

Combining Transitive Trust and Negative Opinions for better Reputation Management in Social Networks

Debora Donato¹

Stefano Leonardi²

Mario Paniccia²

¹Yahoo! Research Barcelona, Spain

²University of Rome, La Sapienza

ABSTRACT

Reputation management is a crucial task in Peer-to-Peer networks, social networks and other decentralized distributed systems. In this work we investigate the role of users' negative opinions in order to devise fully decentralized reputation management mechanisms. Our study is motivated by the limitations of methods proposed in literature that are based on the idea of propagating positive opinions, most notably EigenTrust [9], a cornerstone method for reputation management in decentralized systems. EigenTrust makes use of a transitive definition of trust: a peer tends to trust those peers who have a high reputation in the opinion of trustworthy peers. While EigenTrust has been shown to be effective against a number of threat attacks from coalitions of malicious players, it does not address properly more sophisticated threat attacks.

In this paper we propose a new approach to the design of fully decentralized reputation mechanisms that combine negative and positive opinions expressed by peers to reach a global consensus on trust and distrust values for each peer of the network. We show how these strategies outperform EigenTrust in terms of number of successful transactions against a large set of sophisticated threat attacks posed by coalitions of malicious peers. We also discuss a clustering method that achieves detecting most of the malicious peers with high precision.

Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; H.3.4 [Information Storage and Retrieval]: [Systems and Software - Distributed Systems]

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

General Terms

Algorithms, Security.

Keywords

P2P Networks, Social Networks, Trust, Reputation, Distributed Recommendation Systems.

1. INTRODUCTION

Decentralized environments, such as social networks and peer-to-peer networks, are increasingly spreading over the Internet as premium media for sharing links, data and services. The main drawback of these systems is the lack of a reliable authority certifying the quality of the information and services provided by individual users. However, millions of users are looking forward to principled design guidelines that make social networks accomplish new and more sophisticated tasks like spam resilience. The answer to this question will deeply affect the degree of deployment and the direction of development of these technologies in the near future.

A reputation system is meant to "assist agents in choosing a reliable peer to transact with when one or more have offered the agent a service or resource" [10]. A reputation system in a fully decentralized environment must possess the ability to collect and aggregate the opinions of the users on the quality of the resources and services received by other users. However, when addressing the issues of trust and reputation in fully decentralized environments, it should be assumed that individuals and coalitions may coordinate to provide untruthful information in order to gain some undeserved reputation or simply to disrupt the system. Any mechanism that is run by autonomous untrusted agents must be shown to have essential characteristics like reliability, efficiency, security and robustness against sophisticated attacks orchestrated by malicious peers [10].

Building a reputation mechanism with these requirements is a very challenging task that deserves to be explored since we expect a high impact of this line of research to social network applications that rely on users' feedback to rank users, information and services. As an example, question answer games deeply rely on users rating answers, questions and experts. Social media sites, like news social aggregators and question-answering portals, allow users to submit their own contents, in the form of links, questions, answers, votes. This kind of

systems do not rely on the opinion of a few editors, but aggregate opinions of thousands of them users to decide which news deserves to be promoted, which is the best answer to a question, or which questions are more interesting. Social filtering is an effective information filtering approach and great benefits could be obtained by providing these sites with reputation systems able to identify users that behave in order to defeat the rules that govern such systems.

We develop our methods by considering information sharing on a social network as a benchmark application. We test the reputation system against sophisticated and coordinated attacks orchestrated by a number of malicious peers providing wrong or manipulated information, e.g. corrupted files, and untruthful feedback on the reputation of other peers. The goal is to allow a quick retrieval of the trust ratings of peers, and above all, to update the trust ratings after the feedback of users on the received information is collected. Our objective is to develop a system able to detect most of the malicious peers with high precision. Low precision is unacceptable since it might prevent honest peers rated as malicious from providing any content to other peers.

The final goal of the reputation system is to suggest trusted entities in the system to transact with. This is obtained by maintaining in a decentralized fashion a TrustRank for every candidate peer. The computation of the TrustRank is based on the social network defined by the set of trusted relationships established so far. The objective is to minimize the number of transactions performed with malicious peers.

Our contribution: Our work depart from EigenTrust, a reputation algorithm proposed in [9] for fully decentralized systems and from a number of methods recently developed for detection and demotion of Web spam: Truncated PageRank [3], Bit Propagation [4] and BadRank [1]. In this work we present a number of novel approaches to integrate negative opinion metrics with the notion of transitive trust:

- We show how several methods of aggregation of negative opinions, *Badness*, *Positive Dishonesty* and *Negative Dishonesty*, can be jointly combined with EigenTrust in one single reputation algorithm.
- Our approach is shown very effective in reducing the number of inauthentic downloads from malicious peers. The evaluation is experimentally performed by simulating a large number of threat attacks from coalitions of malicious peers. Some of the attacks are introduced in this work; others, including those not properly addressed by EigenTrust, were already presented in literature [9, 6].
- We address for the first time the issue of detecting malicious peers. We show a clustering method able to detect most of the malicious peers with high precision.

All our algorithms converge, are scalable and implementable in distributed and secure manner using standard techniques as those presented in [9].

The rest of the document is organized as follows. Section 2 describes previous work on reputation management in decentralized systems. In Section 3 we briefly recall the EigenTrust algorithm and we present the performance metrics we evaluate. In Section 4, we introduce our novel taxonomy of threat attacks from malicious coalitions. In Section 5 we present the metrics obtained by aggregating negative opinions. In Section 6, we describe the algorithms obtained by combining transitive trust with negative opinions and the rules of selection of the transacting peers. In Section 7 the simulation framework is described. In Section 8 the proposed strategies are experimentally evaluated against the threat attacks of our taxonomy. Finally, Section 9 presents conclusions and discusses future work.

