A Reputation-Based Trust Management System for P2P Networks

Ali Aydın Selçuk Ersin Uzun Mark Reşat Pariente

Department of Computer Engineering
Bilkent University
Ankara, 06800, Turkey
E-mail: selcuk@cs.bilkent.edu.tr, {euzun,resat}@ug.bilkent.edu.tr

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 describe a reputation-based trust management protocol for P2P networks where users rate the reliability of parties they deal with, and share this information with their peers. The protocol helps establishing trust among good peers as well as identifying the malicious ones.

Results of various simulation experiments show that the proposed system can be highly effective in preventing the spread of malicious content in P2P networks.

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. In this paper, we also focus on P2P file sharing systems and use the term "P2P" mostly to refer to this particular application of the more general concept.

By the nature of its architecture, a P2P file sharing system provides an open, unrestricted environment for content sharing. However, this openness also makes it an ideal environment for attackers to spread malicious content such as the VBS.Gnutella worm [11].

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 the peers in the community [9]. One well-known such system is the rating scheme used by the eBay on-line auction site [5].

In this paper, we propose a reputation-based, distributed trust architecture for P2P networks to identify malicious peers and to prevent the spreading 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 protocol we propose is described in Sections 2– 4. Results of the simulation experiments testing the protocol's effectiveness are presented in Section 5. Earlier protocols with a similar scope and their differences from our proposal are discussed in Section 6. Section 7 concludes the paper with a discussion of the future work necessary for a practical deployment of our protocol.

2 The Basic Protocol

A query in a P2P file sharing system can return many different versions of the queried resource, among which some may be malicious. The aim of our protocol is to distinguish the malicious responses from benign ones by using the reputation of the peers providing them. Since in P2P networks a central server is typically not available, our protocol relies on the P2P infrastructure to obtain the necessary reputation information when it is not locally available at the querying peer.

In this section, we give a high-level description of the basic protocol. The rationale for the design is discussed in Section 3. The security extensions on the basic protocol are described in Section 4. Some relatively insignificant technical details which could not be included in this paper due to the space limitations can be found in the full technical report [10].

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.

0-7803-8430-X/04/$20.00 ©2004 IEEE

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.

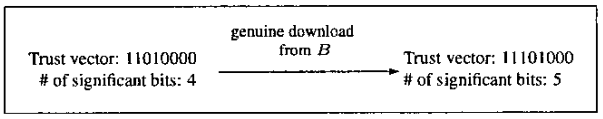


Figure 1: Peer A's update of its trust vector on B after an honest transaction. In this example $\ell = 8$.

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

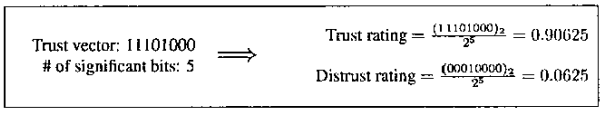


Figure 2: Computation of the trust and distrust ratings from the trust vector.

2.2 The Trust Evaluation Function

The responses to a resource query are grouped according to the file hashes provided. Let G denote a group of peers that provide a certain version of the file, say f_G. The *trust coefficient* of f_G is calculated as the average of the trust ratings of the top θ_T most trusted peers in G, where θ_T is the threshold specifying the number of peers to be considered in a version's trust calculation.

If local trust information is available for fewer than θ_T peers in G, denoted by $known(G)$, then a *trust query* is issued for randomly selected $\theta_T - |known(G)|$ unknown peers in G.

2.3 The Trust Query Process

The trust query process is similar to the file query process except that the subject of the query is a peer about whom trust information is inquired. The responses include the trust and distrust ratings the responders have on the queried subject.

[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.

The responses are sorted and weighted by the *credibility rating* of the responders. Credibility ratings are derived from the *credibility vectors* maintained by the local peer, which are similar to the trust vectors: A 0 in a credibility vector shows a failed judgment from that peer in the past, a 1 shows a successful one.

The threshold θ_C specifies the number of responses to be evaluated for each trust query. The *queried trust rating* is the average of the evaluated trust ratings, weighted by the credibility of their senders. 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, and A's credibility rating for R_i is c_i and R_i's trust rating for B is t_i, then A's queried trust score on B is

