Fuzzy Techniques for Trust and Reputation Management in Anonymous Peer-to-Peer Systems*

R. Aringhieri[1], **E. Damiani**[1],
S. De Capitani Di Vimercati[1], **S. Paraboschi**[2], **P. Samarati**[1]
(1) Dip. di Tecnologie dell'Informazione (2) Dip. di Ing. Gestionale e dell'Informazione
Università di Milano Università di Bergamo
email: {aringhieri,damiani,decapita,samarati}@dti.unimi.it parabosc@unibg.it

Abstract

P2P applications are rapidly gaining acceptance among users of Internet-based services, especially because of their capability of exchanging resources while preserving the anonymity of both requestors and providers. However, concerns have been raised about the possibility that malicious users can exploit the network to spread tampered-with resources (e.g., malicious programs and viruses). A considerable amount of research has then focused on the development of trust and reputation models in P2P networks.

In this paper, we propose to use fuzzy techniques in the design of reputation systems based on collecting and aggregating peers' opinions. Fuzzy techniques are used in the evaluation and synthesis of all the opinions expressed by peers. The behavior of the proposed system is described by comparison with probabilistic approaches.

1 Introduction

Peer-to-Peer (P2P) networks have rapidly achieved an important role in the Internet experience of millions of users [Oram, 2001]. P2P applications allow users to connect directly to other users' machines to exchange files and other resources. Interaction among peers is usually carried out by preserving some degree of anonymity, for example, by using *opaque identifiers* or *nyms* rather than full names or IP network addresses when communicating with each other. For this reason, Internet users increasingly find P2P applications a convenient solution for the anonymous exchange of resources. The success of P2P applications has produced a significant impact in the research community, which has started to investigate the many open issues arising in this new environment, including peers' complete lack of accountability. In a traditional client/server system (like the HTTP-based Web) the provider of a service vouches for its integrity. In anonymous P2P systems, nyms are non-persistent and can be dropped and exchanged at will. In other words, most of the time the peers one encounters when accessing the P2P network are unknown and it is therefore

*A preliminary version of this paper appeared under the title "Fuzzy Logic Techniques for Reputation Management in Anonymous Peer-to-Peer Systems," in *Proc. of the International Conference in Fuzzy Logic and Technology*, Zittau, Germany, September 10-12, 2003 [Damiani et al., 2003a].

difficult to assess their trustworthiness. Questions such as: "how much can I rely on the information or service I am receiving?" are frequently asked by participants in P2P networks. For a user connecting to an anonymous P2P network in particular, information about other peers involves much more than their pseudonyms or even the list of the resources they can provide. Indeed, each peer would like to know *a priori* whether the resources advertised as available by other peers will be actually provided in the end and, above all, the extent to which they will suit the receiver's expectations. This quantitative estimation can be seen as an operational definition for *trust* and is subjective, dynamic, and often uncertain [Rasmusson and Jansson, 1996]. Some approaches [Yahalom et al., 1993] try to deal with multiple types of trust based on a set of different interaction *pragmatics*; others [Yu and Singh, 2000] dynamically assign to each peer a *trust rating*, that is, a reputation based on the peer's previous performance on the network and store it at a suitable place. Any peer wishing to interact with another peer can make an informed decision based on such a rating. Note that there is a clear distinction between *trust* and *reputation*. In general, the trust T in a peer can be computed based on its reputation R, but also on a number of other environmental factors, such as the time t elapsed since when reputation was last modified. If trust and reputation are modeled numerically (see Section 4), then $T = \phi(R, t)$, with the constraint $T = \phi(R, 0) = R$. For instance, a simple functional model for the relation between trust and reputation is $\phi(R, t) = Re^{-(t-t_0)}$, where t_0 is the system's start-up time [Carter et al., 2002]. However, while functional models linking trust and reputations are interesting in theory, in our opinion no closed formula can fully capture the changing views of a user community.

There are therefore two fundamental challenges in designing a reputation management system of anonymous P2P networks.

A first challenge is the design of a protocol able to provide secure placement of (and ready access to) reputation information. From the point of view of collecting and aggregating multiple opinions, P2P network as fundamentally different from centralized environments, inasmuch some basic assumptions, for example, opinion veridicity cannot be made without specific countermeasures against forgery.

Another fundamental challenge is defining a suitable means to represent reputations and synthesizing a set of opinions collected by the protocol in a unique aggregated value. As we shall discuss in the following, most approaches use numbers to express reputation values; but additional assumptions are needed to decide which numerical domain is more suitable for a given application field. In centralized reputation-based systems, for example, reputations can be either positive or negative. In principle, using negative values for reputation seems rather natural, since it mirrors the fact that individuals and businesses have certain roles within society. If society judges that they have met their roles, they are rewarded with a positive reputation. On the other hand, loss of esteem held in society leads to a negative reputation [Carter et al., 2002]. In the P2P anonymous setting, however, the situation is different: no peer can be forced to retain an identity linked to a negative reputation. Therefore, only positive values are meaningful. Zero reputation corresponds either to a newcomer or to a peer who has decided to use a brand new identity, perhaps to avoid being recognized as someone who previously misbehaved. Characterizing what is a good reputation is also non trivial. A possible approach consists in the identification of a threshold that would permit to discriminate the outcome of the poll as representing a *good* or *bad* reputation. However, a crisp value is clearly not adequate, as it would characterize in the same way a positive reputation produced by the collection of only positive opinions by many users and a reputation built with a

limited number of heterogeneous opinions that produce a value immediately above the threshold; the same reasoning can be applied to negative reputations.

In this paper we address the two challenges above-mentioned and present a P2P reputation system based on a two-step-technique: *i)* we poll the community of P2P users for collecting the different reputations on a candidate peer p, and *ii)* we merge the different opinions in a single value by means of a fuzzy aggregation [Klir and Folger, 1988]. Our proposal aims at computing reputations as fuzzy predicates, with a truth value belonging to the interval $[0, 1]$ expressing the reputation one (or the community) has on a given peer. Note that while in this paper we assume reputations to be associated with peers, the approach can also be applied to the exchange of opinions on resources [Damiani et al., 2002] and on many other aspects (e.g., quality of resources, opinions on specified parameters, and so on).

