Personalized Reputation Management in P2P Networks

Paul - Alexandru Chirita[1], Wolfgang Nejdl[1], Mario Schlosser[2], and Oana Scurtu[1]

[1] L3S Research Center / University of Hannover
Deutscher Pavillon Expo Plaza 1
30539 Hannover, Germany
{chirita,nejdl,scurtu}@l3s.de
[2] McKinsey & Company Inc. / Stanford University
schloss@db.stanford.edu

Abstract. P2P networks have become increasingly popular in the recent years. However, their open, distributed and anonymous nature makes them very vulnerable against malicious users who provide bad responses to requests from other peers. Motivated by this observation, various solutions for distributed reputation systems have been presented recently. In this paper, we describe the first reputation system which incorporates both user-individual personalization and global experiences of peers in the network, for the distributed computation of reputation values. We also present a secure method to compute global trust values, thus assuring identification and isolation of malicious peers. Finally, our simulations show that our system is robust even against attacks from groups of malicious peers deliberately cooperating to subvert it.

1 Introduction

P2P networks are powerful distributed infrastructures allowing any peer to search for and offer content and services. They are capable of handling enormous amounts of resources while maintaining an organized and balanced topology. However, because of their open nature, they also require more complex reputation schemes, as malicious users can introduce corrupt data and/or harmful services much easier than in centralized systems. Such reputation algorithms are usually based on aggregating the trustworthiness information collected by each peer with that of several or all other peers, thus generating a micro or macro *web of trust*.

As the size of current P2P networks is continuously increasing, recent research has concentrated on designing more personalized trust management algorithms, while also addressing their robustness against attacks of malicious peers. Some of these techniques employ only local computations on sub-graphs of the entire P2P network [15, 9]. Their output is personalized at the cost of not considering global experiences of peers. [11] tackles this, but only for statements from the Semantic Web. Finally, techniques which include experiences of all peers [8] do not address the issue of personalization.

In this paper we introduce a novel algorithm for computing personalized reputation values in P2P networks and test its robustness against several possible attacks in a simulated P2P environment. We use a real-world experimental setup simulating power-law distributions on peer links as well as content distribution, as can be observed in many current P2P networks.

We start with a short introduction of algorithms related to our research in Section 2. The distributed computation of personalized reputation values is introduced in Section 3, and then extended into a secure version in Section 4. Section 5 presents our experimental results, section 6 contains our conclusions.

2 Background and Previous Work

2.1 Trust and Reputation

Although the area of reputation algorithms has received increased attention recently, some issues are still left open. [15] gives a very good overview over existing approaches. The paper contains the first categorization of trust metrics and a definition of trust elements (model, metrics, etc.) in the Semantic Web. Its main contribution is *Appleseed*, a fixed-point personalized trust algorithm inspired by spreading activation models. Similarly, an important aspect for us is the introduction of ”backward trust propagation” (i.e. virtual edges from every visited node x to the computation seed node s), which solves several rank normalization problems (e.g. distinguish between a peer with whom some peer i did not interact and a peer with whom i had bad experiences).

[11] builds a Web of trust, with each user having to maintain trust values on a small number of other users. The algorithm presented is designed for an application within the context of the Semantic Web, composed of logical assertions. This helps introducing personalization, as each user would have a level of local belief in statements and a level of trust in other users, all of which could then be merged to reflect a global meaning. [9] presents an interesting method based on a quorum of other peers, who are asked about their opinion on some peer p, instead of relying on a fixed-point algorithm. This results in reduced network traffic, but comes at the cost of not achieving a global perspective on the trustworthiness of peer p. A related approach is presented in [5], where the FOAF [3] schema is extended to contain trust assertions between peers. The inferred rating from a source peer to a sink one is (recursively) computed using the weighted average of the neighbors' reputation ratings of the sink. [8] is a fixed-point PageRank-like distributed computation of reputation values in a P2P network. We used its model in designing our algorithm, as well as the investigations about possible attacks from malicious peers. Just as [7] improves [10], our algorithm extends the capabilities of [8] by introducing personalization into the computation.