2. RELATED WORK

EigenTrust [9] is an algorithm that shows how to aggregate the local trust assessments of all peers in the network in an efficient, distributed manner. The adaptation of several metrics originally developed for detecting Web spam [4, 1, 3] to the context of trust management in fully decentralized networks has preliminarily been explored in [6]. The combination of EigenTrust with badness has been shown more effective than EigenTrust on some attacks from malicious coalitions introduced in [9]. In [7] it is proposed to propagate negative opinions through trusted and untrusted peers. This solution cannot be adapted to reputation mechanisms in P2P networks since it relies on negative trust assessments reported by untrustworthy peers.

In [2] several issues related to practical aspects of keeping peers honest in EigenTrust are presented. In [14] it is proposed a novel algorithm, PeerTrust, that combine several parameters such as feedback from peers and credibility of the feedback source into a general trust metric. More recently, in [15], a novel protocol called SybilGuard for limiting the corruptive influences of sybil attacks is based on detecting small cuts in the graph of trust relationships between peers.

A different approach [13] enables every peer to compute a personalized, rather than global, trust rank of peers. The personalized rank of peer a conferred by peer b is computed by comparing the observations made in the past on the performance of other peers. A related approach aimed to incorporate the notion of trust into distributed ranking of Web pages, has also been used in recent works, specifically the concept of trustworthy peer in the meeting selection step of the JXP algorithm [11]. Observe that a more global approach is needed if the reputation system needs to be resilient against the local activity of a set of malicious peers. An excellent taxonomy of concepts in P2P reputation and trust management is given in [10].

3. PRELIMINARIES

EigenTrust. In our model, peers rate each other after each transaction. Each time peer i downloads a file from peer j , it may rate the transaction as positive ($tr(i, j) = 1$) or negative ($tr(i, j) = -1$). We define a local trust value s_{ij} as the sum of the ratings of the trans-

actions that peer i has had with peer j : $s_{ij} = \sum tr_{ij}$. Equivalently, each peer i can record the number of satisfactory transactions it has had with peer j , $sat(i, j)$ and the number of unsatisfactory transactions it has had with peer j , $unsat(i, j)$. Then, s_{ij} is defined:

$$s_{ij} = sat(i, j) - unsat(i, j).$$

Local trust values must be normalized to avoid arbitrarily high assignments from malicious peers. The normalized local trust value c_{ij} is defined as follows:

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

This ensures that all values are between 0 and 1.

A natural way to aggregate the normalized local trust values in a distributed environment is for peer i to ask its acquaintances for their opinions about other peers, and weight such opinions by the trust peer i places in them:

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

where t_{ik} represents the trust that peer i places in peer k based on asking his friends. We can express this in matrix notation: if we define C to be the matrix $[c_{ij}]$ of local trust values and t_i to be the vector containing the values t_{ik} , then $(t_i = C^T c_i)$, with $\sum_j t_{ij} = 1$ as desired. We then ask the friends of the friends, therefore defining $(t = (C^T)^n c_i)$, after a large number n of iterations. Under the assumptions that C is irreducible and aperiodic, the vector t_i will converge to the left principal eigenvector of C , i.e., the vector \mathbf{T} of *global trust* values.

An important role for the convergence of the algorithm is played by the so called *pre-trusted peers*, that is, peers that are known to be trustworthy like the designers of the network. Their presence ensures the convergence of the algorithms. A number of practical issues concerning the computation in a distributed and secure manner of the global EigenTrust vector are discussed in [9]. Similar ideas apply to the algorithms proposed in this work.

Performance metrics. The metrics we use to assess the performances of the P2P system are based on the number of inauthentic file downloads versus the number of authentic file downloads: if the computed global trust values accurately reflected each peer's actual behavior, the number of inauthentic file downloads should be minimized. This goal is obtained by designing methods that are able to discriminate between malicious and honest peers. Honest peers will be preferred to malicious peers when deciding which peer, within a given set of peers, to download from. We define by *true positives* the percentage of malicious peers that are penalized by the method. On the other hand we aim to high precision, i.e., we need to avoid penalizing honest peers in some cases accumulating a negative reputation score. For this reason, we also measure *false positives*, i.e., the percentage of honest peers that are penalized by the method. We ideally aim at achieving 100% of true positives and 0% false positives.

4. THREAT MODELS

Malicious peers can operate in different ways in order to subvert the system and gain high global trust. In [9] several possible threat models are presented. Additional threat models are also introduced in [6]. We propose a taxonomy of threat models that subsume and extend all previous proposals. The first three threat models are equivalent to models A, B and D of [9]:

Threat Model A Malicious peers always provide inauthentic files. Moreover, they report positive opinions on any other malicious peer they meet. This is done for malicious peer i by setting the local trust value $s_{ij} = inauth(j) - auth(j)$ for any other malicious peer j that i meets. In this way malicious peers positively assess inauthentic file downloads instead of authentic ones.

Threat Model B Malicious peers always upload inauthentic files and collude within a M of peers in the network. Each peer in M assigns positive local trust values only to another single malicious peer in the coalition.

Threat Model D Malicious peers are of two different types: part of them acts exactly as in threat model B, the others, called *malicious spies*, answer 0.05% of the most popular queries always uploading authentic files but they assign positive local trust values only to peers of type B. In this way, the spies gain high global trust value that are able to somehow transfer to malicious peers.

While EigenTrust successfully addresses threat models A and B, it performs poorly on the more sophisticated threat model D [9].

Malicious peers can also implement more sophisticated individual and combined attacks with the goal of worsening the ability of the algorithm to detect malicious peers. These attacks are obtained by extending attacks A, B and D with a combination of the following threat features:

Threat Feature α . This feature is called *Camouflaging behind transactions*. Malicious peers do not always upload inauthentic files. They do it in the $f\%$ of the cases.

Threat model C defined in [9] corresponds to threat model B with feature α . It is reported in [9] that EigenTrust performs on threat model C better than on threat model D, but worse than it does on threat models A and B.

Threat Feature β . This feature is called *Camouflaging behind positive judgements*. Malicious peers assign positive local trust value also to good peers. They do it in the $g\%$ of the cases.

Threat model G defined in [6] corresponds to threat model D with feature β , whereas model H, also defined in [6], is obtained by adding feature α to threat model G.

