A Reputation-based Trust Management System for P2P Networks*

Ali Aydın Selçuk[1], Ersin Uzun[2], and Mark Reşat Pariente[3]
(Corresponding author: Ali Aydın Selçuk)

Department of Computer Engineering, Bilkent University[1]
Ankara, 06800, Turkey (E-mail: selcuk@cs.bilkent.edu.tr)
Department of Information and Computer Science, University of California, Irvine, CA 92697-3425, USA[2]
VMware Inc., Palo Alto, CA 94304, USA[3]

(Received Apr. 6, 2006; revised and accepted May 31, 2006 & July 31, 2006)

Abstract

The open and anonymous nature of a P2P network makes it an ideal medium for attackers to spread malicious content. In this paper, we propose a reputation-based trust management system for P2P networks that aims to build confidence among the good members of the community and identify the malicious ones. The proposed system is simple and efficient in design and can be integrated into most first generation P2P systems easily. A diverse set of simulation experiments conducted to test the performance of the system show that it can be highly effective in preventing the spread of malicious content. The proposed system has other potential benefits as well, such as supporting the detection of free riders in a file sharing application.

Keywords: P2P network security, reputation systems, trust management

1 Introduction

A peer-to-peer (P2P) network is a computer network that does not have fixed clients and servers but a number of peer nodes that function as both clients and servers to the other nodes in the network. Although in general any networking technology that uses this model can be considered as P2P, such as the NNTP protocol used for transferring Usenet news or a wireless ad hoc network, the term is most frequently used to refer to file sharing networks over the Internet, such as Gnutella, FastTrack, and Napster. We also focus on this particular application of the more general P2P concept in this paper and use the term to refer to P2P file sharing systems unless otherwise is stated.

By the nature of its architecture, a P2P file sharing system provides an open, unrestricted environment for content sharing. This openness of a P2P network also makes it an ideal environment for attackers to spread malicious content, such as the VBS.Gnutella worm [14].

Reputation-based systems are used to establish trust among members of on-line communities where parties with no prior knowledge of each other use the feedback from their peers to assess the trustworthiness of members of the community [13]. One well-known such system is the rating scheme used by the eBay on-line auction site [8].

In this paper, we propose a reputation-based distributed trust architecture for P2P networks to identify malicious peers and to prevent the spread of malicious content. The protocol is based on the query-response architecture of the first generation P2P networks and is suitable for operation in a Gnutella- or Kazaa-like system. The design is simple and intuitive and, as the simulation results show, it can be highly effective in preventing the malicious content.

The rest of this paper is organized as follows: In Section 2, we describe the protocol in its basic form. The rationale for its design is presented in Section 3, and the security extensions on the basic protocol are discussed in Section 4. Results of the simulation experiments which test the protocol's effectiveness are presented in Section 5. Section 6 reviews the related earlier work in comparison to our proposal. Section 7 concludes the paper with a discussion of the future work necessary for a practical deployment of the proposed protocol.

2 The Basic Protocol

When a resource is queried in a P2P file sharing system, many responses offering various alternative versions may be received, among which some may be malicious. An important security question at this juncture is how to distinguish the malicious alternatives from the benign ones.

*This research is supported in part by the Turkish Scientific and Technological Research Agency (TÜBİTAK), under grant number EEEAG-105E065. This research was done while the authors Ersin Uzun and Mark Reşat Pariente were at Bilkent University.

Our protocol aims to achieve this distinction by evaluating the alternatives according to the reputation of the peers who offer them. When the reputation information is not available locally, the P2P infrastructure is used to query the necessary information over the network.

2.1 Trust Records and Ratings

In our system, the outcomes of past transactions are stored in *trust vectors*, maintained by the peers that make the download. Every peer maintains a trust vector for every other peer it has dealt with in the past.

Trust vectors are constant-length, binary vectors of ℓ bits, where ℓ is typically 8, 16, or 32. A 1 bit represents an honest transaction, a 0 represents a dishonest one. An integer variable accompanies each vector, specifying the number of significant bits in it. The result of a new transaction is written at the most significant bit, shifting the present bits to the right. The process is illustrated in Figure 1.

A trust vector with m significant bits is read as an m-bit integer and divided by 2^m for conversion into a scalar *trust rating* in the $[0, 1)$ interval.[1] A separate *distrust rating* is also computed from the complement of the trust vector, for reasons explained in Section 3. An example computation of the trust and distrust ratings is shown in Figure 2.

Throughout the trust evaluation process, the criterion of *minimum distrust* is given priority in trust comparisons over *maximum trust.* The most trustworthy peer in a group is taken to be the one with the highest trust rating among those who have the lowest rating of distrust.

2.2 Resource Query

When a peer is to query a resource, it issues a query message which contains, among other fields, a *query ID number* (qID) which is a counter value maintained by each peer to identify its queries. In return, the response messages sent by the peers with the requested resource includes,

- ID of the querying peer (QID),
- ID of the responding peer (RID),
- query ID number (qID),
- a one-way hash of the file being offered.

The whole response message is hashed and signed by the responder. The querier is assured of the authenticity and freshness of the response by inclusion of the (QID, RID, qID) triple in the signature. The one-way hash of the file in the response enables the querier to group the identical versions together, which is used to evaluate the trust level of each alternative.

[1]Here, the use of 2^m as the divisor instead of $2^m - 1$ enables distinguishing among the straight-1 trust vectors according to the length m, favoring longer all-honest histories over shorter ones.

2.3 The Trust Evaluation Function

When responses for a resource query arrive, the querying peer organizes them into groups according to the file hash value they contain. Then it calculates a *trust score* for each version according to the reputation of the respondents as described below:

The threshold θ_T specifies the number of peers to be considered for a version's trust calculation. For a group of peers G, $known(G)$ denotes the set of peers in G about whom a trust record is available locally, and $unknown(G)$ denotes $G - known(G)$. We denote the cardinalities of these sets by $n_k(G) = |known(G)|$ and $n_u(G) = |unknown(G)|$.

