Distributed Trust Management [*]

Thomas Repantis
Department of Computer Science & Engineering
University of California, Riverside
trep@cs.ucr.edu

December 6, 2004

Abstract

Peer-to-peer systems are an attractive means of sharing data and services. However, the problem of how to efficiently decide which peers are to be trusted still remains unsolved. We are describing EigenTrust [3], a protocol for reputation management in peer-to-peer networks. We compare it to four more protocols for decentralized trust management we present, namely P2PRep [2], RCertPX [4], P-Grid [1], and TrustMe [5].

1 Distributed Trust Management

Peer-to-peer (P2P) systems have attracted a lot of interest, as a highly dynamic platform that enables autonomous computing nodes to share resources and services. The advantages of peer-to-peer environments, especially of unstructured ones, include their ability for self-organization, for adaptation to different loads, and for resiliency to node failures. Operation of all the peers as both clients and servers, and without a central coordinator eliminates possible bottlenecks in terms of scalability or reliability.

However, in an unstructured and decentralized topology several security issues arise. One of the most challenging problems, that is still being actively

[*]Course Project Report for CS236 - Database Management Systems, Fall 2004.

researched, is how to create a trusted network of peers in the absence of a central trust managing authority. Trust is important when sharing data or processing power, and crucial for e-commerce applications and auctioning.

By saying that peer A puts a level of trust into peer B, we mean that A estimates the probability of B acting in a way that will allow A to achieve a desired level of satisfaction.

One way a peer A can estimate the level of trust to put into another peer B, is by being based on the reputation of peer B. The reputation of peer B is measured from previous interactions of peer A with peer B, or also on previous interactions of other peers with peer B. As the level of trust a peer enjoys is based on its reputation, a peer is motivated to act according to the rules of the network, whether these are to share content, not to cheat, or anything else. Using the peers' opinion to establish a reputation is a process already very popular in the scientific community for example through peer review or citations.

One of the main difficulties in managing reputation-based trust in P2P networks is that information about peer interactions is spread across the network, and no single peer has a complete global view of the peers' reputations. Furthermore, malicious peers might tamper with reputation information while it is stored locally or transmitted, or even try to defame other peers.

Kamvar, Schlosser, and Garcia-Molina present in [3] a distributed and secure method to compute a global trust value for every peer in the network, based on its complete history of uploads. By having access to a peer's trust value, other peers can choose from whom to download a file (with whom to interact). This way, the propagation of inauthentic files or viruses can be combatted and malicious peers can be identified and isolated. Attempting to track down malicious peers is more effective than trying to identify inauthentic files, since malicious peers can generate unlimited inauthentic files, if they are not isolated.

2 EigenTrust

In EigentTrust [3] each peer stores a local trust value for every peer it has interacted with. The global trust value of a peer is generated by aggregating the local trust values of all peers regarding that particular peer in an iterative process. In other words, the reputation of a peer is constructed by aggregating the cumulative opinions all other peers in the network have of it. The reputation of a peer is used to value the probability with which a peer will provide authentic files, as well as the probability with which a peer will provide accurate local trust values for other peers. Hence, in EigenTrust, the global reputation of each peer is given by the local trust values assigned to it by other peers, weighted by the global reputations of the assigning peers.

2.1 Local Trust Values

Each time peer i downloads a file from peer j, it rates the transaction as positive ($tr(i,j) = 1$ for a satisfactory transaction) or negative ($tr(i,j) = -1$ for an unsatisfactory transaction). The local trust value s_{ij} peer i holds for peer j is the sum of the ratings of the individual transactions that peer i has had with peer j:

$$s_{ij} = \sum tr_{ij} = sat(i,j) - unsat(i,j)$$

Before local trust values can be aggregated they have to be normalized. A normalized local trust value is defined as:

$$c_{ij} = \frac{max(s_{ij},0)}{\sum_j max(s_{i,j},0)}$$

The above definition produces a value between 0 and 1 and helps in computing the global trust value without a need for renormalization at each iteration.

