three categories, reputation-based trust systems, policy-based trust systems, and social network-based trust systems. Section 2 will describe trust and trust management systems in further detail.

1.4 Goal and Organization of This Survey

In this survey, we specially look into the area of reputation-based trust management systems. The goal of this survey is to discover the current achievements and future trend of the researches on the P2P trust management. We briefly look into the literature of the trust management, and then we study some remarkable articles that discuss the modeling, computation, maintenance and security concerns of trust management. These articles propose trust management solutions, such as EigenRep [4], DMRep [5], P2PRep [6] and XRep [7]. While EigenRep and DMRep contribute more on proposing a computational model of trust, P2PRep and XRep protocols look into the effect of combining security solution into trust management approach. Advantages and disadvantages of each of those approaches are presented after critical analysis and discussions. Future trend is also given thereafter based on the findings of this survey paper and conclusive analysis.

The rest of this paper is organized as follows. Section 2 shows the works and achievements in the literature of trust management. Section 3 presents a description on computational model for P2P reputation-based trust management. While section 4 presents the detail of EigenRep and DMRep protocols that focus on the computational model of reputation, section 5 talks about the idea of P2PRep and XRep that focus on the security concerns of reputation-based systems. Discussions for each protocol are also presented in section 4 and section 5 respectively. Section 6 points out the future trend of trust management in P2P systems and concludes this paper.

2. LITERATURE REVIEW

Based on the approach adopted to establish and evaluate trust relationship between peers, trust management in P2P system can be classified into 3 categories [8]: credential and policy-based trust management, reputation-based trust management, and social network-based trust management as shown in Figure 1.

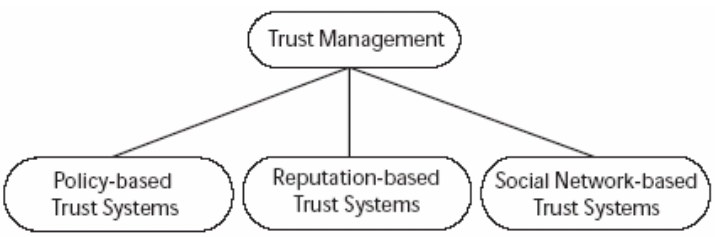

Figure 1: Trust management taxonomy

2.1 Policy-based Trust Management Systems

In credential and policy-based trust management systems, peers use credential verification to establish a trust relationship with other peers [9, 10]. Since the primary goal of such systems is to enable access control, their concept of trust management is limited to verifying credentials and restricting access to resource according to application-defined policies [11].

PolicyMaker [12] is a trust management system that facilitates the development of security features including privacy and authenticity for different kinds of network applications. It provides each peer with local control to specify its policies: using PolicyMaker a peer may grant another peer access to its service if the providing peer can determine that the requesting peer's credentials satisfy the policies.

The policy-based access control trust mechanisms do not incorporate the need of the requesting peer to establish trust in the resource-owner; therefore, they by themselves do not provide a complete generic trust management solution for all decentralized applications.

2.2 Reputation-based Trust Management Systems

Trust management is any mechanism that allows establishing mutual trust. Reputation is a measure that is derived from direct or indirect knowledge on earlier interactions of agents, and it is used to access the level of trust an agent puts into another agent. Thus, reputation-based trust management is one specific form of trust management.

Reputation-based trust management systems on the other hand provide a mechanism, by which a peer requesting a resource may evaluate the trust in the reliability of the resource and the peer providing the resource. Examples of such systems include SPORAS and HISTOS [13], XRep [7], NICE [14], DCRC/CORC [15], EigenRep [4], etc. Peers in such systems establish trust relationships with other peers and assign trust values to these relationships. Trust value assigned to a trust relationship is a function of the combination of the peer's global reputation and the evaluating peer's perception of that peer.

Cornelli et al. [5] proposed an approach to share information about peers' reputation based on the distributed polling algorithm. A node looking for information broadcasts a query and receives back QueryHits. It then chooses some results and asks its peers to vote on the reputation of the nodes that sent it QueryHits. Once votes are received a node contacts voters directly asking to confirm a validity of vote. The data is retrieved from the peer with highest reputation.

Abdul-Rahman et al. [8] proposed a decentralized approach to trust management and a recommendation protocol to compute trust related information. Each entity has its own trust relationship database. They make use of different trust categories (which aspect is trusted), a scale of trust values on recommendations and direct (related to one aspect) trust values. In order to get a recommendation, an entity sends recommendation requests to its trusted recommenders. Results from different paths (from requester to target entity) are collected and averaged. Each path is computed

based on recommender trust value of recommenders and recommended trust value returned.

NICE [14] is a platform for implementing cooperative applications over the Internet. It works in a purely decentralized fashion and each peer stores and controls data that benefits itself. Applications based on NICE barter local resources in exchange for access to remote resources. NICE provides three main services: resource advertisement and location, secure bartering and trading of resources, and distributed trust evaluation. NICE uses two trust mechanisms to protect the integrity of the cooperative groups: trust-based pricing and trust-based trading limits. One of the main contributions of the NICE approach is the ability of good peers to form groups and to isolate malicious peers.

2.3 Social Network-based Trust Management Systems

Social network-based trust management systems utilize social relationships between peers when computing trust and reputation values. In particular, these systems form conclusions about peers through analyzing a social network that represents the relationships within a community.

Marsh [16] is among the first to try to give a formal treatment of trust that could be used in computer science. His model is based on social properties of trust and presents an attempt to integrate all the aspects of trust taken from sociology and psychology. Several limitations exist in his simple trust model: too strong sociological foundation makes the model rather complex and cannot be easily implemented; the agents cannot collectively build a network of trust due to the model puts the emphasis on agent's own experiences.

Other examples of such trust management systems include Regret [17] that identifies groups using the social network, and NodeRanking [18] that identifies experts using the social network.

3. A COMPUTATIONAL MODEL

In this section, we look for the relationships based on trust and reputation, and introduce a computational model [22] based on sociological and biological understanding of trust management.

3.1 Understanding Trust & Reputation

After reviewing several important studies on trust management, reputation and trust have been found to provide useful intuition or services for many systems [19]. It is also proved that reputation-based system (e.g. the feedback rating system used in eBay) does encourage transactions [20]. However, all of these studies model neither how reputation is built nor how trust is derived from reputation.

When facing social dilemmas, trustworthy individuals tend to trust others with a reputation for being trustworthy and shun those deemed less so. It is always the case in real world where everyone in a society might not learn the same norms in all

situations. Only in an environment where individuals "regularly" perform reciprocity norms, there is an incentive to acquire a reputation for reciprocities actions.

The model introduced in this section is built on such an environment where reciprocity norms are expected. The intuition behind this model is inspired by Ostroms' 1998 Presidential Speech [21] to the American Political Society, which proposed a qualitative behavioral model for collective action.

Consider the scenario that agent a_j is evaluating a_i's reputation for being cooperative. Define **embedded social network** of a_j as the set of all the agents that a_j asks for this evaluation. In this way, the reputation of an agent a_i is relative to the particular embedded social network in which a_i is being evaluated. Moreover, to be simplicity, we assume such embedded social networks are taken to be static (i.e. no new agents are expected to join or leave) and the action space is restrict to be {cooperate, defect}.

Figure 2 shows the reinforcing relationships among the three highly related concepts: reciprocity, trust and reputation. The direction of the arrow indicates the direction of influence among the variables.

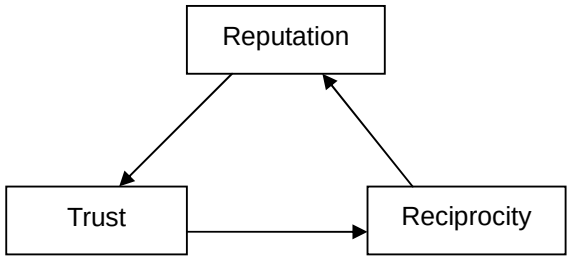

Figure 2: Simple relationship model

For an agent a_j with a embedded social network A: increase a_j's reputation in A should also increase the trust from other agents for a_j, and the increase in a_i's trust of a_j should also increase the likelihood that a_i will reciprocate positively to a_j's action; since a_j's reciprocation action to others in A increased, its reputation in A should also be increased.

