A reputation-based trust management in peer-to-peer network systems

Natalia Stakhanova, Sergio Ferrero, Johnny Wong, Ying Cai
Department of Computer Science
Iowa State University
Ames, Iowa 50011 USA
{ ndubrov, sferrero, wong, yingcai }@iastate.edu

Abstract

Peer-to-peer networks have gained a lot of attention over the last couple of years, mainly due to the popularity of the free multimedia file-sharing program Napster and a legal battle around it. Being open by nature P2P systems represent an ideal environment for various types of malicious intrusions. The problem of securing hosts on P2P network while keeping the openness of the system has been studied extensively over last couple of years but still remains open. Existing solutions based on reputation management either employ centralized algorithms or rely on peers' cooperation on the network. We describe a fully decentralized approach that allows computing peers' reputation based on the traffic between a node and its peers, independently of these peers willingness to cooperate in calculation of their reputation.

1 INTRODUCTION

Peer-to-peer (P2P) network systems are increasingly gaining popularity on the Internet. These networks allow individual hosts (peers) to share and distribute various types of information over the Internet. By their nature, P2P networks are structured in a way that allows an open and unsupervised communication between peers. Therefore, these systems are vulnerable to various types of attacks, among which are denial-of-service attacks (DoS) and distribution of viruses.

To protect themselves from malicious intentions, hosts should be able to identify reliable peers for communication. Identifying these peers is a challenging task in highly dynamic network environments like P2P networks.

In this paper we suggest a solution to this problem based on the notion of trust. We propose a policy for managing traffic in peer-to-peer network based on peers' reputation. We also describe a model for computing this reputation using the trust score based on the peers' interaction with each other. Unlike most existing reputation-based models, our approach does not employ a centralized storage and only produces reputation scores on demand. The proposed solution is fully distributed and does not require any cooperation from the rest of the network.

Our model aims to help users to select the most reliable peers whose past behavior shows willingness to participate in a proper functioning of the P2P system and at the same time to anticipate possible attacks from malicious peers by limiting their access to the victim peer's resources.

In this paper we focus on a particular type of P2P systems, called Gnutella, which proved to be one of the most popular P2P systems. Furthermore, unlike other popular P2P applications such as KaZaa, the source code of Gnutella is freely available for experiments. It should be noted that our approach could be adapted to other types of P2P networks.

2 BASIC DESCRIPTION OF GNUTELLA MODEL

Gnutella is a decentralized peer-to-peer file-sharing model. In this model all peers perform tasks usually associated with both clients and servers. Peers generate queries while at the same time accept queries from other peers, match with local shared files and respond with the results if match is found. To propagate queries and their results, called QueryHits, through the network each peer that receives traffic forwards it to its neighboring peers.

There are two types of peers in Gnutella: end users called local peers (leaf nodes) and group leaders called Ultrapeers. Ultrapeers serve as a shield for their local peers broadcasting traffic that come from them to the network and accept applicable traffic from other Ultrapeers [10].

To connect to a Gnutella network a user starts with a computer that runs a Gnutella client. Once connected, a new Gnutella client will send a request to a Gnutella web server called GwebCache. The web server will reply with a list of IP addresses of Ultrapeers. A new node will announce its existence to those Ultrapeers. Once announced a new node can start searching the data shared on the network [8].

The Gnutella network protocol is built on top of the TCP/IP protocol. After the connection is established, peers communicate with each other by exchanging Gnutella protocol descriptors. Figure 1 presents the descriptors currently defined in Gnutella [10]. Ping and Pong descriptors are used by peers to identify which hosts are currently alive on the network. While descriptors Query and QueryHit carry search request and reply message through discovered active hosts.

Descriptor	Description
Ping	Used to actively discover hosts on the network. A servent receiving a Ping descriptor is expected to respond with one or more Pong descriptors.
Pong	The response to a Ping. Includes the address of a connected Gnutella servent and information regarding the amount of data it is making available to the network.
Query	The primary mechanism for searching the distributed network. A servent receiving a Query descriptor will respond with a QueryHit if a match is found against its local data set.
QueryHit	The response to a Query. This descriptor provides the recipient with enough information to acquire the data matching the corresponding Query.
Push	A mechanism that allows a firewalled servent to contribute file-based data to the network.

Figure 1: Gnutella Descriptors

3 RELATED WORK

Theoretical justifications to the approaches based on trust have been provided by Abdul-Rahman et al. [1].

