The trust management framework for peer-to-peer networks

by

Natalia Stakhanova

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Johnny Wong, Major Professor
Samik Basu
Andrew Miner
Doug Jacobson

Iowa State University

Ames, Iowa

2004

Graduate College
Iowa State University

This is to certify that the master's thesis of

Natalia Stakhanova

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Popularity of peer-to-peer (P2P) networks exposed a number of security vulnerabilities. Among those is a problem of finding reliable communication partners. In this thesis, we present an integrated trust framework for peer-to-peer networks that quantifies the trustworthiness of a peer via reputation-based trust mechanism and anomaly detection techniques. As opposed to other known techniques in P2P networks, our trust management schema is fully decentralized and does not rely on the co-operation of peers. Furthermore, the reputation computation is based on traffic coming from other peers.

We also describe an anomaly detection procedure that analyses peer activity on the network and flags potentially malicious behavior by detecting deviation from peer profile. We present integration of our anomaly detection to trust management scheme and study the performance of reputation-based approach using implementation and performance of trust framework through simulation.

1 INTRODUCTION

Recent years have seen a tremendous growth of peer-to-peer (P2P) networking paradigm on ubiquitous computing in the form of popular P2P file-sharing systems like Napster [21], Gnutella [30] and KaZaA [14]. These systems are large-scale overlay networks that allow file and resources sharing over the Internet. Peers on the network are completely autonomous and can join or leave the system at any time. However, widespread and unrestricted deployment of P2P systems exposed a number of security vulnerabilities and directed major research efforts towards ensuring secure and trustworthy communication between peers in a P2P environment.

In this context, reputation-based techniques have emerged as a natural choice to identify trusted peers and isolate the untrusted ones. These techniques regulate relationships between peers on the basis of the quality of their respective activities on the network. Though promising, reputation-based techniques have several limitations, majority of them employ a centralized storage or rely on peers' cooperation in the network which might not be always available, for example, due to conspiracy of malicious peers. In addition, these approaches may potentially fail to accurately capture a peer's behavior. Specifically, they often lack global focus making decision regarding reputation updates based on current actions. For instance, connection during night time instead of usual afternoon time, or sudden download of system file instead of the usual mp3 files will not necessarily be flagged as suspicious by reputation mechanism. However, these may be signs of possible malicious intention of the user. Therefore, considering current peer's actions within its usual behavior can potentially provide a better understanding of peer's intentions.

This thesis aims to address the mentioned above shortcomings by proposing an integrated trust framework that quantifies the trustworthiness of a peer via decentralized reputation-based trust mechanism and anomaly detection technique.

We developed a distributed reputation computation mechanism which is significantly different from the known techniques of the same class. The primary characteristics are (a) the selected features considered for reputation computation and (b) the mode of deployment which, unlike other approaches, does not rely on the co-operation of peers.

We also introduce the integration of profile-based anomaly detection to trust management that effectively overcomes the inherent shortcoming of reputation-based approach: its general lack of peer's activity global view. The central tenet of our approach is that peer-profile based anomaly detection provides enhancement to reputation-based trust management in P2P networks.

1.1 Basic description of Gnutella model

In this paper we focus on a particular P2P system, called Gnutella, which is currently one of the most widely used P2P systems. In fact there is, ongoing research effort focused on developing a hierarchical model called Gnutella2 [10]. Additionally, the Gnutella protocol is open source and thus source code is readily available on the Internet. Although our framework is based on Gnutella, the proposed trust management schema can also be adapted to other types of unstructured decentralized P2P networks and our approach towards anomaly detection can be applied to any P2P environment.

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 forwards the received traffic it to its neighboring peers.

Gnutella v0.6 presented a concept of Ultrapeers (group leaders), peers that have enough system resources to serve less powerful nodes, called local peers, connected to them directly [28].

Ultrapeers provide a shield for their local peers broadcasting traffic that come from them to the network and accept applicable traffic from other Ultrapeers.

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 [30].

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. Table 1 presents the descriptors currently defined in Gnutella [30]. 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.

Table 1: Gnutella Descriptors

1.2 Contribution

The main contributions of our work can be summarized as follows:

- *Development of reputation-based trust management approach.* We describe a fully decentralized approach that allows computing peers' reputation based on the traffic between a node and its peers

and independently of these peers willingness to cooperate in the calculation of the value of their reputation.

- *Integration of trust management and peer profile-based anomaly detection.* As eluded before, traditionally reputation-based trust computation may not accurately capture peer's behavior. Incorporating peer profile-based anomaly detection, we can effectively overcome this shortcoming. Anomaly detection provides an extra dimension for computing peer-reliability; specifically peer-profile is based on two classes of peer activities, temporal and networking, which are not captured in reputation mechanism. To the best of our knowledge, this is the first effort for integrating reputation computation and anomaly detection in the domain of trust management.
- *On-line reputation computation.* In our framework, peer's reputation is updated while it is in session and also at the end of the session. Updates take into consideration the degree of anomaly which is computed on the basis of carefully selected statistical metrics and distribution of data.
- *Flexibility of trust framework.* The framework is designed and developed in a modular fashion. This de-coupling paves way for easy update and future enhancement of various domain-specific modules, e.g. profile-data collection.
- *Application to P2P networks.* We use the framework in the domain of P2P networks and show the effectiveness of our technique via simulation.

1.3 Roadmap

This thesis is organized as follows. We outline the techniques related to our work in Section 2. Reputation-based trust management approach is discussed in Section 3. We introduce the proposed anomaly detection technique and the integrated trust framework in Section 4. Finally, we conclude the thesis in Section 5.

2 RELATED WORK

The proposed framework aims at enhancing the effectiveness of trust management in P2P networking paradigm by incorporating the reputation computation and peer-profile based anomaly detection. In this section, we briefly outline the techniques related to our work. Section 2.1 discusses the role of trust management and other techniques for classifying peer trustworthiness while Section 2.2 presents an overview of profiling and anomaly detection.

2.1 Identifying peer trustworthiness

Current research efforts in the area of identifying trustworthy peers on P2P network have mainly aimed to identify and penalize malicious nodes. Therefore, most techniques in this area have been focused on trust management. Aside from this, there have been several alternative approaches proposed to enhance security in P2P networks through application of intrusion detection techniques and use of load-balancing policies to limit malicious node damage.

2.1.1 Trust management

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 reputation management. For the purpose of this work we define reputation as numerical characteristic of trust and, for simplicity we will use these terms interchangeably.