3.2 Computation Model

Next, we shall see how the three concepts are defined and how to operationalize this model into mathematical statements.

Reciprocity is defined as mutual exchange of deeds (such as favor or revenge). Two types of reciprocity are considered in this model: direct reciprocity refers to interchange between two concerned agents while indirect reciprocity refers to interchange between two concerned agents interceded by mediating agents in between.

The model defines **reputation** as perception that an agent creates through past actions about its intentions and norms. Mathematically, let $\theta_{ji}(c)$ represent a_i's reputation in an embedded social network of concern to aj for a context c. This value is subjective to every other agent since the embedded social network difference when

a_i connects to different a_j. In this way $\theta_{ji}(c)$ measures the likelihood that a_i reciprocates a_j's actions.

In this model, **trust** is defined as a subjective expectation an agent has about another's future behavior based on the history of their encounters. Thus to evaluate the trustworthiness of a_i, let $D_{ji}(c)$ represents history of encounters that a_j has with a_i within the context c. Moreover, we should take note that trust is a subjective quantity calculated based on the two agents concerned in a dyadic encounter. So we can model trust using $T(c) = E[\theta_{ji}(c) | D_{ji}(c)]$. The higher the trust level for agent a_i, the higher the expectation that a_i will reciprocate agent a_j's action.

We describe the computational model in detail with the following scenario:

Consider two agents *a* and *b*, and assume that they care about each others' actions within a specific context c. In this scenario, we assume that agent *a* always perform "cooperate" actions and that *a* is assessing *b*'s tendency to reciprocate cooperative actions. We assume the notations used for this scenario are:

θ_{ab} : *b*'s reputation in the eyes of *a*

$X_{ab}(i)$: The *ith* encounter between *a* and *b*

$$X_{ab}(i) = \begin{cases} 1 & if\_b's\_action\_is\_cooperate \\ 0 & otherwise \end{cases}$$

D_{ab}: History. The set of *n* previous encounters between *a* and *b*

$D_{ab} = \{Xab\ (1), X_{ab}\ (2), \ldots, X_{ab}\ (n)\}$

Let *p* be the cooperative actions by agent b towards *a* in the *n* previous encounters, *b*'s reputation θ_{ab} could be modeled by a simple proportion function of *p* cooperative actions over *n* encounters. In statistics, a proportion random variable can be modeled as a Beta distribution: $p(\hat{\theta}) = Beta(c1,c2)$ where $\hat{\theta}$ represents an estimator for θ.

If agents *a* and *b* are complete strangers, when they first meet, their estimate for each other's reputation is assumed to be uniformly distributed across the reputation's domain:

$$p(\hat{\theta}) = \begin{cases} 1 & 0 < \hat{\theta} < 1 \\ 0 & otherwise \end{cases}$$

In this model, the beta distribution will be uniform when $c_1=c_2=1$.

Now we have a simple estimator for θ_{ab} which is the proportion of cooperation in *n* finite encounters: $\hat{\theta}_{ab} = p/n$. Assuming that each encounter's cooperation probability is independent of other encounters between *a* and *b*, the likelihood of *p* cooperations and *(n-p)* defections can be modeled as $L(D_{ab} | \hat{\theta}) = \theta^p (1-\theta)^{n-p}$.
Combining the prior and the likelihood, the posterior estimate for $\hat{\theta}$ becomes:
$p(\hat{\theta} | D) = Beta(c_1+p, c_2+n-p)$. As we mentioned previously, trust toward b from a is the conditional expectation of reputation $\hat{\theta}$ so it can be computed by

$$T_{ab} = p(x_{ab}(n+1) = 1 | D) = E[\hat{\theta} | D] = \frac{c_1 + p}{c_1 + c_2 + p}$$

4. REPUTATION-BASED TRUST MANAGEMENT

In this section, two reputation-based trust management models, the DMRep and EigenRep, are described in details. These two models focus on how the reputation values are accessed and computed in a fully decentralized Peer-2-Peer System.

4.1 DMRep

DMRep [5] is an approach that addresses the problem of reputation-based trust management at both the data management and the semantic level. This approach assumes that the probability of cheating within a society is comparably low, and thus it becomes more difficult to hide malicious behavior.

Similarly to the computational model in the previous section, this approach can be interpreted as a simple method of data mining using statistical data analysis of former tractions. It is based on analyzing earlier transaction of peers and deriving from that the reputation of peer. The data necessary for performing the analysis is provided by a decentralized storage method (P-Grid [23]).

4.1.1 Manage Trust in a Decentralized System

In a fully decentralized P2P system, the problem of reputation-based trust management can be defined as follows:

Let P denote the set of all peers. The behavior data B are observations $t(q, p)$ a peer $q \in P$ makes when he interacts with a peer $p \in P$. Based on those observations one can access the behavior of p based on the set $B(p) = \{t(p, q) \text{ or } t(q, p) \mid q \in P\} \subseteq B$. The equation means: we take into account all reports about transactions that are made about p, but as well all reports about transactions that are made by p.

Two problems reputation-based trust management faced in a decentralized environment:

- The semantic question: what is the model that allows to access trust of p based on the data $B(p)$ and B?
- The data management question: how can the necessary data $B(p)$ and B be obtained to compute trust according to the semantic trust model with reasonable effort?

Looking at these two questions more closely, one sees that they cannot be investigated in separation. Next we will show how these two questions are solved in the DMRep Model.

In a fully decentralized environment, a peer q has no access to the global data $B(p)$ and B. Rather, it has to rely on the data obtained from direct interaction, or it can obtain indirectly through a limited number of referrals from witnesses $r \in W_q \subseteq P$. Thus q has information $B_q(p) = \{t(q, p) \mid t(q, p) \in B\}$ and $W_q(p) = \{t(r, p) \mid r \in W_q, t(r, p) \in B\}$ to determine the reputation of a peer p.

Since we assume that usually trust exists and malicious behavior is the exception, information on dishonest interactions can be considered as relevant. Thus a peer p can, in case of malicious behavior of q, file a complaint $c(p, q)$. Complaints are the only

behavioral data B used in the model. To disseminate those behavioral data, a peer forwards its complaints to other peers.

4.1.2 Overview of DMRep

4.1.2.1 Global Trust Model

Let us look at a simple situation where p and q interact and later on r wants to determine the trustworthiness of p and q. We assume that p is cheating and q is honest. After their interaction, q will file a complaint about p, which is perfectly fair. However, q will also file a complaint about p, in order to hide its misbehavior. For an external observer r, it cannot distinguish whether p or q is cheating. This means, a social mechanism to detect dishonest behavior will not work for private interactions.

If p continues to cheat, it will be in trouble. Assume it cheats in another interaction with s. then r will observe that p complaints about both q and s, whereas both q and s complain about p. r can conclude that it is very probable that p is the cheater.

Based on the above scenario, the reputation of a peer can be computed by the equation $\theta(p) = |\{c(p,q) \mid q \in P\}| \times |\{c(q,p) \mid q \in P\}|$. High values of $\theta(p)$ indicate that p is not trustworthy. The reputation is computed based on the global knowledge on complaints. Whereas it is straight forward a peer to collect all information about its own interactions with other peers, it is very difficult for it to obtain all the complaints about any other specific peer. At this point, a perspective of data management is needed in order to solve the problem.

4.1.2.2 Decentralized Data Management

P-Grid method [23] (similar methods like [24] & [25] could also be considered) is used to store data in a P2P network in a scalable way. Figure 3 shows a simple example of a P-Grid. Six peers together support a virtual binary search tree of depth 2. Each peer is associated with one path of the search tree.