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 presents the protocol for collecting distributed opinions which includes a vote checking procedure aimed at providing a degree of assurance on the veridicity of the opinions received in response to a polling. The resulting system thus provides an environment reasonably close to classical decision making based on multiple opinions [Carter et al., 2002]. Section 4 describes our fuzzy-based approach for expressing and aggregating reputations. Section 5 presents experimental results illustrating the behavior of our solution with respect to protection against downloads from malicious peers. The simulation also compares our solution with analogous experiments we have run on a probabilistic approach [Kamvar et al., 2003], where reputation is defined as a probability (e.g., of being deceived) and can be computed by an *event-driven* method using a Bayesian interpretation. Finally, Section 6 presents our conclusions.

2 Related work

A considerable amount of research has focused on the development of trust and reputation models in open environments such as e-commerce and P2P networks [Aberer and Despotovic, 2001, Damiani et al., 2003b, Damiani et al., 2002, Dellarocas, 2000, Dingledine et al., 2001, Dingledine and Syverson, 2002, Friedman and Resnick, 2001, Kamvar et al., 2003, Kinateder and Pearson, 2003, Oram, 2001, Rahman and Hailes, 2000, Wang and Vassileva, 2003, Xiong and Liu, 2003, Yu and Singh, 2000, Zacharia et al., 1999]. Reputations are effectively used in electronic marketplaces as a measure of the reliability of participants [Dellarocas, 2000, Rahman and Hailes, 2000, Yu and Singh, 2000, Zacharia et al., 1999]. For instance, the eBay reputation system (`www.ebay.com`) allows users to express feedback about interactions with each other [eBay Feedback Forum, 2003]. The system is transaction-based, meaning that to express feedback for each other, two users must actually have completed an auction. After the auction ends, the buyer and seller can express a vote (`-1`, `0`, or `1`) on its counterparts. Users' net reputation scores are calculated as the count of distinct users who gave positive feedback minus the count of those who gave negative feedback. The seller's net reputation score is then automatically displayed on the auction page. A user can click on the net score to see a division of the score into positive, negative, and neutral scores over a series of time periods. In systems like eBay, reputations are associated with physical identities and are centrally managed at the eBay server. More in line with the peer-to-peer paradigm, several

proposals worked around the notion of *web of trust* where single peers can rate each other according to the quality of their past transactions with each other. KaZaA (`www.kazaa.com`) is an example of P2P file sharing application which has implemented a reputation management system based on two concepts: *participation levels* and *integrity rating*. Each user rates the integrity of the files it shares as excellent, average, poor, or delete file. The participation level is a value associated with each peer that is based on the quality and amount of files that it shares (see `www.kazaa.com/us/help/glossary.htm`). The participation level is calculated as $\frac{\text{uploads in MB}}{\text{downloads in MB}} * 100$ and ranges from 0 to 1000. Note that in the "uploads in MB field", non integrity rated files are counted as half the size of a rated file. Based on the ratio of Mbytes uploaded and downloaded and the integrity rating of the files, the peers are assigned to three categories: low, medium, and high. A new user starts at a medium participation level of 100. The security aspects in peers modifying their locally stored participation level values are not addressed. Some other approaches are based on the concept of *community-based feedbacks*: peers rate each other according to the quality of their past transactions. Once a peer has been rated, its rating needs to have a way of propagating around the network so that all peers have a wide view of the reputations of all other peers in the network. The proposed approaches use different techniques for combining and propagating the ratings [Aberer and Despotovic, 2001, Damiani et al., 2003b, Damiani et al., 2002, Gupta et al., 2003, Wang and Vassileva, 2003, Xiong and Liu, 2003]. Here we describe a few related examples. In [Aberer and Despotovic, 2001] a trust model is proposed, where, after each transaction, and only in case of malicious behaviors, peers may file a complaint. Before engaging an interaction with others, peers can inquire the network about existing complaints on their counterparts. One limitation of this model is that it is based on a binary trust scale (i.e., an entity is either trustworthy or not). Hence, once there is a complaint filed against a peer p, p is considered untrustworthy even though it has been trustworthy for all previous transactions. In [Wang and Vassileva, 2003] a Bayesian network-based trust model is proposed, where peers are evaluated with respect to different capabilities (e.g., capability in providing music files or movies). Basically, peers develop a naive Bayesian network for each peer with which they have interacted and modify their corresponding Bayesian networks after each interaction. When a peer has no experience with another one, it can ask other peers to make recommendations for it. Such recommendations are partitioned in two groups, recommendations from trustworthy peers and recommendation from unknown peers, and are combined by taking a weighted sum. In [Xiong and Liu, 2003] an adaptive reputation-based trust model for P2P electronic communities is presented. It is based on five trust parameters: feedbacks, number of transactions, credibility of feedbacks, transaction context factor, and community context factor. The trust value associated with a peer is then defined as a weighted sum of two parts. The first part is the average amount of credible satisfaction a peer receives for each transaction. The second part increase or decrease the trust value according to community specific characteristics or situations (e.g., the number of files a peer shares can be seen as a type of community context factor that has to be taken into consideration when evaluating the trustworthiness of a peer). In [Gupta et al., 2003] two schemes are presented: *Debit-Credit Reputation Computation* (DCRC) and *Credit-Only Reputation Computation* (CORC). DCRC uses different parameters for computing the reputation score associated with peers. More precisely, the average query-response message size, the ration of Mbytes uploaded, and the amount of content shared are used to give credits to peers while the

Protocol 1 P2PRep *protocol*

INITIATOR: Servent p
Peers: Participants in the message broadcasting, among which a set O of offerers and a set V of voters