According to the way peers' reputations stored we can classify existing solutions into two major groups, namely approaches employing local storage and those that maintain reputations in a distributed fashion.

Approaches employing local storage

Majority of approaches presented in this area employed special algorithms for collecting peers' reputation values in the P2P network. Generally, peer's reputations are stored by parties that in the past communicated with these peers. However, there are some research works that allow peer to store its own reputation only. We will present approaches belonging to both groups.

Peer stores and maintains reputations of other peers

P2PRep (reputation) schema proposed by Cornelli [5] is based on a distributed polling algorithm which allows peers to access reputation of potential resource providers before initiating a download. Reputation information is requested from other peers on the network and computed using personal peer's experience (number of successful and unsuccessful downloads) and credentials of peers who expressed their opinions. Although this approach addresses many security considerations for P2P networks, there is limitation to it. The approach bases reputation calculation only on the number of downloads which leaves out peers acting mostly as servers (i.e. sharing content in the network).

Similarly, TrustMe [27] introduced a protocol that allows peers to request peer's trust information in a secure and anonymous fashion using key cryptography mechanisms. Since main focus of this work was communication protocol no trust value computation schema was provided. The approach introduced by Selcuk [26] considers both trust computation and collection algorithms. Each peer stores trust and distrust values for peers it communicated in the past as binary vectors. Trust data for unknown peers is requested through trust query. The responses from peers are weighted according to their credibility.

NICE [17] is a reputation-based trust management approach 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. Similarly, EigenTrust model [13] relies on peer's cooperation. The model proposes to compute peer's

global trust value that reflects experiences of all peers in the network with this peer. Although approach can effectively identify malicious peers it is highly dependant on all peers' participation. In similar fashion Papaioannaou and Stamoulis, in [23], suggest to calculate peer's reputation based on the feedback from other peers. However, only a subset of all peers is required for providing feedback. Authors further propose to complement reputation value by reputation-based policies which, on one hand, help peers to find a right resource provider and, on the other, guide resource-providers to identify a peer to be served.

Wang and Vassileva [32] proposed an interesting approach to compute trust using Bayesian networks. The approach differs from the majority of the existing schemas by offering to differentiate aspects of trust according to peer's capability in such areas as download speed, file type and file quality. Trustworthiness of a peer can also be learnt from other peers in the network through recommendation request.

Also differentiated approach to reputation was suggested by Michiardi et al. [20]. Reputation value is considered in three aspects: subjective that is calculated from direct observations, indirect that considers information provided by other nodes and functional that represents global value that aggregates subjective and indirect values with respect to some node's function. Although the approach is designed for mobile ad hoc networks it can be also extended to P2P environment.

Peer stores and maintains its own reputation

The schema presented by Gupta et al. [11] tracks positive peer's contribution to the system using a debit-credit mechanism. Reputation of a peer is periodically recomputed by Reputation Computation Agent (RCA) based on query-response messages processed by this peer. Although approach ensures security and integrity of reputation values, it does not provide mechanisms for decreasing the reputation for malicious behavior.

Approaches R-Chain[18] and RCert[22] are similar transaction-based reputation management systems in a sense that both schemas organize reputation information as a chain and

require third party verification of the transaction. Since each peer maintains and stores only its own reputation such verification ensures integrity of data. In case of transaction, involved parties can request each other's reputation.

Approaches maintaining reputation in distributed manner

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. Peer reputation can be retrieved by following the corresponding path. The model also does not have any preventive mechanism from inserting arbitrary number of false complaints.

In general the existing trust management approaches suffer from one or more of the following serious drawbacks:

- *Necessary cooperation of peers.* Majority of the proposed techniques rely on peers' cooperation in the network which might not be available, for example, due to conspiracy of malicious peers.
- *Generate considerable network traffic.* Schemas require search of the reputation data or other peers' interaction necessary for reputation computation which generates considerable network traffic.
- *Limited to download transactions.* Most of the techniques only consider outcome of download transactions for reputation computation ignoring the rest of the interactions between peers.

We consider a reputation-based trust management approach that addresses mentioned above problems.

2.1.2 Alternative research efforts

The approach proposed by Abimbola [1] employed the Snort intrusion detection system (IDS) to detect malicious nodes in a P2P environment. P2P network traffic was analyzed for unique patterns, which were used to develop Snort signatures. Based on these rules Snort detected malicious traffic in P2P network. Although the results showed a very low false positive rate, as the system was signature-based, the proposed technique is limited to only known intrusions.

A distributed IDS, called Indra, based on trust among peers was proposed by Janakiraman [12] and employed the notion of a "neighborhood watch". Peers on the network watch for intrusion attempts and once an intrusion occurs, the attacked peer notifies its trusted neighbors. This work is currently in progress and experimental results are yet to be reported.

The approach proposed by Daswani and Garcia-Molina [7] 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.

2.2 Profiling and anomaly detection

Profiling is a well known technique that has been applied to many areas such as law enforcement, e-commerce and intrusion detection. Originally, the idea of using user profiling for intrusion detection was proposed by Denning [8]. She described a rule-based intrusion detection model that monitors users' activity logs and reports detected abnormal patterns.

Most of the research efforts in the area of user profiling in intrusion detection have been focused on Unix operating systems [6,15]. In particular, the work done by Dao [6] concentrated on monitoring user activity using login time, host information and command sequences. Concepts related to profiling, as well as some dependencies of user profiles, have been discussed but overall the work was exploratory and only sought to provide a basis for future research.

Further steps in this direction were taken by Lane and Brodley [15] using data mining techniques to perform anomaly detection over user profiles. Sets of shell commands were used to characterize user behavior, which were then evaluated by supervised instance based learning (IBL) algorithm. They also evaluated several data reduction techniques, an important factor in minimizing system overhead. This approach demonstrated good results. However, one of the limitations of IBL algorithm is that it relies on pure "normal" data for training which may be difficult to achieve in a real-time anomaly detection system.

Many researchers have studied anomaly detection techniques as a tool for intrusion detection [9,16,25]. Lazarevich [16] provided a comparative overview of these techniques. Portnoy [25] employed an unsupervised cluster-based algorithm, referred to as single-linkage, which computes distance based on similarity measure between data points and merges them with the closest cluster. Evaluation of the algorithm was performed on a set of real network data and showed on average 40-50% detection rate, where detection rate was defined as the ratio of the number of intrusions detected by the algorithm and the total number of intrusion instances in the data set. In most practical settings, the proposed method with such low detection rate may not be fruitful, even when taking into account its efficiency and ability to use normal and intrusive data for training.