The trust score for a group G is calculated locally if there is sufficient local information; that is, if $n_k(G) \geq \theta_T$. First, the peers in $known(G)$ are sorted by their trust rating, according to the min-distrust-max-trust criterion. The highest ranking θ_T peers are selected and the signatures on their responses are verified. The trust and distrust score of the version offered by G is determined as the average of the trust and distrust ratings of the θ_T selected peers.

If there is not sufficient information on G (i.e., $n_k(G) < \theta_T$), then a set of $\theta_T - n_k(G)$ random peers, denoted by $queried(G)$, are selected from $unknown(G)$ and a *trust query* bearing their IDs is issued.[2] Upon the arrival of the responses to this query, a *queried trust rating* is calculated for each peer in $queried(G)$. The trust and distrust score of the file version offered by G is determined as the average of the trust and distrust ratings of the peers in $known(G)$ and $queried(G)$.

At the end of the evaluation, the file versions are sorted by their trust scores, according to the min-distrust-max-trust criterion, and the highest ranking one is selected for download.

At this point, it would be wise to have another safety check on the trust score of the file to be downloaded since a file offered only by malicious peers may be the highest ranking one, probably due to the lack of any alternative versions. A possible threshold can be set here to allow the download of only those files with a higher trust score than distrust. Or, as a safer alternative, only the download of those files with a zero distrust score may be allowed, which would not be too restrictive provided that the query neighborhoods are sufficiently large.

2.4 The Trust Query Process

As mentioned above, a trust query is issued when there is not enough local information about the peers who offer a file. The contents of a trust query message is similar to that of an ordinary resource query message, and the responses are authenticated in the same fashion.

The credibility of the responses is evaluated according to the past records of the respondents. The results

[2]If $n_u(G) < \theta_T - n_k(G)$, then all peers in $unknown(G)$ are selected.

Trust vector: 11010000
\# of significant bits: 4

genuine download from B $\longrightarrow$

Trust vector: 11101000
\# of significant bits: 5

Figure 1: Peer A's update of its trust vector on B after an honest transaction. In this example $\ell = 8$.

Trust vector:
11101000
\# of significant bits: 5

$\Longrightarrow$

$$\text{Trust rating} = \frac{(11101)_2}{2^5} = 0.90625$$

$$\text{Distrust rating} = \frac{(00010)_2}{2^5} = 0.0625$$

Figure 2: Computation of the trust and distrust ratings from the trust vector

of the past references of a peer are recorded in a binary *credibility vector*. These vectors are managed in the same manner as the trust vectors: A 1 represents a good reference in the past, a 0 represents a bad one. The vectors are maintained as ℓ-bit variables and are converted into scalar *credibility ratings* in $[0, 1)$ by division by 2^m, where m is the number of significant bits. A *discredibility rating* is computed accordingly from the complement of the credibility vector.

The threshold θ_C specifies the number of responses to be considered in a queried trust calculation. When the responses to a trust query arrives, the querying peer sorts the responses by the credibility rating of their senders. Among them, the highest ranking θ_C responses are selected.[3] The signatures on the selected responses are verified.

The main piece of information contained in a trust query response is the respondent's trust and distrust ratings for the queried peer. Once the responses for the calculation are selected and authenticated, the *queried trust rating* is calculated as the average of the trust ratings in these messages, weighted by the net credibility ratings of their senders, where the responses with a higher discredibility than credibility are left out of the calculation. That is, if peer A issued a trust query on peer B, and the responses of peers $R_1, R_2, \ldots, R_k$, $k \leq \theta_C$, qualify for consideration, where the trust rating R_i provides on B is t_i and the credibility and discredibility ratings A has on R_i are c_i and d_i respectively, then A's queried trust score on B is

$$\frac{\sum_{i=1}^{k}(c_i - d_i)t_i}{k}. \quad (1)$$

The queried distrust rating is calculated in the same fashion, using the distrust ratings provided by the respondents.

One last thing to note here is about using the trust information coming from a peer with no credibility record. If the top θ_C trust query responses selected for evaluation contain responses with no credibility rating, which is likely to be the case when the querying peer is new to the system, using the credibility rating of zero as the weight factor would result in the loss of all the information provided in those responses. Here, a constant ϵ factor smaller than the smallest possible positive net credibility rating can be used, such as

$$\epsilon = \frac{0.5}{2^\ell - 1}.$$

The operation of the trust query and evaluation protocol is illustrated in Figure 3.

2.5 File Download

Once the file version to be downloaded is decided, one or more peers among those who has offered that version are chosen as the source of the download. This selection can be made randomly or according to a certain criterion such as the quality of service; however it is important that the selection is not done according to the highest trust ranking, which would result in an overburdening of the trusted peers.

It is possible that a peer who provided the right hash in a query is in fact malicious and will send the malicious file if selected for download. Correctness of the hash can be checked once the file is downloaded completely, but then the attacker will have succeeded in at least wasting the bandwidth of the querier. Moreover, if multiple sources are used for the download, which is a common way of downloading large files, a hash mismatch detected after the file download will not identify the malicious source, possibly ruining the reputation of the honest peers along with that of the malicious ones. To prevent this potential threat, we describe a two-level hash scheme in Section 4.3 that detects falsely reported hashes early in the download.

2.6 Update of Trust and Credibility Ratings

After the file download is complete, a user is asked to judge the file as benign or malicious. If it is rated benign, the trust rating of the peer(s) from whom the file

[3] If there are fewer than θ_C responses in total, all responses are selected.