2.2 Global Trust Value

Using the normalized trust values, a peer i can calculate the trust t_{ik} it places in peer k, by adding all the opinions of its acquaintances about peer k, weighted by the trust peer i places in them:

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

By using matrix notation, where C is the matrix $[c_{ij}]$ and $\vec{t_i}$ the vector containing the values t_{ik} we have:

$$\vec{t_i} = C^T \vec{c_i}$$

This trust values only reflect the experience of peer i and his acquaintances. To get a wider view, peer i will ask its friends' friends ($t_i = (C^T)^2 c_i$). If this continues ($t_i = (C^T)^n c_i$), it will have a complete view of the network after a large number (n) of iterations. If n is large, the trust vector $\vec{t_i}$ will converge to the same vector for every peer i, namely to the left principal eigenvector of C. Hence, $\vec{t}$ is a global trust vector, the elements of which quantify how much trust the system as a whole places in peer j.

2.3 Basic EigenTrust Algorithm

The basic EigenTrust algorithm is just the computation of $\vec{t} = (C^T)^n \vec{e}$), for a large n, where $\vec{e}$ is defined as the m-vector representing a uniform probability distribution over all m peers, $e_i = \frac{1}{m}$. This is equivalent to calculating $\vec{t} = (C^T)^n \vec{c_i}$, where $\vec{c_i}$ is the normalized local trust vector of some peer i, since both $\vec{e}$ and $\vec{c_i}$ converge to the left principal eigenvector of C.

By defining some distribution $\vec{p}$ of pre-trusted peers, the following are achieved:

1. In the presence of malicious peers, $\vec{t} = (C^T)^n \vec{p}$ will converge faster than $\vec{t} = (C^T)^n \vec{e}$.
2. If peer i has no local trust value about any other peer, or if it has rated all other peers with 0, c_{ij} is set equal to p_j instead of being undefined.
3. By taking $t^{(k+1)} = (1-a)C^T t^{(k)} + ap$, where a is some constant less than 1, or equivalently setting the opinion vector for all peers to be $\vec{c_i} = (1-a)\vec{c_i} + a\vec{p}$, each peer is forced to place at least some trust in the peers p that are guaranteed not to be malicious. This helps a peer not to be trapped by a malicious collective of peers, which give high local trust values only to each other. Using $\vec{p}$ also makes the matrix C irreducible and aperiodic, thus guaranteeing that the computation will converge.

Summarizing the above, the basic EigenTrust algorithm is the iterative calculation of $t^{(k+1)} = (1-a)C^T t^{(k)} + ap$, starting from $\vec{t^{(0)}} = \vec{p}$ and ending when the difference between $t^{(k+1)}$ and $t^{(k)}$ is less than a value e.

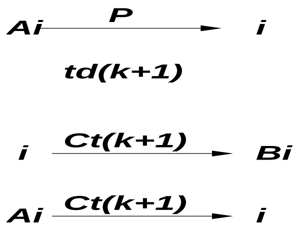

Figure 1: Distributed EigenTrust Algorithm (Ai: set of peers which have downloaded files from peer i, Bi: set of peers from which peer i has downloaded files).

2.4 Distributed EigenTrust Algorithm

In the distributed EigenTrust algorithm each peer stores not only its local trust vector $\vec{c_i}$, but also its own global trust value t_i. Each peer computes its own global trust value:

$t_i^{(k+1)} = (1-a)(c_{1i}t1^{(k)} + ... + c_{ni}tn^{(k)}) + ap_i$

Since peer i has had limited interaction with other peers, many of the components in that equation will be zero. The distributed EigenTrust algorithm can therefore be summarized as follows 1: Each peer i queries all peers which have downloaded files from it, for $t_j^{(0)} = pj$, and then repeatedly:

- calculates $t_i^{(k+1)} = (1-a)(c_{1i}t1^{(k)} + ... + c_{ni}tn^{(k)}) + ap_i$,
- sends $c_{ij}t_i^{(k+1)}$ to all peers j from which it has downloaded files, and
- receives all $c_{ji}t_j^{(k+1)}$ from all peers which have downloaded files from it,

until the difference between $t_i^{(k+1)}$ and $t_i^{(k)}$ is less than a value e. As can be seen, only the pre-trusted peers need to need to know that they are pre-trusted.

2.5 Secure EigenTrust Algorithm

In the secure EigenTrust algorithm, each peer i does not compute and report its own trust value t_i, so that reporting false trust values can be avoided.