Initiator
Phase 1: Resource searching
1.1 Start a search request by broadcasting a `Query` message
`Query`(*min_speed,search_string*)
1.2 Receive a set of offers from offerers O
`QueryHit`(*num_hits,port,IP,speed,Result,trailer,servent_id*$_i$)
Phase 2: Polling
2.1 Select the best offerer $o \in O$
2.2 Generate a pair of public, secret keys ($\text{PK}_{poll}, \text{SK}_{poll}$)
2.3 Poll peers about the reputations of offerer o
`Poll`(o,PK_{poll})
2.4 Receive a set V of votes
`PollReply`($\{(IP, port, \text{PK}_{poll}, Votes)\}_{\text{PK}_{poll}}$)
Phase 3: Vote cleaning
3.1 Remove from V votes that appear suspicious (e.g., checking IP addresses)
3.2 **vote_verification**(V)
Phase 4: Vote aggregation
4.1 **compute_network_reputation**(o, V)
Phase 5: Resource downloading
5.1. If offerer o has a good network reputation, download the resource and update experience repository
PEERS
Q.1 Upon receiving a search request (`Query` message), check if any locally stored files match the query and if so send a `QueryHit` message
Q.2 Broadcast the query through the P2P network

P.1 Upon receiving a poll request (`Poll` message), check if know any of the servents listed in the poll request and express an opinion on them by sending a `PollReply` message
P.2 Broadcast the poll request through the P2P network
P.3 Upon receiving a `TrueVote` message, confirm the votes by sending a `TrueVoteReply`

Figure 1: Sequence of messages and operations in the P2PRep protocol

ratio of Mbytes downloaded is used to give debits. The reputation score is then computed as a weighted sum of credits minus the weighted debits. CORC computes the reputation score in the same way as DCRC; the only difference is that the debit component of DCRC is not used in CORC. Peers that are interested in enrolling in the reputation computations have to generate a (public, private) key pair and register them with a central *Reputation Computation Agent* (RCA). Debits and credits are then computed by contacting the RCA.

3 The protocol for collecting reputations

Before illustrating the protocol it is worth pointing out a few requirements that are peculiar to anonymous P2P environments (in contrast to centralized systems):

- the system has to maintain user anonymity, as this is a fundamental aspect of current P2P

file-sharing networks;

- reputation computation must be as transparent as possible to the user and user participation must be minimized;
- the reputation must be the result of the outcomes of previous interactions with the network;
- the reputation must be robust against malicious attacks that might try to attribute a bad reputation to well-behaved peers and good reputation to malicious ones.

To satisfy these requirements, a few assumptions must be made that are essential to the construction of a working system. Our first hypothesis is that the underlying TCP/IP network architecture is trusted and that it is not possible to falsify the IP address of a party involved in a TCP/IP connection. Although this assumption is not yet realistic on the current Internet (although much more so on corporate WANs), our aim is providing an application-level solution strong enough that the best (and perhaps the only) way to circumvent it is to challenge the infrastructure at an entirely different level. Attacks may be brought below or even above the application level but our solution limits substantially the palette of techniques available to the attacker. We then assume that malicious users may have considerable resources in terms of computational power and bandwidth, but they can control only a limited portion of the global network resources, and in particular a small fraction of the network nodes addresses. Moreover, we consider the goal to represent correctly only the reputations of nodes and resources that are known by a significant number of peers in the network. The protocol must then be able to correctly reconstruct the reputation in presence of malicious attacks.

3.1 P2PRep protocol

P2PRep is a reputation-based protocol, where each peer keeps track and shares with others the reputation of their peers [Damiani et al., 2003b]. Peers are identified using self-appointed opaque identifiers that are the digest of a public key for which only the peer knows the corresponding private key. Figure 1 illustrates the operations executed and the messages exchanged in the context of a P2PRep interaction. The figure reports the operations of the protocol's initiator p and of the other peers in the network.

Peers locate available resources using `Query` and `QueryHit` messages (Phase 1). Upon receiving a set of `QueryHit` messages, the network is polled for any available reputation information on the selected offerer o.[1] `Poll` messages are broadcasted in the same way as `Query` messages. All peers maintain an *experience repository* of their experiences of peers with whom they have interacted. When a peer receives a `Poll` message, it checks its local repository. If it has some information to offer and wants to express an opinion on the selected offerer o, it generates a vote based on its experiences, and returns it to the initiator p of the protocol as a `PollReply` message. As a result of Phase 2, p receives a set V of votes, where some votes can express a good opinion while some others can express a bad opinion.[2] In Phase 3, p evaluates the votes to collapse any set of votes that

[1] The original algorithm assumes a set of top offerers is chosen based on network performances. For simplicity, in the remainder of the paper we assume that a single offerer is chosen.

[2] We will elaborate on this in Section 4.1.

may belong to a clique and explicitly selects a random set of votes for verifying their veridicity (see Section 3.2) [Damiani et al., 2003b]. In Phase 4 the set of collected reputations is then synthesized into an aggregated network reputation value (see Section 4.2). Based on this reputation value, the initiator p can take a decision on whether to initiate a download (Phase 5), and, when a set of offerers is chosen, from which peer to download the resource. After the download, the initiator p can update its local recording of the offerer (recording whether the downloaded resource was satisfactory or not).

While the implementation of the P2PRep protocol requires a certain amount of resources (storage capacity and bandwidth), this cost is limited and justified in many situations. The amount of storage capacity is proportional to the number of servents with which the servent has interacted. With respect to the bandwidth, it is easy to see that the P2PRep protocol increases the traffic of the P2P network by requiring both direct exchanges and broadcast requests. It is however reasonable to assume that the major impact of the protocol on network performance is due to broadcast messages and their answers. To overcome this issue, several optimizations can be applied. For instance, *intelligent routing* techniques can be applied for enabling custom forwarding of poll packets to the "right" peers. Vote caching is another techniques that can be applied to improve the effectiveness of the P2PRep protocol.

