# A Reputation Management and Selection Advisor Schemes for Peer-to-Peer Systems

Loubna Mekouar, Youssef Iraqi, and Raouf Boutaba

University of Waterloo, Waterloo, Canada
{lmekouar, iraqi, rboutaba}@bbcr.uwaterloo.ca

**Abstract.** In this paper we propose a new and simple reputation management scheme for partially decentralized peer-to-peer systems. The reputation scheme helps to build trust between peers based on their past experiences and the feedback from other peers. We also propose two selection advisor algorithms for helping peers select the right peer to download from. The simulation results show that the proposed schemes are able to detect malicious peers and isolate them from the system, hence reducing the amount of inauthentic uploads. Our approach also allows to uniformly distribute the load between non malicious peers.

## 1 Introduction

### 1.1 Background

In a Peer-to-Peer (P2P) file sharing system, peers communicate directly with each other to exchange information and share files. P2P systems can be divided into several categories. Centralized P2P systems (e.g. Napster [1]), use a centralized control server to manage the system. These systems suffer from the single point of failure, scalability and censorship problems. Decentralized P2P systems try to distribute the control over several peers. They can be divided into completely-decentralized and partially-decentralized systems. Completely-decentralized systems (e.g. Gnutella [2]) have absolutely no hierarchical structure between the peers. In other words, all peers have exactly the same role. In partially-decentralized systems (e.g. KaZaa [3], Morpheus [4] and Gnutella2 [5]), peers can have different roles. Some of the peers act as local central indexes for files shared by local peers. These special peers are called "supernodes" or "ultrapeers" and are assigned dynamically [6]. They can be replaced in case of failure or malicious attack. Supernodes index the files shared by peers connected to them, and proxy search requests on behalf of these peers. Queries are therefore sent to supernodes, not to other peers. A supernode typically supports 300 to 500 peers depending on available resources [5]. Partially-decentralized systems are the most popular in P2P systems.

In traditional P2P systems (i.e. without any reputation mechanism), the user is given a list of peers that can provide the requested file. The user has then to choose one peer from which the download will be performed. This process is frustrating to the user as this later struggles to choose the right peer. After the download has finished, the user has to check the received file for malicious content (e.g. viruses[1]) and that it actually corresponds to the requested file (i.e. the requested content). If the file is not good, the user has to start the process again. In traditional P2P systems, little information is given to the user to help in the selection process.

In [8] it is stated that most of the shared content is provided by only 30% of the peers. There should be a mechanism to reward these peers and encourage other peers to share their content. At the same time, there should be a mechanism to punish peers with malicious behavior (i.e. those that provide malicious content or misleading filenames) or at least isolate them from the system.

Reputation-based P2P systems [9–13] were introduced to solve these problems. These systems try to provide a reputation management system that will evaluate the transactions performed by the peers and associate a reputation value to these peers. The reputation values will be used as a selection criteria between peers. These systems differ in the way they compute the reputation values, and in the way they use these values. The following is the life cycle of a peer in a reputation-based P2P system:

1. Send a request for a file
2. Receive a list of candidate peers that have the requested file
3. Select a peer or a set of peers based on a reputation metric
4. Download the file
5. Send a feedback and update the reputation data

### 1.2 Motivation and contribution

Several reputation management systems have been proposed in the literature (please refer to Section 5 for more details). All of these have focused on the completely-decentralized P2P systems. Only KaZaa, a proprietary partially-decentralized P2P system, has introduced basic reputation metric (called “participation level”) for rating peers. Note that the proposed reputation management schemes for completely-decentralized P2P systems cannot be applied in the case of partially-decentralized system. This later relies on the supernodes for control messages exchange (i.e. no direct management messages are allowed between peers.)

In this paper, we propose a reputation management system for partially-decentralized P2P systems. This reputation mechanism will allow a more clear-sighted management of peers and files. Our contribution is in step 3 and 5 of the life cycle of a peer in a reputation-based P2P system (cf. 1.1). The reputation considered in this paper, is for trust (i.e. maliciousness of peers), based on the accuracy and quality of the file received. Good reputation is obtained by having consistent good behavior through several transactions. The reputation criteria is used to distinguish between peers. The goal is to maximize the user satisfaction and decrease the sharing of corrupted files.