Instead, a different peer in the network computes the trust value of peer i. Moreover, in order to prevent malicious peers from reporting false trust values for other peers, the trust value of peer i is computed by more than one peer. Thus, using a DHT each peer is assigned M score managers, which compute the trust value of that peer. If the global trust values returned by the score managers differs, the value reported by the majority is regarded as true. Dynamic leaves and joins of score managers are handled by the DHT, while replication of the trust values guards against possible score manager failures. The secure EigenTrust algorithm can thus be summarized as follows 2: Each peer is assigned a number M of score managers, whose DHT coordinates are determined by applying a set of one-way secure hash functions to the peer's unique identifier. Each score manager maintains the opinion vector c_d^i of the peers whose trust values it is appointed to calculate (its daughters), as well as the set of peers its daughters downloaded files from, the set of peers which downloaded files from its daughters and their trust assessments regarding its daughters. Each peer i:

- submits its local trust values $\vec{c_i}$ to its score managers,
- collects the local trust values of its daughters, and
- also collects the sets of the peers that have downloaded files from them,
- submits its daughters local trust values to other score managers,
- collects the sets of the peers from which its daughters have downloaded files,
- and for each of its daughters it queries all peers which have downloaded files from her for their opinion on her and repeatedly:
  - computes the trust value of each of its daughters d $t_d^{(k+1)} = (1 - a)(c_{1d}t1^{(k)} + ... + c_{nd}tn^{(k)}) + ap_d$,
  - sends $c_{dj}t_d^{(k+1)}$ to all peers j from which its daughter has downloaded files, and
  - receives all $c_{jd}t_j^{(k+1)}$ from all peers which have downloaded files from its daughter,

until the difference between $t_d^{(k+1)}$ and $td^{(k)}$ is less than a value e. The use of hashing guarantees that the score managers do not know whose reputation they calculate and that a peer cannot select its score manager.

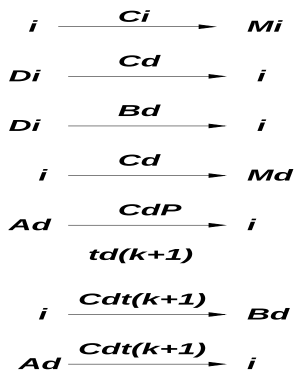

Figure 2: Secure EigenTrust Algorithm (Ai: set of peers which have downloaded files from peer i, Bi: set of peers from which peer i has downloaded files), M: set of Mothers (score managers), D: set of Daughters.

Peers may choose to bias their choice of download by a convex combination of the global trust values and their own local opinions of other peers, i.e. using the trust values given by the vector $\vec{t_{personal}} = d\vec{t} + (1-d)\vec{c}$, where d is a constant between 0 and 1.

In order to avoid a repetitive process, where a peer accumulates reputation by providing files and thus becomes an even more popular download source in the future, and hence accumulates even more reputation, the load is distributed across peers in the network as follows: Among R responses a peer i is chosen as a download source with probability $\frac{t_i}{\sum_{j=0}^{R} t_j}$ while a peer j with trust value $t_j = 0$ is selected with probability 10%. The selection of peers with global trust value of 0 allows newcomers to eventually become trusted members of the network.

2.6 Threat Models

The algorithm is evaluated against a variety of attacks: i) Individual Malicious Peers, which provide inauthentic files and give high trust value to downloads of inauthentic instead of authentic files, ii) Malicious Collectives, where malicious peers provide inauthentic files and assign high trust value only to other malicious peers, iii) Malicious Collectives with Camouflage, where malicious peers also provide a percentage of authentic files, iv) Malicious Spies, where a set of malicious peers provide authentic files so that they can assign high trust values to a set of malicious peers that provide inauthentic files.

3 Comparison With Other Distributed Trust Management Protocols

Several reputation-based, decentralized trust management protocols have already been proposed. We here discuss four major ones and compare them to EigenTrust.

3.1 P2PRep

In P2PRep [2] a polling based protocol is proposed and implemented (on top of Gnutella), that uses public key cryptography for authentication. Every peer stores its opinion of other peers in the network 3. Any peer A that wants to query the trust value of another peer B broadcasts a query to the network. The peers that have had transactions with B reply with their IP and port in a message, encrypted with a public key A provided for that particular query. The encryption of the votes guarantees their confidentiality, as well as the confidentiality of the voters. By using a different public key for every query, the source of the query is not traced back to A. A then individually contacts a random set of the voters and asks them to confirm their votes, so that fake messages are filtered out, and combines the valid votes to make a decision.