$$\frac{\sum_{i=1}^{k} c_i t_i}{k}.$$

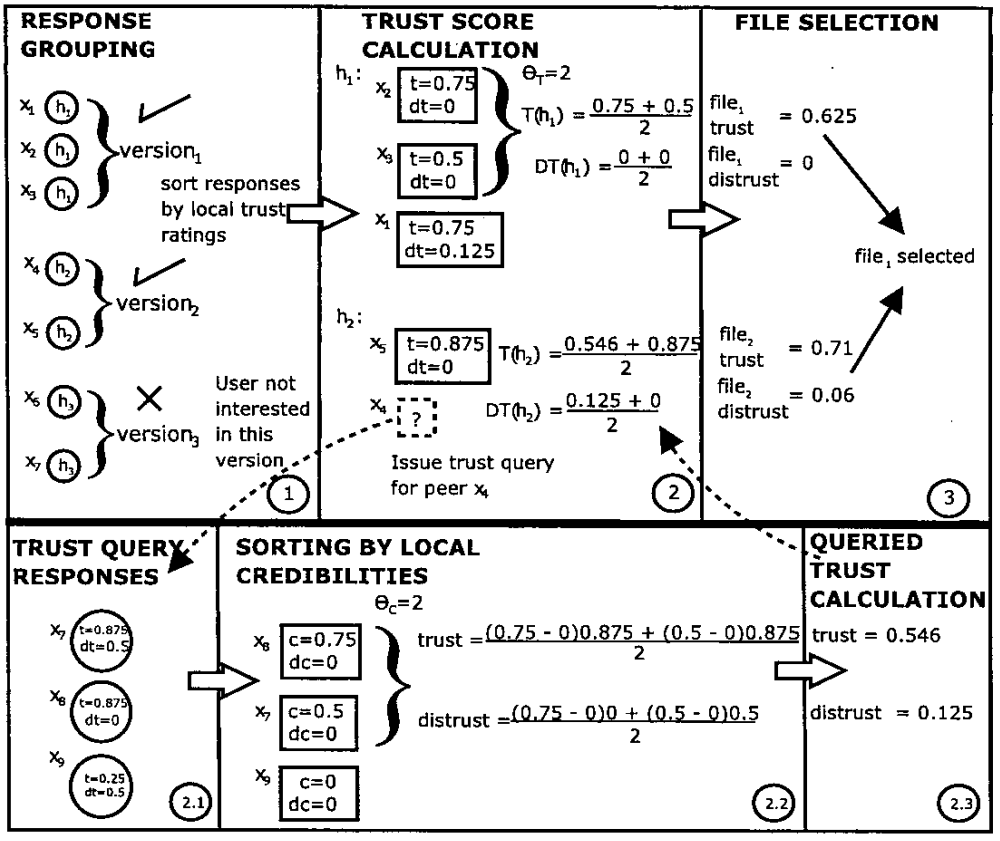
The queried distrust rating is calculated in the same fashion, using the respondents' distrust ratings of B.

2.4 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 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: The rating of a peer who expressed an opinion on a queried peer is updated only if the queried peer's trust rating is updated as a result of the download. A credit 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.



THE LOCAL DATABASE

TRUST RECORDS

Peer	Vector	# of bits significant
x_1	11000000	3
x_2	11000000	2
x_3	10000000	1
x_5	11100000	3

CREDIBILITY RECORDS

Peer	Vector	# of bits significant
x_7	10000000	1
x_8	11000000	2

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.

The operation of our basic protocol is illustrated in Figure 3.

3 Design Rationale

The idea of using the feedback from other peers to assess the trustworthiness of a resource/peer is a fundamental characteristic of reputation systems [9]. 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 credibility. 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 erased 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. Moreover, 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 reasonable amount of 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, proportional to ℓ.

Once the file version to be downloaded is decided, the peer to download it from is selected randomly among those who offered that version, not considering 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.

Another important design decision 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 preclude such attacks.

4 Security Extensions

In the section, we discuss extensions on the basic protocol to provide secure and reliable trust information in the 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 indeed belongs to the claimed party. The classical solution to this problem is by trusted certification authorities, but this may not be an option in a P2P system due to its decentralized nature. In pseudonym-based systems, however, including most P2P systems, the question is to bind the public keys to pseudonyms, not to real-life identities. Hence, a natural solution is to make the pseudonym and the public key of an entity the same thing. 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.[2] 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 Authentication

In order to be a reliable source of information, the responses to be used in trust evaluation must be authenticated. In our protocol, this is obtained by signing the hash of the responses provided. For protection against replay and cut-and-paste attacks, the signed hash covers, among other fields, the ID of the querying and responding peers, a query ID number, and the file hash being offered.

4.3 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 attackers 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 output 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.

4.4 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 if the hash of a downloaded file is checked 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 [8] 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

[2] 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.

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 canceled 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.

4.5 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. Recently, reputation-based solutions are being proposed to overcome this problem (e.g., [6]).

Our trust record system provides a natural infrastructure that can be used for evaluating the service level of the peers: 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—since the most credible respondents may 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. Although it is not possible to exhaust all potential attack types, testing the protocol with a variety of attacks gives an idea on the effectiveness of the protocol. The types of attackers considered in the simulations are,

- *naive,* who responds to every query with a malicious version of the requested file
- *hypocritical,* who acts 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—these attackers are the hardest to detect and their prevention is possible only after honest peers build a sufficient level of trust among themselves.