3.2 Vote verification

The assessment of the correctness of the votes received in a poll is a fundamental step of the P2PRep. P2PRep provides a first protection against fake votes by requiring poll replies to be encrypted with a public key associated with the poll so to guarantee the confidentiality of votes in transit. In addition, the protocol chooses a portion of voters that are then directly contacted to detect possible vote spoofing. The goal of this process is to protect vote veridicity against attacks where malicious peers inject false votes into the network to modify the protocol outcome.

Since in current P2P networks it would be impractical, for performance reasons, to check all the votes, the protocol selects randomly a subset of votes to verify. The fraction of verified votes is a critical configuration parameter for the protocol and a tradeoff has to be considered between bandwidth consumption on one side, that requires to minimize the number of verifications, and protection against attacks on the other, that requires to evaluate a great portion of the votes. Our solution is based on selecting a vote for verification if the extraction of a random value, uniformly distributed between 0 and 1, exceeds a given threshold. More precisely, our approach makes use of two threshold parameters: *Threshold*, which represents the current threshold value; and *Delimiter*, which represents the stable threshold that has been reached by the previous verifications.

Figure 2 illustrates the algorithm for executing the vote verification. Here, $Threshold_{min}$ and $Threshold_{max}$ identify the range of values that are accepted for the threshold, depending on the number n of received votes. In this way the threshold can increase as the number of votes increases (with $Threshold_{max}$ growing faster than $Threshold_{min}$) because the probability that an attack that creates false votes goes undetected becomes lower as the absolute number of verified votes increases. Also, *fast* and *slow* are two constant values that in our experiments were set to 0.05 and 0.005, respectively. Note that *Threshold* reaches the value $Threshold_{min}$ ($Threshold_{max}$, resp.) after several consecutive negative (positive, resp.) verifications. In other words, a single individual negative (positive, resp.) verification has a limited impact. An analysis of the behavior of the algorithm

Algorithm 3.1 (of Phase 3: Vote cleaning)

VOTE_VERIFICATION(V)
/* Verify the reliability of votes V by selecting a subset of them and contacting the corresponding voters directly to check whether they actually expressed that vote */
$Threshold_{min} := 1 - \frac{1}{ln(|V|+e)}$; $Threshold_{max} := 1 - \frac{1}{ln^2(|V|+e)}$
$V_{check} := \{\}$ /* votes to be checked */
For $v_i \in V$ **do** /* i is the voter */
  Generate a random number r
  /* *Threshold* establishes how many verifications are carried out */
  If $r \geq Threshold$ **then** $V_{check} := V_{check} \cup v_i$
$V_{nocheck} := V - V_{check}$ /* votes that will not be checked */
For $v_i \in V_{check}$ **do**
  Check v_i by sending message `TrueVote`(v_i)
  Expect back confirmation message `TrueVoteReply`(*response*)
  Case *response* **of**
    `positive:` **If** $Threshold < Delimiter$ **then** $Threshold := \min(Threshold_{max}, Threshold + fast)$
      else $Threshold := \min(Threshold_{max}, Threshold + slow)$
      /* *Delimiter* is the stable threshold reached by the previous verifications */
      $Delimiter := \min(Delimiter + 0.075, Threshold_{max})$
    `negative:` $Threshold := \max(Threshold - 0.1, Threshold_{min})$
      $Delimiter := \max(Delimiter\text{-}0.075, Threshold_{min})$
      $V_{check} := V_{check} - v_i$
      Remove $\frac{1}{1-Threshold}$ votes from $V_{nocheck}$
$V := V_{check} \cup V_{nocheck}$
return V /* returns votes explicitly verified or not removed due to negative verifications */

Figure 2: Vote verification

shows that when the number of votes is large, the percentage of the verified votes is reduced to a relatively small value (e.g., it can reach 2% after 1000 votes); at first sight, this could appear a low value, but it is important to observe that the quantity of the verified votes is significant, enough to draw a conclusion on vote reliability. Note that if a vote verification fails, we assume that it is a false vote and we subtract from the number of the votes that had been expressed that vote multiplied by the inverse of the difference between 1 and the threshold value used for the random choice. For instance, let us suppose we received the 100-th vote, which is positive; the *Threshold* is equal to 0.95; the random value is greater than 0.95 and the vote is verified. If the vote is not successfully verified, we mark the vote as invalid and remove 20 positive votes from the total $(1/(1 - 0.95))$. The goal is to make vote falsification a zero-sum game, that statistically neither improves nor decreases the outcome of the poll.

4 Reputation model

As already noticed, in our approach, reputations are viewed at two levels: *local* and *network reputation*. Local reputations represent direct experience of interactions between two peers, where network reputations represent the synthesis resulting by aggregating multiple opinions about a peer. In the remainder of this section, we illustrate our fuzzy-based approach to computing local and network reputations.

4.1 Local Reputation

Local reputation $r_{i,j}$ is the result of direct interactions between peer i and peer j, where i downloads a resource from j, and expresses i's satisfaction for past interactions with j. We use a Boolean value to express the satisfaction of each single transaction. Specifically, for each download we model the transaction outcome $t_{i,j}^{(n)}$ as follows: $t_{i,j}^{(n)} = 1$ if the outcome was satisfactory, $t_{i,j}^{(n)} = 0$ otherwise. By contrast, we use a fuzzy value to express local reputations to take into consideration the fact that transactions can be heterogeneous for importance, resource value, and so on. Each local reputation is initialized after the first interaction by taking the value of $t_{i,j}^{(1)}$. At any time $n > 1$, local reputation is updated based on the outcome of the n-th transaction as follows.

$$r_{i,j}^{(n)} = \begin{cases} t_{i,j}^{(n)} & \text{if } n = 1 \\ \alpha^{(n)} r_{i,j}^{(n-1)} + (1 - \alpha^{(n)}) t_{i,j}^{(n)} & \text{if } n \geq 2 \end{cases} \quad (1)$$

where $\alpha^{(n)}$, a value between 0 and 1, is the aggregation *freshness*,[3] that is, the importance of past transactions in relation with the last one. If $\alpha^{(n)} \simeq 1$ past experience will have a very high importance and the last transaction has a little role in reputation evaluation; if $\alpha^{(n)} \simeq 0$ then last experience is merely forgotten. Note that our freshness value is not static, but can change at any single interaction, according to circumstances. In particular, for the first experiences of i with j, (i.e., for low values of n), freshness should remain high while it can decrease as n grows and therefore i has acquired enough experience on j. Note however that freshness should never become too low, since this would mean having a blind trust in other peers.