In an enhanced version of the protocol 4, each peer stores not only reputation information, regarding the resources offered by peers, but also credibility information, regarding the votes peers express. The credibility information is built by comparing after a transaction the peer's own opinion about a peer, to the reputation information other peers provided. The credibility information is used to properly weight votes coming from the same peers in the future.

A modified version of the protocol is suggested for low-bandwidth networks, where reputation message exchanges are reduced by providing a server-based functionality, whereby peers keep a record of positive votes for them stated by others. A peer can provide those credentials to interested peers. Obviously that information has to be signed by the voters that expressed it.

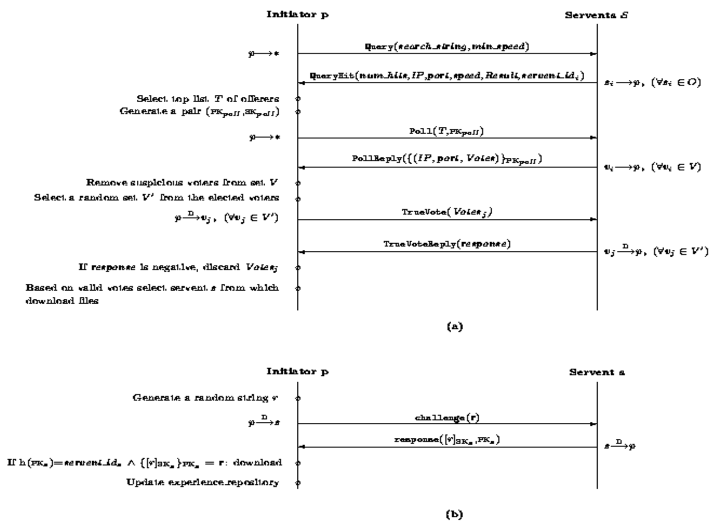
Initiator p
Servents S
p ⟶ *
Query(search_string, min_speed)
QueryHit(num_hits, IP, port, speed, Result, servent_id_i)
s_i ⟶ p, (∀s_i ∈ O)
Select top list T of offerers
Generate a pair (PK_poll, SK_poll)
p ⟶ *
Poll(T, PK_poll)
PollReply({(IP, port, Votes)}_PK_poll)
v_i ⟶ p, (∀v_i ∈ V)
Remove suspicious voters from set V
Select a random set V′ from the elected voters
p ⟶D v_j, (∀v_j ∈ V′)
TrueVote(Votes_j)
TrueVoteReply(response)
v_j ⟶D p, (∀v_j ∈ V′)
If response is negative, discard Votes_j
Based on valid votes select servent s from which download files
(a)
Initiator p
Servent s
Generate a random string r
p ⟶D s
challenge(r)
response([r]_SK_s, PK_s)
s ⟶D p
If h(PK_s)=servent_id_s ∧ {[r]_SK_s}_PK_s = r: download
Update experience repository
(b)

Figure 3: Sequence of messages and operations in the basic polling protocol of P2PRep (a) and download of files from the selected servent (b) [2].