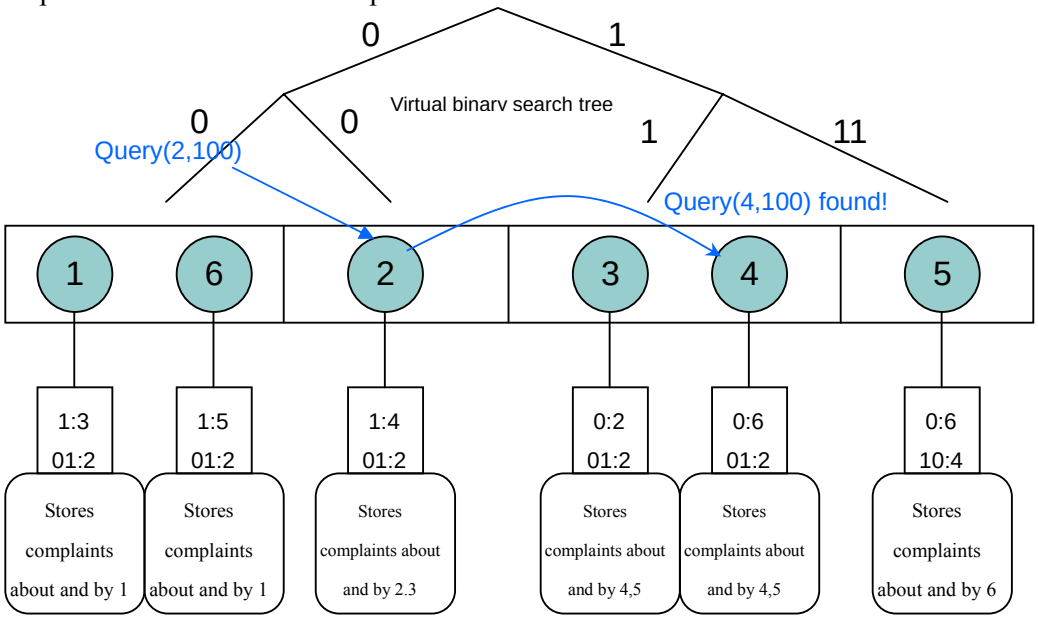

Figure 3: Example P-Grid

It stores data items for which the associated path is a prefix of the data key. For the trust management application this are the complaints indexed by the peer number. Each peer can serve any search request. Either the requested key has as prefix the search path associated with the peer processing the request, or the peer can use its routing table to forward the query to another peer that is responsible for the complementary part of the search tree. We demonstrate in the figure the processing of one sample query *query(2,100)* using this search structure. As peer 2 is not associated with keys start with 1 it looks up in its routing table peer 4, who can answer the query, as it stores all data with keys, that start with *10*.

This access method is organized in a peer-to-peer fashion, i.e. there exists no central database. Two request forms are allowed in this model:

- *Insert(a,k,v),* where *a* is an arbitrary peer in the network, *k* is the key value to be searched for, and *v* is the data value associated with the key.
- *Query(a, k): v*, where *a* is an arbitrary peer in the network, which returns the data values *v* for a corresponding query *k*

The following properties that P-Grid satisfies enable it to be the access structure:

1. There exists an efficient decentralized bootstrap algorithm which creates the access structure without central control.
2. The search algorithm consists of randomly forwarding the requests from one peer to the other, according to routing tables that have been constructed in the bootstrap algorithm.
3. All algorithms scale gracefully. Time and space complexity are both *O(logn)* where *n* is the number of peers.

4.1.2.3 Local Trust Model

Since every peer p can file a complaint about q at any time, it stores the complaint by sending messages *insert(a_1, key(p), c(p, q))* and *insert(a_2, key(q), c(p, q))* to arbitrary peers: a_1 and a_2 in the P-Grid storage structure. The insertion algorithm of P-Grid forwards the complaints to one or more peers storing complaints about *p* respectively *q*. In this way the desired re-aggregation of the complaint data is achieved.

As described previously, when a peer *p* wants to evaluate the trustworthiness of another peer *q*, it starts to search for complaints on *q*. Based on the data obtained, *p* uses the trust function $\theta(q)$ to decide upon trustworthiness.

However, there are two problems when the complaints data are obtained. First, the witness peers might also be malicious and will provide wrong *t(r, p)* values. This can be dealt with by checking the trustworthiness of the witness in the next step. However eventually this would lead to the exploration of the whole network, which is clearly undesirable. Second, even if the referring peer is honest, it may not be reachable reliably over the network. This might distort the quality of the data received about the behavior of other peers.

To address the above problems, DMRep proceeds as follows:

DMRep assumes that the peers are only malicious with a certain probability $\pi \le \pi_{\max} < 1$. Then it configures the storage infrastructure, such that the

number r of replicas satisfies on average $\pi^{r}_{\max} < \varepsilon$, where ε is an acceptable fault-tolerance.

Thus, if the same data about a specific peer is received from a sufficient number of replicas, no further checks are need. If the data is insufficient or contradictory, DMRep will continue to check. In addition, DMRep also limits the depth of the exploration of trustworthiness of peers to limit the search space, and might end up in situations, where no clear decision can be made. However, these cases should be rare.

4.1.3 DMRep Algorithms

As shown in figure 4, when a peer p evaluates the trustworthiness of a peer *q*, it retrieves from the decentralized storage complaint data by submitting messages *query(a, key(q))* to arbitrary peers a. In order to obtain multiple referrals it will do this repeatedly for a number (*n*) of times. As a result it obtains a set:

$W = \{(cr_i(q), cf_i(q), s_i, f_i) \mid i = 1...w\}$

where

- w is the number of different witness found
- s_i is the identifier of the *ith* witness,
- f_i is the frequency with which witness si is founded.
- $cr_i(q)$ and $cf_i(q)$ are number of complaints q received and filed respectively.

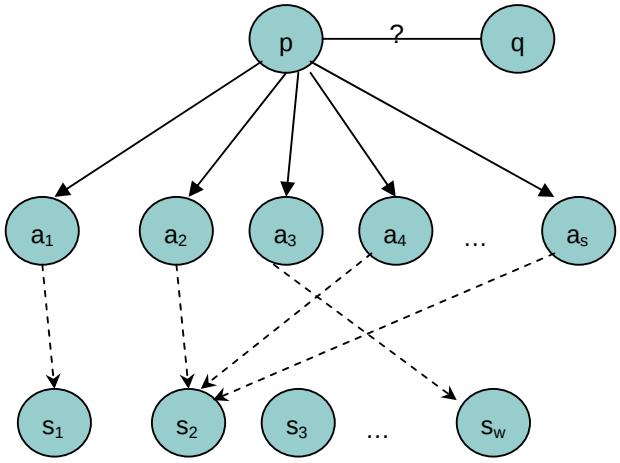

Figure 4: Check complaints

Different frequencies f_i indicate that not all witnesses are found with the same probability due to the non-uniformity of the P-Grid structure. This non-uniformity impacts not only query message but also storage messages. Thus witnesses found less frequently will probably also not receive as many storage messages when complaints are filed. Thus the number of complaints they report will tend to be low. Therefore we normalize the values by using the frequencies observed during querying. The following function compensates for the variable probability of a peer to be found,

$$cr_i^{norm}(q) = cr_i(q)(1-(\frac{s-f_i}{s})^s), i = 1,...,w$$

$$cf_i^{norm}(q) = cf_i(q)(1-(\frac{s-f_i}{s})^s), i = 1,...,w$$

The factor $1-(\frac{s-f_i}{s})^s$ corresponds to the probability of not finding witness *i* in *s* attempts. This probability is high when the f_i value is high and vice versa, which leads to the desired compensation effect.