Trust and willingness of peers to act honestly seem to be natural mechanisms to prevent security breaches in P2P systems. There have been several approaches proposed to enhance security in P2P networks based on trust and reputation management. Certainly the lack of central authority in Gnutella makes it difficult to track the reputations of all peers. Therefore, most solutions in this area can be divided into those that add centralization to the existing system [5] and decentralized solutions that are based on the peers' cooperation for reputation computation [3, 5,7].

3.1 Centralized approaches

The approach presented by Gupta et al. [5] is an example of centralized approaches. The proposed model tracks positive peer's contribution to the system using a credit-debit mechanism. Based on its activity each peer computes and stores its reputation locally. To ensure secure and distributed access to the reputation scores, Reputation Computation Agent (RCA) periodically collects reputations from peers using a {public, private} key pair. This approach does not provide mechanisms for decreasing the reputation score for malicious behavior.

3.2 Decentralized approached

The second group of the approaches is based on the peers' cooperation. One of such approaches is proposed by K.Aberer and Z.Despotovic[2]. All peers support a P-Grid-virtual binary search tree where each node is associated with certain path and stores complaints (reputation) about other agents. The model also does not have any preventive mechanism from inserting arbitrary number of false complaints.

NICE [7] is another decentralized reputation-based approach to trust management where reputation is stored in the form of cookies expressing peer satisfaction about the transactions. Before initiating a transaction a peer checks a local cookie to ensure that a targeted peer can be trusted. However, if no cookie is available for that peer, cooperation of other peers in acquiring that information is necessary.

Cornelli et al. [3] also proposed an approach to share information about peers' reputation based on a distributed polling algorithm. When a node receives QueryHits it chooses a download peer with a highest reputation based on the opinion of other peers. Although this approach addresses many security considerations for P2P networks, there are limitations to this approach. Based only on the number of downloads, the approach does not consider peers acting mostly as servers (i.e. sharing content in the network). This approach is also highly dependent on the other peers' cooperation. In addition, extensive traffic generated by polling algorithm can add significant load to the traffic in the network, which might discourage the adoption of the protocol.

3.3 Other approaches

The approach proposed by Daswani and Garcia-Molina [4] is a radically different solution, which focuses on managing traffic between peers based on load-balancing policies rather than peers' reputation. These policies allow "fair" sharing of the available resources by all clients and therefore, help peers to cope with a particular type of attacks on P2P networks – DoS attacks. The simulations were run on different network topologies under various policies to evaluate damage caused by malicious node in the network. The results showed that the cumulative network damage can be greatly reduced using particular policies and network topologies.

Our proposed approach is built on the features of the above mentioned approaches. In the next section we introduce the details of our reputation-based model.

4 REPUTATION-BASED TRUST MANAGEMENT MODEL

The approach we are presenting in this paper is reputation-based. Reputations about peers are stored and managed locally which will not create excessive traffic in Gnutella network. It integrates easily with the original Gnutella protocol and can be viewed as an extension to it.

As mentioned above a peer searching for information in Gnutella environment broadcasts a Query message and receives responses from peers having matching resources. Among those responses a peer is selected from which information is downloaded. The choice about the download peer is usually based on the quality of the offered file (file size, file name) as well as an uploading speed of the offerer. The rest of the traffic in Gnutella network is accepted without any consideration.

We suggest assessing the reputation of peers before accepting any kind of traffic from them. When traffic arrives at a peer it looks up the sender's reputation in the local reputation repository and makes a decision, to

accept or reject traffic, based on the adopted trust threshold value.

Since trust thresholds vary from peer to peer this makes it difficult for the malicious node to change its behavior in such a way so that its harmful traffic is accepted at the target peer.

In our model we consider as a malicious peer a node that performs the following irresponsible actions: generating as many queries as possible and not offering any service to others. Service in this case means forwarding queries and offering (sharing) its own resources on the network.

Since our approach relies on a peer's reputation, persistence of peer's ID would definitely enhance the model. Although, this does not exist in current versions of Gnutella network it can be easily implemented. As have been already noted by some researchers [5,3], maintaining the same ID also allows peers to maintain reputation scores across online sessions, which benefits "good" peers. At the same time malicious peers would constantly try to change their IDs in order to update the reputation.

4.1 Reputation computation

The reputation of a peer is determined by its contribution to the functioning of the P2P network. Based on this we can distinguish factors indicating a peer's behavior contributing to a proper functioning of the network and factors destructing it. Among these factors are *resource search, resource upload, resource download* and *traffic extensiveness*.

