Peer-to-Peer's Most Wanted: Malicious Peers*

Loubna Mekouar, Youssef Iraqi and Raouf Boutaba
University of Waterloo, Waterloo, Canada
{lmekouar, iraqi, rboutaba}@bbcr.uwaterloo.ca

In this paper, we propose a reputation management scheme for partially decentralized peer-to-peer systems. The reputation scheme helps to build trust among peers based on their past experiences and feedback from other peers. Two selection advisor algorithms are proposed for helping peers to select the most trustworthy peer to download from. The proposed algorithms can detect malicious peers sending inauthentic files. The *Malicious Detector Algorithm* is also proposed to detect liar peers that send the wrong feedback to subvert the reputation system. The new concept of *Suspicious Transactions* is introduced and explained. Simulation results confirm the capability of the proposed algorithms to effectively detect malicious peers and isolate them from the system, hence reducing the amount of inauthentic uploads, increasing peers' satisfaction, and preserving network resources.

Keywords: Reputation Management, Suspicious Transactions, Authentic Behavior, Credibility Behavior, Partially Decentralized Peer-to-Peer Systems.

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 (Figure 1): Centralized P2P systems (e.g., Napster [1]) use a centralized control server to manage the systems. These systems suffer from a single point of failure, scalability, and censorship problems. Decentralized P2P systems try to distribute 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 among 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 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. For peers connected to supernodes, the supernodes index shared files 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]. Figure 2 depicts the paths of both control and data messages in partially decentralized systems. Partially decentralized systems are the most popular; for example, KaZaA supports more than four million simultaneous peers, and the amount of data available for download is more than 5281.54 TB.

In traditional P2P systems (i.e., without any reputation mechanism), a 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 because this latter struggles to choose the most trustworthy peer. After the download has finished, the user has to check the received file for malicious content (e.g., viruses[2])

*The present work was partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC). A preliminary version was published in the IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM), 2004, and in the IEEE Consumer Communications and Networking Conference (CCNC), 2005.

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

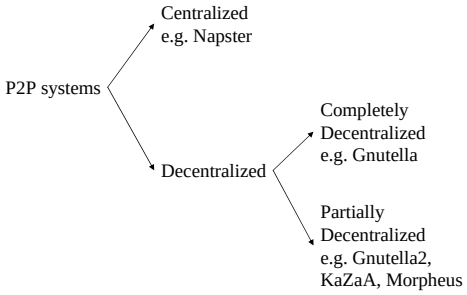

Figure 1. P2P systems

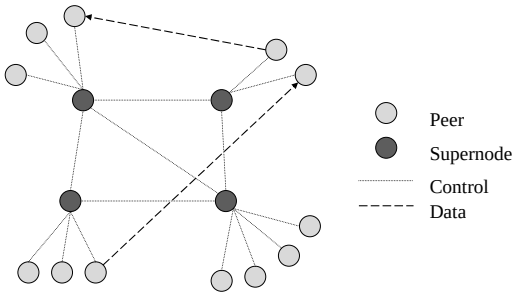

Figure 2. Example topology in a partially decentralized P2P system

and verify that the received file corresponds to the requested file (i.e., the requested content). If the file is not acceptable, the user has to restart the process[3]. In traditional P2P systems, little information is given to the user to help in the selection process.

The following is the life cycle of a peer in a traditional P2P system (see Figure 3):

1. Send a file request (either to the supernode or to other peers, depending on the P2P system)

[3]The download can take few days for large files

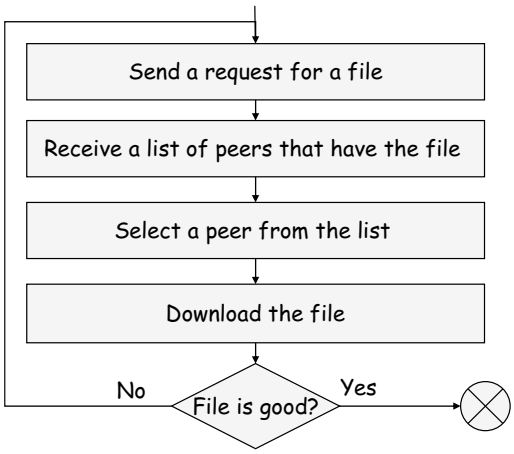

Figure 3. Life cycle in a traditional P2P system

2. Receive a list of peers that have the requested file
3. Select a peer (performed by the human user)
4. Download the file

In [8] it is stated that most of the shared content is provided by only 30% of peers. A mechanism is needed to reward these peers and encourage other peers to share their content. At the same time, a mechanism is needed to punish peers that exhibit 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 peers and associate a reputation value to these peers. The reputation values will be used as selection criteria among peers. These systems differ in how they compute reputation values, and how they use these values.