Threat Feature γ . This feature is called *Camouflaging behind negative judgements*. Malicious peers assign negative local trust also to malicious peers. They do it in the $h\%$ of the cases. This feature defines a new type of attack never addressed before.

In [9] threat models E and F are also defined but not addressed by EigenTrust. We should also not assume

honest peers always reporting authentic files. In our simulations honest peers supply corrupted files in 2% of the cases.

5. NEGATIVE OPINION METRICS

To overcome the limitations of EigenTrust, we integrate in the reputation mechanism a new class of metrics that aggregate the *negative opinions* expressed by peers: *Badness*, *Positive Dishonesty* and *Negative Dishonesty*.

5.1 Badness

Badness is meant to identify those peers that report inauthentic files. Exploiting the fact that each peer is prone to trust the opinions of peers that have already built a positive reputation, we can say that if i trusts j and j distrusts k , then i should regard k as untrustworthy as well. Following the same arguments used to build the global EigenTrust vector \mathbf{T} , we define the *Badness* vector as:

$$\mathbf{negT} = \mathbf{D}^\top * \mathbf{T}$$

where D is the normalized negative opinion matrix whose elements are $negC_{ik}$, $k = 1, \dots, n$:

$$negC_{ik} = \frac{\min(s_{ik}, 0)}{\sum_k \min(s_{ik}, 0)}. \quad (1)$$

Each peer i has global Badness

$$negT_i = \sum_{j=1}^n negC_{ji} * T_j,$$

which is obtained by propagating the negative opinion values by the positive opinion matrix. Global badness is therefore converging to a steady state since the EigenTrust vector \mathbf{T} does.

The Badness of peers in threat models A, B, B α and D is shown in Figure 1(a). Honest peers have Badness greater than 0 since they supply corrupted files in 2% of the cases. Spy peers of attack D always have Badness values equal to 0 because of the strong assumption that they never provide corrupted files. *Badness fails to detect spy peers*. For this reason we introduce a second notion called *Dishonesty*.

5.2 Dishonesty

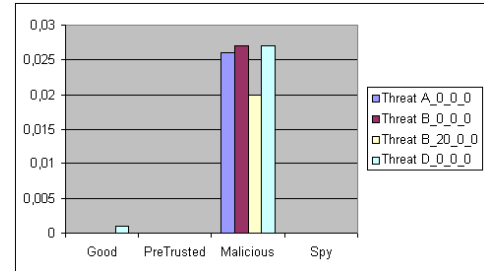
Dishonesty is meant to detect those peers that lie in reporting their opinion on the service received from other peers. These are either peers conferring trust to malicious peers or peers conferring distrust to honest peers. The spy peers of threat model D are an example of peers that confer trust to malicious peers.

Let P_i be the set of peers that i has positively assessed. The notion of *Positive Dishonesty* is defined as

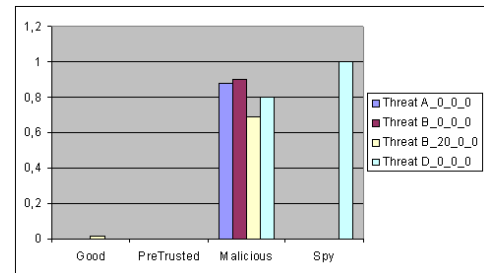
$$Dishonesty_i^+ = \sum_{j \in P_i} negT_j.$$

Let N_i be the set of peers that have been negatively assessed by peer i . The notion of *Negative Dishonesty* is defined as

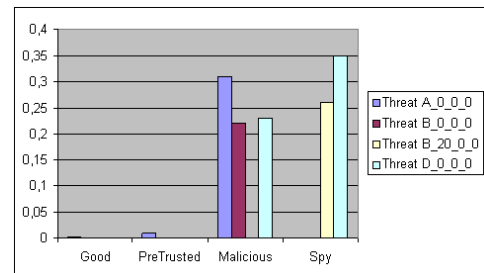
$$Dishonesty_i^- = \sum_{j \in N_i} T_j.$$



(a)



(b)



(c)

Figure 1: (a) Badness (b) Positive Dishonesty (c) Negative Dishonesty

Positive Dishonesty is high for those peers that assess positively peers with high Badness. Negative Dishonesty is high for those peers that assess negatively peers with high EigenTrust. As shown in Figure 1(b) and Figure 1(c), this measure is able to detect peers that are cheating in all the attacks presented in [9].

6. INTEGRATION ALGORITHMS

In this section we present the integration of the metrics based on negative opinions into the reputation mechanism defined by EigenTrust. EigenTrust implements a probabilistic choice to select the peer to transact with. Let $\{T_1, T_2, \dots, T_R\}$ be the global trust values of the peers that answer a query. The probability sel_j^{Eig} that peer j is selected is defined as follows:

$$sel_j^{Eig} = \begin{cases} 90\% * \frac{T_j}{\sum_{i=1}^R T_i} & \text{if } T_j > 0 \\ 10\% * \frac{1}{T_j^0} & \text{if } T_j = 0 \end{cases}$$

where T_j^0 is the number of peers with $T_j = 0$. Probabilistic selection is meant to balance the load in the network and to introduce redundancy in the selection of peers. Negative opinion metrics should be used very carefully and in combination with a global trust metric. The reason is that good peers also have a non-zero value of Badness and Dishonesty. A selection strategy based only on negative opinion metrics not integrated with a global trust metric might lead to a large number of false positives.