Resource search is essentially willingness of a peer to forward traffic (Queries and QueryHits) passing through it. Each peer that forwards the query adds its ID to the "trailer" which is an addition to Query message. Once a peer finds resources matching the request, it forms a QueryHit and transfers a trailer from Query to QueryHit. In this way the peer that originated the Query receives the trailer with information about peers that behaved "well" and updates its local reputation repository.

Resource upload indicates another peer's interest in the shared resource and therefore its willingness to function properly on the network. A file uploaded completely is considered a successful upload.

Resource download reflects the quality of the downloaded information. A peer can decide through GUI interaction that a download is unsuccessful if, for example, the file was unreadable, contained harmful content or did not match the query request.

The concept of *traffic extensiveness* helps to evaluate the traffic load coming from all connected peers based on the average amount of traffic received until this point. Assuming that n peers are being connected to the peer i at a particular moment and each of them have sent l_j bytes, average load is determined by:

$$\sum_{j=1}^{n} l_j / n$$

The traffic can be considered extensive if a current peer's load exceeds the average amount by a user pre-defined threshold, where a threshold is a factor of the average amount of traffic. In other words, let L_{cK} be a current load from peer k and t be a threshold, then the traffic from peer k is extensive if the following holds

$$L_{cK} > \sum_{j=1}^{n} l_j / n * t$$

We will refer to the factors mentioned above as actions and will distinguish bad and good actions as actions that failed and actions that succeeded, respectively.

Traffic from a particular peer can be accepted or rejected depending on its reputation (trust score) and trust threshold scale at a particular period of time. *Trust score* is calculated based on good actions (GA) and bad actions (BA). We define trust score as a percent of bad actions happened during that period of time. Let trust score of peer i be a ratio R_i *and* total number of considered actions for this peer be TA_i, *then*

$$R_i = BA_i / TA_i$$

Therefore, the smaller the percent of bad actions (trust score) the better peer's behavior on the network and, thus, the better peer's reputation.

Each peer maintains a local reputation repository in a tuple of the form (*peer_id, total number of actions, number of bad_actions*). In addition to this, each peer defines its trust thresholds x_1 and x_2 in the range from 0 to 100, which indicate percent of bad actions acceptable by the peer. Trust thresholds are presented in figure 2.

Trust Threshold	*Meaning*	*Description*
Greater than x_1	Distrust	Peer is completely untrustworthy.
Between x_1 and x_2	Average	Peer is trustworthy.
Less than x_2	Full trust	Peer has a complete trust.

Figure 2: Trust thresholds

The correspondence between trust thresholds and trust score is presented in figure 3.

$R_i => x_1$	$x_1 > R_i > x_2$	$R_i =< x_2$
No traffic is accepted	x_1–(R_i–x_2) percent of the traffic from peer i is accepted for a period of time k.	All traffic is accepted

Figure 3: The correspondence between trust thresholds and trust score

The correspondence shows that high values of x_1 and x_2 thresholds will cause majority of traffic to be accepted despite of sender's bad reputation while low threshold values will limit traffic to a minimum.

Based on these policies a peer decides how many messages to process during each time period k. For instance, let x_1=30, x_2=4 then if a percent of bad actions is 13, then a peer has a trust range of "average" and amount

of traffic to be processed is determined by the formula: $x_1-(R_i - x_2)$, 30-(13-4) = 21, so 21% of all its traffic is accepted within period k.

Such correspondence between trust thresholds and trust score allows a peer to balance its available resources and incoming traffic according to the reputation of other peers, i.e peers with "good" reputation will still be able to access necessary resources freely while malicious peers will have a restricted service. Of course, a malicious peer can "behave well" until it gains good reputation. However, since each peer adopts its own trust thresholds it will be hard for a malicious peer to determine its trust scale.

If a peer is deemed malicious then no traffic is accepted from it. A peer can gain trust back by continuing to forward queries in the network and correctly processing queries sent to him.

It is also important to provide a mechanism to support initial reputation for newcomers. As the system oriented on a successful functioning it should not assume good intentions of the new peers. Therefore, minimum average trust given to a newcomer will provide a start for good peers and at the same time it should help to expose malicious intentions from the beginning.

5 DESIGN AND IMPLEMENTATION

The architecture of our reputation-based model is comprised of the several components that are displayed in figure 4.

Security Manager is the front component that is responsible for authorizing the incoming traffic. Each new incoming request is handed to a reputation management component to identify whether this request can be granted, and if granted, whether there are any restrictions for it.

The reputation management block is the main module where the current peer's trust score is compared with the existing trust thresholds and the decision about the current peer's request is made.