The following is the life cycle of a peer in a reputation-based P2P system (Figure 4):

1. Send a file request

2. Receive a list of 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 feedback and update the reputation data

1.2. Motivation and contribution

Partially decentralized P2P systems have been proposed to reduce the control overhead needed to run the P2P system. They also provide lower discovery time because the discovery process involves only supernodes. Several reputation management systems have been proposed in the literature (Section 9 provides more details). They have focused on completely decentralized P2P systems. Only KaZaA, a proprietary partially decentralized P2P system, has introduced basic reputation metric (called "participation level") for rating peers. The proposed reputation management schemes for completely decentralized P2P systems cannot be applied in the case of a partially decentralized system. The latter relies on supernodes for control message exchange (i.e., no direct management messages are allowed among peers.)

In completely decentralized P2P systems, each peer may record information on past experiences with all peers it has interacted with and may use a voting system to request peers' opinions regarding the peers that have the requested file. The goal of this paper is to take advantage of the use of supernodes in partially decentralized P2P systems for reputation management. In addition to being used for control message exchange, a supernode will also be used to store reputation data for peers it serves, update this data, and provide it to other supernodes. Since each peer interacts only with one supernode at a time (an assumption that can be justified by the use of FastTrack[4]), the proposed approach achieves more accuracy and reduces message overhead significantly. In fact, once a supernode sends a search request on behalf a peer, the supernode will get a list of peers that have the requested file and their reputations from their corresponding supernodes. In this case, no voting system is needed since the reputation will represent all past experiences from all the peers that had interacted with these peers. A supernode will be also used to provide service differentiation based on reputation data of the peers it is responsible for.

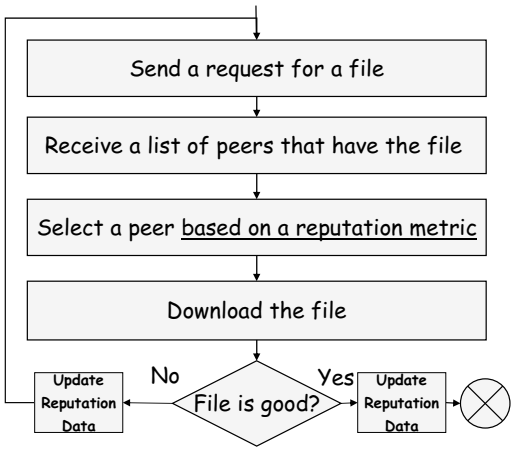

Figure 4. Life cycle in a reputation-based P2P system

In this paper, we propose a reputation management system for partially decentralized P2P systems. This reputation mechanism allows more clearsighted 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). Good reputation is obtained by having consistent good behavior through several transactions. The reputation criteria is used to distinguish among peers. The goal is to maximize the user satisfaction, and decrease the sharing of corrupted files. This algorithm detects malicious peers that are sending inauthentic files and isolates them from the system.

In this paper, we also propose a novel scheme,

[4]Although in this paper, it is assumed that each peer interacts only with one supernode at a time, the proposed mechanism can easily be extended to handle the case when a peer can connect to several supernodes at the same time.

that in addition to detecting and punishing inauthentic peers, detects liar peers and punishes them. This scheme reduces considerably the amount of malicious uploads and protects the health of the system.

The reputation considered in this paper, is for trust (i.e. maliciousness of peers), based on the accuracy and quality of the file received. We also consider that peers can lie while sending feedback. We assume no other malicious behaviors such as Denial of Service Attacks etc. Such malicious behaviors are outside of the scope of this paper and left for future work.

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 an analysis of peers' behavior. Section 5 discusses the proposed approaches to detect malicious peers. Section 6 discusses service differentiation issues in the considered P2P systems. Section 7 presents the performance evaluation of the proposed schemes and Section 8 presents some open issues, while Section 9 presents the related works. Finally, Section 10 concludes the paper.