[1] such as the VBS.Gnutella Worm [7]

The paper is organized as follows. In Section 2, we introduce the new reputation management schemes considered in this paper. Section 3, describes the proposed selection advisor mechanisms. Section 4 presents the performance evaluation of the proposed schemes while Section 5 presents the related works. Finally, Section 6 concludes the paper.

## 2 Reputation Management

In this section, we introduce the new reputation management schemes. The following notations will be used.

### 2.1 Notations and Assumptions

– Let $P_i$ denotes peer $i$
– Let $D_{i,j}$ be the units of downloads performed from peer $P_j$ by peer $P_i$
– Let $D_{i,*}$ denotes the units of downloads performed by peer $P_i$
– Let $D_{*,j}$ denotes the units of uploads by peer $P_j$
– Let $A^F_{i,j}$ be the appreciation of peer $P_i$ for downloading the file $F$ from $P_j$.
– Let $Sup(i)$ denotes the supernode of peer $i$

In this paper, we assume that supernodes are selected from a set of trusted peers. This means that supernodes are trusted to manipulate the reputation data. The mechanism used to do so is outside the scope of this paper and will be addressed in the future. We also assume that the supernodes share a secret key that will be used to digitally sign data. The reader is referred to [14] for a survey on key management for secure group communication. We also assume the use of public key encryption to provide integrity and confidentiality of message exchanges.

### 2.2 The Reputation Management Scheme

After downloading a file $F$ from peer $P_j$, peer $P_i$ will value this download. If the file received corresponds to the requested file and has good quality, then we set $A^F_{i,j} = 1$. If not, we set $A^F_{i,j} = -1$. In this case, either the file has the same title as the requested file but different content, or that its quality is not acceptable. Note that we can set the appreciation as a real number between $-1$ and 1, if we want to support different levels of appreciations. Note that a null appreciation can be used, for example, if a faulty communication occurred during the file transfer.

Each peer $P_i$ in the system has four values, called *reputation data* ($REP_{P_i}$), stored by its supernode $Sup(i)$:

1. $D^+_{i,*}$: Appreciated downloads of peer $P_i$ from other peers,
2. $D^-_{i,*}$: Non-appreciated downloads of peer $P_i$ from other peers,
3. $D^+_{*,i}$: Successful downloads by other peers from peer $P_i$,

4. $D^-_{*,i}$: Failed downloads by other peers from peer $P_i$

$D^+_{i,*}$ and $D^-_{i,*}$ provide an idea about the health of the system (i.e. satisfaction of the peers). $D^+_{*,i}$ and $D^-_{*,i}$ provide an idea about the amount of uploads provided by the peer. They can for example help detect free riders. Keeping track of $D^-_{*,i}$ will also help detecting malicious peers (i.e. those peers who are providing corrupted files and misleading filenames). Note that we have the following relationships:

$$\begin{array}{c} D^+_{i,*} + D^-_{i,*} = D_{i,*}\ \forall i \\ D^+_{*,i} + D^-_{*,i} = D_{*,i}\ \forall i \end{array} \tag{1}$$

Keeping track of these values is important. They will be used as an indication of the reputation and the satisfaction of the peers. Figure 1 depicts the steps performed after receiving a file.

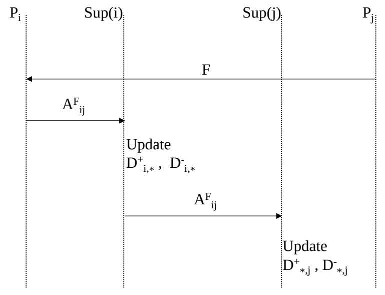


**Fig. 1.** Reputation update steps

When receiving the appreciation (i.e. $A^F_{i,j}$) of peer $P_i$, its supernode $Sup(i)$ will update the values of $D^+_{i,*}$ and $D^-_{i,*}$. The appreciation is then sent to the supernode of peer $P_j$ to update the values of $D^+_{*,j}$ and $D^-_{*,j}$. The way these values are updated is explained in the following two subsections 2.3 and 2.4.