Later Eskin [9] also explored algorithms for unsupervised anomaly detection over real network data and system calls using clustering, a k-nearest neighbor algorithm and support vector machines. The results showed a significant trade-off between detection and false positive rates i.e.

increase in the percentage of correctly detected intrusion instances corresponded to an increase in the number of false positives (normal instances incorrectly identified as malicious).

In our framework, we employed unsupervised anomaly detection over peer profile data in P2P environment.

3 REPUTATION-BASED TRUST MANAGEMENT MODEL

The approach we are presenting in this section 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 in section 1.1 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 peer providing file is usually based on the characteristics of the offered file (file size, file name) as well as an uploading speed of the offering peer. The rest of the traffic in Gnutella network is processed without such detailed 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, for example, generating as many queries as possible, uploading corrupted files 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 [5,11], 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.

3.1 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 intended services and factors destructing it. Among these factors are *resource search, resource upload, resource download* and *traffic extensiveness.*

Resource search is essentially the 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 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_k be a current load from peer k and t be a threshold, then the traffic from peer k is extensive if the following holds

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

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* of peer i (R_i) at specified time period is defined as a percentage of bad actions (BA_i) performed by the peer at that given time period and calculated as follows:

$$R_i = 100*BA_i/TA_i$$

where TA_i denotes total number of actions by peer 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 Table 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 is completely trustworthy.

Table 2: Trust thresholds

The correspondence between trust thresholds and trust score is presented in Table 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

Table 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 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. 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 and correctly processing queries sent to it.

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, allowing it to increase its reputation by behaving properly on the network. At the same time minimum average trust should help to expose malicious intentions from the beginning: peer's bad behavior will immediately results in its trust score decrease and, therefore, limit malicious peer's ability to harm network.

3.2 Design and implementation

The architecture of our reputation-based model is comprised of the several components that are displayed in Figure 1. 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 component 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.

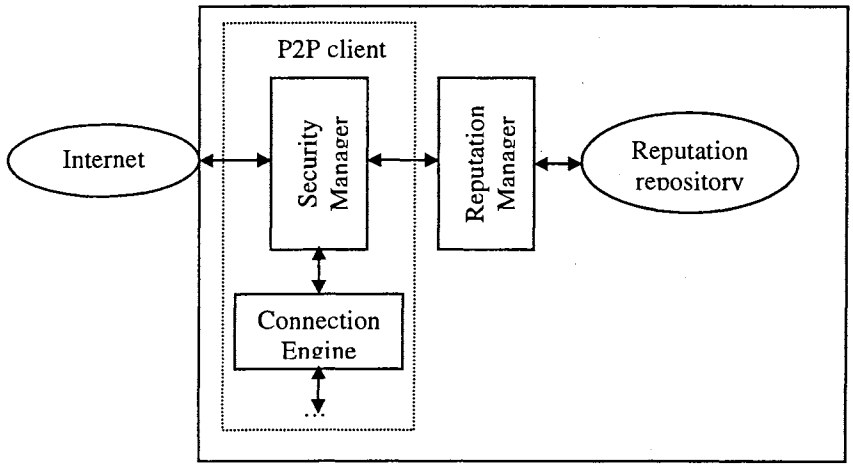


Figure 1: System overview

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 [24]. However, this architecture can be applied to other P2P clients. For implementation of the local reputation repository the MySQL database management system was used.

3.2.1 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.

3.2.2 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 2.

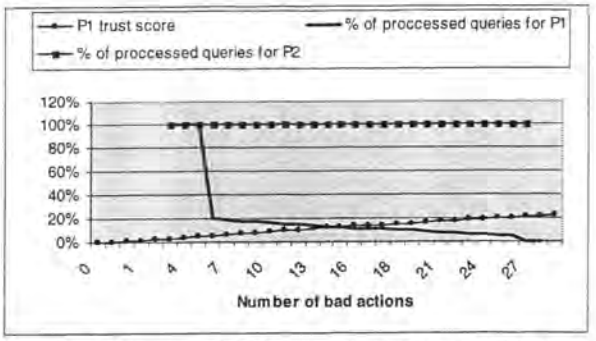


Figure 2: Decrease of full reputation when peer P1 starts "acting" maliciously

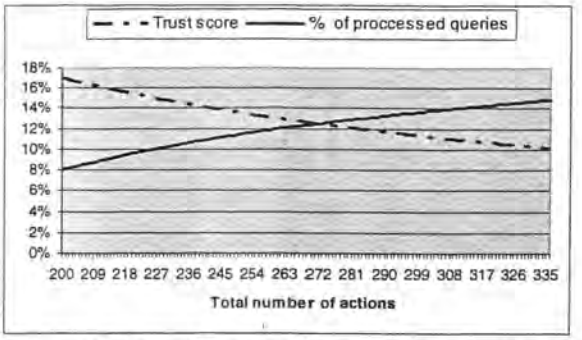


Figure 3: Reputation gain when peer starts "acting" properly

Trust score increases rapidly as the number of harmful actions (BA) increases. As the experiment showed the 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 lowest average reputation is able to gain the trust back if it starts behaving properly.

The reputation gain is presented in Figure 3. 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 2) while almost 100 actions is needed to bring reputation from bad to average level. Figure 3 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.

3.3 Discussion

In this section we present a comparative analysis of the proposed approach with existing reputation–based algorithms. Table 4 summarizes the main factors considered for assessing peers' reputation in four different models.

Factors taking into consideration for reputation computation	Debit-Credit Reputation Computation (DCRC)[11]	NICE [17]	P2P Rep [5]	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	

Table 4: Comparison of different reputation-based calculation schemes

Unlike our approach Debit-Credit Reputation Computation (DCRC) schema [11] 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 maintaining a good reputation.

P2P Rep (reputation) [5] 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 [17] 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.

We anticipate that 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.

4 INTEGRATED P2P TRUST FRAMEWORK

As we mentioned in introduction reputation-based approach alone might not capture a peer's behavior accurately. This shortcoming can be effectively avoided by complementing reputation trust management by anomaly detection technique. In this section, we describe an anomaly detection technique based on the peer profile and integrate it into our trusted management scheme presented in the previous section.