2. Reputation Management

In this section, we describe the proposed 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 by peer P_i from peer P_j
- 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 after downloading the file F from peer P_j.
- Let $Sup(i)$ denotes the supernode of peer i

We assume that supernodes are selected from a set of trusted peers. This means that supernodes are trusted to manipulate the reputation data. This trust is similar to the trust given to these supernodes for control messages exchange.

2.2. The Reputation Management Scheme

After downloading a file F from peer P_j, peer P_i will evaluate this download. If the file received corresponds to the requested file, 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 if we want to support different levels of appreciations, we can set the appreciation as a real number between -1 and 1. Note also 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. $D^+_{i,*}$: Satisfied downloads of peer P_i from other peers,
2. $D^-_{i,*}$: Unsatisfied downloads of peer P_i from other peers,
3. $D^+_{*,i}$: Satisfied uploads from peer P_i to other peers,
4. $D^-_{*,i}$: Unsatisfied uploads from peer P_i to other peers

$D^+_{i,*}$ and $D^-_{i,*}$ provide an idea about the health of the system (i.e., satisfaction of peers) while $D^+_{*,i}$ and $D^-_{*,i}$ provide an idea about the amount and quality of uploads provided by peer P_i. 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/or misleading filenames). Note that we have the following relationships:

$$\begin{array}{ll} D^+_{i,*} + D^-_{i,*} = D_{i,*} & \forall i \\ D^+_{*,i} + D^-_{*,i} = D_{*,i} & \forall i \end{array} \quad (1)$$

Keeping track of these values is important. They will be used as an indication of the reputation and the satisfaction of peers.

Figure 5 depicts the steps performed after receiving a file. When receiving the appreciation (i.e. $A^F_{i,j}$) of peer P_i, its supernode $Sup(i)$ will update $D^+_{i,*}$ and $D^-_{i,*}$ values. The appreciation is then sent to the supernode of peer P_j to update $D^+_{*,j}$ and $D^-_{*,j}$ values. The way these values are updated is explained in the following subsections 2.3 and 2.4.

When peer P_i joins the system for the first time, all values of its *reputation data* REP_{P_i} are initialized to zero[5]. Based on peer's 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. The *reputation data* can be used to compute important reputation parameters as presented in section 3.

To prevent tampering with REP_{P_i}, we assume that supernodes share a secret key that will be used to digitally sign REP_{P_i}. The mechanism used to do so is outside the scope of this paper. 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 of message exchanges.

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 among 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.

[5]This is a neutral reputation value.

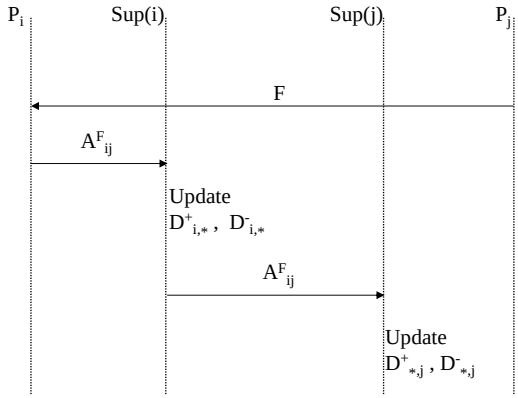

Figure 5. Reputation update steps

2.4. The Size Based Appreciation Scheme

A better approach 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 peer P_j) is also sent[6]. The reputation data of P_i and P_j will be updated based on the amount of data downloaded.

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 $D^+_{*,j} = D^+_{*,j} + Size(F)$,

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

else $D^-_{*,j} = D^-_{*,j} + Size(F)$.

If we want to include the content of files in the rating scheme, 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 upload operation (i.e. processing, bandwidth, time, etc.).

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 and their corresponding reputation values. These values are computed at their corresponding supernodes and sent to the supernode of the peer requesting the file (i.e., P_i). Peer P_i has to use the reputation value as a selection criterion among these peers and choose the right peer to download from. Note that the four values of the reputation data are not sent to P_i. Only one reputation value for each peer that has the requested file will be received. Note also 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.