When a peer $P_i$ joins the system for the first time, all values of its *reputation data* $REP_{P_i}$ are initialized to zero. Based on the peer transactions of uploads and downloads, these values are updated. Periodically, the supernode of peer $P_i$ sends $REP_{P_i}$ to the peer. This period of time is not too long to preserve accuracy and not too short to avoid extra overhead. The peer will keep a copy of $REP_{P_i}$ to be used the next time the peer joins the system or if its supernode changes. To prevent tempering with $REP_{P_i}$, the supernode digitally signs $REP_{P_i}$. The *reputation data* can be used to compute important reputation parameters as presented in section 3.

### 2.3 The Number Based Appreciation Scheme

In this first scheme, we will use the number of downloads as an indication of the amount downloaded. This means that $D_{*,j}$ will indicate the number of downloads from peer $P_j$, i.e. the number of uploads by peer $P_j$. In this case, after each download transaction by peer $P_i$ from peer $P_j$, $Sup(i)$ will perform the following operation:

If $A^F_{i,j} = 1$ then $D^+_{i,*} ++$, else $D^-_{i,*} ++$.

and $Sup(j)$ will perform the following operation:

If $A^F_{i,j} = 1$ then $D^+_{*,j} ++$, else $D^-_{*,j} ++$.

This scheme allows to rate peers according to the number of transactions performed. However, since it does not take into consideration the size of the downloads, this scheme makes no difference between peers who are uploading large files and those who are uploading small files. This may rise a fairness issue between the peers as uploading large files necessitates the dedication of more resources. Also, some malicious peers may artificially increase their reputation by uploading a large number of small files to a malicious partner.

### 2.4 The Size Based Appreciation Scheme

An alternative for the proposed algorithm is to take into consideration the size of the download. Once the peer sends its appreciation, the size of the download $Size(F)$ (the amount of bytes downloaded by the peer $P_i$ from the peer $P_j$) is also sent[2]. The reputation data of $P_i$ and $P_j$ will be updated based on the amount of data downloaded (MBytes).

In this case, after each download transaction by peer $P_i$ from peer $P_j$, $Sup(i)$ will perform the following operation:

If $A^F_{i,j} = 1$ then $D^+_{i,*} = D^+_{i,*} + Size(F)$,
else $D^-_{i,*} = D^-_{i,*} + Size(F)$.

and $Sup(j)$ will perform the following operation:

If $A^F_{i,j} = 1$ then $D^+_{*,j} = D^+_{*,j} + Size(F)$,
else $D^-_{*,j} = D^-_{*,j} + Size(F)$.

If we want to include the content of files in the rating, it is possible to attribute a coefficient for each file. For example, in the case that the file is rare, the uploading peer could be rewarded by increasing its satisfied uploads with more than just the size of the file. Eventually, instead of using the size of the download, we can use the amount of resources dedicated by the uploading peer to this download operation.

## 3 The Selection Advisor Algorithm

In this section we assume that peer $P_i$ has received a list of peers $P_j$ that have the requested file. Peer $P_i$ has to use the reputation data of these peers to choose the right peer to download from. Note that the selection operation can be performed at the level of the supernode, i.e. the supernode can, for example, filter malicious peers from the list given to peer $P_i$.

[2] Alternatively the supernode can know the size of the file from the information received as a response to the peer's request.

The following is the life cycle of a peer $P_i$ in the proposed reputation-based P2P system:

1. Send a request for a file $F$ to the supernode $Sup(i)$
2. Receive a list of candidate peers that have the requested file
3. Select a peer or a set of peers $P_j$ based on a reputation metric (The reputation algorithms are presented in the following subsections 3.1 and 3.2)
4. Download the file $F$
5. Send the feedback $A^F_{i,j}$. $Sup(i)$ and $Sup(j)$ will update the reputation data $REP_{P_i}$ and $REP_{P_j}$ respectively

The following subsections describe two alternative selection algorithms. Anyone of these algorithms can be based on one of the appreciation schemes presented in section 2.3 and 2.4.

## 3.1 The Difference Based Algorithm

In this scheme, we compute the Difference-Based (DB) behavior of a peer $P_j$ as:

$$DB_j = D^+_{*,j} - D^-_{*,j} \tag{2}$$

This value gives an idea about the aggregate behavior of the peer. Note that the reputation as defined in equation 2 can be negative. This reputation value gives preference to peers who did more good uploads than bad ones.