The following decision criterion is used to decide when to consider a peer trustworthy.

$$decide_p(cr_i^{norm}(q), cf_i^{norm}(q)) =$$

if

$$cr_i^{norm}(q)cf_i^{norm}(q) \le (\frac{1}{2} + \frac{4}{\sqrt{cr_p^{avg} cf_p^{avg}}})^2 cr_p^{avg} cf_p^{avg}$$

then 1 else -1

where cr_p^{avg} and cf_p^{avg} are the average number of complaints received and complaints filed.

This criterion is a heuristics. It is based on the argument that, if an observed value for complaints exceeds the general average of the trust measure too much, the peer must be dishonest. And the factor is determined by preformed a probabilistic analysis by modeling peer interaction as Poisson processes.

Two strategies are proposed in the DMRep Model to determine trust. The simplest strategy *(ExploreTrustSimple (p, q)*) is to take a majority decision and in case of a tie return "undecided". A more sophisticated strategy (*ExploreTrustComplext(p, q, l))* includes the checking of the witness. The underlying assumption for this strategy is that the probability π that a peer is not trustworthy is higher than the tolerance for a wrong assessment, i.e. $\pi \geq \varepsilon$, but that two witnesses giving the same assessment are acceptable, i.e. $\pi^2 \leq \varepsilon$, therefore, in case of a single witness, it always needs to be checked. If after the check, a majority decision can be made, it is accepted.

4.1.4 DMRep Discussion

DMRep is an approach that addresses the problem at both the data management and the semantic level. The principal advantage of this approach is that it has an efficient way of storing and retrieving trust data and does not flood every peer in the system with queries about other peers, thus limiting storage and bandwidth costs. It is thus more scalable than approaches that broadcast trust queries to all peers in the system.

The main disadvantage, however, is that a peer is forced to store data owned by other peers and does not have local control over the treatment of that data. Therefore, the system is not truly decentralized because peers have to implicitly agree to not alter data owned by others. It also does not employ any kind of mechanism to authenticate messages or explicitly protect the identity of peers. And DMRep assumes that usually trust exists and malicious behavior is the exception, thus it is not suitable for the environment with high cheating rates.

Since DMRep uses complaints to report behaviors of peers and so relies on a negative reputation-based scheme. Global trust information in the form of complaints is stored across peers. This leads to two problems. The first is that the entry and departure of peers from the system may result in important trust information being lost, resulting in a decrease of the fault-tolerance ability of the system. The second affects reliability since it is possible that a peer may end up storing complaints about itself which it may be motivated to alter or destroy. P-Grid addresses both these concerns by making trust data redundant across peers improving both fault-tolerance and reliability. However, DMRep always regards a new peer as a good peer since there is no complaint data associated with it. It cannot prevent the case that a malicious peer frequently leaves the network and re-join as a new peer.

4.2 EigenRep

4.2.1 Reputation Values

4.2.1.1 Local Reputation Values

The evaluation of a peer to another peer can simply rely on the transactions between the two peers. For example, peer *i* may have a positive opinion on peer *j* if it successfully download an authentic file from peer *j*. On the other hand, it will have a negative opinion if the file downloaded is inauthentic, or if the download fails. In this way, EigenRep defines a local reputation value s_{ij} as the sum of the ratings of the individual transactions that peer *i* has downloaded from peer *j,* where $s_{ij} = sat(i, j) - unsat(i, j)$. Here, *sat(i, j)* represents the number of satisfactory transactions peer *i* has had with peer *j*. While *unsat(i, j)* represents the number of unsatisfactory transactions. In this way, a peer evaluates the other peers by its own experience. However, its own experience is very limited because a peer usually has transactions with only a small number of peers in the system. How should a peer evaluate another peer on which it has no experience? It should get evaluation from other peers which may be familiar with the unknown peer. In this way, peers share their experience (local evaluation values) with each other.

Since there is no centralized peer that stores and manages all the local reputation values in P2P system, it gives rise to a challenge that how to aggregate these local reputation values. EigenRep presents an efficient way to manage and use the reputation values with minimal overhead in terms of message traffic in the system.

The local reputation values are the basis of a reputation system. The numbers of

transactions may vary greatly in different peers, which may result in the large variation in local reputation values. Consequently, it will be better to normalize the reputation values before peers sharing them. EigenRep defines a normalized local reputation value c_{ij}, where $c_{ij} = \frac{\max(s_{ij},0)}{\sum_j \max(s_{ij},0)}$.

This definition maps all local reputation values s_{ij} to a value from 0 to 1. Furthermore, peer i has a normalized local reputation vector, $\vec{c}_i = (c_{i1},...,c_{iN})^T$, which contains all the normalized local reputation values of peer i to the peers in the system. As the limited experience of a peer, most entries of its normalized local reputation vector will be zeros. The advantage of this definition is that, $\|\vec{c}_i\|_1 = 1$, *i.e.* $\sum_{j=1}^{N} c_{ij} = 1$. It determines the convergence of iterations in the following algorithms in EigenRep. However, the definition still has some drawbacks. First, it does not distinguish between the peers that peer i has no experience and the peers that peer i has poor experience since the normalized local reputation values of them are all zeros. Second, for new peers, which has no experience on the other peers, $\sum_j \max(s_{ij},0) = 0$. So the normalized local reputation value can not be defined in this case.

4.2.1.2 Global Reputation Values

Based on the normalized local reputation values, peer i can get experience on peer k by ask its acquaintances about their opinions on peer k. And it weights its acquaintances' opinions by the trust peer i places in them.

$$t_{ik} = \sum_j c_{ij} c_{jk}$$

Similarly, peer i can ask its friends (acquaintances) about their opinions on all peers.

$$\vec{t}_i = C^T \vec{c}_i \text{, i. e., } \begin{pmatrix} t_{i1} \\ \cdots \\ t_{ik} \\ \cdots \\ t_{iN} \end{pmatrix} = \begin{pmatrix} c_{11} & \cdots & c_{k1} & \cdots & c_{N1} \\ \cdots & & & & \cdots \\ c_{1k} & \cdots & c_{kk} & \cdots & c_{Nk} \\ \cdots & & & & \cdots \\ c_{1N} & \cdots & c_{kN} & \cdots & c_{NN} \end{pmatrix} \begin{pmatrix} c_{i1} \\ \cdots \\ c_{ik} \\ \cdots \\ c_{iN} \end{pmatrix}$$

In this way, peer i broadens its own experience. However, $\vec{t}_i$ only reflects the experience of its friends, which may still be limited. To broaden the view further, peer i can ask its friend again, and weights their opinions using the new experience it gets last time ($\vec{t}_i^{(2)} = (C^T)^2 \vec{c}_i$). This is can be interpreted as peer i ask its friends' friends. If it continues in this manner, $\vec{t}_i^{(k+1)} = C^T \vec{t}_i^{(k)}$ and $\vec{t}_i^{(k)} = (C^T)^k \vec{c}_i$. If k is large enough, $\vec{t}_i^{(k)}$ will eventually converge to the left principle eigenvector of C regardless of $\vec{c}_i$ ($\vec{c}_i \neq 0$). This is determined by C is irreducible and aperiodic, which can be deduced from the definition of $\vec{c}_i$. We call the converged vector, $\vec{t}$, global reputation vector. An element of $\vec{t}$, for example, t_j, quantifies how much trust the system as a

whole places peer *j*.

If one peer can collect all the normalized local reputation vectors, which means it has the whole information of matrix *C*, it will be able to calculate the global reputation vector $\vec{t}$ using a non-distributed algorithm.

$$\vec{t}^{(0)} = \vec{e};$$
$$repeat$$
$$\quad \vec{t}^{(k+1)} = C^T \vec{t}^{(k)};$$
$$\quad \delta = \left\| \vec{t}^{(k+1)} - \vec{t}^{(k)} \right\|$$
$$until\ \delta < \varepsilon$$

4.2.1.3 Practical Issues

In a P2P system, there are some peers that firstly join the system, and they are usually known to be trusted by other peers. These peers are pre-trusted peers which can be assigned some per-trust values in the process of calculation. This can be done through a pre-trusted vector, $\vec{p}$.

