Trust And Reputation in Peer-to-Peer Systems

Leonardo L. Fernandes

University of Trento
leonardo.fernandes@dit.unitn.it

Abstract. Peer-to-peer (P2P) systems are distributed systems in which nodes act as peers, in the sense that all of them have similar roles in the network, being both providers and consumers of services. Such systems are becoming very popular in applications like file sharing. In this kind of architecture, trust and reputation are fundamental requirements for the reliability of the system. Since services are provided by each user, a reputation scheme is needed to properly inform other users, helping them to decide in which peers to trust. Means to avoid malicious users to sabotage the system by providing unauthentic files or services, or by consuming system resources without offering anything to peers in exchange are necessary. Frequently, such systems must also provide reputation information preserving users anonymity, a feature of some popular P2P applications. In this paper, the challenges involved in trust and reputation management on P2P systems are introduced, and some approaches to the problem found in the literature are discussed.

1 Introduction

Peer-to-Peer (P2P) systems have been demonstrating a growing popularity in innumerous distributed applications. Different from the more traditional client/server paradigm, on P2P systems all the nodes on the network are considered peers. In other words, all nodes are identical. Peers should be able to provide/request similar services to/from each other. Ideally there should be no difference at all between peers implementation and potentialities.

In many current P2P applications such as Gnutella [3], BitTorrent [1] and Kazaa [6], among others, thousands of peers are already deployed all over the Internet. Such systems provide a huge infrastructure for file sharing among users all over the world. Highly popular materials can be downloaded by large numbers of users simultaneously without overloading any central server, assuming there are enough copies of the resource distributed in the network. Many other applications for peer-to-peer systems can be envisioned. Peer-to-peer databases, computeer games and multimedia broadcasting are just a few of the innumerous possibilities of the new paradigm.

Unfortunately, there is a fundamental problem with P2P large networks. It is not feasible to trust on all peers in the network as good providers of services. On a client/server structure, a user can decide to trust in the organization that owns and runs the server application and assume that such organization is taking the required security measures. On a peer-to-peer architecture, in the other hand, there is not one central server provider to be trusted or not. System services are provided by a large number of peers possibly distributed all over the world. It is clearly not wise to trust unknown users to provide services as they are expected to. Of course, most of the users tend to cooperate and do their part for the system to work, but still the opportunities for malicious peers to cause damage are countless. Just to cite a couple of examples, malicious peers participating in a file sharing service can distribute unauthentic files or files infected by virus. Other peers can try to take advantage of system resources without providing any service to the system. Furthermore, many services in peer-to-peer networks actually rely on the existance of trusted peers. Peer sampling service [4] is just one example of such kind of service.

Trust and reputation systems aim to provide mechanisms that allow users to have at least an idea about the expected behavior of other nodes. By keeping track of peers past behavior, nodes can make informed decisions on weather to trust a given peer or not. Usually there is a reputation, or a trust value, associated with any entity participating in the system. By choosing to interact with peers with better trust values, users can take advantage of the services of peers that are likely to be authentic, since such reputation should be earned through a reasonable time of good service to the system.

Kamvar et al. [5] propose five design guidelines for the development of trust architectures for peer to peer systems:

- The system should be self-policing, in the sense that peers themselves should be responsible for defining and enforcing the trust mechanism.
- The system should maintain anonymity, since it is a desirable feature in many P2P systems.
- The system should not assign any profit to newcomers. This is necessary because otherwise malicious peers can take advantage of such profit by constantly rejoining the system as a new user. For this reason usually the initial reputation of peers is low.
- The system should have minimal overhead in terms of computation, infrastructure, storage, and message complexity.
- The system should be robust to malicious collectives of peers who attempt to, as a group, subvert the system.

This paper presents and discusses some approaches found in the literature to deal with the problem of trust in the peer-to-peer paradigm.