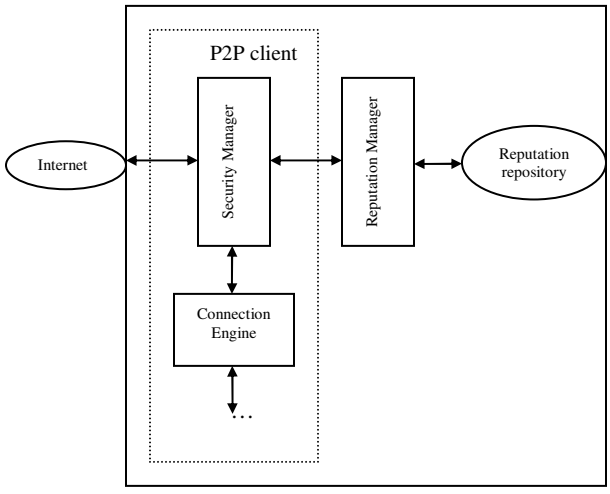

Figure 4: System overview

The reputation management component is connected to a local reputation repository, managed by a relational database system.

The decision on the incoming request is sent back to a Security Manager. If the request is granted, corresponding engines are triggered through the Connection Engine component. The declined request is being ignored.

Our design and implementation were based on Phex version 0.9.5.54, a java-based Gnutella client. However, this architecture can be applied to other P2P clients. For implementation of the local reputation repository the MySQL database system was used.

5.1 Experimental setup

Our evaluations were run on a small network consisting of three PCs running the Phex P2P clients. Two peers were configured as Ultrapeers to be able to forward traffic in the network. One peer represented a malicious node and therefore carried out harmful functions such as generating extensive traffic, responding to queries with "bad" files and not forwarding the queries.

We generated the input queries for our tests interactively, therefore each node was given a processing capacity of 20 queries per time period k, where k=5 sec. Extensive traffic threshold was set to 1.7. We experimented with the thresholds in the range from 1 to 3. Setting thresholds value closer to 1 made system very sensitive to even small deviations from the average amount of traffic. While threshold value 3 gave no effect on the trust score. Since the traffic for our experiments was generated manually, the threshold value of 1.7 is determined to be optimal in our experiments.

Trust thresholds were set as follows x_1=20 and x_2=5. Initial reputation values for peers were set up manually.

5.2 Results

In our experiments we examined the dependence of peer performance from its reputation in the following two scenarios.

1. **A highly trusted peer starts acting maliciously**.

This situation is possible if malicious peer intentionally integrates into network to gain a high reputation and archive a greater damage as a result of its malicious actions later. The loss of reputation is presented in figure 5.

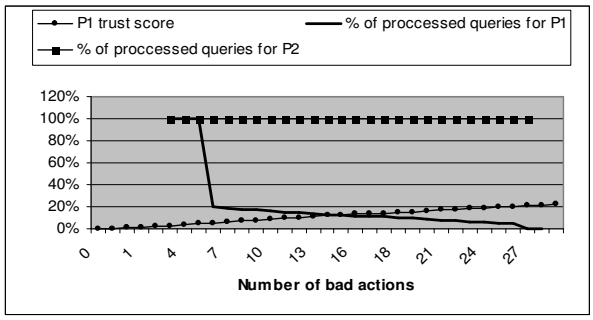

Figure 5: Decrease of full reputation when peer P1 starts "acting" maliciously

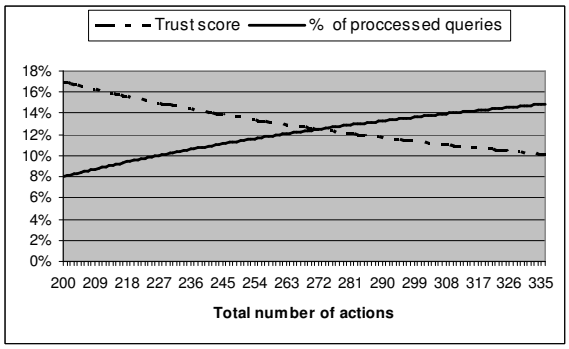

Figure 6: Reputation gain when peer starts "acting" properly

Trust score increases rapidly as the number of harmful actions (BA) increases. As the experiment shows number of malicious node's queries processed decreases with a decrease of its reputation. The results also show that as reputation decreases a malicious node is able to receive some service from the victim peer until it becomes distrusted (trust score reaches 20%, which is the distrust threshold) although it does not prevent the rest of the peers to be serviced. For our tests malicious peer P1 and a "good" peer P2 were producing queries within the processing capacity of the victim host.