An element of $\vec{p}$, $p_i = \begin{cases} 1/|P|, & if\ \ i \in P \\ 0, & otherwise \end{cases}$

$\vec{p}$ can be used in three ways.

First, it can be used as the starting vector of non-distributed algorithm.

Second, for new peers, their normalized local reputation values can not be defined since $\sum_j \max(s_{ij}, 0) = 0$. In this case, we can use $\vec{p}$ as the new peers' local reputation values.

Third, to weaken the effects of malicious collectives, the calculation of global reputation vector can place some trust in peers that are not part of collective, for example, pre-trust peers.

$$\vec{t}^{(k+1)} = (1-a)C^T \vec{t}^{(k)} + a\vec{p}$$

According to these practical issues, the non-distributed algorithm can be modified as follows.

$$\vec{t}^{(0)} = \vec{p};$$
$$repeat$$
$$\quad \vec{t}^{(k+1)} = C^T \vec{t}^{(k)};$$
$$\quad \vec{t}^{(k+1)} = (1-a)C^T \vec{t}^{(k+1)} + a\vec{p}$$
$$\quad \delta = \left\| \vec{t}^{(k+1)} - \vec{t}^{(k)} \right\|$$
$$until\ \delta < \varepsilon$$

4.2.2 Reputation Algorithm

4.2.2.1 Distributed Algorithm

However, in a P2P system, no peer can collect all the normalized local reputation vectors from the other peers. Consequently, non-distributed algorithm is not practical. Therefore, peers need to cooperate to compute global trust vector based on distributed algorithms. EigenRep provides such a distributed algorithm which costs minimal computation, storage, and message overhead.

The distributed algorithm is based on the component-wise version of $\vec{t}^{(k+1)} = (1-a)C^T\vec{t}^{(k)} + a\vec{p}$.

$$t_i^{(k+1)} = (1-a)(c_{1i}t_1^{(k)} + ... + c_{Ni}t_N^{(k)}) + ap_i$$

Here, $t_i^{(k+1)}$ represents the $k+1$ times of global reputation value of peer i. It is calculated and stored by peer i itself. The distributed algorithm is shown as follows:

A_i: set of peers who have downloaded files from peer i;

B_i: set of peers from which peer i has downloaded files;

Each peer i do{

Query all peers $j \in A_i$ for $t_j^{(0)} = p_j$;

Repeat

Compute $t_i^{(k+1)} = (1-a)(c_{1i}t_1^{(k)} + ... + c_{Ni}t_N^{(k)}) + ap_i$;

Send $c_{ij}t_i^{(k+1)}$ to all peers $j \in B_i$;

Compute $\delta = \left|t_i^{(k+1)} - t_i^{(k)}\right|$;

Wait for all peers $j \in A_i$ to return $c_{ji}t_i^{(k+1)}$

Until $\delta < \varepsilon$;

}

The algorithm can be interpreted as figure 5.

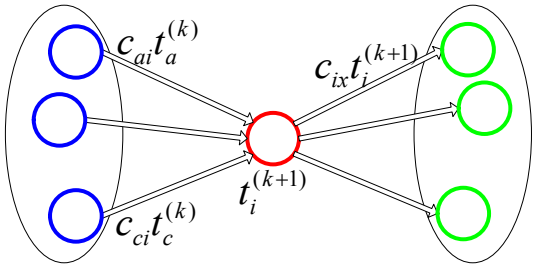

Figure 5: calculating global reputation value

Experiment shows that for a P2P system with 1000 peers, the algorithm has converged after less than 10 iterations. Actually, each peer has limited interaction with other peers in P2P systems. As a result, most c_{ij} in $t_i^{(k+1)} = (1-a)(c_{1i}t_1^{(k)} + ... + c_{Ni}t_N^{(k)}) + ap_i$ are zero; the sizes of Ai and Bi are also small. All these factors determine that a limited number of messages used in the algorithm. If the average number of acquaintances per peer is m, average number of iteration is k, the mean number of messages per peer will be *O(mk)*.

4.2.2.2 Secure Algorithm

Considering the existence of malicious peers, computing and storing global reputation values by peers themselves will cause two problems: first, malicious peers report false trust values of their own; second, malicious peers provide false reputation values to others. Consequently, distributed algorithms can not guarantee the security because malicious peers can easily cheat. Thus EigenRep provide a secure algorithm to resolve this problem.

The basic idea of secure algorithm is that the trust value of one peer is computed by some other peers. Those peers are called mothers which are responsible for computing their daughters' global reputation values. The reason for using more than one other peer to compute a peer's reputation value is that some mothers may be malicious peers and they report false trust values for their daughters.

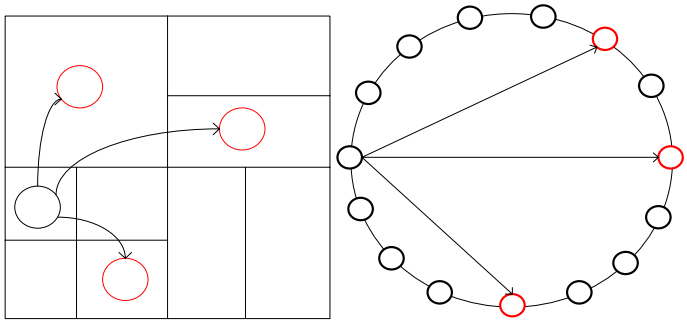

Figure 6: finding mothers using multiple DHTs

As shown in the Figure 6, the mothers of a peer can be assigned using multiple DHTs. If a peer needs the trust value of peer *i*, it can send queries to *mother(i)* (mothers of peer *i*). A majority vote on the trust value will determine the trust value of peer *i*.

EigenRep gives a secure algorithm. A mother of peer *i* calculates the global reputation value for peer *i* as follows.

Query all peers $j \in A_i$ for $t_j^{(0)} = c_{ji} p_j$;

Repeat

Compute $t_i^{(k+1)} = (1-a)(c_{1i} t_1^k + ... + c_{Ni} t_N^k) + a p_i$;

Send $c_{ij} t_i^{(k+1)}$ to all peers $j \in B_i$;

Compute $\delta = \left| t_i^{(k+1)} - t_i^{(k)} \right|$;

Wait for all peers $j \in A_i$ to return $c_{ji} t_i^{(k+1)}$

Until $\delta < \varepsilon$;

Mother(i) needs to get A_i, B_i and peer *i*'s opinions on B_i from peer *i,* before it starts to calculate the global trust value of peer *i*. In the process of calculating, after *mother(i)* collects all reputation values from A_i for the *k+1* times of computing, it calculates the *k+1* times of global trust value of peer *i*. Then it send its new reputation value (with the weight they trust B_i) to B_i. *Mother(i)* continues computing in this way until the global trust value of peer *i* converges to a certain value.