## 3.2 The Real Behavior Based Algorithm

In the previous scheme, only the difference between $D^+_{*,j}$ and $D^-_{*,j}$ is considered. This may not be able to give a real idea about the behavior of the peers.

**Example** If peer $P_1$ and peer $P_2$ have the reputation data as follows: $D^+_{*,1} = 40$, $D^-_{*,1} = 20$, $D^+_{*,2} = 20$ and $D^-_{*,2} = 0$. Then according to Difference-Based reputation (cf. equation 2) and the Number Based Appreciation scheme, we have $DB_1 = 40 - 20 = 20$ and $DB_2 = 20 - 0 = 20$. In this case, both peers have the same reputation. However, from the user perspective, peer $P_2$ is more preferable than peer $P_1$. Indeed, peer $P_2$ has not uploaded any malicious files.

To solve this problem, we propose to take into consideration not only the difference between $D^+_{*,j}$ and $D^-_{*,j}$ but also the sum of these values. In this scheme, we compute the real behavior of a peer $P_j$ as:

$$\begin{array}{ll} RB_j = \frac{D^+_{*,j} - D^-_{*,j}}{D^+_{*,j} + D^-_{*,j}} = \frac{D^+_{*,j} - D^-_{*,j}}{D_{*,j}} & \text{if } D_{*,j} \neq 0 \\ RB_j = 0 & \text{otherwise} \end{array} \tag{3}$$

Note that the reputation as defined in equation 3 can vary from $-1$ to 1. If we go back to example 3.2, then we have $RB_1 = (40 - 20)/60 = 1/3$ and $RB_2 = (20 - 0)/20 = 1$. The Real Behavior Based scheme will choose peer $P_2$.

When using this reputation scheme, the peer can choose the peer $P_j$ with the maximum value of $RB_j$.

In addition of being used as a selection criteria, the reputation data can be used by the supernode to perform service differentiation. Periodically, the supernode can check the reputation data of its peers and assign priorities to them. Peers with high reputation will have higher priority while those with lower reputation will receive a low priority. For example, by comparing the values of $D_{*,i}$ and $D_{i,*}$ one can have a real characterization of the peer's behavior. If $D_{i,*} >> D_{*,i}$, then this peer can be considered as a *free rider*. Its supernode can reduce or stop providing services to this peer. This will encourage and motivate *free riders* to share more with others. In addition, the supernode can enforce additional management policies to protect the system from malicious peers. It is also possible to implement mechanisms to prevent malicious peers from downloading in addition to prevent them from uploading.

## 4 Performance evaluation

### 4.1 Simulated Algorithms

We will simulate the two selection advisor algorithms proposed in this paper (cf. section 3.1 and 3.2) namely, the Difference-Based ($DB$) algorithm and the Real-Behavior-Based ($RB$) algorithm. Both schemes use the Size-Based Appreciation Scheme proposed in section 2.4. We will compare the performance of these two algorithms with the following two schemes.

In KaZaa [3], the peer participation level is computed as follows:

$(uploaded/downloaded) \times 100$, i.e. using our notation (cf. section 2.1) the participation level is $(D_{*,j}/D_{j,*}) \times 100$. We will consider the scheme where each peer uses the participation level of other peers as a selection criteria and we will refer to it as the KaZaa-Based algorithm ($KB$).

We will also simulate a system without reputation management. This means that the selection is done in a random way. We will refer to this algorithm as the Random Way algorithm ($RW$). Table 1 presents the list of considered algorithms.

| Algorithm | Acronym |
|---|---|
| Difference-Based algorithm | DB |
| Real-Behavior-Based algorithm | RB |
| KaZaa-Based algorithm | KB |
| Random Way algorithm | RW |

**Table 1.**

### 4.2 Simulation Parameters

We use the following simulation parameters:

- We simulate a system with 1000 peers. Compared to the actual populations of current P2P networks, the number of peers may seem small. However, our main goal in this preliminary evaluation is to observe the effectiveness of the proposed selection advisor schemes.
- The number of files is 1000.
- File sizes are uniformly distributed between 10MB and 150MB.
- At the beginning of the simulation, each peer has one of the files randomly and each file has one owner.
- As observed by [15], KaZaa files' requests do not follow the Zipf's law distribution. In our simulations, file requests follow the real life distribution observed in [15]. This means that each peer can ask for a file with a Zipf distribution over all the files that the peer does not already have. The Zipf distribution parameter is chosen close to 1 as assumed in [15]
- The probability of malicious peers is 50%. Recall that our goal is to assess the capability of the selection algorithms to isolate the malicious peers.
- The probability of a malicious peer to upload an inauthentic file is 80%
- Only 80% of all peers with the requested file are found in each request.
- We simulate 30000 requests. This means that each peer performs an average of 30 requests. For this reason we did not specify a storage capacity limit.
- The simulations were repeated 10 times over which the results are averaged.

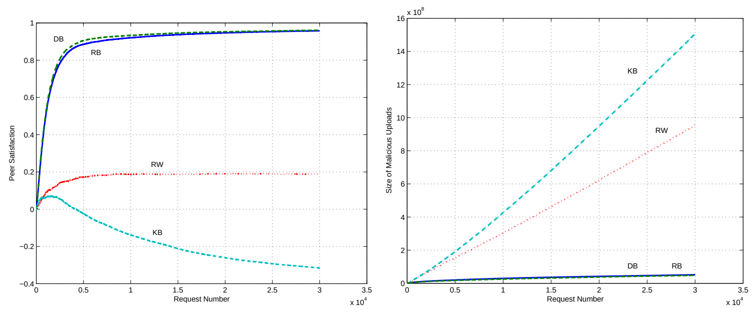
### 4.3 Performance Parameters

In our simulations we will mainly focus on the following performance parameters:

1. The peer satisfaction: computed as the difference of non-malicious downloads and malicious ones over the sum of all the downloads performed by the peer. Using our notation (cf. section 2.2) the peer satisfaction is: $(D^+_{i,*} - D^-_{i,*})/(D^+_{i,*} + D^-_{i,*})$. The peer satisfaction is averaged over all peers.
2. The size of malicious uploads: computed as the sum of the size of all malicious uploads performed by all peers during the simulation. Using our notation this can be computed as: $\sum_j D^-_{*,j}$.
3. Peer load share: we would like to know the impact of the selection advisor algorithm on the load distribution among the peers. The peer load share is computed as the normalized load supported by the peer. This is computed as the sum of all uploads performed by the peer over all the uploads in the system.

### 4.4 Simulation Results

Figure 2 (a) depicts the peer satisfaction achieved by the four considered schemes. The $X$ axis represents the number of requests while the $Y$ axis represents the peer satisfaction. Note that the maximum peer satisfaction that can be achieved



(a) (b)

**Fig. 2.** (a) Peer Satisfaction, (b) Size of malicious uploads

is 1. Note also that the peer satisfaction can be negative. According to the figure, it is clear that the *DB* and *RB* schemes outperform the *RW* and *KB* schemes in terms of peer satisfaction. The bad performance of *KB* can be explained by the fact that it does not distinguish between malicious and non-malicious peers. As long as the peer has the highest participation level, it is chosen regardless of its behavior. Our schemes (*DB* and *RB*) make the distinction and do not chose a peer if it is detected as malicious. The *RW* scheme chooses peers randomly and hence the results observed from the simulations (i.e. 20% satisfaction) can be explained as follows. With 50% malicious peers and 80% probability to upload an inauthentic file, we can expect to have 60% of authentic uploads and 40% inauthentic uploads in average. As the peer satisfaction is computed as the difference of non-malicious downloads and malicious ones over the sum of all the downloads performed by the peer. We can expect a peer satisfaction of $(60-40)/(60+40) = 20\%$.

Figure 2 (b) shows the size of malicious uploads, i.e. the size of inauthentic file uploads. As in *RW* scheme peers are chosen randomly, we can expect to see a steady increase of the size of malicious uploads. On the other hand, our proposed schemes *DB* and *RB* can quickly detect malicious peers and avoid choosing them for uploads. This isolates the malicious peers and controls the size of malicious uploads. This, of course, results in using the network bandwidth more efficiently and higher peer satisfaction as shown in figure 2 (a). In *KB* scheme, the peer with the highest participation level is chosen. The chosen peer will see its participation level increases according to the amount of the requested upload. This will further increase the probability of being chosen again in the future. If the chosen peer happens to be malicious, the size of malicious uploads will increase dramatically as malicious peers are chosen again and again. This is reflected in figure 2 (b) where *KB* has worse results than *RW*.