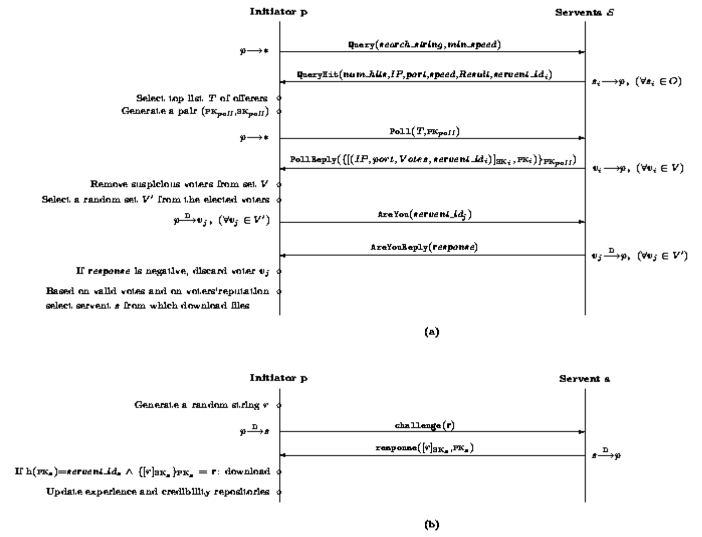
Initiator p
Servents S
p ⟶ *
Query(search_string, min_speed)
QueryHit(num_hits, IP, port, speed, Result, servent_id_i)
s_i ⟶ p, (∀s_i ∈ O)
Select top list T of offerers
Generate a pair (PK_poll, SK_poll)
p ⟶ *
Poll(T, PK_poll)
PollReply({[(IP, port, Votes, servent_id_i)]_SK_i, PK_i)}_PK_poll)
v_i ⟶ p, (∀v_i ∈ V)
Remove suspicious voters from set V
Select a random set V′ from the elected voters
p ⟶D v_j, (∀v_j ∈ V′)
AreYou(servent_id_j)
AreYouReply(response)
v_j ⟶D p, (∀v_j ∈ V′)
If response is negative, discard voter v_j
Based on valid votes and on voters'reputation select servent s from which download files
(a)
Initiator p
Servent s
Generate a random string r
p ⟶D s
challenge(r)
response([r]_SK_s, PK_s)
s ⟶D p
If h(PK_s)=servent_id_s ∧ {[r]_SK_s}_PK_s = r: download
Update experience and credibility repositories
(b)

Figure 4: Sequence of messages and operations in the enhanced polling protocol of P2PRep (a) and interaction with the selected servent (b) [2].

Apparently, the polling procedure described in P2PRep requires a lot of additional messages and generates large amounts of network traffic, while also delaying the process of downloading a file. To reduce the generated traffic the horizon of a peer might be shortened. In that way a peer will only poll a small portion of the network. This however will enable the poller to only get a reasonable number of votes for peers that are very active. Not only is the polling mechanism limited to the reviews of the peers that are reached, but the reviews of peers that are not currently present in the system are obviously not taken into account. This is not the case in EigenTrust, the iterative process of which shall however also produce a significant amount of traffic. P2PRep offers the advantage of not requiring any particular network structure, whereas EigenTrust relies on a DHT to assign score managers to peers.

3.2 RCertPX

In RCertPX [4] a reputation certificate (RCert) 5 is created and is stored in the peer that it refers to. After each transaction the reputation certificate is updated. In order to eliminate the possibility of tampering with the reputation certificate, the last rater always digitally signs the whole certificate. When a peer wants to interact with another peer, it requests its reputation certificate, and it contacts the last rater, in order to verify the correctness of the certificate. If the last rater is unavailable, the one before it is contacted. Since every rater revokes the previous reputation certificate, a peer can only use the latest reputation certificate, that includes all the ratings. When a peer wants to decide with which peer it should do a transaction, it contacts all peers that offer the service/data and they provide it with their reputation certificates. The peer then contacts the latest rater of each peer to determine if its reputation certificate is still valid 6.

The network traffic produced by RCertPX is relatively low, as well as the decision delay. Furthermore, even if some peers have left, their ratings are still available. Also important is the fact that the ratings cannot be changed once they are determined. In that way, a peer will not be able to change its rating of another peer if it becomes a competitor. However, the protocol is complicated, especially when the last rater is offline, which is often the case in P2P systems. In that case, a group of previous raters for each peer, instead of a single rater, has to be contacted. The protocol uses ratings even from untrusted peers, which might be even more critical in the case a rater and

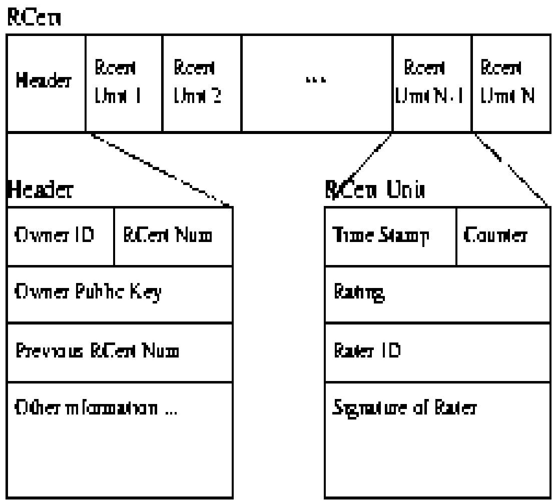

Figure 5: Format of the reputation certificate (RCert) [4].