6.1 Integration of Badness with EigenTrust

We first show how to combine EigenTrust with Badness in order to reduce the probability that a peer with high Badness is selected as source for downloads. The basic idea is to classify a peer as malicious if it has Badness value above the *Average Badness* $AB = \frac{1}{R} \sum_i negT_i$. Peers apply the selection strategy on a set $\{T'_1, T'_2, \dots, T'_R\}$ of new global trust values

$$T'_j = \begin{cases} h_{bad} & \text{if } badness_j - d_{bad} * T_j > c_{bad} * AB \\ T_j & \text{otherwise} \end{cases}$$

with $h_{bad} \in \{-1, 0\}$ (a discussion between the two cases follows later in the section), $c_{bad} \in [0, 1]$ and $d_{bad} \geq 1$. The modified selection strategy is:

$$sel_j^{Badness} = \begin{cases} 0\% & \text{if } T'_j = -1 \\ 90\% * \frac{T'_j}{\sum_{i=1}^R \max\{T'_i, 0\}} & \text{if } T'_j > 0 \\ 10\% * \frac{1}{T_j^0} & \text{if } T'_j = 0 \end{cases}$$

Parameter c_{bad} controls the activation of the selection process. The number of true positives critically depends on the value chosen for c_{bad} . The lower it is, the higher is the number of true positives. Whenever we consider peers with the same value of Badness, we also need to penalize peers with low global trust more than peers with high global trust. With this aim we classify peers as malicious if the difference between the Badness value and d_{bad} times the global trust is higher than the Badness threshold. A suitable tuning of d_{bad} is very effective for reducing the number of false positives in the system, especially when the number of malicious peers in the system is small.

The value chosen for h_{bad} affects the number of false positives and the percentage of inauthentic downloads. If $h_{bad} = 0$, honest peers erroneously classified as malicious still have 10% of probability to be chosen and prove their goodness, but also malicious peers classified as malicious have 10% of probability of uploading an inauthentic file. The net result is a lower number of false positives and a higher number of inauthentic downloads. If $h_{bad} = -1$ we ignore all peers marked as malicious, i.e., $T'_j = -1$, as source of download. The drawback of this second alternative is that good peers erroneously classified as malicious are prevented from uploading files.

This results in higher number of false positives but fewer inauthentic downloads.

In both cases, this selection strategy results in a very high number of false positives when only very few malicious peers are in the system. A good indicator of the percentage of malicious peers in the system is actually given by the ratio between the AB and the *average global trust* $AGT = \frac{1}{R} \sum_i T_i$. In order to contrast this problem we inactivate the selection strategy with Badness once this ratio is below a value that we call r_{bad} . The final selection strategy is:

$$sel_j^{Badness} = \begin{cases} sel_j^{Eig} & \text{if } \frac{AB}{AGT} < r_{bad} \\ sel_j^{Badness} & \text{if } \frac{AB}{AGT} \geq r_{bad} \end{cases}$$

We observe that the new global trust values T' exist only locally at each peer during the selection phase; they don't replace the global trust values T in the system, although they affect the way the global trust is computed.

6.2 Integration of Dishonesty with EigenTrust and Badness

In this paragraph we present the integration of Dishonesty with EigenTrust and Badness. The goal is to reduce the probability that a peer with high Dishonesty is selected as source for downloads. We present here a first variant of the algorithm.

6.2.1 Integration with Exact Mean Global Dishonesty Value

In this case we use the average Dishonesty of all peers in the network to set the threshold. Moreover, we inactivate the use of the Dishonesty when the deviations DEV_{dis}^+ and DEV_{dis}^- are lower than some specific value d^+ and d^- , respectively, in order to avoid a large number of false positives.

$$T_j''^+ = \begin{cases} -1 & \text{if } Dishonesty_j^+ > c_{dis}^+ * AD^+ \text{ and} \\ & DEV_{dis}^+ \geq d^+ \\ T_j' & \text{otherwise} \end{cases}$$

$$T_j''^- = \begin{cases} -1 & \text{if } Dishonesty_j^- > c_{dis}^- * AD^- \text{ and} \\ & DEV_{dis}^- \geq d^- \\ T_j' & \text{otherwise,} \end{cases}$$

with $c_{dis}^+ \geq 1$ increasing function of $\frac{AD^+}{AGT}$ and $c_{dis}^- \geq 1$ decreasing function of $\frac{AD^-}{AGT}$.

The selection strategy is

$$sel_j^{Dishonesty} = \begin{cases} sel_j^{Badness} & \text{if } DEV_{dis}^+ < d^+ \\ & \text{and } DEV_{dis}^- < d^- \\ 0\% & \text{if } T_j'' = -1 \\ 90\% * \frac{T_j''}{\sum_{i=1}^R \max\{T_i'', 0\}} & \text{if } T_j'' > 0 \\ 10\% * \frac{1}{T_j^0} & \text{if } T_j'' = 0 \end{cases}$$

6.3 Clustering

In order to further improve the number of inauthentic downloads and, at the same time, decrease the rate of

false positives, we also propose the use of negative opinion metrics as features of an automatic clustering tool able to discern between good and malicious peers. The first approach consists in using three metrics (*Badness*, *Dishonesty*⁺ and *Dishonesty*⁻) as coordinates of an unsupervised classification algorithm that identifies the clusters $C_{malicious}$ and C_{good} . We adopt the well-known and widely used K -means algorithm [8] that can also be implemented in a decentralized fashion [5]. Once the clusters are computed, modified global trust values are defined as follows:

$$T_j^{cl} = \begin{cases} -1 & \text{if } j \in C_{malicious} \\ T_j & \text{if } j \in C_{good} \end{cases}$$

The cluster of malicious peers is easily separated from the cluster of good peers since the corresponding centroid holds the higher value of Dishonesty or Badness. The candidate peers for the download are only chosen within C_{good} . The modified version of the Probabilistic selection strategy is shown below:

$$sel_j^{clustering} = \begin{cases} sel_j^{Badness} & \text{without clusters} \\ 0\% & \text{if } T_j^{cl} = -1 \text{ and clusters} \\ 90\% * \frac{T_j^{cl}}{\sum_{i=1}^R T_i^{cl}} & \text{if } T_j^{cl} > 0 \text{ and clusters} \\ 10\% * \frac{T_j^{cl}}{\sum_{i=1}^R T_i^{cl}} & \text{if } T_j^{cl} = 0 \text{ and clusters} \end{cases}$$