4.2.2.3 Improved Secure Algorithm

However, there is a bug in this algorithm. Actually, *mother(i)* should not collect all reputation values from A_i, but the mothers of peers in A_i. Similarly, *mother(i)* should not send its reputation value to B_i, but the mothers of peers in B_i. This is because A_i are not the peers which send reputation values to *mother(i)*, and B_i are not the peers which need the reputation value from *mother(i)*. The reputation values should be transferred among the mothers of A_i, B_i, and peer i. According to this, we give a modified secure algorithm as follows:

for each peer i do
  send $A_i, B_i, \vec{c}_i$ to its mothers M_i;
  collect $A_d, B_d, \vec{c}_d$ from its daughters D_i;
  for each $d \in D_i$ do
    $i = Hash_t(d)$;
    $k = 0$;
    send $c_{dj} t_d^{(k)} = c_{dj} p_d$ to all peers $Hash_t(j)$, for $j \in B_i$;
    repeat
      wai t for all peers $Hash_t(j)$, for $j \in A_d$ to return $c_{jd} t_j^{(k)}$;
      compute $t_d^{(k+1)} = (1-a)(c_{1d} t_1^{(k)} + ... + c_{Nd} t_N^{(k)}) + ap_d$;
      send $c_{dj} t_d^{(k+1)}$ to all peers $Hash_t(j)$, for $j \in B_i$;
      compute $\delta = \left| t_i^{(k+1)} - t_i^{(k)} \right|$;
      $k = k + 1$;
    until $\delta < \varepsilon$;
  end
end

In our modified secure algorithm, we assume that each peer has the same number of mothers. To cut down the message traffic, a mother of peer i will only communicate with the mothers of A_i and B_i which use the same hash function as it. In this way, the mean number of messages per peer will be $O(tmk)$, where t is the number of mothers for one peer, m is the mean number of acquaintances per peer, k is the mean number of iteration in calculating a peer's global trust value.

4.2.3 Discussion of EigenRep

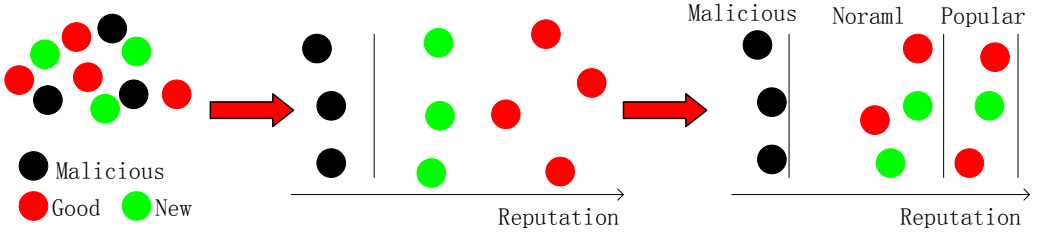

Figure 7: the use of EigenRep

Basically, a trust management solution in P2P system needs to do the following things:

Isolate malicious peers: malicious peers usually try to enhance their own reputations and reduce others' reputation. It is difficult to avoid malicious peers from doing these especially when some malicious peers cheat in collectives. A good reputation management solution should have the mechanism to detect the malicious peers and isolate them from the others. EigenRep succeeds in this by using multiple mothers to calculate and store reputation values for a peer. It also biases the users to download files from reputable peers.

Encourage peers to share file: P2P system should be able to encourage peers to share their authentic files. EigenRep achieves this by rewarding reputation to those peers which provide good services. The more authentic a peer shares to others, the more positive transactions others may have with the peer, and the more reputation the peer gains.

Allow the new peers to build trust: As mentioned in the local reputation values section, peers can not distinguish between new peers and malicious peers because the normalized local reputation values to those peers are all zero. As a result, the global trust value of them will be also zero. This causes a problem that, like malicious peers, new peers will hardly be selected because of their poor reputation. To allow the new peers to build trust, EigenRep provides a probability of 10% for new peers to be selected. However, other methods need to be used to distinguish between new peers and malicious peers before assigned some probability to be selected to new peers. Another way to help new peers build trust may be to reward them greatly for their good behaviors, so that they can keep up with good peers quickly.

Balance the load: Reputable peers have a high probability to be chosen because of their high reputations. Consequently, more transactions may be done in these peers, which will enhance their reputation further. This may lead them to be overloaded. A good reputation system should avoid this by balancing the load among peers. Some strategies may be used to achieve this goal. One way is to download probabilistically so that low reputation peers still have chance to be selected. The other is to set up maximum reputation values so that reputable peers will not be overloaded.

4.2.4 Limitations of EigenRep

First, malicious peers can still cheat in collectives. Since a peer reports A_i and B_i to its mothers directly, it can report other malicious peers as its A_i and B_i. So still they can cooperate to cheat. EigenRep seems do not solve this problem.

Second: the flexibility of calculating global reputation value. Since peers join and leave frequently in P2P system, and they may also be off line for a long time. EigenRep does not provide solutions to handle this.

Third, update global reputation values. After a certain time, the local reputation values of some peers may change greatly due to the transactions during this time. The global reputation values need to be recalculated in this case. How to determine whether recalculation need to be done or not? How to minimize the message traffic in the process of recalculation? EigenRep does not give answers.

Fourth: anonymous. EigenRep promises that it is not possible for a peer at a specific coordinate to find out which peer ID exactly it computes for. However, since

a peer may calculate many other peers, how does it distinguish the received reputation messages? It does not know for which peer is the reputation message. To solve this, the mother peer has to know its daughters' ID. As a result, anonymity loses.

5. SECURITY CONCERNS

As discussed in earlier section, security concerns are raised out in reputation based trust management systems. In this section we will look into two papers [6, 7], which firstly point out that anonymity opens the door to possible misuses and abuses by resource providers while it enables P2P information sharing systems to gain popularity. Motivated by that, the papers look into the area of trust management with reputation-based approach, which aims to complement the security weakness of P2P protocols. However, the focus is not the modeling, computation or maintenance of reputations. Instead they propose two P2P reputation-based protocols that take care of the security concerns of reputation-based trust management. We studied these two papers with the critical goal of discovering the weaknesses and advantages of the protocols by comparing, questioning and analyzing. The following section presents a clearer detail of the two protocols, with discussions and conclusions enclosed thereafter.

5.1 Basic Description of Gnutella

Gnutella is used as the testbed for their research, for Gnutella is an open protocol and simple open source implementations are available that permit to experiment with their protocol variants.

Gnutella offers a fully peer-to-peer decentralized infrastructure for information sharing, where all servents act both as clients and servers and as routers propagating incoming messages to neighbors. Messages, that can be broadcast or unicast, are labeled by a unique identifier, used by the recipient to detect where the message comes from. This feature allows replies to broadcast messages to be unicast when needed. To reduce network congestion, all the packets exchanged on the network are characterized by a given TTL (Time to Live) that creates a horizon of visibility for each node on the network. To search for a particular file, a servent p sends a broadcast Query message to every node linked directly to it. Servents that receive the query and have in their repository the file requested, answer with a QueryHit unicast packet that contains a ResultSet plus their IP address and the port number of a server process from which the files can be downloaded using the HTTP protocol. Finally, communication with servents located behind firewalls is ensured by means of Push messages. A Push message behaves more or less like passive communication in traditional protocols such as FTP, inasmuch it requires the "pushed" servent to initiate the connection for downloading.

5.2 Security Threats to Gnutella

Gnutella, a pure distributed architecture there is no central authority to trust, is a good tested for security provisions, as its current architecture provides an almost ideal

environment for the spread of self-replicating malicious peers. In an ordinary Internet-based transaction, if malicious content is discovered on a server the administrator can be notified. In Gnutella, malicious nodes have relatively small chance of detection due to the self-appointed opaque identifiers. Another security threat to Gnutella is man-in-the-middle-attack, where a QueryHit is modified by some malicious node in the path. The modified QueryHit directs the downloading request to a non-existent node or an unreliable or a malicious node.

The two protocols proposed by [6, 7], named P2PRep and XRep, aim for novel provisions preventing the attacks against the above security threats. The following sections discuss common basis for the two protocols, and different perspectives of the two protocols will be described thereafter.

5.3 P2PRep and XRep

5.3.1 Basics of P2PRep and XRep

Both P2PRep and XRep share the following common requirements. Each servent has associated a self-appointed servent id, which can be communicated to others when interacting. The servent id of a party can change at any instantiation or remain persistent. The two protocols encourage persistence as the only way to maintain history of a servent id across transactions.

Both protocols assume the usage of public key encryption to provide integrity and confidentiality of message exchanged. Whether persistent or fresh at each transaction, each servent_id is required to be a digest of a public key, for which the servent knows the corresponding private key. The public key is obtained using a secure hash function.