Anomaly detection relies on flagging potentially malicious peer activity by detecting deviation of peer behavior from its normal profile. Behavior of peer, for the purpose of this work, is defined using a set of data, peer session data, while the peer is online. In our framework, anomalous peer behavior affects its trust value in two phases: (a) once at the end of the session and (b) at user predefined regular intervals during the session. While the former update, gives a global view of the peer's reliability, the latter is required to update trust scores on-the-fly so that malicious activity does not go un-noticed while in session.

We will describe the peer profiling technique in section 4.1 followed by our anomaly detection in section 4.2 and 4.3.

4.1 Peer profiling

A peer profile (user profile) as it relates to P2P networks is a collection of information that establishes the user's typical behavior on the network. Some users connect to the network first thing in the morning to browse for new MP3 files while others may stay up late downloading large video files. Other users might vary their pattern greatly over the course of time. Regardless of the patterns, their behavior can be used to create a profile of normal usage.

We identify six features to characterize peer behavior in P2P framework. These features can be broadly classified into two groups reflecting the temporal behavior and the network activity of each peer respectively.

(1)Temporal features:

(a) connection time - Most users tend to connect to the P2P network at particular period of time, which may be explained by their regular life schedules as well as personal preferences. Therefore, a user who always connects to the network at 10pm is unlikely to connect at 9am in the morning. If this happens it can be considered an anomalous event.

(b) connection duration – Connection duration is defined as a duration of the online session from the time user connects to a monitoring peer until the time it disconnects. Similar to the first group, connection duration depends on individual user preferences.

(2)Network activity:

(a) number of search requests, (b) number of file downloads and (c) number of file uploads during connection - These parameters characterize the level of user activity on the network. Number of bytes received from a remote peer is strongly correlated with the first three parameters.

(d) number of bytes uploaded by a peer - Number of uploaded bytes determines the search preferences of the user. If the normal peer behavior includes upload of several multimedia files which are usually quite large then a sudden upload of system files will be considered an anomaly and will be reflected in the number of bytes uploaded by this peer.

The information related to the above features is collected for each peer throughout its online session and is referred to as *peer's session data.*

4.2 Anomaly detection

In general, anomaly detection techniques build a model through training data and then detect deviations in new data sets. Our definition of an anomaly for the purposes of this paper is any event that does not fit the normal behavioral profile of the peer. For example, it would be unusual for a peer that only downloads MP3s to suddenly upload executable files to other clients.

There are two approaches to anomaly detection based on data mining techniques: supervised and unsupervised learning. The main difference between unsupervised and supervised techniques becomes clear when compared in terms of input data. Supervised learning requires input data labeling or some sort of input template to which the output can be matched to determine whether the output data is classified correctly or not. Unsupervised learning algorithms do not require any indication of how the output should look and therefore do not need any external interference. These algorithms learn from data as it becomes available and thus the technique is often referred to as adaptive.

Unsupervised learning is especially useful when prior data does not exist and therefore no template can be formed. Often in practice pure normal data is not available unless such data is gathered through simulations. Real data obtained from the network cannot be guaranteed to be free of anomalous events. Additionally, a model trained on simulated data does not reflect the reality and is limited to the anomalies found in the simulations. For these reasons, in our anomaly detection framework we will employ unsupervised algorithm.

Another important feature of anomaly detection in our framework is incremental learning. These algorithms incrementally process new data as it becomes available [3]. In dynamic environment such as P2P, network size of the training data can increase exponentially. Therefore, it is not practical to store the entire data set including incoming new data throughout the learning process. It is desirable to train the algorithm on the new data set only and thus dispose data that has already been analyzed; this process is called incremental learning. Although the algorithm evaluated

herein is a batch learning algorithm, requiring that the entire data set be processed whenever new data is added, incremental version of it is available [3].

The algorithm we employ in our framework is one-class support vector machine (SVM). This is an unsupervised version of the classic supervised two-class SVM. As opposed to the classic algorithm, which attempts to break data points into two classes by a hyperplane, one-class SVM maximally separates all data from the origin, also using a hyperplane. The majority of data points on one side of the hyperplane are classified as normal while outliers on the other side are labeled abnormal.

4.3 Anomaly Detection procedure

Anomaly detection component in our framework performs two main functions. First, it analyzes a peer's session data to reveal hidden abnormal information using anomaly detection algorithm. Second, if the data is established as anomalous it determines the degree of anomaly and estimates amount of changes to be done to the peer's reputation score.

Analysis of peer's session data was performed using one-class SVM algorithm from libsvm tool [4]. Although the anomaly detection component employs support vector machines, the design of the framework is independent of a particular algorithm and, therefore, can be easily adapted to any other unsupervised anomaly detection algorithm.

We determine the degree of anomaly for the peer's session data based on the two most commonly used statistical metrics: mean and standard deviation. Mean indicates the central tendency of the data, while standard deviation provides a measure of variability of the data. Since we cannot guarantee the normal distribution of the data[1] we use these metrics according to the Chebyshev's rule [19], which can be applied regardless of the shape of the distribution. According to this rule, at least

[1] Data distribution often depends on a particular user behavior which can be radical and unstable.

$1\text{-}1/k^2$ of data points will fall within k standard deviations from the mean. As such, majority of data, at least 89%, falls within three standard deviations from the mean.

We calculate these metrics for each peer profile to obtain full descriptions of the data sets. Based on these calculations we can determine how far the anomalous data is from the normal data mean and measure this distance in standard deviations. Since about 89% of data are at most three standard deviations away from the mean, we will consider data beyond the range of three standard deviations. However, larger number of standard deviations can be used to consider bigger range of data as normal.

$\{x_1,..,x_i,..x_n\}$	- an anomalous session data, where $x_{i,i=1..n}$ are the peer profile features
$\{M_1, ...M_i, ...M_n\}$	- mean values of profile data set
$\{stdDev_1,... stdDev_i... stdDev_n\}$	- standard deviations values of profile data set

Formally, we define *Distance* ($Dist_i$) as number of standard deviations x_i lies from the mean value M_i.

$$Dist_i = (|(x_i - M_i)| / stdDev_i) - 3$$

The degree of anomaly is calculated for the entire session data based on the feature distances and will indicate the total deviation of the session from the normal behavior. Let Da_p be a degree of anomaly of peer's p session data and n be a number of features, then

$$Da_p = 1/n \sum_{i=1}^{n} Dist_i$$

The final step in the anomaly detection procedure is to determine the change in the reputation score depending on the calculated degree of anomaly for the anomalous session and anomaly thresholds. We establish two anomaly thresholds ATh_1 and ATh_2 that indicate the level of anomaly acceptable by a peer. Since we only consider data lying beyond the range of three standard deviations, anomaly thresholds are intended to determine how big the deviation is from this cut-off point. The corresponding reputation update is presented in Table 5.