The clustering algorithm is effective against threat models A, B, D and $B\alpha$, but its performances decrease with the increase of the degree of camouflaging behind positive and negative judgments. In this case in fact the average value of both types of Dishonesty decreases with the increase of the degree of camouflaging in the system. In the extreme case in which malicious peers always assess positively good transactions, Positive Dishonesty is only due to good peers, thus resulting in an incidental rise of false positives. A similar argument also applies to Negative Dishonesty. We therefore define the following adaptive rules in order to activate the use of Positive and Negative Dishonesty as clustering features:

$$Featured_{dis+} = \begin{cases} 1 & \text{if } \frac{AD_{nospy}^+}{AGT} > h_d^+ \\ 0 & \text{otherwise} \end{cases}$$

and

$$Featured_{dis-} = \begin{cases} 1 & \text{if } \frac{AD_{nospy}^-}{AGT} > h_d^- \\ 0 & \text{otherwise} \end{cases}$$

where AD_{nospy}^+ and AD_{nospy}^- , respectively, denote the average positive and negative Dishonesty computed without the contribution of the spies as defined in attack D, AGT the average global trust in the network, and h_d^+ , h_d^- are suitable constants. When we drop both Positive Dishonesty and Negative Dishonesty as features of the clustering, the clustering itself is no longer effective. In this case, we directly resort to the selection strategy described in Section 6.1 that uses the only integration of Badness with EigenTrust.

6.4 How to choose the integration algorithm

We have presented two methods to integrate the global trust with negative opinions:

| T | | C | |
|-----------|------|-----------|-----|
| h_{bad} | 0 | h_{bad} | -1 |
| d_{bad} | 5 | d_{bad} | 5 |
| c_{bad} | 0.5 | c_{bad} | 0.5 |
| r_{bad} | 0.5 | r_{bad} | 0.5 |
| d^+ | 0.05 | h_d^+ | 2 |
| d^- | 0.05 | h_d^- | 1.5 |

Table 1: Integration algorithms parameters used for T and C

| Malicious | Good | PreTrusted | Malicious | Spy |
|-----------|------|------------|-----------|-----|
| 5% | 100 | 5 | 6 | 1 |
| 10% | 100 | 5 | 12 | 2 |
| 25% | 100 | 5 | 35 | 4 |
| 50% | 100 | 5 | 105 | 12 |
| 70% | 100 | 5 | 245 | 24 |

Table 2: Network configurations used in the simulations

- Integration with Badness and Clustering;
- Integration with Exact Mean Global Dishonesty Value

The two methods are characterized by a clear trade-off between true positives and false positives. While the method based on clustering is computationally more expensive, it is the one that guarantees the lowest number of false positives. The threshold-based methods for integrating Dishonesty are more flexible thanks to the possibility to set the thresholds.

7. THE SIMULATION PLATFORM

We briefly describe the simulation platform proposed in [12] and adopted in [9] for evaluating the performances of EigenTrust. The simulator [12] is based on a P2P network model with peers sending and responding queries for files, and files transferred between peers to conclude a search query. Queries are broadcasted with usual Gnutella-like hop-count horizon. Peers which receive a query forward it and check if they are able to answer it. Peers are interconnected by a power-law network to model the topology observed in real-world P2P networks.

The content distribution model is defined by a probabilistic interaction between peers. Peers are assumed to be interested in a subset of the total available content in the network, i.e., each peer initially picks a number of content categories and shares files only in these categories. We also assume that files with different popularities exist within one content category. Popularity is governed by a Zipf distribution. When the simulator generates a query, it draws from a Zipf distribution the category and rank (or degree of popularity) of the file that will satisfy the query. Each peer that receives the query checks if it supports the category and if it shares the file. Files are distributed between peers at the initialization with a probabilistic process based on the content categories assigned to the peers and on the popularity of the files. We use exactly the same setting proposed in [12].

7.1 Simulation execution

The simulation proceeds in cycles. Each simulation cycle is subdivided into a number of query cycles. In each query cycle, a peer i may be *actively issuing a query*, *inactive*, or even *down* and not responding to queries passing by. Upon issuing a query, a peer waits for incoming responses, selects a download source among those nodes that responded, and starts downloading the file. The latter two steps are repeated until the peer has properly received a good copy of the file. Upon the conclusion of each simulation cycle, the global trust value computation is kicked off. Each experiment is run several times and the results of all runs are averaged, until there is convergence to a steady state. The initial transient states of the simulation are excluded from the results.

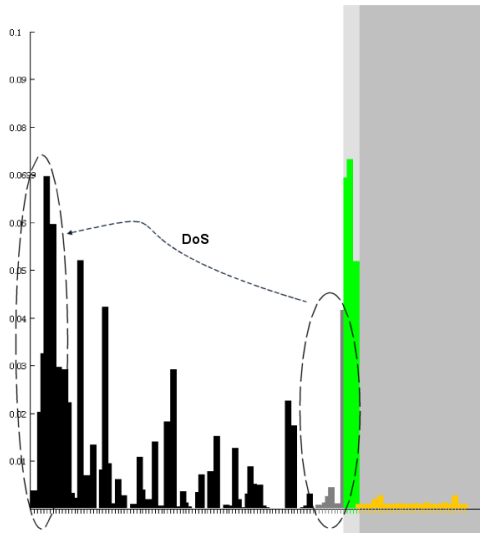


Figure 2: Global Trust Values of peers in Dos

8. EXPERIMENTAL RESULTS

The experimental evaluation has been done using the simulation platform introduced in [12] and briefly described in Section 7. We simulate the behavior of all our algorithms over a network with 100 good peers and 5 pre-trusted peers. In all the possible scenarios we consider, the good peers are assigned with a probability of supplying corrupted files equal to 2%.

For all the threat models, we compare our algorithms:

- Integration with Badness and Clustering (C)
- Integration with Exact Mean Global Dishonesty Value (T)

against EigenTrust (E). The evaluation is based on the comparison of the percentage of inauthentic downloads. For our algorithms we also consider the percentage of True Positives and False Positives of malicious peers (Colluding and Spy).