In our approach, freshness evolution relies on feedback, by checking whether the current reputation value of a peer would give an accurate prediction of the result of the next transaction with it. If this is the case, reputation is just right and freshness should not increase; otherwise, the current reputation is considered unreliable and α is incremented. Our approach follows a well-known heuristic technique for feedback control, that quickly stabilizes to a fair and efficient setting [Jacobson, 1988]. While other feedback strategies such as MIMD (Multiplicative Increase/Multiplicative Decrease) could be adopted, they are known to be less robust and need careful tuning to avoid oscillatory behavior [Chiu and Jain, 1989].

More specifically, if we consider the reputation $r_{i,j}^{(n-1)}$ to be a prediction of the outcome of i's next download from j, the *accuracy* of this prediction can be computed with a Boolean function which returns 0 (wrong prediction) if the value of $r_{i,j}^{(n-1)}$ differs from the actual outcome $t_{i,j}^{(n)}$ more than a given error threshold E, and returns 1 (right prediction) otherwise. Namely,

$$\mathrm{Acc}_{i,j}^{(n)} = \begin{cases} 1 & \text{if } \mid r_{i,j}^{(n-1)} - t_{i,j}^{(n)} \mid < E \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

This accuracy value is then used to determine a coefficient $\beta^{(n)}$ taking into account past experience and the outcome of the last transaction as follows. The coefficient is initialized to $\beta^{(0)} = 0$ and is updated at subsequent times as:

[3] Different values of freshness are used for different peers and therefore a peer i will use a different α for each different peer j. For readability, we omit the subscripts when they are clear from the context.

$$\beta^{(n)} = \frac{\beta^{(n-1)} + Acc_{i,j}^{(n)}}{2} \tag{3}$$

Finally, $\alpha^{(n)} = \frac{\beta^{(n)}}{2}$.

If the degree of similarity $Acc_{i,j}^{(n)} \simeq 1$, $\alpha^{(n)}$ will increase granting more importance to past history. Note that, of course, the accuracy could also be defined as a fuzzy function, for example, by considering that not all transactions have the same importance. We shall not elaborate on this possibility in this paper, assuming that the β coefficient summarizes all context representation.

4.2 Network Reputation

If a peer has no previous experience on a peer j or the local reputation is still considered not mature (i.e., for low values of n), the peer will, by using P2PRep, run a poll and inquire other peers about j's reputation for them. Here, we assume the vote expressed by each peer k participating in the poll on j to be its local reputation $r_{k,j}$ of j. The question now is how the poller should aggregate the different votes received to produce a synthesized value.[4] A basic requirement for aggregating opinions is that if the pool peers is stable and they maintain identical beliefs across all transactions from a given instant t_0 onwards, then all interactions will asymptotically conform to these beliefs. In other words, if the majority of peers considers that peer j has a bad reputation, then j will be in the end excluded from all transactions.[5] This property, called *unanimity*, is often considered a minimal standard of acceptability for an opinion pool aggregator, and is held both by the weighted mean and the geometric mean [Aczél and Alsina, 1986]. The simplest aggregation available is the arithmetic average of the votes received. However, arithmetic average performs a rough compensation between high and low values, not taking into account different variations between individual opinions that may characterize different polls. Luckily, arithmetic means are not the only aggregation functions usually used in opinion pooling and many other combination methods had been studied.

The *OWA (Ordered Weighted Average)* operator, introduced by Yager in [Yager, 1988], allows the decision maker to give different importance to the values of a criteria. The main difference between OWA and the arithmetic means consists in the *separability* of the aggregation function: OWA considers that the influence of each contribution on the result is not directly separable, but depends on the other contributions. For instance, a very good reputation value in the midst of low ones should be treated differently than a good value accompanied in the poll by fair ones. Technically, an OWA operator is a weighted average that acts on an ordered list of arguments and applies a set of weights to tune their impact on the final result. Namely, in our setting,

$$\lambda_{OWA} = \frac{\sum_{k=1}^{n} w_k r_{t_k,j}}{\sum_{k=1}^{n} w_k} \tag{4}$$

where n is the number of reputations to be aggregated considered in decreasing order, that is, assuming $r_{t_1,j} \geq r_{t_2,j} \geq ... \geq r_{t_n,j}$ and $[w_1\ w_2 \dots\ w_n]$ is a weighting vector. The behavior of this

[4]Note that since the individual's reputations are fuzzy, their fuzzy aggregation will also be a value in the unit interval.