2 Trust Mechanisms

2.1 P2Prep

P2PRep [2] is a relative simple protocol aiming to be an extension to the popular Gnutella [3] peer-to-peer file sharing system. The protocol is relatively simple. Nodes just need to keep local trust values for the peers they have interacted with.

Gnutella works in a very simple way. Each node is connected to a limited number of nodes in the Gnutella network. When a user wants to download a file, the host broadcasts a query to its *k-hop* neighborhood, being k a parameter of the protocol. The nodes in this neighborhood that can serve the query send a reply message back to the requester. The requesting node than arbitrarily chooses one of the peers to download from, establishing a straight connection to the chosen node.

P2PRep does not change the query and reply mechanism of Gnutella. The only addition is the inclusion of two new phases in the protocol, namely polling and vote confirmation. After the peer making the request receives the replies from the nodes that host the desired file, the polling phase starts. The requester sends a poll message asking for information about the nodes that are offering their service. Such poll message is also broadcast k hops away. Every node that receives the poll message and that has previously interacted with any of the queried peers should reply with a vote message. Such vote contains a trust value for one or more peers. By analyzing such votes the requesting user can decide to download from the peer with the best reputation among the neighborhood instead of choosing arbitrarily.

2.2 TrustMe

TrustMe [7] is a more secure approach. The protocol requires one or more trusted bootstrap servers (BS). A bootstrap server server takes care of adding new nodes to the system. For each new node in the system, a group of trust holding agents (THA's) is randomly selected among the available peers. Such agents have the duty of keeping the trust information about the entering peer. Furthermore, the BS generates a pair of public and private special encryption keys (SP and SB) to be used only by the THA's. The server generates a pair of keys to each node in the system and make it available only for the respective THA's.

When a node i wants to interact with a peer j, it broadcasts to the whole network a message asking about the trust value of j. Usually such broadcast includes requests for more than one node in order to make a choice between them. When THA nodes for j receive such request, they encrypt the appropriate trust value using the special private key SP_j that they share and send it back to the requesting node through the network. Based on the received trust value, node i can then decide weather or not to interact with j. If the decision is affirmative, then i interacts with j (e.g. downloads a file from it) and collects a proof of such interaction. Such proof is a message signed by j confirming the interaction happened. After the interaction is done, i broadcasts a message encrypted with the special public key SB_j, so that only j's THA's can read it, containing the result (e.g. weather it was satisfactory or not) and the proof of the interaction. The trust holding agents for j can then update j's trust value accordingly.

The important detail is that using this process, and assuming the forwarding scheme does not allow for identifying message initiators, the THA's, as well as the peers, remain completely anonymous. Also, the fact that the trust holding agents are chosen randomly makes collective attacks very unlikely. Assuming the attackers do not control the majority of the network nodes, they have a low probability of having a majority in THA groups for any node. Even less probable is for malicious nodes to have a majority of the THA's of another malicious node.

2.3 EigenTrust

This protocol [5] is based on the calculation of the left principal eigenvector of a matrix of local trust values. Local trust values are obtained by subtracting the number of unsatisfactory transactions between each pair of nodes from the number of satisfactory ones. Such values are then normalized so that each peer has a local trust value c between 0 and 1 for each of the other peers.

A global trust value t_{ij} representing how much does i trust j could then be obtained by the following formula:

$$t_{ij} = \sum_k c_{ik} c_{kj} \tag{1}$$

Note that the equation considers node i's opinion about every node k as well as the opinions of every node k about j. The result is a global trust value that expresses how much i trusts j. If we express (1) in a vectorial form we have:

$$\overrightarrow{t_i} = C^T \overrightarrow{c_i} \tag{2}$$

Where C is the matrix containing the local trust values. And by executing n interactions of (2) we obtain:

$$\overrightarrow{t} = (C^T)^n \overrightarrow{c} \tag{3}$$

The vector t is the left eigenvector of C. The interesting about this vector is that it represents a global trust value which is common to all peers. All solutions of (2) for all peers converge to t if n is large enough.