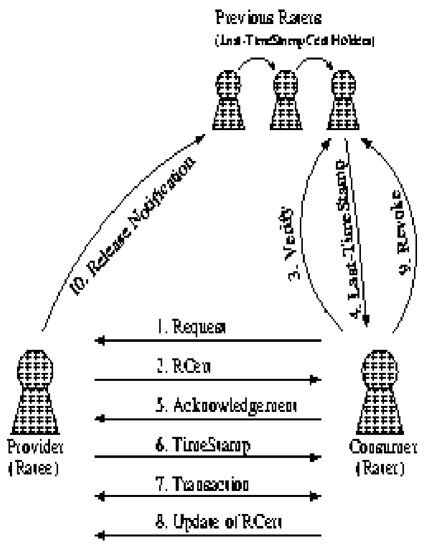

Figure 6: RCertPX Protocol [4].

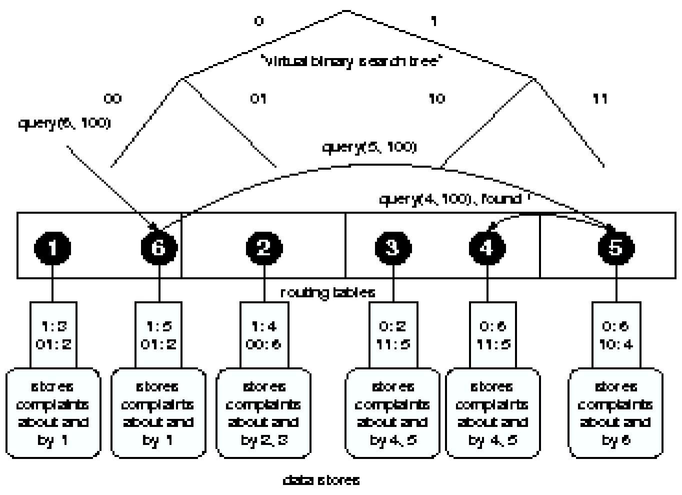
0
1
'virtual binary search tree'
00
01
10
11
query(6, 100)
query(5, 100)
query(4, 100), found !
1
6
2
3
4
5
routing tables
1:3
01:2
1:5
01:2
1:4
00:6
0:2
11:5
0:6
11:5
0:6
10:4
stores complaints about and by 1
stores complaints about and by 1
stores complaints about and by 2, 3
stores complaints about and by 4, 5
stores complaints about and by 4, 5
stores complaints about and by 6
data stores

Figure 7: Virtual binary search tree provided by P-Grid example [1].

a ratee collude to change the ratings. A peer can collude with another peer so that previous certificates will not be revoked and therefore bad ratings may be lost. Even though the protocol is complicated, to ensure that a peer will not tamper with its reputation, it does not rely on a particular network structure, unlike EigenTrust.

3.3 P-Grid

In [1] a trust managing system on top of the P2P system P-Grid is described. The reputation of a peer in this system is expressed as the number of complaints he has (not) received from other peers. Complaints can be filed from a peer receiving or providing a service or file. Hence, when investigating complaints regarding a peer, both complaints about the peer acting as a service provider as well as a service user are taken into account. If an observed value of complaints exceeds the general average of the trust measure too much, the peer is regarded as dishonest. The complaints regarding a peer are stored in a virtual binary search tree 7. These include both complaints by and about a peer. The P-Grid infrastructure governs the insertion and querying of complaints.

Replication in storage satisfies the integrity of the stored complaints in a probabilistic manner. This means that complaints that are related to a peer are stored in and provided by more than one peer. The complaints are replicated to such an extent, that the probability of getting inauthentic complaints by malicious peers is less than an acceptable fault-tolerance threshold. Hence, this scheme provides only probabilistic guarantees regarding the accuracy of the rating. Moreover, peers might store complaints about themselves. Even though this might be a rare case there is no provision for it. Furthermore no measures are taken against malicious peers which might tamper with ratings while they are transferred. This problem is left to be bypassed by the unrestricted connectivity a network like the Internet provides. Similar to EigenTrust, P-Grid requires a network structure to be maintained, for the reputation information to be stored and retrieved.