$Da < ATh_1$	Bad actions value is incremented by b_0[2]
$ATh_1 < Da < ATh_2$	Bad actions value is incremented by b_1
$ATh_2 < Da$	Bad actions value is incremented by b_2

Table 5: Correspondence between degree of anomaly and reputation update

Peer's session data is evaluated at the end of its session as well as during the session. While peer is online, its feature values are monitored for deviations from the profile data at regular user-predefined intervals (*checkPeriod)*. If deviation occurs, features are analyzed against peer profile by anomaly detection algorithm.

4.4 Design and implementation

The architecture of our framework is comprised of two main components: the user-profile based peer-to-peer simulator and an anomaly detector. The architecture is presented in Figure 4.

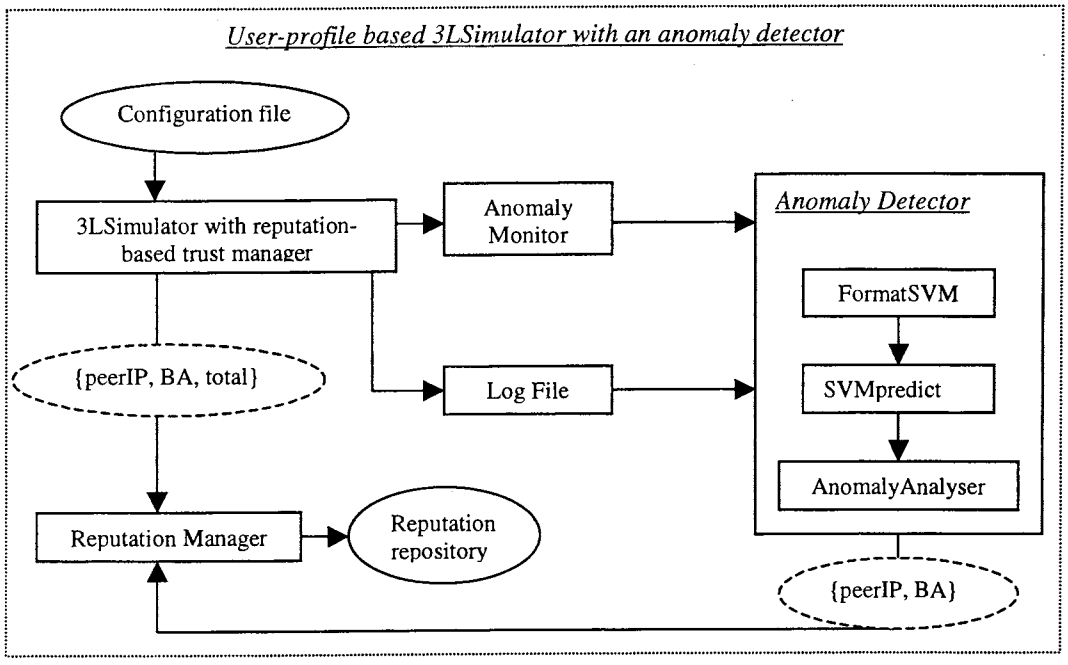


Figure 4: Framework architecture

[2] Parameters b_0,b_1,b_2 can be configured by system administrator.

The core module of the system is 3LSimulator that is a generic peer-to-peer simulator supporting Gnutella protocol v.4.0 [30]. It was modified to simulate peer behavior and capture necessary information about peers during their sessions into the log file. The simulator also includes a reputation-based trust management mechanism employed in our framework.

Upon termination of all peers' connections, the log file is written to disk and then analyzed by the Anomaly Detector module. Anomaly Detector component contains three main parts, namely, FormatSVM, SVMpredict and AnomalyAnalyser.

FormatSVM module normalizes a log data and converts it to a SVM format. Converted data are evaluated by SVMpredict which employs libsvm tool [4]. The output of SVMpredict is analyzed by the last module, AnomalyAnalyser to determine if the log records are anomalous and, in this case the degree of anomaly. Based on the anomaly degree, request to update a trust score containing peer's IP address and number of BA is sent to Reputation Manager.

In addition to these modules, framework includes Anomaly Monitor component whose main function is to determine possible anomalies in peers' activities while the peers are online. Anomaly Monitor periodically examines behavior of connected peers to identify deviations from normal peer behavior. If such deviations are found, the peer session data is transferred to Anomaly Detector to determine if anomaly exists and to request a reputation update for a peer.

4.4.1 Data sets

As a preliminary step, we generated data sets using the user-profile based peer-to-peer simulator for five peers. Each peer's data set was split into two sets: train data set, consisted of approximately 200 records and test data set containing 60 records. In order to make our train data set more realistic we added simulated abnormal instances so that the resulting set contained 5% to 10% of the anomalies. Train and test data sets were normalized with libsvm's included normalization tool.

Anomalous instances were generated by changing the configuration parameters of the simulator. The configuration parameters include original settings for peers' profiles such as connection time, connection duration, number of query requests and number of uploaded bytes. During simulation initial feature values were modified based on uniform random distribution within a range specified for each peer's profile.

To generate abnormal data instances based on the normal initial configuration parameters we use standard deviation value of the train data sets by increasing each of the specified parameters by four standard deviations.

4.4.2 Results

We evaluated our framework on a completely connected network of five peers. We were interested in peers' trust score change in five scenarios: increase of connection time, connection duration, number of query requests, number of uploaded bytes and increase of all mentioned features. These scenarios were run in two modes: full mode that represented our trust framework with anomaly detection mechanism and trust mode that included reputation-based approach that we employed in the framework without anomaly detection mechanism. Simulation settings for our experiments are presented in Table 6.