The basic idea is as follows. Upon receiving QueryHits from fellow peers, requestor p will enquire about the reputation of offerers by polling its peers before deciding from where to download the resource. *p* polls its peers by broadcasting a message requesting their opinion about the selected servents. All peers can respond to the poll with their opinions about the reputation of each of such servents. The poller *p* can use the opinions expressed by these voters to make its decision.

In particular, the two protocols need to ensure authenticity of servents acting as offerers or voters and the quality of the poll. The next two sections will show how P2PRep and XRep ensure the quality of the poll by ensuring the integrity of each single vote.

5.3.2 P2PRep

P2PRep has two flavors, basic polling and enhanced polling. The former does not require the voters responding to the poll to provide their servent_id, whereas the latter requires the declaration of the servent_id to facilitate weighting of votes by p.

5.3.2.1 Basic Polling

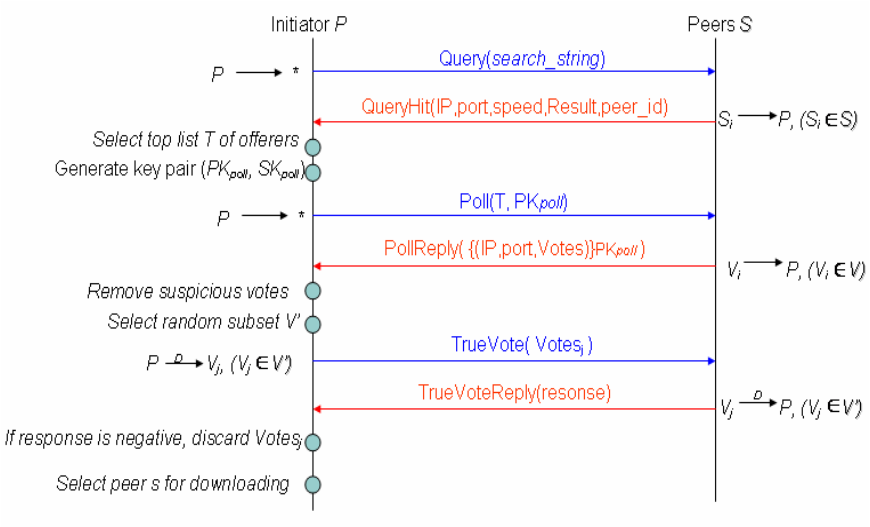

Figure 8: Basic polling of P2PRep

The basic polling solution, illustrated in Figure 8, works as follows. The servent p looking for a resource broadcasts a Query. Servents who receive the query and are willing to offer the requested resource, send back a QueryHit message indicating its IP, port, and servent id. Then, p selects its top list of servents T and polls its peers about the reputations of these servents. In the poll request, p includes the set T of servent ids and a public key generated on the fly for the poll request, with which responses to the poll will need to be encrypted. Peers receiving the poll request and willing to express an opinion on any of the servents in the list, send back a PollReply declaring their votes, IP and port. The PollReply is encrypted with the public key provided by p to ensure its confidentiality. Before making decision based on the received votes, p needs to test the reliability of the votes. Thus, p first uses decryption to detect tampered with votes and discards them, and then selects a set of voters and checks whether they actually expressed that vote by direction connection. If a negative reply is received, p will discard that vote. Upon assessing correctness of the votes received, p can finally select the offerer it judges as its best choice.

Now p needs to challenge the selected offerer s to assess whether it corresponds to the declared servent id before initiating the actual download. Servent s needs to respond with a message containing its public key PK_s and the challenge signed with its private key SK_s as shown in figure 9. If challenge-response exchange succeeds and the PK_s's digest is equal to the servent id declared by *s*, then *p* will know that it is actually talking to *s*. With the authenticity of the counterpart established, *p* can initiate the download and, depending on its satisfaction for the operation, update its reputation for *s*.

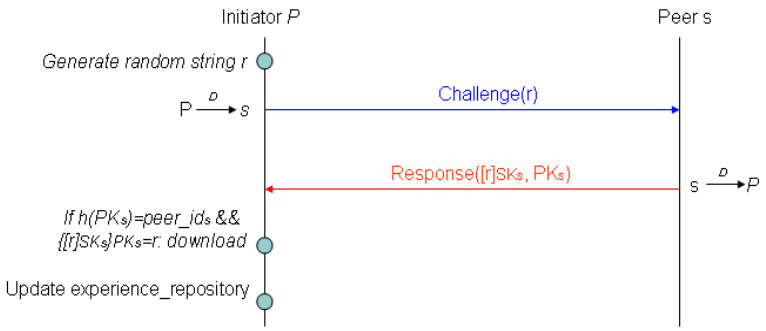

Figure 9: Challenge-response phase

5.3.2.2 Enhanced Polling

As mentioned before, the enhanced polling protocol differs from the basic solution by requesting voters to provide their servent id. Intuitively, while in the previous approach a servent only maintains a local recording of its peer's reputation, in the enhanced solution each servent also maintains track of the credibility of its peers, which it will use to properly weight the votes they express.

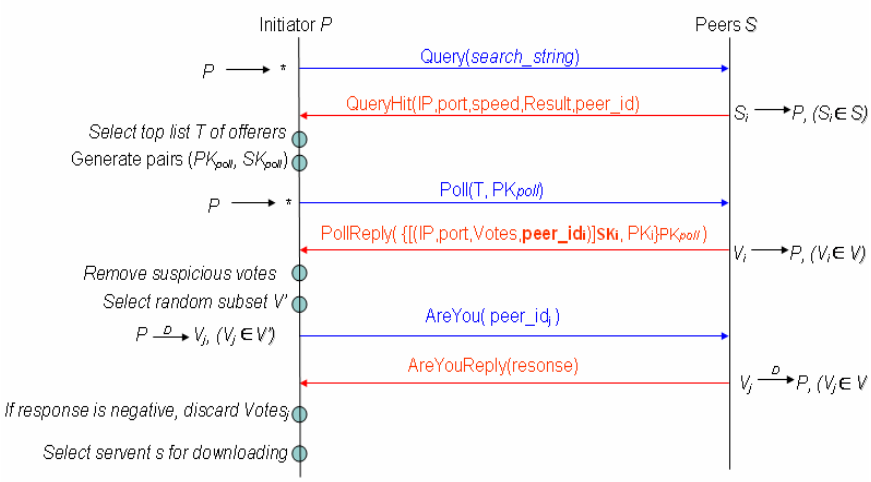

Figure 10: Enhanced polling of P2PRep

The approach, illustrated in Figure 10, works as follows. A voter responds to the poll with a PollReply message, in which it also reports its servent id unlike the basic polling. More precisely, PollReply encrypted with the public key PK_{poll}, and its vote signed with its own private key SK_i. Like in the basic polling, after collecting all the replies to the poll, p carries out an analysis of the votes received removing suspicious votes and then selects a set of voters to be contacted directly to assess the correct origin of votes. This time, the direct connection is needed to avoid servent id to declare fake IPs via an AreYou message (there is no need anymore to check the integrity of the vote as the vote's signature guarantees it). Servent p can now evaluate the votes received in order to select an offerer to download. While in the basic polling all votes were considered equal, the knowledge about the servent ids of the voters allows p to weight the votes based on who expressed them. This distinction is based on credibility information maintained by p. Like for the basic case, before

downloading, a challenge-response exchange is executed.

5.3.3 XRep

Extended from P2PRep for which reputation is solely associated with servents, XRep combines servent-based reputation and resource-based reputation together to allow votings on a node, or a particular resource, or both of the two. As in P2PRep, public key encryption is used in the same way as P2PRep to ensure the authenticity of the voter and votes, as well as the integrity of the messages exchanged. The XRep protocol may also have two flavors, basic polling and enhanced polling. While basic polling does not require voter to declare its servent id, enhanced polling needs the voter to declare their servent id in PollReply messages.