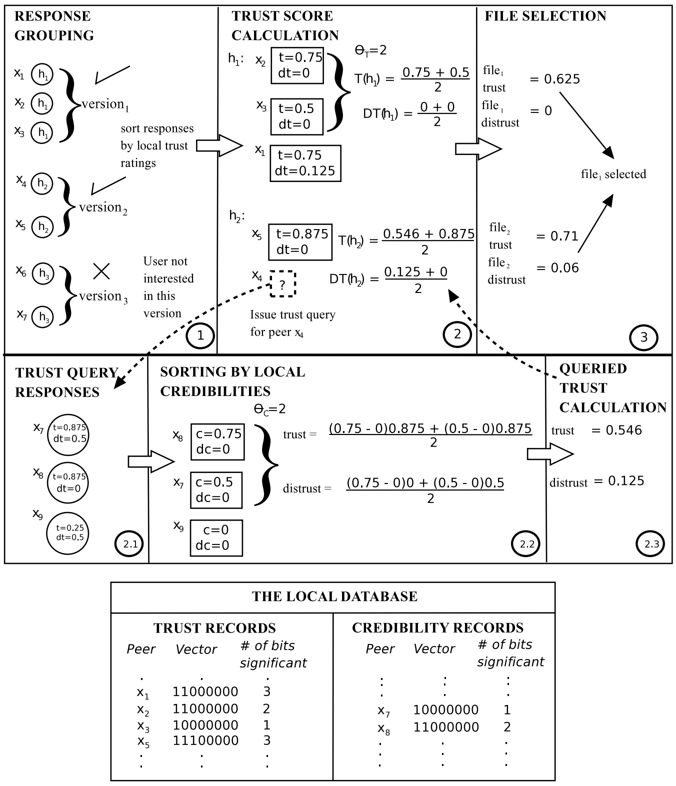


Figure 3: An illustration of the trust evaluation protocol. In response to a file query, three different replies are received among which the querier is interested in the first two. A trust comparison among these two versions follows. In this process, sufficient information is not available locally on the providers of the second version. Hence, a trust query is issued for peer x_4. At the end of the calculations, the first version turns out to be the one with a better trust score and will be downloaded from some subset of the peers $\{x_1, x_2, x_3\}$.

is downloaded is upgraded. Otherwise, the rating of the peer who sent the malicious content and the rating of those who contributed to its selection are downgraded. The difference between the two cases is due to the following fact: A malicious peer may well offer a right hash during a query in the hope of being selected and, if selected, sends the malicious content. Therefore, merely a reference for a good file is not sufficient for upgrade of the trust rating. On the other hand, if a downloaded file turns out to be malicious, all peers who offered that file can be assumed to be malicious.

The update of the credibility ratings is slightly more complex: A peer's credibility rating is updated at the end of a file download if that peer has given an authenticated opinion on a peer whose trust rating ended up being updated as the result of that download.

A credibility rating update's direction (i.e., its being negative or positive) is determined according to the opinion given and the direction of the trust rating that is updated: If a peer's trust rating is upgraded and some peer gave a positive opinion on that peer, or if both the trust rating update and the opinion were negative, then the credibility of the referring peer is upgraded. Otherwise, it is downgraded.

Another important point here is how an opinion is classified as positive or negative. Since the distrust rating has priority in evaluation over the trust rating, an opinion with a non-zero distrust rating is considered a negative one. An opinion with a positive trust rating with zero distrust on the other hand, which implies a trust rating of 0.5 or higher, is considered a positive opinion.

3 Design Rationale

3.1 Basic Trust Evaluation Process

The idea of using the feedback from other peers to assess the trustworthiness of a resource/peer is a fundamental characteristic of reputation systems [13]. In our protocol, this process is carried out in a distributed fashion due to the lack of a centralized server in P2P systems in general.

In our trust rating calculations, opinions of peers are weighted by their trustworthiness. Moreover, the evaluation is restricted to a few (θ_T or θ_C) most trusted responses. This has the purpose of preventing some low-trust responses discrediting a reliable resource/peer supported by sufficiently many trusted peers, as well as limiting the number of responses to be authenticated, which, unless restricted in number, can be a performance bottleneck.

A special feature of our trust evaluation function is the separate treatment of the distrust ratings. Although both the trust and distrust ratings are derived from the same trust vectors, handling the distrust ratings separately has the additional feature of not letting a dishonest dealing be covered up easily by a few honest transactions, which closely models real-life trust relations where a single dishonest transaction in someone's history is a more significant indicator than several honest transactions.

An important factor to be considered in reputation-based systems is temporal adaptivity; that is the ability to respond rapidly to changing behavioral patterns. Our trust rating design with binary vectors makes an efficient exponential aging scheme with an aging factor of 0.5. Besides, implementing the aging scheme by fixed-length registers rather than floating point arithmetic has the desirable feature of enabling peers to cleanse their history by doing a certain amount of honest community service after a bad deed. Note that this service must be done to the same person who was cheated, and hence a bad transaction on record will take some time to be erased completely. The number of faithful transactions required to cleanse a bad record is determined by the length of the trust vector, ℓ. If it takes a considerable amount of time to have two transactions happen between the same pair of users, $\ell = 8$ could be a reasonable choice. Higher values of ℓ could be preferred for highly active systems or in systems where cheating is considered a major offense.

3.2 Queried Trust Evaluation

A fundamental decision in our design was to use a credibility rating system separate from the trust ratings. The main risk of using the trust ratings for credibility evaluation comes from coordinated attacks where some malicious peers do as much faithful public service as they can and build a strong reputation, and then use their credibility for supporting others who spread malicious content. Having separate trust and credibility rating systems precludes such attacks.