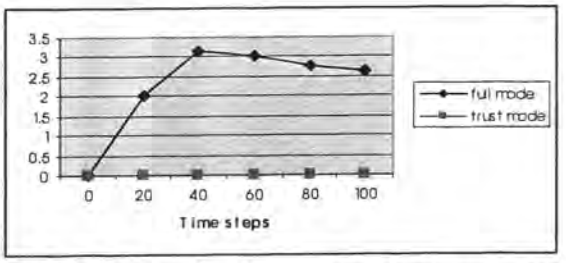
time period	2 time steps
extensive traffic threshold	1.7
trust thresholds	x_1=20 and x_2=5
anomaly thresholds	ATh_1=1 and ATh_2=2
SVM –n parameter	0.1
initial peers' trust score	0
checkPeriod	20 time steps
b_0,b_1,b_2	1,2,3

Table 6: Simulation settings

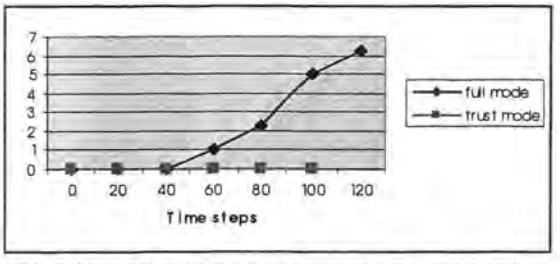
We present our results in two parts. First, we analyze the effectiveness of anomaly detection procedure to reflect trust scores of peers on the network in general. Then we look at the particular peer's trust score patterns.

General results

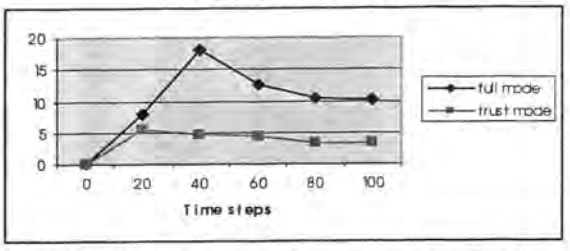
This first set of the results in Figure 5 shows average trust score value of the peers in the system. Parameters "connection time", "connection duration" and "number of uploaded bytes" are not considered in the trust score calculation in the trust mode, therefore, as we expected, trust score in those experiments (a, b, d) remains the same. However, in the full mode we see that trust score takes different patterns which reflect deviation in the peers' behavior from their profiles. It is interesting to note that even though averaging trust score we negate the effect of some trust score changes that deviate from average value, the full mode always reflects changes in peers' behavior better than trust mode alone.



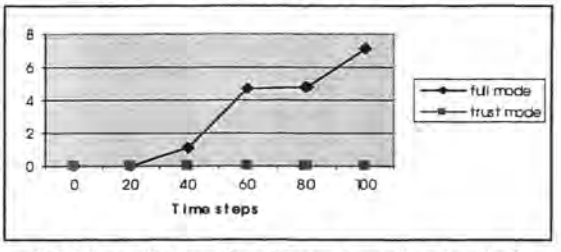
a) Effect of changing "connection time" parameter on peers trust score



b) Effect of changing "connection duration" parameter on peers trust score



c) Effect of changing "number of requests" parameter on peers trust score



d) Effect of changing "number of uploaded bytes" parameter on peers trust score

Figure 5: Average trust score change in four scenarios

Particular patterns

As the experiments results showed, all peers displayed similar patterns, therefore, for brevity we only introduce representative trust score patterns from peer 4's view on its neighboring peers.

1. Change of "connection time" & "connection duration" parameters

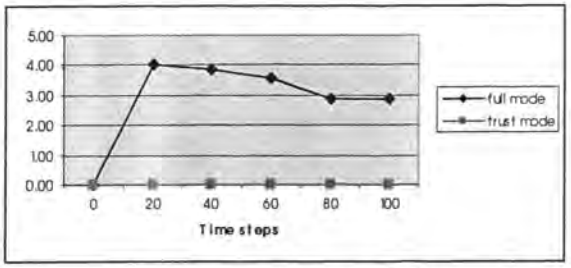


Figure 6: Trust score pattern with "connection time" parameter change

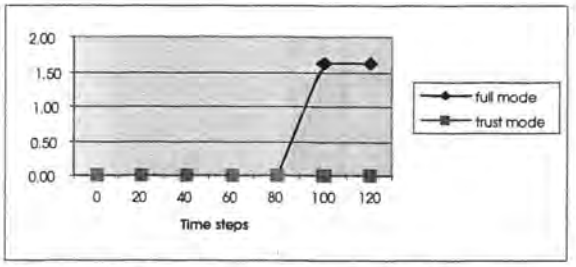
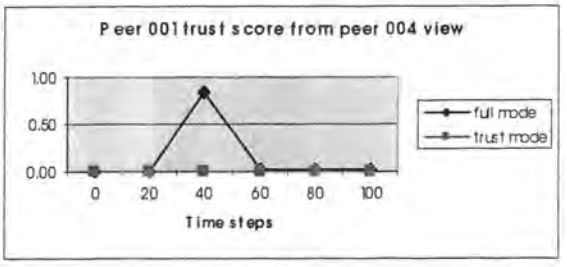


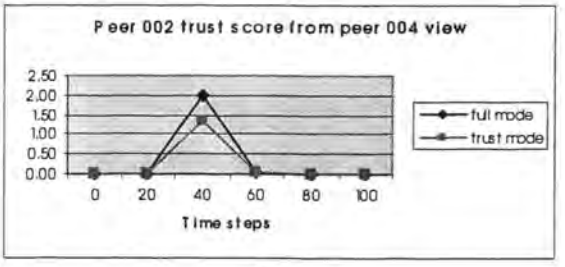
Figure 7: Trust score pattern with "connection duration" parameter change

Deviation due to unusual connection time can be detected at the beginning of the peer session. Depending on the abnormality of this feature initial increase of trust score can be more or less rapid. After this increase, if no other deviations from the peer's profile are detected throughout the session, peer's trust score decreases. Rate of decrease depends on subsequent peer's behavior, the better its behavior the faster its trust score decreases. We see similar results for "connection duration" parameter change. Note that, abnormal connection duration is visible only if peer stays online longer than its normal time or disconnects much earlier than usual.