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 issued by a peer is propagated over these links for a certain number of hops specified by the TTL. The simulations are run with the following common parameters:

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%

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. 10% malicious ratio 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. 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.

It has been observed that the user behavior in P2P file sharing systems show a 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 [7]. Our simulations can be expected to give better results when run with a Zipf 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 Figure 4, where the performance metric used is

Φ_1: Ratio of malicious to all downloads.

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 it 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 $\theta = 2$ helps 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.

In Figure 4, we have $\theta = 2$, $\ell = 32$. More extensive results with different performance metrics and different values of θ and ℓ can be found in the full technical report [10].

6 Comparison to 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 the protocol by Aberer and Despotovic [1] which aims to identify dishonest peers by a complaint-based system. A shortcoming of this protocol is that it maintains only the negative feedbacks, providing no means for a trustworthy peer to be distinguished from a newcomer. The trust evaluation is also rather simplistic, classifying every peer either as trustworthy or untrustworthy. Moreover, maintenance of a "P-Grid" architecture is required on top of the existing P2P structure.

Another protocol is the EigenTrust scheme proposed by Kamvar et al. [7], which evaluates the trust information provided by peers according to their trustworthiness (i.e., using the trust ratings for credibility). 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 properties, this normalization may result in the loss of important trust information. E.g., if there are n identical trust ratings in the database, their normalized value will be $1/n$, whether the originals were the highest or the lowest possible value.

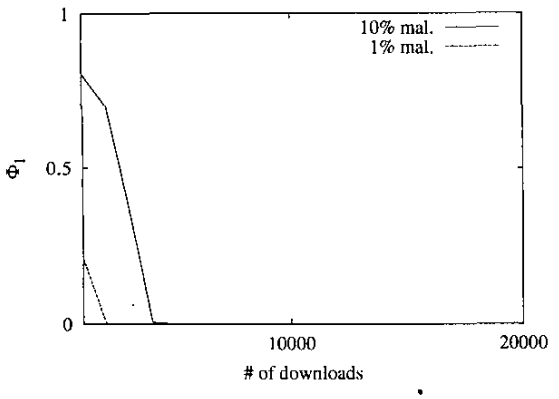
Another proposal with a similar scope is the protocol of Damiani et al. [4], which assesses the trustability of a file to be downloaded by "voting" of the peers. The protocol makes no distinction between the votes from trustworthy and non-trustworthy peers, and there is no authentication of the vote messages. Also, no quantitative trust metric is specified for choosing among alternative versions. An important idea of [4] is to maintain reputations for resources as well as for peers.

A study with a different but relevant scope is a recent paper of Xiong and Liu [12] on trust evaluation in P2P e-commerce communities. Although they do not deal with the details of trust evaluation functions, they run an experimental system which utilizes a modification of the P-Grid scheme of [1].

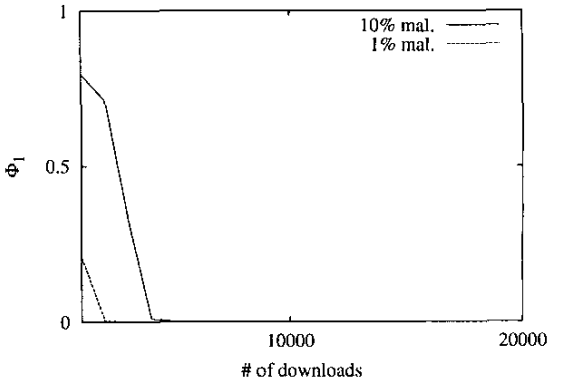
7 Final Considerations

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.

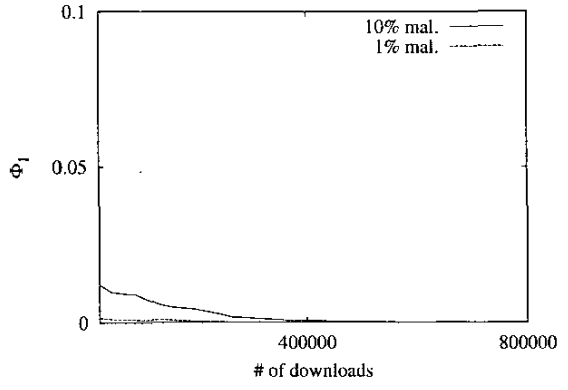
A potential improvement on the basic protocol may be realized by preserving the hashes of the malicious files downloaded. These hashes can later be used to send a warning to the querying peer when a relevant query is received. This idea was originally proposed in [4] in a similar context. Our protocol 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