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 feedback $A^F_{i,j}$. $Sup(i)$ and $Sup(j)$ will update the reputation data REP_{P_i} and REP_{P_j} respectively.

j	1	2
$D^+_{*,j}$	40	20
$D^-_{*,j}$	20	0

Table 1
Example of Reputation Data

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. Note that only the satisfied and unsatisfied units of uploads (i.e. $D^+_{*,j}$, $D^-_{*,j}$) are considered for computing the reputation of a peer. The satisfied and unsatisfied units of downloads (i.e. $D^+_{i,*}$, $D^-_{i,*}$) will be taken into consideration for service differentiation.

3.1. The Difference Based Algorithm

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

$$DB_j = D^+_{*,j} - D^-_{*,j} \quad (2)$$

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 performed more good uploads than bad ones.

3.2. The Inauthentic Detector Algorithm

In the previous scheme, only the difference between $D^+_{*,j}$ and $D^-_{*,j}$ is considered. This may not give a real idea about the behavior of the peers as shown in the following example. If peer P_1 and peer P_2 have the reputation data as in Table 1, and assume we are using the Size Based Appreciation scheme, then according to Difference-Based algorithm (cf. equation 2) we have $DB_1 = 40 - 20 = 20$ MB and $DB_2 = 20 - 0 = 20$ MB. In this case, both peers have the same reputation. However, from the user's 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 *Authentic Behavior* of a peer P_j

as:

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

Note that the reputation as defined in equation 3 is a real number between -1 (if $D^+_{*,j} = 0$) and 1 (if $D^-_{*,j} = 0$).

If we go back to the example in table 1, then we have $AB_1 = (40 - 20)/60 = 1/3$ and $AB_2 = (20-0)/20 = 1$. The Inauthentic Detector scheme will choose peer P_2.

When using this reputation scheme, the peer can do one of the following:

1. Choose peer P_j with the maximum value of AB_j, or
2. Choose the set of peers P_j such that $AB_j \geq AB_{threshold}$, where $AB_{threshold}$ is a parameter set by the peer P_i or by the system. The latter case can be used if the P2P system supports swarmed retrieval (i.e. the system is able to download several pieces of the file simultaneously from different peers.) Note that our proposed schemes as presented in this paper can be easily adapted to systems with swarmed retrieval. In this case, each peer participating in the upload operation, will see its reputation data updated according to the amount it uploaded.

Note that in the case where there are several peers with a maximum reputation value, one peer can be selected randomly or other parameters can be used to distinguish among peers. One can use the amount of available resources as a selection criterion. For example nodes with higher outgoing bandwidth will be preferable.

As it will be shown in the performance evaluation section 7, the proposed schemes are able to isolate malicious peers from the system by not requesting them for uploads. This will prevent malicious peers from increasing their reputation. We call this approach the *implicit service differentiation*. In addition of being used as a selection criterion, the reputation data can be used by the supernode to perform service differentiation. In this case, we call this approach the *explicit service differentiation*. Service differentiation will be discussed in section 6.

4. Peer Behavior Understanding

In all previously proposed feedback-based reputation management schemes for P2P systems, the emphasize was on detecting and punishing peers who are sending inauthentic files. No special mechanism was proposed to detect and punish peers that send wrong feedbacks. Although some of the proposed feedback-based reputation schemes take this behavior into consideration, those schemes rely on peers' reputation for their peer-selection process. In this paper, we introduce the concept of *suspicious transactions* to detect liar peers.

In a P2P system, peers can lie in their feedbacks. Such liar peers can subvert the reputation system by affecting badly the reputation of other peers (increase the reputation of malicious peers, and/or decrease the reputation of good peers). These malicious peers may not be detected if they are not sending inauthentic files and, hence, their reputation can be high and they will be wrongfully trusted by the system.

Malicious peers send inauthentic files to satisfy their needs in propagating malicious content: e.g. viruses and misleading filenames. Malicious peers can also lie in their feedbacks to satisfy their needs to subvert the reputation system. Acting maliciously has bad effects on any Peer-to-Peer system. We believe that it is of paramount importance to detect liar peers and prevent them from affecting the system.

In this context, free riders [8] will not be considered as malicious peers if they do not affect directly the reputation of other peers.