For global ranking algorithms, we distinguish three targets based on fixed-point iterations: (1) Web pages [10, 7, 6], (2) linked documents in P2P networks [12, 14, 1], and (3) peers in P2P networks [8]. In all cases, input data can be represented as a directed graph with the targets of the algorithm as nodes. For ranking Web pages or documents, graph edges are the hyperlinks, whereas for building reputation values they resemble peers' experiences with other peers. These approaches are summarized in table 1.

2.2 Personalized PageRank

Description. [7] is the most recent investigation towards personalized page ranks. We give a more detailed description of this algorithm in the following paragraph, as we will extend this algorithm to compute personalized trust values in a distributed way.

Outcome	Node in graph	Edge in graph
Web page ranks [10]	Web page	Hyperlink
Personalized Web page ranks [7, 6]	Web page	Hyperlink
Document ranks (distributed) [12, 14]	Document on a peer	Hyperlink between two documents
Personalized document ranks (distributed) [1]	Document on a peer	Hyperlink between two documents
Reputation values (distributed) [8]	Peer	Download experience of peers
Personalized reputation values (distributed) - this paper	Peer	Download experience of peers

Table 1. Hyperlink structures used by different ranking algorithms

[7] introduces personalized PageRank Vectors (PPV) computed for each user. The personalization aspect of this algorithm stems from a *set of hubs* (H), and each user has to select her *preferred pages* from this set. PPVs can be expressed as a linear combination of basis vectors (PPVs for preference vectors with a single non-zero entry corresponding to each of the pages from P, the preference set), which could be selected from the precomputed basis hub vectors, one for each page from H. To avoid the massive storage resources basis hub vectors would use, they are decomposed into partial vectors (which encode the part unique to each page, computed at run-time) and the hub skeleton (which captures the interrelationships among hub vectors, stored off-line).

Algorithm. In the first part of the paper, the authors present three different algorithms for computing basis vectors: ”Basic Dynamic Programming”, ”Selective Expansion” and ”Repeated Squaring”. In the second part, specializations of these algorithms are combined into a general algorithm for computing PPVs, as depicted below.

Algorithm 1. Personalized PageRank in a centralized fashion.

Let $D[p]$ be the approximation of p’s basis vector, and $E[p]$ the error of its computation.
1.(Selective Expansion) Compute the partial vectors using
$Q_0(p) = V$ and $Q_k(p) = V \setminus H$, for $k > 0$, in the formulas below:

$$\mathbf{D_{k+1}}[\mathbf{p}] = \mathbf{D_k}[\mathbf{p}] + \sum_{q \in Q_k(p)} c \cdot E_k[p](q)\mathbf{x_q}$$

$$\mathbf{E_{k+1}}[\mathbf{p}] = \mathbf{E_k}[\mathbf{p}] - \sum_{q \in Q_k(p)} E_k[p](q)\mathbf{x_q} + \sum_{q \in Q_k(p)} \frac{1-c}{|O(q)|} \sum_{i=1}^{|O(q)|} E_k[p](q)\mathbf{x}_{O_i(q)}$$

Under this choice, $D_k[p] + c * E_k[p]$ will converge to $\mathbf{r_p} - \mathbf{r_p^H}$,
the partial vector corresponding to page p.

2.(Repeated squaring) Having the results from the first step as input, one can now compute the hubs skeleton ($r_p(H)$). This is represented by the final $D[p]$ vectors calculated using $Q_k(p) = H$ into:

$$\mathbf{D_{2k}}[\mathbf{p}] = \mathbf{D_k}[\mathbf{p}] + \sum_{q \in Q_k(p)} E_k[p](q) * D_k[q]$$

$$\mathbf{E_{2k}}[\mathbf{p}] = \mathbf{E_k}[\mathbf{p}] - \sum_{q \in Q_k(p)} E_k[p](q)\mathbf{x_q} + \sum_{q \in Q_k(p)} E_k[p](q)E_k[q]$$

3. Let $u = \alpha_1 p_1 + \cdots + \alpha_z p_z$, $p_i \in H$, $i = 1, 2, .., z$, be a preference vector, and let:

$$r_u(h) = \sum_{i=1}^{z} \alpha_i (r_{p_i}(h) - c * x_{p_i}(h)),\ h \in H,$$ computable from the hubs skeleton.