To investigate the distribution of loads between the peers for the considered schemes, we plotted the normalized load supported by each peer in the simulation. Figure 3 and 4 depict the results. Note that we organized the peers into two categories, malicious peers from 1 to 500 and non malicious peers from 501 to 1000. As expected, the *RW* scheme distributes the load uniformly among the peers (malicious and non malicious). The *KB* scheme does not distribute the load uniformly. Instead, few peers are always chosen to upload the requested files. In addition, the *KB* scheme cannot distinguish between malicious and non malicious peers, and in this particular case, the malicious peer number 280 has been chosen to perform most of the requested uploads.

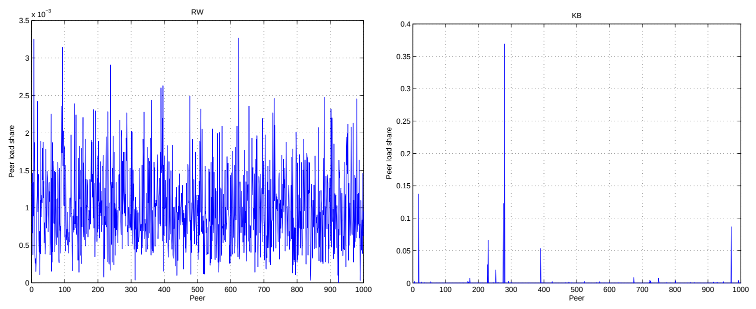


**Fig. 3.** Peer load share for RW and KB

In figure 4 the results for the proposed schemes are presented. We can note that in both schemes malicious peers are isolated from the system by not being requested to perform uploads. This explains the fact that the normalized loads of malicious peers (peers from 1 to 500) is very small. This also explains why the load supported by non malicious peers is higher than the one in the *RW* and *KB* scenarios. Indeed, since none of the malicious peers is involved in uploading the requested files[3], almost all the load (of the 30000 requests) is supported by the non malicious peers.

According to the figure, we can observe that even if the two proposed schemes *DB* and *RB* are able to detect malicious peers and isolate them from the system, they do not distribute the load among non malicious peers in the same manner. Indeed, the *RB* scheme distributes the load more uniformly among the non malicious peers than the *DB* scheme. The *DB* scheme tends to concentrate the load on a small number of peers. This can be explained by the way each scheme computes the reputation of the peers. As explained in sections 3.1 and 3.2, the *DB* scheme computes the reputation of a peer $P_j$ as shown in equation 2 based on a difference between non malicious uploads and malicious ones. The *RB* scheme, on the other hand, computes a ratio as shown in equation 3. The fact that *DB* is based on a difference, makes it choose the peer with the highest difference. This in turn will make this peer more likely to be chosen again in the future. This is why, in figure 4, the load is not distributed uniformly.

[3] after that these malicious peers are detected by the proposed schemes

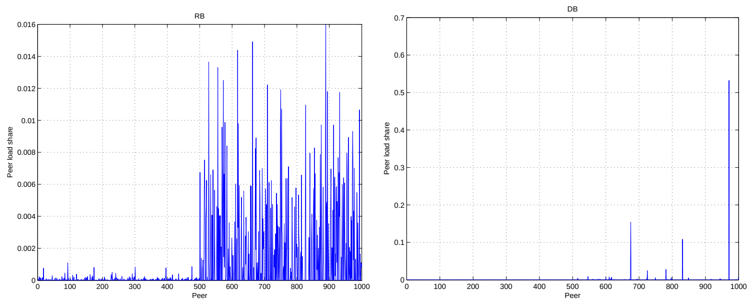


**Fig. 4.** Peer load share for RB and DB

The *RB* scheme, focuses on the ratio of the difference between non malicious uploads and malicious ones over the sum of all uploads performed by the peer (cf. eq. 3). This does not give any preference to peers with higher difference. Since in our simulations we did not consider any free riders, we can expect to have a uniform load distribution between the peers as depicted by figure 4. If free riders are considered, the reputation mechanisms will not be affected since reputation data is based on the uploads of peers. Obviously, the load distribution will be different.