The problem of implementing this algorithm in a peer-to-peer setting is that there is no central server to collect all of the values that compose C and make the calculations. Distributed EigenTrust is proposed. On the distributed version, each peer can calculate its own trust value by obtaining just the relevant row of C. It can be done by any given node i by collecting the values from all the peers that have interacted with i in the past. For the remaining nodes the value can be set to zero, since nodes that have never interacted with i can't have an opinion about it. The calculation needs to be synchronized, and in each of the n iterations, intermediate values need to be exchanged again.

Since it is not wise to let nodes calculate their own trust values for security reasons, one last extension to EigenTrust is provided. In secure EigenTrust, the trust value of each node is calculated by a group of other nodes.

The protocol relies on a small number x of initially trusted peers. All other nodes start with a trust value of zero while the trusted peer start with reputation $\frac{1}{x}$. If there are no such special nodes, the algorithm is vulnerable to collective attacks. Even small groups of malicious nodes can dominate the C matrix if all peers start at an equal condition. With initially trusted peers that does not happen, since the strong influence of such peers prevail.

3 Discussion

The protocols presented are all reasonable solutions to the trust problem in P2P systems.

The main advantage of P2PRep is that it is a straightforward extension to a very popular and widely deployed system. Another advantage is the relative cheap cost in both communication and computation of the protocol. The problem with the approach is that it is vulnerable to collective malicious peers attacks. Malicious peers connected in neighboring areas of the Gnutella network can easily manipulate the election scheme proposed by P2PRep.

The fact that anonymity is preserved is a major advantage of both TrustMe and EigenTrust.

TrustMe provides a great level of security, since even a large number of collective attackers are not likely to be able to subvert the trusting system significantly. One drawback of the protocol is its overhead. Since the THA's are completely anonymous, every trust value request and interaction report must be done through broadcasting to the whole network. This can severely limit the protocol performance on large networks. Another problem is the need for a bootstrap server, which is against the P2P paradigm.

EigenTrust provides a very consistent trust mechanism. The protocol achieves the goal of having a completely distributed algorithm for finding global trust values that take in to account the knowledge of all non malicious peers in the system. The authors of the protocol [5] demonstrate that the procedure to find eigenvectors converge reasonably fast, and since most values of the matrix tend to be zero in large networks (because most nodes never interact with each other), the protocol is not very expensive in terms of both communication and computation. Unfortunately, the requirement of initially trusted peers for security goes against the pure P2P idea, because such peers are not equal to the others. Also if a malicious user has access to one or more of such trusted peers, he can jeopardize the trust system.

4 Conclusion

This work demonstrates that it is possible to implement a reasonable level of trust management in peer-to-peer systems. Assuming most nodes cooperate, the protocols discussed tend to achieve the goals to which they were proposed in real world applications. However, the studied approaches, for providing appropriate security, still rely on some kind of central authority or initially trusted peers, which are against the standard peer-to-peer idea, where all nodes in the system should be identical peers with the same capabilities, potentialities and limitations. Due to this fact we believe that the problem of trust in "pure" peer-to-peer systems is still an open research matter.

References

1. Bram Cohen. Incentives build robustness in bittorrent, 2003.

2. E. Damiani, De Capitani, S. Paraboschi, and P. Samarati. Managing and sharing servants' reputations in p2p systems. *Knowledge and Data Engineering, IEEE Transactions on*, 15(4):840–854, 2003.
3. Gnutella. http://gnutella.wego.com.
4. Mark Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In *Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 79–98, New York, NY, USA, 2004. Springer-Verlag New York, Inc.
5. Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM Press.
6. Kazaa. http://www.kazaa.com.
7. Aameek Singh and Ling Liu. Trustme: Anonymous management of trust relationships in decentralized p2p systems. In *P2P '03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, page 142, Washington, DC, USA, 2003. IEEE Computer Society.