The PPV v for u can then be constructed as:

$$v = \sum_{i=1}^{z} \alpha_i (r_{p_i} - r_{p_i}^H) + \frac{1}{c} \sum_{h \in H\ r_u(h) > 0} r_u(h) * \left[(\mathbf{r_h} - \mathbf{r_h^H}) - c * x_h \right]$$

3 Personalized Reputation Values in P2P Networks

3.1 Introduction

In our distributed algorithm for computing personalized reputation values we start from a set H of pre-trusted peers (called *hub peers* hereafter). Each peer will have its own preference set $P \subset H$. Even though all hub peers are highly trusted, each peer trusts some of them (those from P) more than it trusts the others. Generally, this happens because they provide better quality of service or faster and more downloads, in a specific area of interest. In a services network, for example, a peer might prefer the abilities of one or several hub peers, because they supply very good results in the specific service domain it is interested in.

What is the intuition behind personalized trust values in a P2P network, and, even more importantly, in which way does personalization of trust values improve on global trust rating systems, where only one trust value is established per peer, encompassing all other peers' interactions with this peer? The notion of personally selected pre-trusted peers gives an elegant answer: When a peer in the network selects a subset of pre-trusted peers which it trusts most, it does not necessarily consider these peers as more trustworthy than other pre-trusted peers - in fact, all pre-trusted peers should be entirely trustworthy, i.e., always strive to provide authentic file uploads or non- malicious services. Rather, a peer selects those pre-trusted peers whose trust ratings and experiences with peers in the network are most *relevant* to this peer's operations within the P2P network. The peer may operate in a content domain in which certain peers have provided seamless and perfectly trustworthy service - even though they would have a low global trust rating due to other peers being much more active in responding to popular queries. Hence, personalization of trust values does not only establish a web of trust, it creates personal webs of trust in which peers with similar interests cooperate to choose the most trustworthy peers in their peer group.

But what if a peer does not prefer *any* hub peer, but some other peer h'? There are several solutions which cope with this (see [2]), the simplest one being a breadth-first search around h' with the minimum radius such that at least a hub peer is discovered, thus selecting the hub peer(s) closest to h'.

We divide the algorithm in three parts, as presented in section 2.2: One part focuses mostly on peers outside the set of pre-trusted peers, $V \setminus H$ (Algorithms 3.1.1 and 3.1.2), one on peers within the set of pre-trusted peers H (Algorithm 3.2), and the final algorithmic step ties everything together (Algorithm 3.3).

3.2 Partial Vectors

This part of the algorithm consists of one special initialization step and several succeeding steps. Even though its focus is on peers from $V \setminus H$, peers $p \in H$ also gather their components of $\mathbf{D}$ and $\mathbf{E}$. All peers normalize their trust values as $\gamma_q(i) = \gamma_q(i)/\sum_i \gamma_q(i)$. In the first step, each peer $q \in V$ computes $E[p](q)$. As peers $p \in H$ are known, each peer $q \in V$ can set its initial values $D_0[p](q)$ and $E_0[p](q)$ by itself to:

$$D_0[p](q) = \begin{cases} c \ , q \in H \\ 0 \ , otherwise \end{cases} ; E_0[p](q) = T_0[p](q) = \begin{cases} 1 \ , q \in H \\ 0 \ , otherwise \end{cases} \quad (1)$$

Non-hub peers. After this initialization, the following operations will be executed in parallel by each peer $q \in V \setminus H$ for each $p \in H$:

Algorithm 3.1.1. Distributed computation of partial vectors by peers in $V \setminus H$.

```
1: Send $\frac{1-c}{|\mathbf{O(q)}|} \cdot T_k[p](q) \cdot \gamma_q(i)$ to all peers $i$ from which $q$ has
     downloaded files, including those from $H$.
2: Wait from all peers which downloaded files from $p$ their $T_k[p](*)$ (at this step $k$)
3: After all $T_k[p](*)$ values have been received, compute $T_{k+1}[p](q)$ as:
     $T_{k+1}[p](q) = \sum_{v \in \mathbf{I(q)}} T_k[p](v)$
4: Compute:
     $D_{k+1}[p](q) = D_k[p](q) + c \cdot T_k[p](q)$
     $N$ being the number of steps we apply the Selective Expansion algorithm.
5: If there are more iterations left, go to 1
6: Each peer $q \in V \setminus H$ computes $(r_p - r_p^H)(q) = D_N[p](q) + c \cdot T_N[p](q)$,
     its component of the partial vector corresponding to $p$.
```

Theorem 1. *In the distributed version of the Selective Expansion algorithm, for $p \in H$ each peer $q \in V \setminus H$ has to compute:*

$$T_{k+1}[p](q) = \sum_{v \in \mathbf{I(q)}} \frac{1-c}{|\mathbf{O(v)}|} \cdot E_k[p](v) \cdot \gamma_v(q) \tag{2}$$

Proof. Due to space limitations, we refer the reader to [1] for the proof of this theorem.

Hub peers. In the special first step, a peer $p \in H$ will send $\frac{1-c}{|\mathbf{O(q)}|} \cdot T_k[p](p) \cdot \gamma_p(i)$ to all peers i from which it has downloaded files, including those from H. After that, it will execute the following operations:

Algorithm 3.1.2. Distributed computation of partial vectors by peers in H.

```
1: Wait from all peers which downloaded files from $p$ their $T_k[p](*)$ (at this step $k$)
2: After all $T_k[p](*)$ values have been received, do $T_{k+1}[p](p) = \sum_{v \in \mathbf{I(q)}} T_k[p](v)$
3: If there are more iterations left, go to 1
4: Compute: $E[p](p) = \sum_{k=1}^{N} T_k[p](p)$
5: Set $D_N[p](p)$ to $c$
6: Each peer $p \in H$ computes $(r_p - r_p^H)(p)$, its component of its partial vector.
```

Algorithms 3.1.1 and 3.1.2 perform a power-iteration, and they would therefore converge also in the presence of loops in G (see [7, 10] for details), whereas the high level of dynamics of a P2P network (peers skipping participation in one or more iterations because entering or leaving the network, etc.) would only result in some additional iterations until convergence.

3.3 Hub Skeleton

In the second phase of the algorithm, each hub peer ($p \in H$) has to calculate its hub skeleton ($\mathbf{r_p}(\mathbf{H})$) using as input the results from the previous stage. The output is stored in the values $\mathbf{D_{2k}}[\mathbf{p}]$ obtained after the last iteration of the following operations:

Algorithm 3.2. Distributed computation of hub skeleton (only in H).

```
1: Calculate $\mathbf{D_{2k}}[\mathbf{p}]$ and $\mathbf{E_{2k}}[\mathbf{p}]$, using the centralized version formulas
2: Multicast the results to all other peers $q \in H$, possibly using a minimum
     spanning tree for that.
3: If there are more iterations left, go to 1.
4: Every hub peer broadcasts its $\mathbf{D_{2N}}[\mathbf{p}]$ sub-vector (only the components
     regarding pages from $H$).
```

As this step refers to hub-peers only, the computation of $\mathbf{D_{2k}}[\mathbf{p}]$ and $\mathbf{E_{2k}}[\mathbf{p}]$ can consider *only* the components regarding pages from H.

3.4 PPV

As the $\mathbf{D_{2N}}[\mathbf{p}]$ sub-vectors have been broadcast, *any* peer $v \in V$ can now determine its $\mathbf{r_v}(\mathbf{P})$ locally, using the original formula (see sec. 2.2). It can also calculate its partial PPV containing its reputation and the reputation of any other peers from its own point of view. If *NB* is the set of peers whose rank v wants to compute, it must do the following:

Algorithm 3.3. Computation of the Personalized Reputation Vector.

```
1: Request the components of $\mathbf{r_p} - \mathbf{r_p^H}$ for all $p \in H$ from all peers from $NB$.
2: Compute the components of the PPV using the original formula (see section 2.2).
```

Of course, if later v wants to compute the reputation value of another peer, it would only need to ask the new peer about its components of $\mathbf{r_p} - \mathbf{r_p^H}$.