3.4 TrustMe

TrustMe [5] identifies anonymity of both the trust host and the trust querying peer as an important feature of trust-managing systems. Anonymity can protect both peers reporting poor trust values and peers asking for the trust values of other peers. Public-private key pairs are used by TrustMe to preserve anonymity. The trust rating of each peer is placed at a set of random (and equally distributed amongst all nodes of the network) peers, which replies to all queries for the trust values it holds. A peer can anonymously issue a query and get the trust value without needing to know where that value is stored. Similarly only the trust holding agents of a peer are able to identify rating messages regarding that peer. In more detail, the procedure includes the bootstrapping of the system, where peers become trust holding agents for other peers, querying about a peer's reputation, receiving replies by the trust holding agents of that peer, collecting proof-of-interaction after two peers interact, and finally reporting a rating to be stored by the trust holding agent of a peer, after the interaction with that particular peer 8. The use of more than one trust holding agents for a peer reduces the possibility of manipulating rating messages. The use timestamps prevents peers from replaying old rating messages. As already mentioned messages are encrypted and signed to ensure anonymity and integrity.

Like EigenTrust, TrustMe takes into account reports even of peers that have left the system and also achieves small decision time. Unlike EigenTrust, it also provides secure communication between the peers hosting the trust value of a peer and the peers querying it, in addition to anonymity of both. Moreover TrustMe is not vulnerable to group threats, in which malicious groups of nodes can monitor where trust holding agents are hashed.

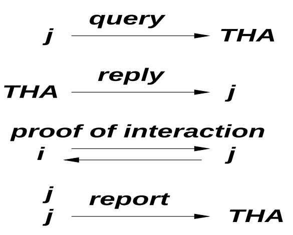

Figure 8: Messages exchanged in TrustMe.

Finally, since TrustMe does not rely on a DHT infrastructure, it is not vulnerable to DHT threats like malicious routing information tampering and malicious lookup replies. However, TrustMe suffers from a serious drawback, namely that it relies on broadcasting for both querying and reporting trust values. The simulations show that TrustMe is twice as expensive in generated messages as a polling based system like P2PRep, where only the query is broadcasted. Caching trust values for a certain amount of time can reduce the protocol overhead and the response times of the querying peers. The reliance on broadcasting, with the amount of network traffic that this generates, makes TrustMe unacceptable for large-scale, unstructured networks.

4 Conclusion

We have presented EigenTrust, a distributed and secure algorithm for computing global trust values of peers in a peer-to-peer network. The global trust value of a peer is computed by calculating the left principal eigenvector of a matrix of normalized local trust values, that summarize the entire system's history with the particular peer.

Even though EigenTrust relies on a DHT to assign peers to score managers, we can envision an unstructured version of the algorithm, where trusted peers in a neighborhood are responsible for storing the trust values of peers in their vicinity, and the trust values are weighted according to credibility of the raters or the similarity in the interests of the peers.

We have also presented four more protocols for decentralized trust management, namely P2PRep, RCertPX, P-Grid, and TrustMe, and compared them with EigenTrust. The major parameters of the comparison have been the reliance on a particular network infrastructure, the amount of traffic generated, which can be regarded as the protocol overhead, the persistence of the protocol, meaning whether it is able to take into account the opinions of nodes that have left the system, and the defense the protocol provides against a variety of known attacks.

References

[1] K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of the International Conference on Information and Knowledge Management, CIKM*, 2001.

[2] F. Cornelli, E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Choosing reputable servents in a P2P network. In *Proceedings of the International World Wide Web Conference, WWW*, 2002.

[3] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proceedings of the International World Wide Web Conference, WWW*, 2003.

[4] B. C. Ooi, C. Y. Liau, and K. L. Tau. Managing trust in peer-to-peer systems using reputation-based techniques. In *Proceedings of the International Conference on Web Age Information Management, WAIM*, 2003.

[5] A. Singh and L. Liu. TrustMe: Anonymous management of trust relationships in decentralized P2P systems. In *Proceedings of the International Conference on Peer-to-Peer Computing, P2P*, 2003.