For each algorithm presented in Section 6, we run a total number of 6 simulations of 50 cycles each. We

consider the average ratio between the number of inauthentic downloads and the total number of downloads, the average percentage of true positives and false positives. The first 15 cycles of each run are considered initial transient states and are excluded from the data.

We run different simulations varying the fraction of malicious peers from 5% to 70% and using different combinations of α , β and γ . We consider only two values for α , that is $\{0\%, 20\%\}$. As shown in [9], malicious peers would not have any advantage considering values greater than 20%, which indeed are counterproductive for them. For all the models we let β, γ vary in 0%, 33%, 66%, 100%. We use the notation $X_{i,j,k}$ to indicate the attack $X \in [A, B, D]$ with $\alpha = i$, $\beta = j$ and $\gamma = k$.

Due to space constraints, we do not report on all the cases we investigate. We just show the results for the “worst cases”, i.e. for each attack $X \in [A, B, D]$, we plot the results for the combinations of features that are more difficult to handle. In each plot, we use the following metrics (reported on the vertical axes as **evaluated metrics**):

- **inauthentic downloads**: visualized with continue lines (blue for E, red for T and green for C)
- **true positives**: visualized with dotted lines (blue for E, red for T and green for C)
- **false positives**: visualized with dashed lines (blue for E, red for T and green for C)

We can observe the following:

- With respect to the inauthentic downloads rate, all our methods always outperform EigenTrust with the only exception of the attack $A_{0,1,0,1,0}$ for a percentage of malicious peers equal to the 70% of the global number of peers in the system.
- For threat models B and C, all the methods are able to achieve nearly zero percentage of inauthentic files downloads.

The performances are more sensitive to variations of α rather than of β and γ . When $\alpha = 0$, we observe, in general, better performances in terms of true and false positive rates (respectively close to 1.0 and 0.0). The attacks $A_{0,0,0,66,1,0}$ and $A_{0,0,1,0,1,0}$, shown in Figure 3, are the most difficult ones for our algorithms. Nevertheless, even in these extreme cases, the inauthentic downloads never exceed 10%. In the case of $\alpha = 0.2$, we experiment better performances in terms of inauthentic downloads but worse true and false positive rates. We want to remark that all the cases not shown in the Section ?? are characterized by true and false positive rates close to the ideal case.

It is interesting to observe how the integration of negative opinions with EigenTrust is, at some extent, less effective on attack A, the one that should in principle be the easiest to handle. The reason lies in the absence of colluding malicious peers that report untruthful judgments on the other peers in the coalition. Camouflaging

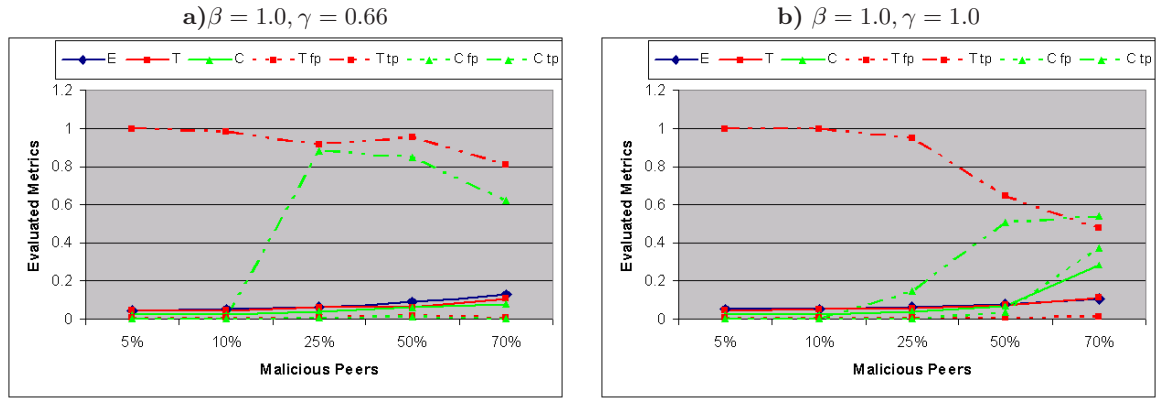


Figure 3: Threat attack A with $\alpha = 0.0$.