4 Secure Computation of Personalized Reputation Values

From a security point of view, the algorithm as described above contains two critical components. First, pre-trusted peers play an important role in computing trust values. Hence an implementation of the algorithm has to make sure that they behave correctly. Fortunately, only very few pre-trusted peers are required in a network (see also Section 5) such that the administrative task of ensuring this behavior will present little overhead. Second, non-pre-trusted peers calculate trust values for themselves, or at least contribute to their calculation. This gives each peer major power over his own trust rating. A secure version of the algorithm thus has to ensure that malicious peers in the network have limited or no impact on the computation of their own trust ratings.

We achieve this in two steps: First, by having another, deterministically chosen peer in the network take over a peer's calculation job on his own trust value, becoming this peer's proxy calculation peer. Second, by adding redundancy to the calculations through having each calculation being performed by several peers in parallel. Each peer can be queried for the results of his calculations, and the final result of a calculation is determined through a majority vote among the collaborating peers.

We organize the network into a distributed hash table, using a scheme such as Chord [13]. Each peer's IP address is mapped into a logical hash space. Through random assignment upon joining the network, each peer covers a part of the hash space, becoming the proxy calculation peer for all peers whose network address has been mapped into its domain in the hash space. Several proxy peers can be associated with a peer by applying several different hash functions. Each hash function is deterministic, i.e., each peer in the network can infer the location of a proxy calculation peer on the DHT grid and route requests for calculation results appropriately.

5 Experimental Results

We tested our algorithms on simulated P2P networks which exhibit a power-law connectivity distribution, because popular real-world networks (e.g. Gnutella [4]) are structured as such. Peers were selected as download source with a probability proportional to their rank: Due to the power-law distribution, highly ranked peers were selected much more often than other peers, which reflects real-world P2P traffic distributions. Unless stated differently, the set of pre-trusted peers contained 5 (hub) peers, and the preference set of each ordinary peer contained 2-3 hub peers randomly selected. We always ran algorithms 3.1.* for 6 iterations, and algorithm 3.2 for 10 iterations, as in [7].

Resources used. We first generated a graph with 10,000 nodes in order to assess the amount of network traffic required by the algorithm. The set of pre-trusted peers contained 150 (hub) peers, while the preference set of each ordinary peer contained 30 hub peers randomly selected. Results are summarized in tables 2 and 3.

In our statistics, we distinguish between ordinary peers and hub peers. Hub peers generate significantly more traffic because of the second phase of the computation (algorithm 3.2), in which they need to multicast their intermediate results to all other hub peers. However, there is usually a small number of such peers compared to the size of the network, and they also control more resources (e.g. bandwidth or processing power).

	Max. Size	**Avg. Size**
All Data	1,989,072	1,979,695
$\neq 0$ **Data**	1,944,764	1,938,796

Table 2. Data transferred by hub peers (nb. of real values sent)

	Max. Size	**Avg. Size**
All Data	101,232	4,231.3
$\neq 0$ **Data**	5,360	84.69
All Data, 1 iteration	665	27.91
$\neq 0$ **Data, 1 iteration**	171	2.043

Table 3. Data transfered by ordinary peers (nb. of real values sent)

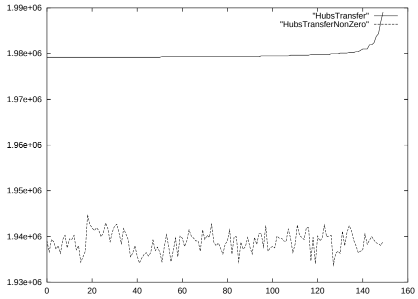

Fig. 1. Data transfered by hub peers (nb. of real values sent)