## 5 Related Works

eBay [16] uses the feedback profile for rating their members and establishing the members' reputation. Members rate their trading partners with a Positive, Negative or Neutral feedback, and explain briefly why. eBay suffers from the single point of failure problem as it is based on a centralized server for reputation management.

In [10], a distributed polling algorithm is used to allow a peer looking for a resource to enquire about the reputation of offerers by polling other peers. The polling is performed by broadcasting a message asking all other peers to give their opinion about the reputation of the servants. In [11], the EigenTrust algorithm assigns to each peer in the system a trust value. This value is based on its past uploads and reflects the experiences of all peers with the peer.

The two previous schemes are reactive, They require reputations to be computed *on-demand* which requires cooperation from a large number of peers in performing computations. As this is performed for each peer having the requested file with the cooperation of all other peers, this will introduce additional latency and overhead. Most of the proposed reputation management schemes for completely decentralized P2P systems are reactive and hence suffer from this drawback.

## 6 Conclusion

In this paper, we proposed a new reputation management scheme for partially decentralized peer-to-peer systems. Our scheme is based on four simple values associated to each peer and stored at the supernode level. We also propose two selection advisor algorithms for assisting peers in selecting the right peer to download from. Performance evaluation shows that our schemes are able to detect and isolate malicious peers from the system. Our reputation management scheme is simple, proactive and has minimal overhead in terms of computation, infrastructure, storage and message complexity. Furthermore, it does not require any synchronization between the peers. Our scheme is designed to reward those who are practicing good P2P behavior, and punish those who are not. Important aspects that we will investigate in future work include mechanisms to give incentives for peers to provide appreciations after performing downloads, and countermeasures for peers who provide faked values for appreciations.

## References

1. A.Oram: 2. In: Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly Books (2001) 21–37
2. (Clip2) http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf.
3. (KaZaa) http://www.kazaa.com/us/help/glossary.htm.
4. (Morphus) http://www.morphus.com/morphus.htm.
5. (Specification, G.) http://www.gnutella2.com/.
6. Androutsellis-Theotokis, S.: A Survey of Peer-to-Peer File Sharing Technologies. Technical report, ELTRUN (2002)
7. (VBS/GWV.A) http://www.commandsoftware.com/virus/gnutella.html.
8. Adar, E., Huberman, B.A.: Free Riding on Gnutella. Technical report, HP (2000) http://www.hpl.hp.com/research/idl/papers/gnutella/.
9. Aberer, K., Despotovic, Z.: Managing Trust in a Peer-to-Peer Information System. In: International Conference on Information and Knowledge Management. (2001)
10. Cornelli, F., Damiani, E., di Vimercati, S.D.C., Paraboschi, S., Samarati, P.: Choosing Reputable Servents in a P2P Network. In: The Eleventh International World Wide Web Conference, Honolulu, Hawaii, USA (2002)
11. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The EigenTrust Algorithm for Reputation Management in P2P Networks. In: the Twelfth International World Wide Web Conference, Budapest, Hungary (2003)
12. Gupta, M., Judge, P., Ammar, M.: A Reputation System for Peer-to-Peer Networks. In: ACM 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video, Monterey, California, USA (2003)
13. Xiong, L., Liu, L.: PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. IEEE Transactions on Knowledge and Data Engineering, Special Issue on Peer-to-Peer Based Data Management (2004)
14. Rafaeli, S., Hutchison, D.: A survey of key management for secure group communication. ACM Computing Surveys **35** (2003) 309–329
15. Gummadi, K., Dunn, R.J., Saroiu, S., Gribble, S.D., Levy, H.M., Zahorjan, J.: Measurement, Modeling, and analysis of a Peer-to-Peer File Sharing Workload. In: ACM Symposium on Operating Systems Principles, New York, USA (2003)
16. (eBay Feedback) http://pages.ebay.com/help/feedback/reputation-ov.html.