[5]Of course, the delay can be arbitrarily long if communications on the network are slow.

operator is largely determined by the choice of weights. For instance, the result could be based on the most frequent values simply by assigning lower weights to extreme ones. Alternatively, high weights can be given to extreme values to increase the operator responsiveness to them.[6] In our case, we set the OWA weights *asymmetrically*, since our aggregation operator needs to be biased toward the lower end of the interval, increasing the impact of low local reputations on the overall result. The reason is that we assume that peers are usually trustworthiness and a malicious behavior is the exception. According to this assumption, low local reputations should be considered as relevant and their impact on the overall reputation should be significant. Of course, there are many ways to do this, for example, by increasing weights linearly or non-linearly with the position k of the corresponding opinion $r_{t_k,j}$ in the OWA ordered set of arguments. Also, it is possible to give a bonus to multiple occurrences of the same weight (*group votes*). This can be done by defining the operator on a (usually small) set of different reputation values d rather than on the (usually large) number of peers, as follows:

$$\lambda = \frac{\sum_{i=1}^{d} w_i (v_i)^{1/|V_i|}}{\sum_{i=1}^{d} \mid V_i \mid w_i} \tag{5}$$

where $V_1, \ldots, V_d$ is a partitioning of the set of votes grouping together votes with the same value v_i, and where v_i are considered ordered, that is, $v_1 > \ldots > v_d$.

Since our local opinions range in the unit interval, in principle there is no reason to favor group votes; however, computational efficiency and values' rounding to a fixed number of decimal may make this a viable solution for practical implementations. In the algorithm shown below, we simply partition the unit interval in $d+1$ sub-intervals and use their extreme values (discarding 0 and 1) as a (linearly increasing) set of weights for aggregating the d distinct reputation values to be aggregated via the OWA operator. In other words, we set

$$\lambda = \frac{\sum_{i=1}^{d} \frac{i}{d+1} v_i \mid V_i \mid}{\sum_{i=1}^{d} \frac{i}{d+1} \mid V_i \mid} \tag{6}$$

We include local reputation in the computation by adding a new class $V_{d+1} = r_{p,j}$ and associate with it the highest possible weight, with weights now computed for $d+1$ values. Our operator is now defined as follows:

$$R_{p,j} = \frac{\sum_{i=1}^{d+1} \frac{i}{d+2} v_i \mid V_i \mid}{\sum_{i=1}^{d+1} \frac{i}{d+2} \mid V_i \mid} \tag{7}$$

Figure 3 illustrates the algorithm performing this computation.

[6] With the OWA operator the decision makers can express their preferences in relation of the values of the criteria, they cannot express preferences between criteria. This drawback is important when criteria are heterogeneous and is usually solved using the *WOWA (Weighted OWA)* operator [Torra, 1997] instead of OWA. Here, however, we are in a homogeneous scenario, so using OWA looks perfectly safe.

Algorithm 4.1 (of Phase 4: Vote aggregation)

COMPUTE_NETWORK_REPUTATION(o, V)
/* Compute the network reputation of the offerer o by aggregating the received votes V*/
Compute the number d of distinct values in V
Let $v_1, \ldots, v_d$ be the order sequence of distinct values in V $(\forall i, k \in [1, 2, \ldots, d] : i < k \Longrightarrow v_i > v_k)$
Partition the set V of votes into d subsets $V_1, \ldots, V_d$ such that $V_i = \{v \mid v \in V \wedge v = v_i\}$ and $V = \bigcup_{i=1}^{d} V_i$
$V_{d+1} := \{r_{p,o}\}$
$R_{p,o} := \frac{\sum_{i=1}^{d+1} \frac{i}{d+2} v_i |V_i|}{\sum_{i=1}^{d+1} \frac{i}{d+2} |V_i|}$

Figure 3: Computing network reputation

5 Evaluation

In this section, we investigate the behavior of our solution by highlighting its features with respect to EigenTrust [Kamvar et al., 2003], a probabilistic approach to trust computation.

5.1 Overview of EigenTrust

In EigenTrust, each peer i rates another peer j from which it tries to download a file by keeping track of the numbers of successful *sat*(*i*,*j*) and unsuccessful *unsat*(*i*,*j*) downloads. A *local trust value* s_{ij} is then defined as the difference between *sat*(*i*,*j*) and *unsat*(*i*,*j*). To aggregate these local trust values around the P2P network, they are normalized so that malicious peers will not be able to assign arbitrarily high trust values to other malicious peers and subvert the EigenTrust algorithm. The normalized local trust value c_{ij} is defined as:

$$c_{ij} = \frac{\max(s_{ij}, 0)}{\sum_j \max(s_{ij}, 0)} \tag{8}$$

The normalized trust values are aggregated using the concept of *transitive trust*: peer i can know about the trust of peer k by asking all peers j with which peer i has interacted. However, since not all peers j are trustworthy, their opinions is weighed with the trust peer i places in them:

$$t_{ik} = \sum_j c_{ij} c_{jk} \tag{9}$$

In other words, if C is defined as the matrix $[c_{ij}]$, c_i is defined as a local trust vector containing c_{ij} values for all peers j that peer i has interacted with, and $\overrightarrow{t_i}$ is the vector containing the values t_{ik}, then $\overrightarrow{t_i} = C^T \overrightarrow{c_i}$. By generalizing this approach to include the opinions that "friends of our friends" have about peer k and so on, we obtain a trust vector $\overrightarrow{t} = (C^T)^n \overrightarrow{c_i}$. As n gets larger, the global trust vector $\overrightarrow{t}$ for all peers converges to the same value. To compute the global trust value $\overrightarrow{t}$, the authors of EigenTrust propose two approaches: *i)* a basic non-distributed EigenTrust algorithm and a *ii)* distributed EigenTrust algorithm. The first solution is based on the observation that the global trust vector does not depend on i and therefore c_i can be replaced with any distribution. A uniform distribution $\overrightarrow{e}$, where $e_i = \frac{1}{m}$ and m is the number of peers in the P2P network is chosen. The definition of the global trust vector $\overrightarrow{t}$ is then $\overrightarrow{t} = (C^T)^n \overrightarrow{e}$. The Basic EigenTrust algorithm

consists in setting $\overrightarrow{t}^{(0)}$ equals to $\overrightarrow{e}$ and computing $\overrightarrow{t}^{(k+1)} = (C^T)^n \overrightarrow{t}^{(k)}$ until it converges. This simple algorithm does not take into consideration two important issues: the lack of a prior notion of trust, and the fact that malicious peers can form a group and can try to increase the trust associated with them and to decrease the trust associated with all the other peers. To overcome these drawbacks, the authors propose to introduce the notion of *pre-trusted peers*, that is, peers in the network that can be trusted. Let P be the set of pre-trusted peers, the global trust vector can be computed as $\overrightarrow{t}^{(k+1)} = (1-a)(C^T)^n \overrightarrow{t}^{(k)} + a\overrightarrow{p}$, where a is a constant less than 1 and $\overrightarrow{p}$ is a distribution with $p_i = \frac{1}{|P|}$ if $i \in P$; $p_i = 0$, otherwise. By writing this equation component-wise, the following formula is obtained:

$$t_i^{(k+1)} = (1-a)(c_{1i}t_1^{(k)} + \ldots + c_{ni}t_n^{(k)}) + ap_i \tag{10}$$