2. **A peer with a low reputation is able to gain the trust back if it starts behaving properly**.

The reputation gain is presented in figure 6.

As opposed to reputation loss, reputation gain happens very slowly. It takes around 27 malicious actions dropped the reputation from full trust to distrust (see figure 5) while almost 100 actions is needed to bring reputation from bad to average level.

Figure 6 shows the impact of reputation on the amount of service that peer receives. The number of queries processed at any moment is proportional to the reputation. A peer receives better service as its reputation increases.

5.3 Discussion

In this section we present a comparative analysis of the proposed approach with existing reputation–based algorithms.

Figure 7 summarizes the main factors considered for assessing peers' reputation in four different models.

Unlike our approach Debit-Credit Reputation Computation (DCRC) schema is a system-oriented approach. Peers compute reputation information based on their own activity on the network, while the RCA agent represents a global repository of reputation values. Therefore, the system is more focused on providing a global view of the reputations for all peers in the network. As opposed to this approach our schema is user oriented. Each peer monitors the activity of the connected peers and makes trust decisions based on their individual thresholds. This gives users more autonomy from the rest of the network.

The other advantage of our approach is that it considers not only positive experience of the peer but also negative ones such as the downloading of viruses. DCRC schema does not distinguish successful and unsuccessful experiences and thus can lead to a situation where malicious peer freely uploads viruses while still

Factors taking into consideration for reputation computation	Debit-Credit Reputation Computation (DCRC)[5]	NICE [7]	P2P Rep [3]	Our approach
Size of QueryHit	x			
Size of uploaded file	x			
Size of downloaded file	x			
Sharing hard-to-find content	x			
Bandwidth	x			
Time factor	x	x		x
QueryHit		x		x
File Upload		x		x
File Download		x	x	x
Amount of incoming traffic				x
Location of reputation computation for other peers	Reputation Computation Agent	Local peer	Local peer	Local peer
Presence of centralized storage	x			
Presence of protocol/algorithm for collecting reputations		x	x	

Figure 7: Comparison of different reputation-based calculation schemes

maintaining a good reputation.

P2P Rep (reputation) model considers both negative and positive experiences; however it bases reputation calculation on the number of file downloads which leaves out peers only sharing files on the network. Since our model monitors all activities of peers connected at the moment, a reputation value will be computed for each of them.

P2P Rep and NICE approaches require cooperation of peers in reputation calculation. However, such cooperation might not be always available, for example, due to conspiracy of malicious peers. Therefore, our approach employs methods that allow a peer to calculate a reputation independently of other peers' willingness to cooperate.

The proposed approach does not demand high system overhead. This becomes an important factor considering fast development of wireless technologies and their application in P2P networks.

6 CONCLUSION

In this paper we have described a solution to reputation management of peers on P2P networks using a reputation-based trust model based on the traffic between peers. The approach is fully decentralized, requires no peers' cooperation and employs only on-demand calculations.

As for future research we envision enhancement of the model through user profiling techniques. This will allow peers to have better knowledge about other peers' typical behavior and therefore, consider deviations from it as anomalies. This work is currently in progress.

7 REFERENCES

[1] A. Abdul-Rahman and S. Hailes, "A distributed trust model", Proceedings of the 1997 New Security Paradigms Workshop, pp 48--60, 1997.

[2] K. Aberer and Z. Despotovic, "Managing trust in a Peer-to-Peer Information System", Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM 2001), 2001.

[3] Cornelli, Damiani, di Vimercati, Paraboschi, Samarati, "Choosing Reputable Servents in P2P network", Proc. of WWW, 2002.

[4] N. Daswani and H. Garcia-Molina, "Query-Flood DoS Attacks in Gnutella", ACM Conference on Computer and Communications Security, 2002.

[5] M. Gupta, P. Judge, and M. Ammar, "A Reputation System for Peer-to-Peer Networks", Proceedings of NOSSDAV, 2003.

[6] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks", Proceedings of the Twelfth International WWW Conference, 2003.

[7] S. Lee, R. Sherwood, and B. Bhattacharjee, "Cooperative Peer Groups in NICE", Proceedings of IEEE INFOCOM, 2003.

[8] LimeWire: Running on the Gnutella Network. http://www.limewire.com/english/content/p2p.shtml

[9] Phex- the Gnutella P2P filesharing client. http://phex.kouk.de/

[10] The Gnutella Protocol Specifications v0.4. Document Revision 1.2. http://www.clip2.com