One aspect different in the treatment of the credibility and the trust ratings is the way they are used when ranking the file or trust query responses. The trust ratings are ranked by the min-distrust-max-trust criterion whereas the credibility ratings are ranked simply by the rating values. This difference is due to the difference in the significance of a negative entry on the vectors: A negative entry on Alice's trust vector for Bob is due to a problematic file served by Bob in the past. On the other hand, a negative entry on Alice's credibility vector for Bob does not necessarily imply a wrongdoing on Bob's part but may simply be due to Alice and Bob's having different experiences with a third peer Charlie who may be demonstrating inconsistent behavior, possibly with the specific aim of creating discredibility among good peers. It is due to this difference in reliability that the responses to a trust query are ranked by the credibility ratings alone, instead of using a min-discredibility-max-credibility ranking.

Like the safety check discussed at the end of Section 2.3 against the low-trust responses that may enter into the top θ_T in the local trust evaluation, we decided that the top θ_C responses in a trust query should be evaluated only if they have a credibility rating higher than discredibility. Accordingly, the factor for weighting the trust query responses in Equation (1) is taken as the net credibility ratings, $c_i - d_i$, rather than the credibility ratings alone.

3.3 File Download and Update of the Ratings

Once the file version to be downloaded is decided, the peer to download it from is selected randomly among those who offered that version without regard to the trust ratings. This way of selection has the desirable feature of enabling new peers to build a reputation as well as not overloading the trusted peers.

We have already explained in Section 2 why the update of trust ratings after a download is limited to the peers from whom the file was downloaded and, in case the file was malicious, to those who were authenticated references for the file. The main reason was that we could not decide conclusively about the other peers involved. It is due to the same reason that the credibility ratings are updated only for those peers who are authenticated references for someone whose trust rating is updated as a result of the download.

Note that, in a trust query, the top θ_C responses are authenticated regardless of their net credibility rating's being positive or negative, despite the fact that the responses with a negative rating would not be used in the calculations. This is necessary to give the peers with a zero or negative net credibility rating a chance to upgrade their ratings. Otherwise, if such an opportunity were not present, it would not be possible for the new peers to build a credibility, and the credibility system would be totally useless. Similarly, it would not be possible for the good peers who have somehow got a negative entry on their credibility history to turn their ratings to positive again.

4 Security Extensions

In this section, we discuss the extensions done on the basic protocol to provide secure and reliable trust information in presence of active attackers.

4.1 Key Management

Our system makes use of digital signatures for authentication of critical messages. The core trust issue in public key systems is to ascertain that a public key received on-line belongs indeed to the claimed party. The classical solution to this problem is by trusted certification authorities, which may not be an option in the P2P systems that are totally decentralized. On the other hand, most P2P systems are pseudonym-based systems, where the question is to bind the public keys to pseudonyms, not to real-life identities. A well-known natural solution here is to make the public key of a peer also its pseudonym. That is, in an RSA-based system for example, the public exponent-modulus pair (e, n) can be taken as the pseudonym of the entity using it.[4] In such a system, there will be no question of the public key's authenticity when the trust information from a certain pseudonym is to be verified.

4.2 Denial of Service Protection

The requirement of responding to every relevant query with a digital signature is likely to be an excessive burden on the peers. Moreover, it can easily be exploited for denial of service attacks by continually issuing many high-match queries. To protect against this threat, a puzzle scheme is used adding an extra round to the protocol: In the initial response, the file hash is sent without any signature. Instead, the responding peer includes a puzzle to be solved by the querier, such as finding a string whose MD5 hash matches a certain value [2], which should be answered correctly before a signature is issued. Then the querying peer decides on which file versions he is genuinely interested in and solves the puzzles of a limited number of the respondents for each version.

[4] If the pseudonyms are desired to be of uniform length such as an ID number, a one-way hash of the public key can be used, as was proposed by Damiani et al. [6].

4.3 Avoiding Fake File Downloads

Another avenue of attack for sending malicious files is to provide the hash of a benign file during the query-response process but, if selected as the download source, to send the malicious file during the download. Such attacks can be detected by checking the hash of a downloaded file before opening. However, the time and bandwidth of the downloader would be wasted, which is exactly the purpose of certain attacks such as the "decoy files" [3].

A more effective protection is to compare the hash of the blocks of the file while the download is in progress. Merkle hash trees [12] provide a solution of this sort. An alternative hash scheme is also possible that is more suitable for our protocol. In this alternative scheme, the hash of a file is computed in two stages: First, the file is divided into segments of a certain size and the hash of each segment is computed separately; then the hash of the file is computed as the hash of these segment hashes. The only computational overhead of this method is the extra hash computation over the segment hashes, which would be insignificant given that the segments sizes are reasonably large. We believe that a segment size in the 100KB–1MB range is a reasonable choice for most P2P networks.

Our trust evaluation protocol can be made to work with this new hashing scheme by a simple modification: Once the file version for download is selected, the querier contacts one of the peers who provided the selected hash and requests the detailed hash of the file. Upon receiving the response and verifying its correctness, the peer proceeds to download the file, possibly from multiple sources. During the download, the hash is checked after every downloaded segment and the connection is cancelled if a mismatch occurs.

Note that if an attacker sends the fake segment later in the download to delay detection, the benign segments downloaded until that point can be used without any problem, saving the time and bandwidth spent.

4.4 The Problem of Free Riders

A problem with a quite different theme but which may nevertheless benefit from our architecture is the problem of "free riders"; that is, the peers who use the P2P system only to download content but do not serve to other peers. Many users of Kazaa-like file sharing systems use the system as free riders. To tackle this problem and to discourage free riding, some systems determine the priority of the service reception of a peer according to the amount of service the peer has provided in the past. However, this service information is typically provided by the software of the client peer, which is easily hacked to always send the highest possible value. Alternatively, centralized solutions have been proposed where a server keeps track of the amount of service provided and received by each peer (e.g., [4, 9]); but this may not be a possible option for the P2P systems that are totally decentralized.