Therefore, each peer needs its local trust vector and its global trust value. This observation results in a distributed EigenTrust algorithm that works as follows.

Foreach peer i do:

1. query all peers j who have downloaded files from i for their opinions about him ($t_j^{(0)} = p_j$)

2. repeat

2a. compute i's current global trust value $t_i^{(k+1)} = (1-a)(c_{1i}t_1^{(k)} \ldots c_{ni}t_n^{(k)} + ap_i$

2b. send opinion $c_{ij}t_i^{(k+1)}$ to peers j from which i has downloaded files

2c. wait for all such peers j to send their updated trust values $c_{ji}t_j^{(k+1)}$

until $\mid t_i^{(k+1)} - t_i^{(k)} \mid < \epsilon$.

5.2 Experiment setting and results

We now describe the experiments we performed to evaluate the behavior of our P2PRep protocol and EigenTrust, as a representative of probabilistic approaches, against downloads from malicious peers. Our simulation model refers to a P2P network where each peer is reachable from all others[7] and does not take into account delays due to message routing. In other words, all peers are considered to be at one hop from one another. Over this network, we simulate a set of queries, each asking for a randomly chosen resource. For each query, the peer querying the network is randomly chosen (with a uniform probability distribution) over all available peers. Then, a preferred offerer o is selected, randomly choosing some peers among those having the resource required. In our simulation, a malicious peer is more likely to be selected as o than a well-behaved one.

The main settings of our experiments are as following:

- number of peers P in the network: uniformly distributed in $[300, 400]$;
- number of malicious peers $M \subset P$: 40% of $\mid P \mid$;
- number of pre-trusted peers (for EigenTrust): 5% of $\mid P \mid$;
- number of different kinds of resources: 20;

[7] In real P2P systems this condition is only verified within a fixed *horizon*.

Reputation model	Number of queries									
	1.000	*2.000*	*3.000*	*4.000*	*5.000*	*6.000*	*7.000*	*8.000*	*9.000*	*10.000*
Random	37.41	38.28	38.04	37.81	38.38	38.59	37.24	37.23	37.49	37.78
EigenTrust	16.82	18.54	17.36	15.92	16.49	16.70	15.58	16.33	15.72	17.49
P2PRep	30.60	27.84	22.87	21.01	18.55	16.59	14.92	14.41	12.82	12.98

Table 1: Percentage of malicious downloads

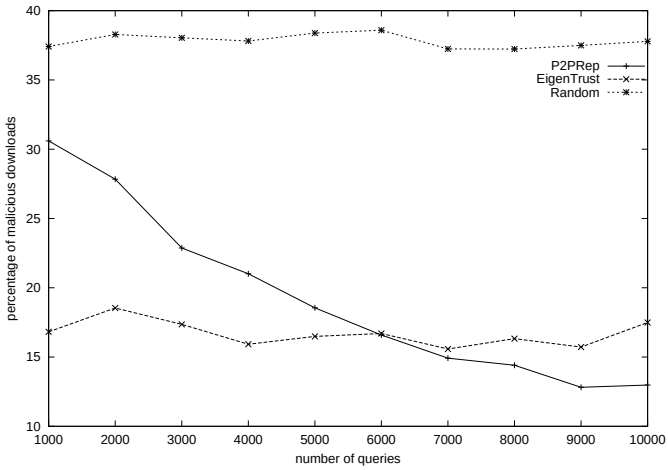

Figure 4: Percentage of malicious downloads for the random, P2PRep, and EigenTrust reputation model

- max poll cardinality (for P2PRep): uniformly distributed in $[5, 15]$.

We run the experiments for three different approaches: *random*, P2PRep, and *EigenTrust*. The random policy simply chooses the offerer randomly and does not perform any reputation check; while the other two approaches are as illustrated in the paper. As for P2PRep, we assume all genuine peers i participate in a poll on offerer o by returning the local reputation $r_{i,o}$ if such a value is recorded; no response is returned otherwise. Moreover, we modeled the behavior of malicious peers in M by assuming that: *1)* malicious peers provide only malicious resources; *2)* malicious peers respond to the polling on a peer o by always providing a (malicious) 1 reputation if $o \in M$, and by providing a genuine opinion otherwise.

Our simulation consists of a number of repeated experiments, each one evaluating a different, randomly generated scenario. In this paper, we have set the total number of experiments to 50 while the number of queries for each experiment ranges from $1,000$ to $10,000$ with an increment of $1,000$. Higher values are not infrequent on real P2P systems like Gnutella[Damiani et al., 2002]; however we chose this range to enforce the assumption that the set of peers remains more or less stable across the experiments.