The XRep protocol differs from P2PRep as follows. In the PollReply messages, a voter can express its opinions on some nodes, or some recourses. Additionally, to support the resource-based reputation, resources must have some unique identifiers to distinguish from other resources. More precisely, in XRep, resource-based reputations are tightly coupled to the resource's content via digest, which is derived by applying a secure hash function to its content, similar in forming the identifiers of servents. To facilitate XRep, each peer is required to maintain two experience repositories, a resource repository and a servent repository for resource reputation and servent reputation respectively.

5.4 Discussions on P2PRep and XRep

5.4.1 Security Improvements

Section 5.2 described the two main security threats for Gnutella system. This section will analyze how P2PRep and XRep eliminate those two threats.

5.4.1.1 Distribution of Tampered with Information

The simplest version of this attack is based on the fact that there is virtually no way to verify the source or contents of a message. A particularly nasty attack is for a malicious node to simply respond providing a fake resource. Both the simple and enhanced versions of P2PRep and XRep protocol aim at solving the problem of impersonation. When p discovers the potentially harmful content of the information it downloaded from q, p will update q's reputation, thus preventing further interaction with q. Subsequently p will become a material witness against q in all polling procedures called by others.

5.4.1.2 Man in the Middle Attack

This kind of attacks takes advantage of the fact that the malicious user m can be in the path between p and q. The basic version of the attack goes as follows. Before q's QueryHit message reaches p, t rewrites q's QueryHit message with t's IP and port. M then will download the original file from q, modify the content and pass it to p. Both P2PRep and XRep protocols address this problem by including a challenge-response phase just before downloading. In order to impersonate q in this phase, m should

know q's private key and be able to design a public key whose digest is q's identifier. Therefore, both versions of this attack are successfully prevented.

5.4.2 XRep's Combined Reputation

As described in 5.3.3, XRep supports combined reputations of servents and resources. Combination of the two provides more informative pollings overcoming the limitations of servent-based only reputations. Comparisons of the two reputations are presented in table 1 in terms of life cycle, cold start, and performance bottleneck.

	Servent-based	**Resource-based**
Reputation's life cycle	shorter due to peer_id changes	good resource always recognizable
Cold start	avoids cold start for new resource	avoids cold start for new offerers
Performance bottleneck	may direct all downloads to most reputable peers	avoids bottleneck for most reputable peers

Table 1: Comparison of servent reputation and resource reputation

5.4.3 Some Weaknesses

P2PRep and XRep encourage a persistent identifier (and its good reputation) through several transactions in order to keep experience repositories so that reputation-based trust management works. However, this restricts the flexibility of servents choosing its own ids. Moreover, voters reveal their IP addresses in PollReply messages, which lead to weak anonymities.

6. CONCLUSION

In P2P systems, it is important to detect the malicious peers and harmful resources before a peer starts downloading. Reputation-based trust management is used to promote honest and cooperative behaviors, and thus the overall credibility of the P2P network can be maintained at an expected level. We have looked into a computation model in this report that explains the relationships between trust and reputation. It shows that trust management can be implemented by calculating reputation scores. Subsequently we studied several reputation-based trust management protocols, DMRep and EigenRep, through which the computation and maintenance of reputation scores are analyzed. These protocols assume that the probability of cheating within a society is comparably low. Security concerns involved in reputation-based protocols are also studied by two protocols P2Prep and XRep, which use public key encryption to provide authenticity and integration of reputation scores exchanged. However, the effect of P2Prep and XRep much depends on the underlying reputation model.

A number of issues for future studies remain open. First, more extensive evaluation methods over wider parameters are needed. Second, robust methods are needed to avoid the malicious peers cheat in collectives, as the current works are based on the assumption that the probability of cheating within a society is comparably low. Third, security protocols in self-regulating system still need to be

studied to prevent various malicious behaviors, as the public key encryption requires a trustworthy third party functioning as certificate authority and the current proposed public key verification by direct connection is still subject to long-time testing.

REFERENCE

[1] Napster http://www.napster.com.

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

[3] P. Resnick, R. Zeckhauser (2000). "Reputation Systems." Communications of the ACM 43(12): 45-48.

[4] S. Kamvar, M. Schlosser (2003). “The EigenTrust Algorithm for Reputation Management in P2P Networks”, *WWW, Budapest, Hungary*.

[5] K. Aberer, Z. Despotovic (2001) “Managing Trust in a Peer-2-Peer Information System”, *In Proc. of the IX International Conference on Information and Knowledge Management,* Atlanta, Georgia.

[6] F. Cornelli, E. Damiani, S.C. Vimercati, S. Paraboschi, and P. Samarati (2002). A reputation-based approach for choosing reliable resources in peer-to-peer networks. In CCS’02, Washington DC, USA.

[7] F. Cornelli, E. Damiani, S.C. Vimercati, S. Paraboschi, and P. Samarati (2002). Choosing reputable servents in a P2P network. In Proceedings of the eleventh international conference on World Wide Web, Honolulu, Hawaii, USA.

[8] G. Suryanarayana, R. N. Taylor (2004) “A Survey of Trust Management and Resource Discovery Technologies in Peer-to-Peer Applications”, *ISR Technical Report # UCI-ISR-04-6.*

[9] M. Blaze, J.Feigenbaum (1996). “Decentralized Trust Management”, *IEEE Symposium on Security and Privacy*.

[10] L. Kagal, S. Cost (2001). “A framework for distributed trust Management”, *Second Workshop on Norms and Institutions in MAS,* Autonomous Agents.

[11] T. Grandison, M. Sloman (2000). "A Survey of Trust in Internet Applications", *IEEE Communications Surveys 3(4).*

[12] M.Blaze, J. Feigenbaum (1999b) “The Role of Trust Management in Distributed Systems Security”, *Secure Internet Programming.*

[13] G.. Zacharia, P. Maes, (2000) "Trust Management Through Reputation Mechanisms." *Applied Artificial Intelligence 14: 881-907*.

[14] S. Lee, R. Sherwood (2003). “Cooperative peer groups in NICE”, *IEEE Infocom*, San Francisco, USA.

[15] M. Gupta, P. Judge (2003). “A Reputation System for Peer-to-Peer Networks”, *Thirteenth ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Monterey, California.

[16] S. Marsh (1994). “Formalising Trust as a Computational Concept”, Ph.D. Thesis, University of Stirling.

[17] J. Sabater, C. Sierra, (2002). “Reputation and social network analysis in multi-agent systems”, *First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy.

[18] J. Pujol, R. Sanguesa (2002). “Extracting reputation in multi agent systems by

means of social network topology", *First International Joint Conference on Autonomous Agents and Multi-Agent Systems,* Bologna, Italy.

[19] J. Rouchier, M. O'Connor, F. Bousquet (2001) "The Creation of a Reputation in an Artificial Society Organized by a Gift System", *Journal of Artificial Societies and Social Simulations, 4(2)*.

[20] P. Resnick, R. Zeckhauser (2000b) "Trust among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System", *Working paper for the NBER Workshop on Empirical Studies of Electronic commerce.*

[21] E. Ostrom (1998) "A behavioral Approach to the Rational-Choice Theory of Collective Action", *American Political Science Review, 92(1),* pp. 1-22.

[22] L.Mui, M. Mohtashemi, A.Halberstadt (2001) "A Computational Model of Trust and Repuation",

[23] K. Aberer (2001) "P-Grid: A self-organizing access structure for P2P information systems", *In Proc. Of the Ninth International Conference on Cooperative Information Systems.*

[24] S. Ratnasamy, P. Francis, M. Handley, R.Karp, S. Shenker(2001) "A Scalable Content-Addressable Network", *In Proc. of the ACM SIGCOMM.*

[25] I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan (2001) "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", *In Proc. of the ACM SIGCOMM*.