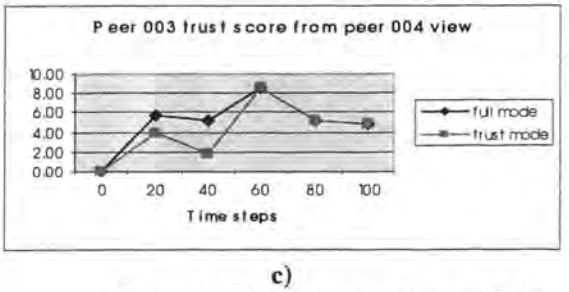
3. Change of "number of query requests" parameter



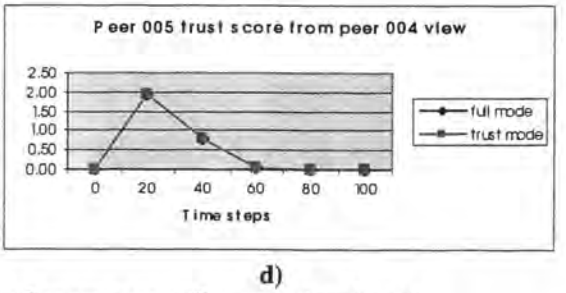
a)



b)



c)

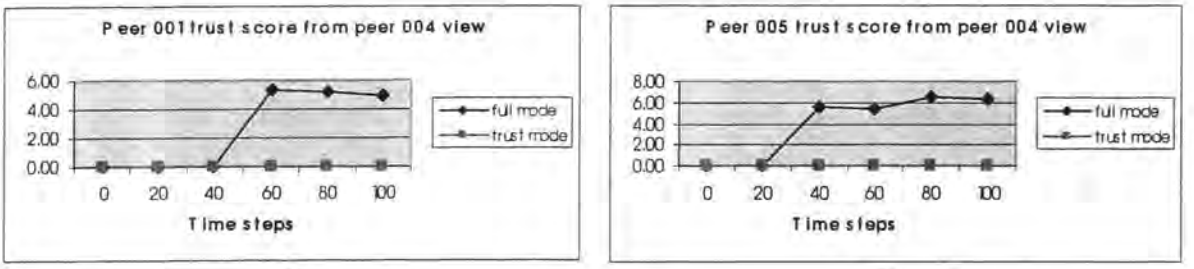


d)

Figure 8: Trust score pattern with "number of query request" parameter change

Figure 8 shows the impact of the number of query requests on the trust score in full and trust modes. In the trust mode this parameter is not considered in the trust score computation. However, increased number of queries may result in the extensive traffic and therefore, can be expressed in the trust score computation through traffic extensiveness factor. On the other hand, equal increase in the number of requests might create approximately equal amount of traffic and is not reflected in the trust score (Figure 8, a). This can potentially be used by malicious peers conspired to damage a victim node. Meanwhile, full mode is able to detect an unusual behavior. If large number of requests is distributed throughout a peer's session then full mode keeps the trust score high (c). However, if the requests are more localized and anomaly detection does not see continuous increase in the traffic then effect of trust score increase by full mode is negated by the total number of actions. In other words, if a peer starts session with a malicious behavior but quickly drops it, its trust score will reflect that.

4. Change of "number of uploaded bytes" parameter



a) **b)**

Figure 9: Trust score pattern with "number of uploaded bytes" parameter change

Number of uploaded bytes parameter is also not considered in trust mode, therefore changes in this feature does not impact peer trust score. Full mode, as we expected, shows detected deviation from the peer profile in the current actions. This abnormal behavior is detected later in the session when peers exchanged queries and determined nodes to upload desired files.

5. Change of all parameters

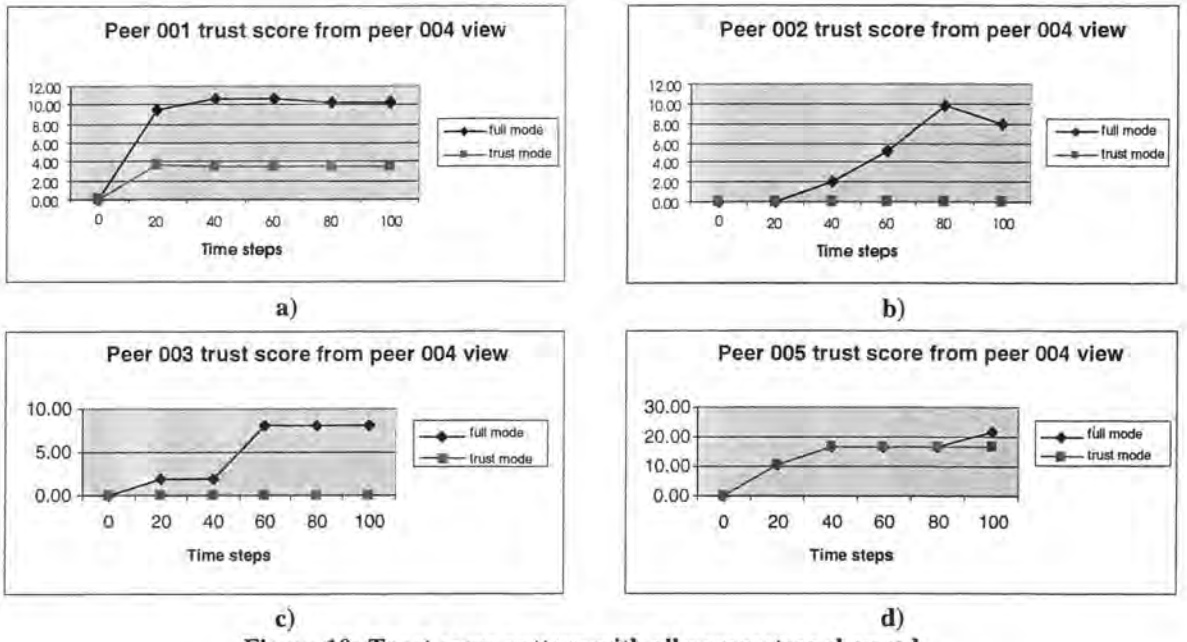


Figure 10: Trust score pattern with all parameters changed

Figure 10 illustrates that full mode is always able to detect malicious peer behavior even if this behavior is not flagged as "bad" by reputation-based trust mechanism. Similar to "number of query requests" parameter increase, if bad actions are spread throughout the peer's session, trust score in full mode rises rapidly.

Majority of our results showed clear difference of trust score between full and trust modes, however, on few occasions we noticed that full mode was not able to detect anomaly. One potential reason is false negative response of SVM algorithm to anomalous data. Although, the detection rate of one-class SVM is generally high, 100% performance is not desirable since it becomes an indicator of overfitted data and, therefore, inability of algorithm to detect unknown anomalies.

5 CONCLUSION

5.1 Summary

The trust framework we presented in this work integrates reputation-based trust management scheme and peer-profile based anomaly detection technique in P2P environment. The trust management scheme considers peers' activity on the network and computes peer's reputation value based on the following factors: resource search, resource upload, resource download and traffic extensiveness. The characteristics of our approach are

- decentralized: reputations are stored and computed locally.
- independence: reputation is computed independent from peers' willingness to cooperate in calculation of their reputation, and
- on-demand: the approach employs only on-demand calculations.