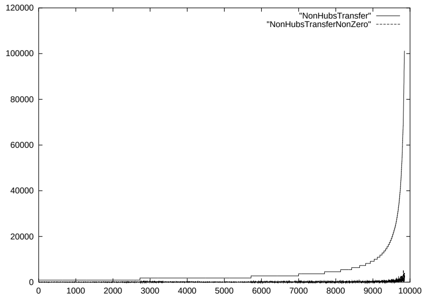

Fig. 2. Data transfered by ordinary peers (nb. of real values sent)

We observed only reduced communication for ordinary peers, which means that for the majority of the network, the computation needs only very small bandwidth. We also noticed that many peers often send the value ”0” through the network (as a correct intermediate value), and therefore a further modification of the algorithm which avoids sending it would reduce the traffic even more. Figures 1 and 2 present the traffic generated by each peer in the network. As we are dealing with a power-law network, we can observe the power-law distribution of data size among the ordinary peers. This is because the more neighbors a peer has, the more data it needs to exchange.

Robustness. An important aspect of a reputation algorithm is to be robust against attacks of malicious peers, which try to subvert the network by uploading inauthentic files or providing faulty services. For these tests, we simulated possible attacks as described in [8]. As our algorithm is also based on a power-iteration, the percentages of inauthentic files malicious peers are able to upload in the presence of the reputation system should be similar to those from [8]. In all experiments, we assumed the network to have an initial structure, i.e. peers have some initial trusts in other peers, generated randomly following a power-law distribution. We ran 30 query cycles, each of them consisting of sending 50 queries through the network. After each query cycle one more iteration of the selective expansion took place, as well as an update of the reputation vectors at each peer. The results of repeated squaring were updated only once in 5 cycles, as they need more computational resources.

Threat Model A. Malicious peers always provide an inauthentic file when selected as download source. They set their local trust values to 1 - s_{ij}, i.e. the opposite of their real trust value. The network consists of 63 good peers and 0, 7, 14, 25, 37, and 60 malicious peers respectively. Each good peer answers a corrupt file with 5% probability.

Threat Model B. Malicious peers of type A form a malicious collective. They set their local trust values to 1 - sij for good peers, and to 1 for any other malicious peers (i.e. complete trust in the other malicious peers). The peer distribution has the same characteristics as in threat model A.

Discussion. In both cases, the average fraction of inauthentic downloads is about 11% and does not exceed 13.5%, which represents a significant decrease against a net-

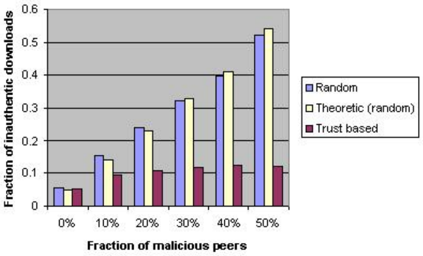

Fig. 3. Threat A: Mal. peers provide corrupt files and set their trust opposed to the real values.

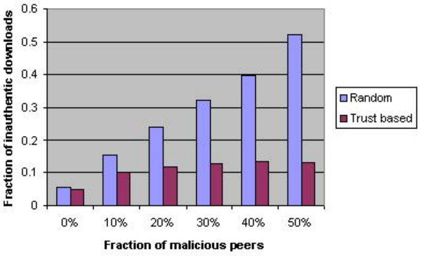

Fig. 4. Threat B: Malicious peers of type A also form a collective.

work without a reputation system in place. Moreover, malicious collectives are broken by our algorithm, as they only add 1-2% extra inauthentic downloads.

Threat Model C. Malicious peers of type B provide an inauthentic file in $f\%$ of all cases when selected as download source, with an f varying from 0 to 100 in steps of 10. The network consists of 53 good peers and 20 malicious ones.

Threat Model D. A first group of malicious peers of type D provide authentic files with 95% probability (just as the good peers), but additionally assign a trust of 1 to all malicious peers of the second type, B. There are 63 good peers and 40 malicious ones, the latter ones having different roles in different experiments, as depicted in figure 6.