The results of our experiments are reported in Table 1 as well as in graphical form in Figure 4. The EigenTrust algorithm uses the eigenvalues of the matrix C to take transitivity into account when computing the trustworthiness of a candidate offerer. The low complexity of this closure

computation ensures good performance of the EigenTrust algorithm even for a small number of transactions. However, being the overall knowledge over the peers' trustworthiness stored in a matrix, a single variation of s_{ij} has a small impact in Equation (8) and the algorithm cannot improve much over time. The fuzzy solution has a slower start but the percentage of malicious downloads steadily decreases for the fuzzy solution as the quality of the network reputation increases following the diffusion of information about malicious peers.[8]

6 Conclusions

We have introduced a fuzzy reputation approach for P2P capable to increase the overall accountability of the network without imposing a high computational burden. The behavior of our algorithm has been assessed in comparison with the probabilistic approach of [Kamvar et al., 2003]. Further research will be aimed at improving the proposed fuzzy technique by taking into account other aspects relevant for computing the reputation (e.g., the timing) and at investigating the effects of the high mortality rates typical of P2P environments and assuming more complex behavior of malicious peers.

Acknowledgments

This work was supported in part by the European Union within the PRIME Project in the FP6/IST Programme under contract IST-2002-507591 and by the Italian MIUR within the KIWI and MAPS projects.

References

[Aberer and Despotovic, 2001] Aberer, K. and Despotovic, Z. (2001). Managing trust in a peer-2-peer information system. In *Proc. of the Tenth International Conference on Information and Knowledge Management (CIKM 2001)*, Atlanta, Georgia.

[Aczél and Alsina, 1986] Aczél, J. and Alsina, C. (1986). On synthesis of judgements. *Socio-Econom Planning Science*, 20/6:333–339.

[Carter et al., 2002] Carter, J., Bitting, E., and Ghorbani, A. (2002). Reputation formalization within information sharing multiagent architectures. *Computational Intelligence*, 18(4):515–534.

[Chiu and Jain, 1989] Chiu, D. and Jain, R. (1989). Analysis of the increase and decrease algorithms for congestion avoidance. *Journal of Computer Networks*, 17(1):1–14.

[Damiani et al., 2003a] Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Pesenti, M., Samarati, P., and Zara, S. (2003a). Fuzzy logic techniques for reputation management in anonymous peer-to-peer systems. In *Proc. of the Third International Conference in Fuzzy Logic and Technology*, Zittau, Germany.

[8] The slope of this descent is likely to be less steep if a network delay is introduced to slow down the spreading of information.

[Damiani et al., 2003b] Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., and Samarati, P. (2003b). Managing and sharing servents' reputations in P2P systems. *IEEE Transactions on Data and Knowledge Engineering*, 15(4):840–854.

[Damiani et al., 2002] Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P., and Violante, F. (2002). A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proc. of the 9th ACM Conference on Computer and Communications Security*, Washington, DC, USA.

[Dellarocas, 2000] Dellarocas, C. (2000). Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *Proc. of the 2nd ACM Conference on Electronic Commerce*, Minneapolis, MN, USA.

[Dingledine et al., 2001] Dingledine, R., Freedman, M. J., Hopwood, D., and Molnar, D. (2001). A reputation system to increase MIX-net reliability. *Lecture Notes in Computer Science*, 2137:126+.

[Dingledine and Syverson, 2002] Dingledine, R. and Syverson, P. (2002). Reliable MIX cascade networks through reputation. In *Proc. of Financial Cryptography*.

[eBay Feedback Forum, 2003] eBay Feedback Forum (2003). eBay Feedback Forum. http://pages.ebay.com/services/forum/feedback.html?ssPageName=STRK:SRVC:038.

[Friedman and Resnick, 2001] Friedman, E. and Resnick, P. (2001). The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199.

[Gupta et al., 2003] Gupta, M., Judge, P., and Ammar, M. (2003). A reputation system for peer-to-peer networks. In *Proc. of the ACM 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Monterey, California, USA.

[Jacobson, 1988] Jacobson, V. (1988). Congestion avoidance and control. *ACM SIGCOMM Computer Communication Review*, 18(4):314–329.

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

[Kinateder and Pearson, 2003] Kinateder, M. and Pearson, S. (2003). A privacy-enhanced peer-to-peer reputation system. In *Proc. of the 4th International Conference of E-Commerce and Web Technologies*, Prague, Czech Republic.

[Klir and Folger, 1988] Klir, G. and Folger, T. (1988). *Fuzzy Sets, Uncertainty, and Information*. Prentice-Hall.

[Oram, 2001] Oram, A., editor (2001). *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates.

[Rahman and Hailes, 2000] Rahman, A. and Hailes, S. (2000). Supporting trust in virtual communities. In *Proc. of the IEEE Hawaii International Conference on System Sciences*, Maui, Hawaii.

[Rasmusson and Jansson, 1996] Rasmusson, L. and Jansson, S. (1996). Simulated social control for secure internet commerce. In *Proc. of the New Security Paradigms Workshop*, Lake Arrowhead, CA, USA.

[Torra, 1997] Torra, V. (1997). The weighted owa operator. *International Journal of Intelligent Systems*, 12(2):153–166.

[Wang and Vassileva, 2003] Wang, Y. and Vassileva, J. (2003). Trust and reputation model in peer-to-peer networks. In *Proc. of the Third International Conference on Peer-to-Peer Computing*, Linköping, Sweden.

[Xiong and Liu, 2003] Xiong, L. and Liu, L. (2003). A reputation-based trust model for peer-to-peer ecommerce communities. In *Proc. of the IEEE International Conference on E-Commerce*, Newport Beach, California, USA.

[Yager, 1988] Yager, R. (1988). On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):183–190.

[Yahalom et al., 1993] Yahalom, R., Klein, B., and Beth, T. (1993). Trust relationships in secure systems. In *Proc. of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, USA.

[Yu and Singh, 2000] Yu, B. and Singh, M. (2000). A social mechanism for reputation management in electronic communities. In *Proc. of the 4th International Workshop on Cooperative Information Agents*, Boston, USA.

[Zacharia et al., 1999] Zacharia, G., Moukas, A., and Maes, P. (1999). Collaborative reputation mechanisms in electronic marketplaces. In *Proc. of the 32nd Hawaii International Conference on System Sciences*, Maui, Hawaii.