Our trust record system provides a natural distributed infrastructure that can also be used to assess the service level of a peer: At the time of a download, the priority of the download is determined according to the number of 1s in the trust vector the server peer maintains for the client peer. When the local information is insufficient, a trust query can be issued and a "service score" can be calculated from some top few responses. Here, unlike in the trust score calculation, the ranking of the responses should not be based solely on the credibility of the sources—the most credible respondents may possibly have not received any service from the client peer. Instead, a combination of the credibility ratings and the provided service scores should be used.

5 Simulation Experiments

We tested the performance of our protocol with simulations on various attack scenarios with the following types of attackers:

- *naive,* who respond to every query with a malicious version of the requested file.
- *hypocritical,* who act like a reliable peer most of the time but occasionally tries to send a malicious file.
- *collaborative,* who collaborate with each other in trust queries, expressing a positive opinion for malicious peers and a negative opinion for others.
- *pseudospoofing,* who change their pseudonym periodically to escape recognition (i.e., the "Sybil attack" [7])—these attackers are the hardest to detect and their prevention is possible only after honest peers build a sufficient level of trust among themselves.
- *pseudospoofing with collaborators,* where the pseudospoofing peers are supported by a group of "collaborators" who normally act as trustworthy peers and build trust in their communities, but give their strongest support to their malicious peers when they receive a relevant trust query.[5]

The simulated P2P networks operate with a Gnutella-like decentralized routing structure. Every peer is linked to a certain number of neighbors, and a query message is propagated over these links for a certain number of hops specified by the TTL. The simulations are run with the following parameters:

[5] This attack scenario is more meaningful than collaborators alone, since for malicious support in trust queries to be effective, the peer to receive the support must have a clean history in its neighborhood, hence must be changing its identity periodically. Otherwise, that peer would have been identified as malicious and the support to be given in trust queries would be irrelevant, as observed in Figure 5.

number of peers:	1000
number of distinct files:	1000
number of files each peer initially holds:	10
number of links per peer:	3
TTL:	3
ratio of malicious peers:	1–10%
length of trust vector:	8 bits

Here, the number of peers and files in the network are determined according to the capacity of our system. The number of connections per peer and the TTL are chosen to make the area covered by a peer's reachable neighborhood a reasonable fraction of the whole network—about 2% in this case. As for the malicious ratio in the network, 10% represents a high concentration of malicious peers, whereas 1% is the scenario that is probably closer to a real-life situation.

In a simulation run, regular users make file requests periodically, according to a uniform distribution. If the requested file is available locally, no further action is taken. Otherwise, a resource query message is issued, and the protocol proceeds as described in Section 2. Malicious peers may also issue file queries, basically for obtaining genuine files to be used for confidence building. Malicious peers are limited to their databases to send genuine file responses, but they are free to respond to any query maliciously.

It has been observed that the user behavior in P2P file sharing systems shows a skewed, Zipf-like distribution where users can be grouped into several categories according to their interests, and within each category there are a few highly popular files along with a large number of less popular ones [10]. Our simulations can be expected to give better results when run with such a skewed distribution since positive correlation among users' behavior would result in a more rapid trust establishment among the users in the same category. We preferred to stick to the uniform distribution which favors our protocol the least, since the file requests in a uniform distribution can come from anywhere in the domain and in our system it is only the attackers who are able to respond to all queries unrestrictedly.

5.1 Simulation Results

Results of our simulations are shown in Figures 4–8, where the metric used to evaluate the performance is,

Φ_1: Ratio of malicious downloads among all downloads

In the figures, every point shows the value of the statistics measured since the last plotted point (i.e., not cumulative), and the progress of the system is shown in terms of the total number of file downloads.

Throughout the simulations, we take $\theta_T = \theta_C$, denoted by θ. The *inter-query time*, or *iqt*, is the average time between two consecutive file queries of a peer and is used as the basic unit of the simulation time.

The main characteristics demonstrated by the experiments can be summarized as follows:

- The protocol is quite effective in preventing the malicious downloads and can reduce an attacks' effectiveness to zero within a short time depending on the sophistication of the attackers.
- A large degree of protection can be obtained by just evaluating one most trusted response, i.e., $\theta = 1$. Setting $\theta = 2$ may help against sophisticated attackers. The gain from $\theta > 2$ appears to be negligible.
- The protocol is similarly effective for both 1% and 10% malicious peer density.

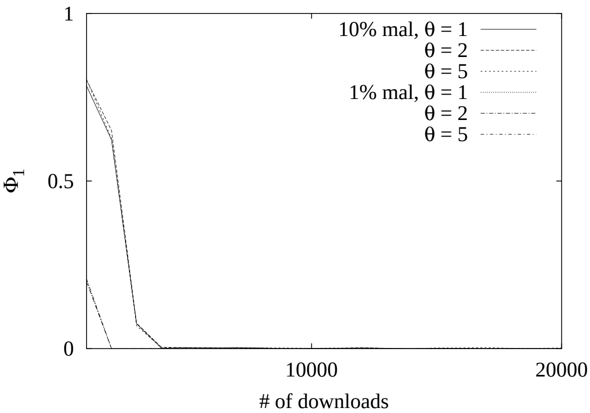


Figure 4: Simulation results for the naive attacker type. The attackers are identified and inhibited rapidly. The performance does not depend on θ.