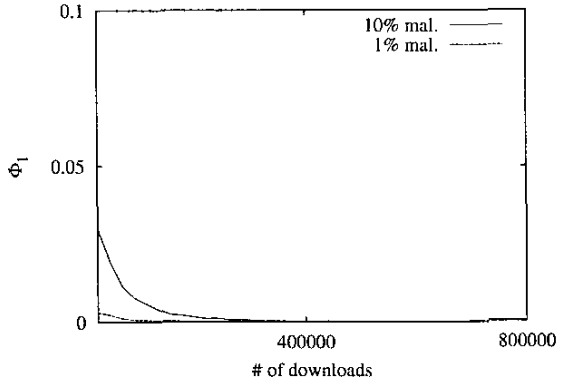
(a) Naive



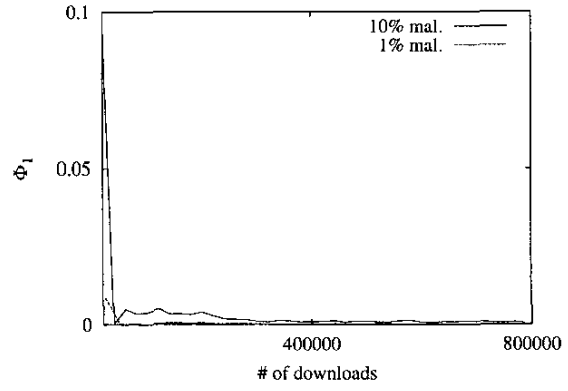
(b) Collaborative



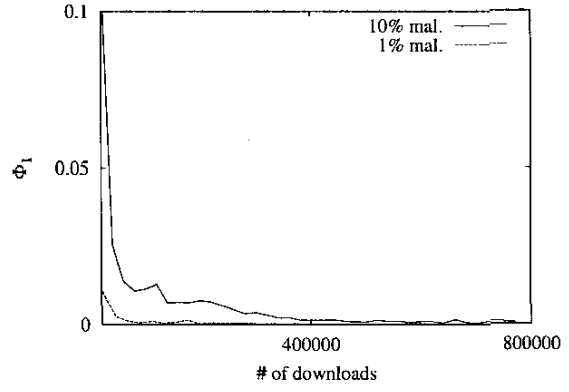
(c) Hypocritical, 10% cheating



(d) Hypocritical, 25% cheating



(e) Pseudospoofing, at 250 iqt



(f) Pseudospoofing, at 100 iqt

Figure 4: A graphical summary of the simulation results. Naive attackers can be detected and contained quite rapidly. Hypocritical attackers, operating at a much lower effectiveness level, can evade detection longer; but their activity is also contained once a sufficient level of trust is established among the good peers. Pseudospoofing attackers are also able to continue spreading malicious content for a while but become ineffective as the good peers establish trust among themselves. Collaboration does not seem to be a significant source of benefit to naive attackers.

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. Therefore, it can be effective only against attacks that are discernible by the user. Nevertheless, many attacks in P2P systems fall into this category, such as the common decoy files attacks [3].

Another point to note is that our protocol does not distinguish between malicious peers and the peers that spread malicious content due to their carelessness, which we believe is the right way to deal with careless peers from a practical point of view. On the other hand, if careless users change their attitude, they always have the ability to improve their reputation by serving a sufficient number of good files.

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 are changed constantly to provide rapid distribution of the content across the network, 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 neighborhood. 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 that consider this kind of specifics of the network where the protocol is to be deployed, and with a more sophisticated modeling of the attackers according to the network's possible vulnerabilities, would be needed to get a more realistic evaluation of the proposed architecture for deployment in an actual system.

Acknowledgments

We would like to thank Ezhan Karaşan for kindly letting us use the Information Networks Lab's high performance workstation for our simulations.

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)*. 2001.

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

[3] BBC-Online. http://news.bbc.co.uk/2/hi/entertainment/-2093931.stm.

[4] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. Reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proc. of the 9th ACM Conference on Computer and Communications Security*. 2002.

[5] EBay. http://www.ebay.com.

[6] M. Gupta, P. Judge, and M. Ammar. A reputation system for peer-to-peer networks. In *Proc. of NOSSDAV'03*. 2003.

[7] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *Proc. of the Twelfth International World Wide Web Conference (WWW2003)*. 2003.

[8] R. Merkle. Protocols for public key cryptosystems. In *Proceedings of the 1980 IEEE Symposium on Security and Privacy*. 1980.

[9] P. Resnickand, R. Zeckhauserand, E. Friedman, and K. Kuwabara. Reputation systems. *Communications of the ACM*, 43(12), 2000.

[10] A. A. Selçuk, E. Uzun, and M. R. Pariente. Reputation-based trust management for P2P networks. Technical Report BU-CE-0402, Department of Computer Engineering, Bilkent University, 2004.

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

[12] 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.