Peer-profile based anomaly detection in our framework provides an enhancement to reputation-based trust management. Monitoring two classes of peer activities, temporal activities and network activities, anomaly detection captures behavior not considered by reputation mechanism.

Using simulation we demonstrated that our framework is able to capture peer's behavior more accurately than a reputation-based approach alone. These results provide strong testimony that the proposed framework can be used in many settings including intrusion detection and e-commerce applications.

5.2 Future work

The future directions of this work include:

- *Detailed evaluation of system efficiency.* Although we did not specifically evaluated system overhead of the proposed model, we anticipate that it will not be significantly affected in our trust framework by the addition of anomaly detection.
- *Application of our trust framework to hierarchical setting through super-node concept.* In particular, reputation computation can be distributed among trusted nodes in groups headed by a super-node. Such distribution of trust calculation in large networks is essential for real-time anomaly response.

6 REFERENCES

[1] Abimbola, A., Shi, Q. and Merabti, M. Using Intrusion Detection to Detect Malicious Peer-to-Peer Network Traffic. *PGNet*, Liverpool, UK, 2003.

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

[3] Caragea, D., Silvescu, A., and Honavar, V. Agents That Learn from Distributed Dynamic Data Sources. *In Proceedings of the ECML 2000/Agents 2000 Workshop on Learning Agents.* Barcelona, Spain, 2000.

[4] Chang, C.C. and Lin, C.J. Libsvm: a library for support vector machines (version 2.31). Accessed October 28, 2004. http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[5] Cornelli, F., Damiani, E., di Vimercati, S., Paraboschi, S. and Samarati, P. Choosing Reputable Servents in P2P network. *In Proceedings of WWW*, Honolulu, Hawaii, 2002.

[6] Dao, V. and Vemuri, V. Profiling Users in the UNIX OS Environment. *International ICSC Conference on Intelligent Systems and Applications*, University of Wollongong, Australia, 2000.

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

[8] Denning, D. E. An intrusion-detection model. *IEEE Transactions on Software Engineering,* vol. SE-13, 1987.

[9] Eskin, E., Arnold, A., Prerau, M., Portnoy, L. and Stolfo, S. A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data. *Applications of Data Mining in Computer Security*, Kluwer, 2002.

[10] g2dn:Gnutella2 Developers' Network. Accessed October 28, 2004. http://www.gnutella2.com.

[11] Gupta, M., Judge, P. and Ammar, M. A Reputation System for Peer-to-Peer Networks. *In the Proceedings of NOSSDAV*, Monterey, USA, 2003.

[12] Janakiraman, R., Waldvogel, M. and Zhang, Q. Indra: A Peer-to-Peer Approach to Network Intrusion Detection and Prevention. *In Proceedings of IEEE WETICE 2003 Workshop on Enterprise Security*, Linz, Austria, 2003.

[13] Kamvar, S. D., Schlosser, M. T., and Garcia-Molina H.. The EigenTrust Algorithm for Reputation Management in P2P Networks. *In Proceedings of the Twelfth International World Wide Web Conference*, Budapest, Hungary, 2003.

[14] KaZaA. Accessed October 28, 2004. http://www.kazaa.com/us/index.htm

[15] Lane, T. and Brodley, C. Temporal sequence learning and data reduction for anomaly detection. *In 5th ACM Conference on Computer & Communications Security*, San Francisco, USA, 1998.

[16] Lazarevic, A., Ertoz, L., Ozgur, A., Srivastava, J. and Kumar V. A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection. *In Proceedings of SIAM Conference on Data Mining*, San Francisco, USA, 2003.

[17] Lee, S., Sherwood, R. and Bhattacharjee, B. Cooperative Peer Groups in NICE. *Proceedings of IEEE INFOCOM*, San Francisco, USA, 2003.

[18] Liu, L., Zhang, S., Ryu, K.D. and Dasgupta, P.. R-Chain: A Self-Maintained Reputation Management System in P2P Networks. *17th International Conference on Parallel and Distributed Computing Systems,* San Francisco, USA, 2004.

[19] McClave, J. and Dietrich, F. Statistics. Dellen Publishing Company, 4nd edition, 1988.

[20] Michiardi, P. and Molva, R.. CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. *In Proceedings IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security, Portoroz, Slovenia*, 2002.

[21] Napster. Accessed October 28, 2004. http://www.napster.com.

[22] Ooi, B.C., Liau C.Y. and Tan, K.L.. Managing Trust in Peer-to-Peer Systems Using Reputation-Based Techniques. The 4th International Conference on Web Age Information Management, Chengdu, China, 2003.

[23] Papaioannou, G. Thanasis and Stamoulis, D. George. Effective Use of Reputation in Peer-to-Peer Environments. Proc. of IEEE/ACM CCGRID 2004 (Workshop on Global P2P Computing), Chicago, USA, 2004.

[24] Phex - the Gnutella P2P filesharing client. Accessed October 28, 2004. http://phex.kouk.de.

[25] Portnoy, L., Eskin, E. and Stolfo, S. Intrusion Detection with Unlabeled Data using Clustering. *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*, Philadelphia, USA, 2001.

[26] Selcuk, A., Uzun, E. and Pariente, M. . A Reputation-Based Trust Management System for P2P Networks. *CCGRID2004: 4th IEEE/ACM International Symposium on Cluster Computing and the Grid,* Chicago, USA, *2004.*

[27] Singh, A. and Liu, L. TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems. *Third International Conference on Peer-to-Peer Computing (P2P'03)*, Linkoping, Sweden, 2003.

[28] Single, A., Rohrs, C. Ultrapeers: Another Step Towards Gnutella Scalability. Accessed October 28, 2004. http://www.limewire.com/developer/Ultrapeers.html.

[29] Stakhanova, N., Ferrero, S., Wong, J. and Cai, Y.. A reputation-based trust management in peer-to-peer network systems. *2004 International Workshop on Security in Parallel and Distributed Systems,* San Francisco, USA, 2004.

[30] The Gnutella Protocol Specifications v0.4. Accessed October 28, 2004. http://cnscenter.future.co.kr/resource/hot-topic/p2p/GnutellaProtocol04.pdf.

[31] 3LSimulator. Accessed October 28, 2004. http://bistrica.usask.ca/madmuc/Grads/Nyik/3LS/.

[32] Wang, Y. and Vassileva, J..Trust and Reputation Model in Peer-to-Peer Networks. *Third International Conference on Peer-to-Peer Computing (P2P'03)*, Linkoping, Sweden, 2003.