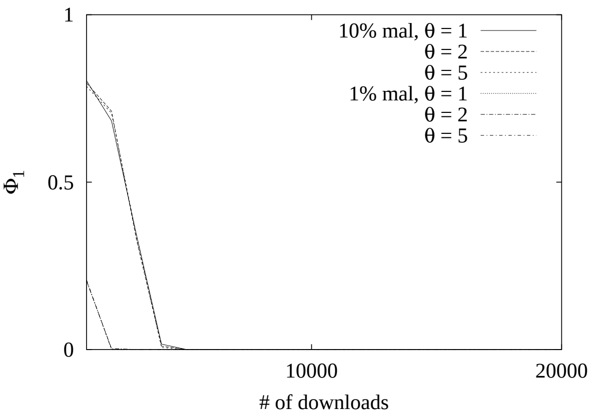
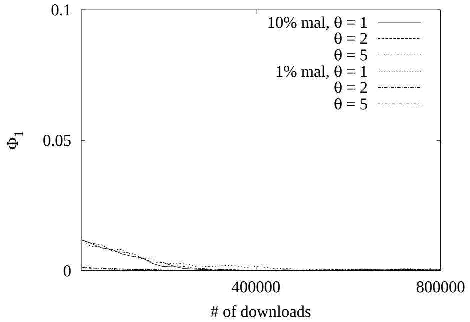
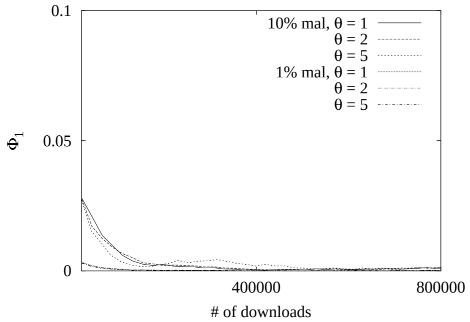


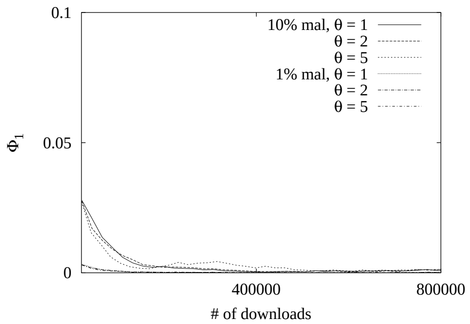
Figure 5: Simulation results for the collaborative attacker type. Their performance is only marginally better than the naive attackers for reasons discussed earlier in this section. (That is, for the support to be given in trust queries to be effective, the peer to receive the support must have a clean history.) A more effective attack scenario can be seen in Figure 8 where collaboration is carried out in coordination with pseudospoofing.



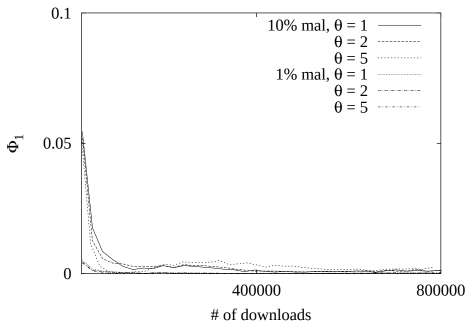
(a) $h = 0.10$



(b) $h = 0.20$

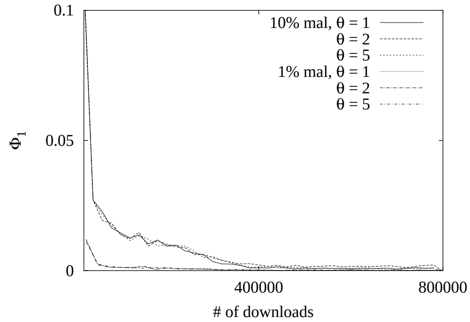


(c) $h = 0.25$

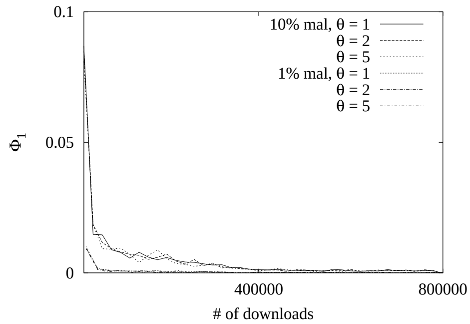


(d) $h = 0.50$

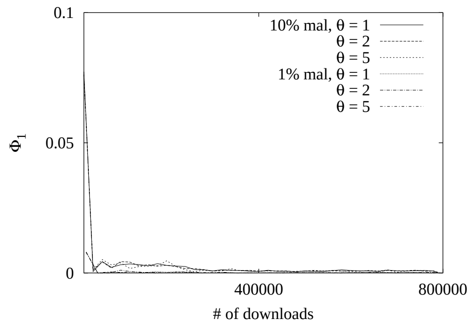
Figure 6: Simulation results for the hypocritical attackers who try to send a malicious file after a certain number of honest uploads. The parameter h specifies the dishonesty rate of the attackers. (E.g., for $h = 0.10$, an attacker tries to send a malicious file after every nine honest uploads.) The results show that detection of the hypocritical attackers takes longer than other attacker types, but their level of effectiveness is also significantly lower. Among the h values tested, the best attack performance is obtained for $h = 0.5$