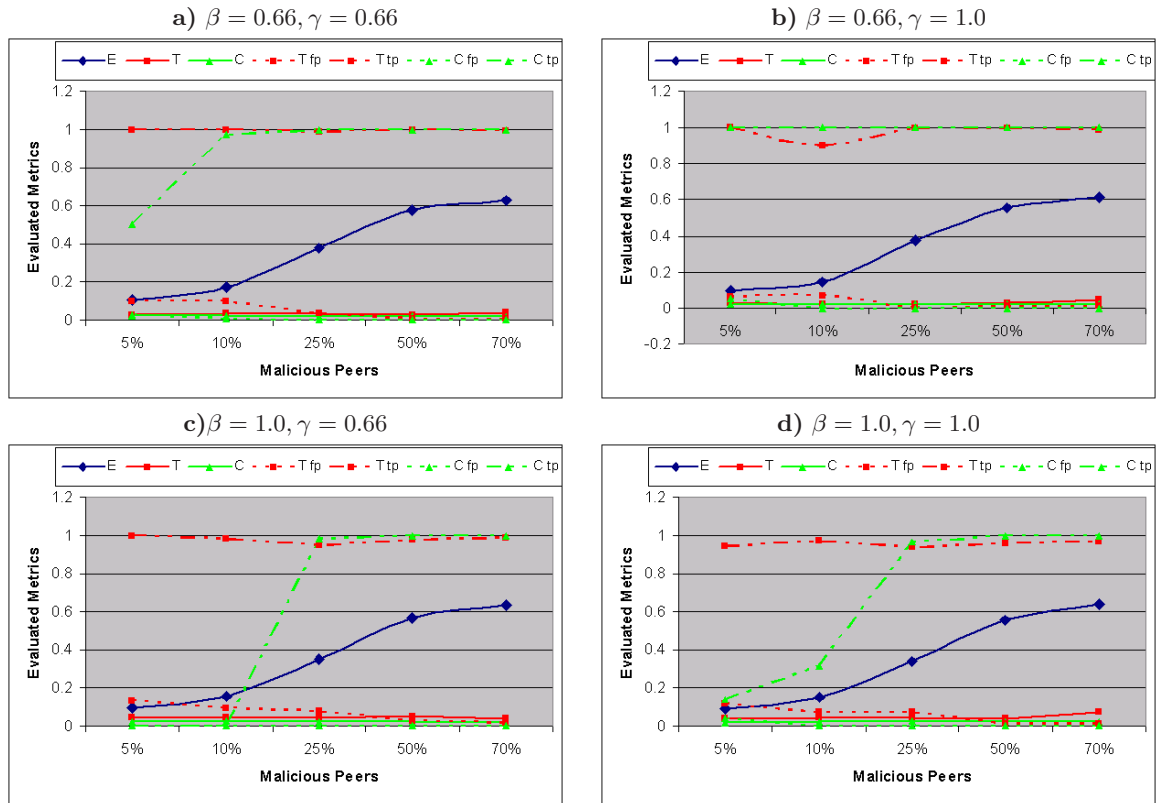


Figure 4: Threat attack A with $\alpha = 0.2$.

behind positive opinions conferred to good peers is in this case a rather challenging attack given that malicious peers are hard to detect since they do not positively assess other malicious peers. Algorithms C and T definitely outperform EigenTrust in terms of the percentage of inauthentic files that are downloaded on attacks B and D combined with different threat features. Dishonesty is able to single out, after few cycles, all the peers that are lying in assessing the other peers. Hence this method can penalize in the attack D both the malicious peers and the spies.

Positive and negative opinions are normalized during

the computations. This means that peers only have got one unit of positive and negative opinions to distribute. In the attacks examined so far, malicious peers spread this unit uniformly over all good peers. A different type of threat can be operated by malicious peers whenever they concentrate their negative opinions on a subset of good peers. This threat is known as Denial of Service (DoS). If simple malicious peers organize this threat, their attempt will be useless, since they have low global trust value, that is, low credibility. We investigate what would happen if spy peers with high credibility, organize the DoS. Our simulations show that DoS is ineffective,

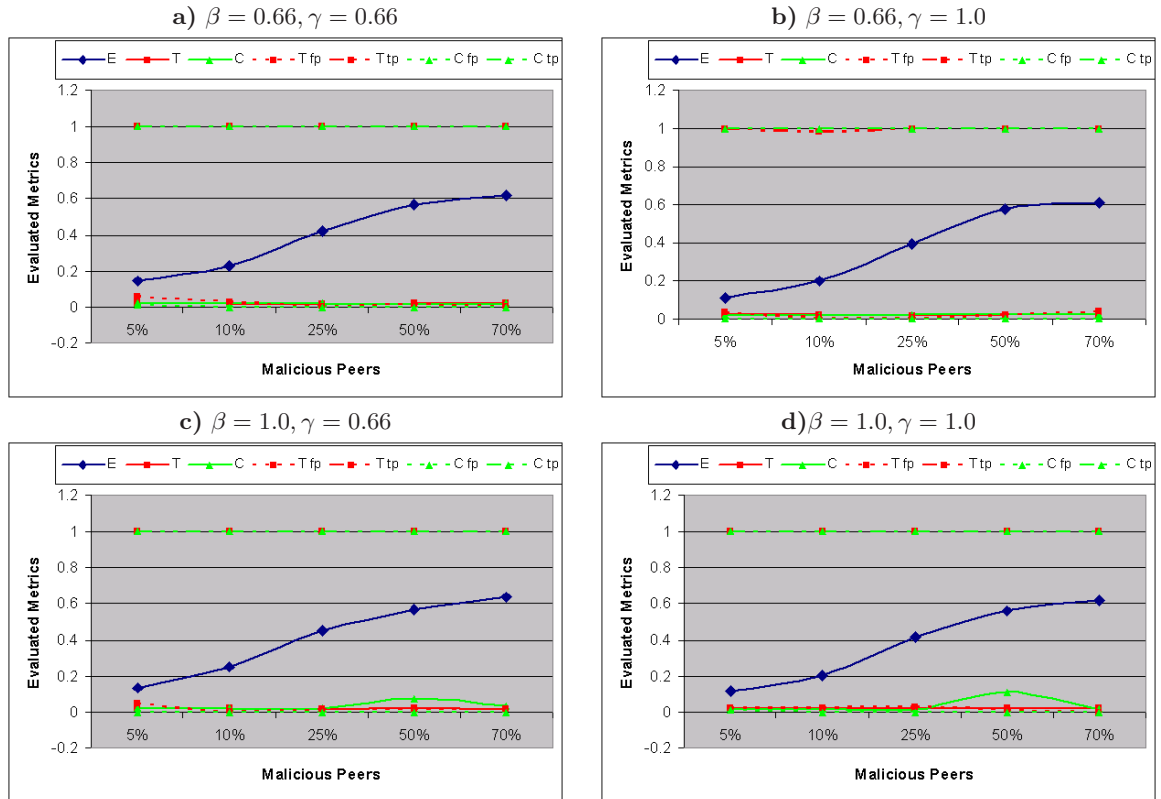


Figure 5: Threat attack B with $\alpha = 0.2$.

even when a large number of spies act against a small number of malicious. Figure 2 shows global trust values of all the peers in a simulated network. The global trust of good, malicious peers and the new type of spy, realizing the DoS attack, are respectively in black, yellow and grey. As indicated, good peers under DoS (in the left circle) are not prevented from reaching high global trust values, even higher than the global trust of spies acting the DoS (in the right circle).

9. CONCLUSIONS

In this work we propose more effective strategies for reputation management in decentralized environments based on the integration of the notion of transitive trust defined by EigenTrust [9] with metrics that aggregate the negative opinions expressed between peers. Simulations prove that our approach is very effective and to outperform EigenTrust on several sophisticated attacks coming from coalitions of malicious peers. We also propose an approach based on the use of negative opinion metrics as features of a clustering tool able to classify peers into good and malicious peers with a high rate of true positives and very few false positives.