Discussion. Here, malicious peers try to increase their rating by also providing good files. In model C, the maximum amount of inauthentic files they insert in the network is 17.6%, achieved when providing 50% good answers. For $f = 70\%$ (e.g. 365 good answers and 1215 bad ones), there were only 7.3% corrupt downloads in the entire network. Finally, the increase of bad downloads is also small when some peers acting correctly are trying to boost the reputation of the malicious ones. Although results in the right side of figure 6 are comparable to the random case, they require too much effort from malicious peers. For example, with 15 B peers and 25 D peers, they need to upload 1420 good files altogether in order to distribute 1197 inauthentic ones.

Generally, we can conclude that the robustness of our algorithm against malicious peers is very similar to that of EigenTrust [8] and thus the addition of personalization into a fixed-point reputation algorithm does not make it more vulnerable.

6 Conclusions

We have presented an algorithm which computes personalized global reputation values in P2P networks based on a fixed-point iteration and having peers' previous experiences as input. We also described a secure method to compute global trust values, in order to assure identification and isolation of malicious peers. We showed how to implement the reputation system in a scalable and distributed manner and simulated several possible subversion attacks. Our algorithm proved to be as robust against them as [8], while adding personalization to the global ranks computed by each peer.

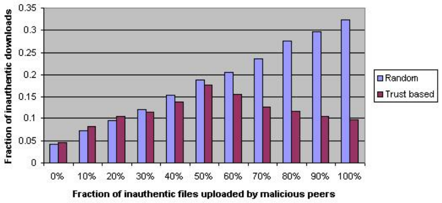

Fig. 5. Threat C: Malicious collective providing f% correct answers.

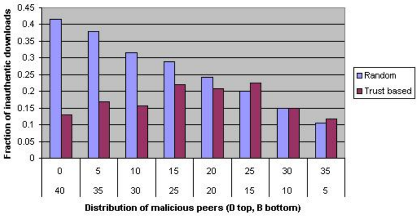

Fig. 6. Threat D: Malicious group boosting the reputation of a malicious collective of type B.

References

1. Paul-Alexandru Chirita, Wolfgang Nejdl, and Oana Scurtu. Knowing where to search: Personalized search strategies for peers in p2p networks. In *Proceedings of the P2P Information Retrieval Workshop held at the 27th International ACM SIGIR Conference*, 2004.
2. Paul-Alexandru Chirita, Daniel Olmedilla, and Wolfgang Nejdl. Pros: A personalized ranking platform for web search. In *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, Aug 2004.
3. E. Dumbill. Finding friends with xml and rdf.
4. Gnutella web page: http://www.gnutella.com/.
5. J. Golbeck, B. Parsia, and J. Hendler. Trust networks on the semantic web. In *Proceedings of Cooperative Intelligent Agents*, 2003.
6. T. Haveliwala. Topic-sensitive pagerank. In *In Proceedings of the Eleventh International World Wide Web Conference, Honolulu, Hawaii*, May 2002.
7. G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th International World Wide Web Conference*, 2003.
8. S. Kamvar, M. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th Intl. WWW Conference*, 2003.
9. S. Marti and H. Garcia-Molina. Limited reputation sharing in p2p systems. In *Proceedings of ACM Conference on Electronic Commerce (EC04)*, 2004.
10. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.
11. M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *Proceedings of the 2nd International Semantic Web Conference*, 2003.
12. K. Sankaralingam, S. Sethumadhavan, and J. C. Browne. Distributed pagerank for p2p systems. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, 2003.
13. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup lervice for internet applications. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, San Diego, USA, August 2001.
14. A. Yamamoto, D. Asahara, T. Itao, S. Tanaka, and T. Suda. Distributed pagerank: A distributed reputation model for open peer-to-peer networks. In *Proceedings of the 2004 Symposium on Applications and the Internet-Workshops*, 2004.
15. C. Ziegler and G. Lausen. Spreading activation models for trust propagation. In *Proceedings of the IEEE International Conference on e-Technology, e-Commerce, and e-Service*, 2004.