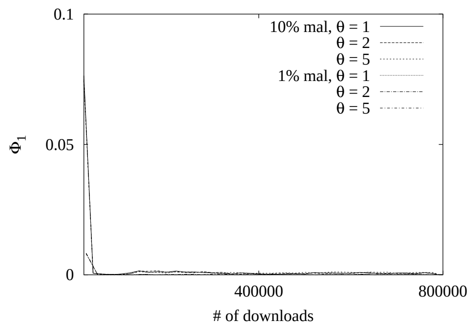
(a) $p = 50$



(b) $p = 100$

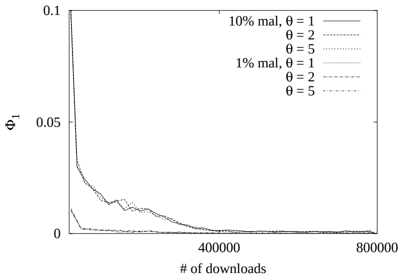


(c) $p = 250$

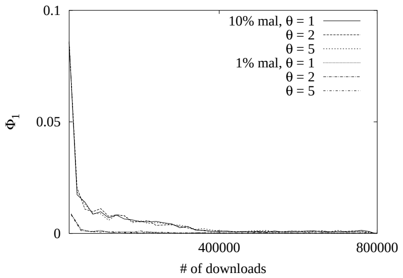


(d) $p = 500$

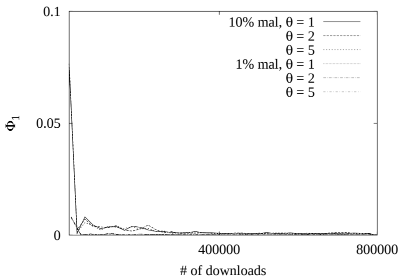
Figure 7: Simulation results for the pseudospoofing attackers who adopt a new identity periodically. The parameter p specifies the period of the identity change. (E.g., for $p = 100$, an attacker adopts a new identity at every 100 iqt.) This attacker type is the hardest to detect and prevent. Nevertheless, their ability to spread malicious content converges to zero as good peers get to know each other and build trust among themselves.



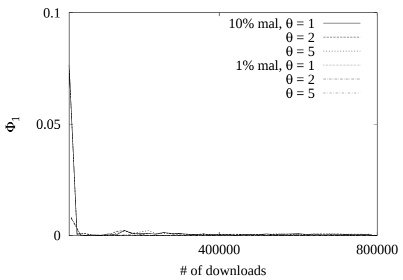
(a) $p = 50$



(b) $p = 100$



(c) $p = 250$



(d) $p = 500$

Figure 8: Simulation results for the pseudospoofing attackers with collaborators. Here, 1–10% pseudospoofing malicious peers try to spread malicious content where another 10% of the peers are *collaborators* who normally act as trustworthy peers and build confidence in their communities, but give their full support to the pseudospoofing malicious peers in trust queries—the 10% collaborator ratio, which is probably too high to be realistic, is chosen to guarantee the presence of at least one collaborator in the neighborhood of every pseudospoofing attacker in order to make the attacks more effective. This attack type is somewhat more effective than pseudospoofing alone, but their effectiveness also converge to zero as good peers build trust with each other.

6 Related Earlier Work

A number of protocols have been proposed recently for reputation-based trust management in P2P systems. In this section, we discuss them briefly in comparison to our protocol.

One of the earliest works in this area is a protocol by Aberer and Despotovic [1], where dishonest peers are identified by a complaint-based system. The protocol maintains the negative feedbacks only and, hence, a trustworthy peer cannot be distinguished from a newcomer. At the trust evaluation process, every peer is classified as either trustworthy or untrustworthy. Operation of the proposed protocol requires the maintenance of a *P-Grid* structure on top of the existing P2P network.

Another protocol in the area is the *EigenTrust* scheme proposed by Kamvar et al. [10], which evaluates the trust information provided by peers according to their trustworthiness (i.e., using the trust ratings for credibility). At the core of the protocol is a special normalization process where the trust ratings held by a peer are normalized to have their sum equal to 1. Although it has some interesting mathematical properties, a drawback of this normalization is that important trust information may get lost during the process. (E.g., when there are n identical, non-zero trust ratings in a peer's database, their normalized values will be $1/n$, regardless of the original values which could be very high or very low.)

A recent protocol that specifically addresses the issue of assessing the trustability of alternative file versions obtained in a P2P query is the *P2PRep* protocol of Damiani et al. [5] (formerly, the *XRep* protocol [6]). This protocol is similar to our proposal in scope but differs in some major aspects, including the way the trust scores are calculated, the way the ratings are maintained and updated, and the way the protocol messages are authenticated. A detailed description of how P2PRep can be integrated into the Gnutella protocol is given in [5]. An interesting idea in [6] is to maintain reputation information for resources as well as for peers, which we discuss how can be incorporated into our system in Section 7.

A study with a rather different but nevertheless relevant scope is the *PeerTrust* system of Xiong and Liu [15, 16] on trust evaluation in P2P e-commerce communities. The paper emphasizes two points for P2P reputation systems:

- In a rating scheme, the complaints, or any other opinions for that matter, should be evaluated according to the credibility of their providers.
- A peer's behavior in different contexts should be evaluated according to the context. (E.g., feedbacks from small and large transactions should be weighted differently, according to the size of the transaction.)

The paper does not deal with the specifics of the protocol to be used in trust evaluation. Nevertheless, they do simulations with an experimental rating scheme modified from the protocol of Aberer and Despotovic [1].

Other recent works with a relevant scope include [17] where a reputation management mechanism is proposed for a community of software agents, [18] where a reputation mechanism is discussed for P2P applications where a binary rating would not be sufficient (e.g., for P2P commerce), and [11] where a reputation management scheme is discussed for partially-decentralized P2P systems—i.e., systems with a set of "supernodes" or "ultrapeers".

7 Conclusions

In this paper, we presented a reputation system for establishing trust in P2P networks that helps preventing the spread of malicious content. The system works by the users' evaluating the outcome of their past transactions and sharing this information with their peers over the P2P infrastructure when requested. Simulation results show that the proposed system can be highly effective in establishing trust among good peers and inhibiting the malicious ones. The simulation results are further significant in that they show that evaluating the one or two most trusted responses in a query is sufficient to have most of the potential benefit, and, hence, the proposed system can be used with a relatively small overhead.

Several improvements are possible on the basic protocol to make it more efficient. For example, a timer mechanism can be used to detect and remove the trust vectors belonging to peers that are no longer active. Trust queries can be made more efficient by combining all IDs to be queried into a single query message, reducing the number of query and response messages to be handled. Although the protocol maintains ℓ-bit trust and credibility vectors, it might be more suitable to use only part of that information for different purposes. For example, when evaluating trust responses that come from peers, it may be more desirable to take only the most recent few transactions into account.