Our approach consists of propagating negative opinions through positive trust relationships between peers. This is a methodological sound approach. It is unclear how to adapt other recent proposals along this line to our setting [7]. Reputation management has induced an increase in the number of applications for recommenda-

tion systems, ranking users and content. The growth of applications affected all kind of decentralized environments. We believe that the methods presented in this paper are suitable to provide novel solution concepts for managing trust and reputation in on-line social networks and systems for collaborative ranking. An interesting problem left open from this work is showing a formal theoretical setting that our approach always outperforms EigenTrust on all reasonable attacks from coalitions of malicious peers.

10. REFERENCES

- [1] <http://ewwws.com/pr/przero.php>. PR0 - Google's PageRank 0 Penalty.
- [2] Z. Abrams, R. Mcgreg, and S. Plotkin. A non-manipulable trust system based on eigentrust, July 2005.
- [3] R. Baeza-Yates, P. Boldi, and C. Castillo. Generalizing PageRank: Damping functions for link-based ranking algorithms. In *Proc. of SIGIR*, Seattle, Washington, USA, August 2006. ACM Press.
- [4] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Using rank propagation and probabilistic counting for link-based spam detection. In *Proc. of WebKDD*. ACM Press, August 2006.
- [5] Souptik D., Chris G., and Hillol K. K-means clustering over a large, dynamic network. In *Proc.*

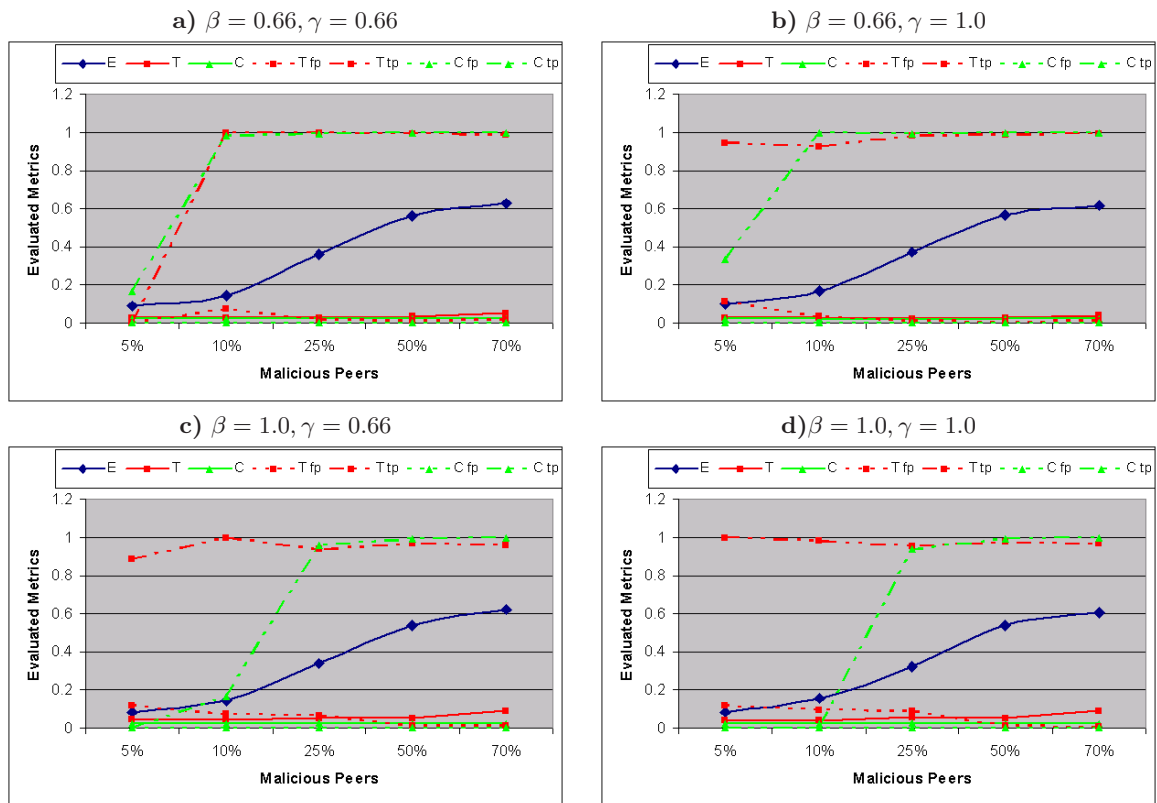


Figure 6: Threat attack D with $\alpha = 0.2$.

- of the Siam Conference on Data Mining, SDM, 2006.
- [6] D. Donato, M. Panicia, M. Selis, C. Castillo, G. Cortese, and S. Leonardi. New metrics for reputation management in p2p networks. In *Proc. of AIRWeb*, volume 215, pages 73 – 80, 2007.
 - [7] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Proc. of WWW2004*, 2004.
 - [8] J. A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 1975.
 - [9] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *In Proc. of WWW '03*, pages 640–651. ACM Press, 2003.
 - [10] S. Marti and H. Garcia-Molina. Taxonomy of trust: Categorizing p2p reputation systems. *Computer Networks*, 50(4):472–484, March 2006.
 - [11] Xavier J. Parreira, D. Donato, S. Michel, and S. Weikum. Efficient and decentralized pagerank approximation in a peer-to-peer web search network. In *32 nd International Conference on Very Large Data Bases*.
 - [12] M. Schlosser and S. Kamvar. Simulating a p2p file-sharing network, 2003.
 - [13] Kevin Walsh and Emin G. Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *NSDI'06: Proceedings of the 3rd conference on 3rd Symposium on Networked Systems Design & Implementation*, pages 1–1, Berkeley, CA, USA, 2006. USENIX Association.
 - [14] L. Xiong and L. Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transaction On Knowledge and Data Engineering*, 16(7):843–857, July 2004.
 - [15] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. In *Proc. of the Conference on Applications, technologies, architectures, and protocols for computer communications*, pages 267–278. ACM Press, 2006.