A potential improvement on the basic protocol may be realized by preserving the hashes of the malicious files downloaded. These hashes can later be used to send a warning to the querying peer when a relevant query is received. This idea was originally proposed by Damiani et al. [6] in a similar context. Our protocol can be enhanced to include this feature with the following modifications: The warning messages received in a query are grouped along with the normal responses according to their file hash value. If selected into the top θ_T for trust evaluation, a warning message's trust and distrust ratings are reversed in the trust score calculation, contributing a significant distrust factor to the average.

The limitations of our protocol must also be noted. Being a reputation-based protocol, our system in the end relies on the judgment of its users and, hence, is effective only against attacks that are perceivable by the user. Nevertheless, many attacks in P2P systems fall into this category, not the least the common decoy files attacks [3].

Another point to note is that our protocol does not distinguish between malicious peers and careless peers who spread malicious content, which we believe is the right way to deal with careless peers from a practical point of view. A careless peer always has the ability to improve its reputation by serving a sufficient number of good files once it corrects its attitude.

Our protocol is designed to be compatible with most first generation P2P systems. However, certain optimizations would be needed to obtain the best performance when integrating it with a specific system. For example, in a Gnutella-like network where a peer's connections change constantly, building a reliable reputation base can take too long and a malicious peer can escape recognition for a long time due to the constantly changing neighborhoods. In such a system, a connection scheme where some of the neighbors of a peer change continually for content distribution and others, which are possibly determined by a longest prefix match on the ID, remain relatively stable for trust management, could be more effective for faster trust establishment. More detailed simulations with this kind of specifics can be carried out to get a more detailed view of the protocol's performance in particular environments.

References

[1] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *Ninth International Conference on Information and Knowledge Management (CIKM'01)*, 2001.

[2] T. Aura, P. Nikander, and J. Leiwo, "DOS-resistant authentication with client puzzles," in *8th International Workshop on Security Protocols*, Springer-Verlag, 2000.

[3] BBC-Online, *Record Industry Spoofs Net Pirates.* (http://news.bbc.co.uk/2/hi/entertainment/-2093931.stm)

[4] B. Cohen, "Incentives build robustness in BitTorrent," in *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[5] E. Damiani, S. D. C. D. Vimercati, S. Paraboschi, and P. Samarati, "Managing and sharing servants' reputations in P2P systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 4, pp. 840-854, July-Aug. 2003.

[6] E. Damiani, S. D. C. D. Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A Reputation-based approach for choosing reliable resources in peer-to-peer networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 207-216, 2002.

[7] J. R. Douceur, "The Sybil attack," in *1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, pp. 251-260, Springer-Verlag, 2002.

[8] EBay. (http://www.ebay.com)

[9] M. Gupta, P. Judge, and M. Ammar, "A reputation system for peer-to-peer networks," in *Proceedings of the 13th International Workshop on Network*

and Operating Systems Support for Digital Audio and Video (NOSSDAV'03), pp. 144-152, 2003.

[10] S. D. Kamvar, M. T. Schlosser, and H. G.-Molina, "The eigentrust algorithm for reputation management in P2P networks," in *Proceedings of the 12th International Conference on World Wide Web (WWW2003)*, pp. 640-651, 2003.

[11] L. Mekouar, Y. Iraqi, and R. Boutaba, "A reputation management and selection advisor schemes for peer-to-peer systems," in *15th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM'04)*, Springer-Verlag, pp. 208-219, 2004.

[12] R. Merkle, "Protocols for public key cryptosystems," in *Proceedings of the 1980 IEEE Symposium on Security and Privacy*, 1980.

[13] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara, "Reputation systems," *Communications of the ACM*, vol. 43, no. 12, pp. 45-48, 2000.

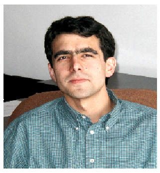
[14] Symantec. (http://securityresponse.symantec.com/-avcenter/venc/data/vbs.gnutella.html)

[15] L. Xiong and L. Liu, "A reputation-based trust model for peer-to-peer ecommerce communities," in *IEEE Conference on E-Commerce (CEC'03)*, 2003.

[16] L. Xiong and L. Liu, "PeerTrust: Supporting reputation-based trust in peer-to-peer communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843-857, July 2004.

[17] B. Yu and M. P. Singh, "A social mechanism of reputation management in electronic communities," in *Proceedings of the 4th International Workshop on Cooperative Information Agents IV, The Future of Information Agents in Cyberspace*, pp. 154-165, 2000.

[18] B. Yu, M. P. Singh, and K. Sycara, "Developing trust in large-scale peer-to-peer systems," in *First IEEE Symposium on Multi-Agent Security and Survivability*, pp. 1-10, 2004.



Ali Aydin Selcuk received his BS degree from Middle East Technical University in 1993, MS from Bilkent University in 1995, and PhD from University of Maryland, Baltimore County, in 2001. Currently he is an assistant professor at Bilkent Univeristy. His previous work experience includes RSA Data Security, Novell, and the Network Systems Lab of Purdue University. His research interests are in cryptography and network security, with an emphasis on secure communication protocols, secure multiparty computation, and cryptanalysis of block ciphers.

Ersin Uzun is a PhD student in the Networked Systems Department at University of California, Irvine. He had his BS degree in computer engineering at Bilkent University and his MS degree in networked systems at University of California, Irvine. His research interests include networking and systems security, usable security and applied cryptography.

Mark Pariente is currently working as a software engineer in VMware Inc. He got his BS degree in Computer Engineering from Bilkent University in Ankara, Turkey and his MS degree from the Information Networking Institute at Carnegie Mellon University. His research interests include operating systems, networking and network security.