4.1. Peer Behavior Categorization

In a P2P system, we consider two general behaviors of peers: good and malicious. Good peers are those that send authentic files and do not lie in their feedbacks (Type $T1$ in Table 2). Malicious peers can be divided into three categories: 1) peers that send inauthentic files and do not lie in their feedbacks (Type $T2$), 2) peers that send

Type	Peer	Authentic Behavior	Credibility Behavior
$T1$	Good	High	High
$T2$	Malicious: Inauthentic	Low	High
$T3$	Malicious: Liar	High	Low
$T4$	Malicious: Inauthentic and Liar	Low	Low

Table 2
Peer Behavior

authentic files and do lie in their feedbacks (Type $T3$), and 3) peers that send inauthentic files and do lie in their feedbacks (Type $T4$).

A liar peer is one that after receiving an authentic file, instead of giving an appreciation equal to 1, the peer sends an appreciation equal to -1 to decrease the reputation of the peer uploading the file. Or, if the peer receives an inauthentic file, it sends a positive appreciation to increase the reputation of other malicious peers. Note that we consider the consistent behaviors of peers. This means that most of the time a peer behavior is consistent with the category it belongs to (i.e. $T1$, $T2$, $T3$, or $T4$). For example, a good peer can sometimes (on purpose or by mistake) send inauthentic files. Note also that peers can change their behavior over time and hence can jump from one category to another. A free rider can belong to one of the categories described in Table 2 and the system will deal with it accordingly.

4.2. Effect On Reputation

Peers can have positive or negative reputations. A good peer usually has a positive reputation since he is behaving well, but since malicious peers can lie, his reputation can decrease and even get negative. In contrast, malicious peers will have negative reputation values since they are sending inauthentic files. However, their reputation values can increase and even get positive if some other malicious peers send positive appreciations even if they receive inauthentic files. This happens in systems where liar peers are not detected nor punished.

5. Detecting Malicious Peers

Let's assume that peer P_i downloads file F from peer P_j. We focus on the *Authentic Behavior* (sending authentic or inauthentic files) of peer P_j since it is sending the file, and the *Credibility Behavior* (lying or not in the feedback) of peer P_i since it is sending the appreciation that will affect the reputation of peer P_j. If we want to take the appropriate actions after this transaction, we have to detect if peer P_j belongs to any of the categories $T2$ and $T4$, and if peer P_i belongs to any of the categories $T3$ and $T4$.

IDA (section 3.2) allows us to detect peers sending inauthentic files. The goal now is to detect peers that send wrong feedbacks and diminish their impact on the reputation-based system.

5.1. First Approach

One approach is to say that malicious peers have a low reputation than good peers. One way of reducing the impact of peers having a low reputation is to take this later into account when updating the reputation of other peers.

We can then change the operations in section 2.4, to:

$$\begin{aligned} \text{If } A^F_{i,j} = 1 \quad & \text{then } D^+_{*,j} = D^+_{*,j} + \tfrac{1+AB_i}{2} Size(F) \\ & \text{else } D^-_{*,j} = D^-_{*,j} + \tfrac{1+AB_i}{2} Size(F) \end{aligned} \quad (4)$$

Using this approach[7], the impact of peer P_i on the reputation of peer P_j is related to the trust given to peer P_i. The trust is expressed by the value of AB_i. If peer P_i has a good reputation (usually above zero), it is trusted more and it will impact the reputation of peer P_j, but, if its reputation is low (usually negative), only a small

[7] In operation (4), $\frac{1+AB_i}{2}$ can be replaced by any function of AB_i that is strictly increasing from 0 to 1.

fraction of the file size is considered hence reducing the impact on the reputation of peer P_j.

In case peer P_i is new, his reputation is null and since we do not know yet if he is a good or a malicious peer, only half of the size of the uploaded file F is affecting the reputation of the peer uploading the file (i.e. peer P_j).

The problem with this approach appears in the following example. Assume that some peers belong to category $T3$. Those peers always send authentic files, but send also wrong appreciations. Most of the time, and according to operation (4), those peers will have a high reputation since they always send authentic files and hence will receive good feedbacks[8]. Those peers will be trusted by the system and will affect badly the reputations of other peers and may eventually brake the system. The performance evaluation section assesses the effect of liar peers on the reputation of other peers.

5.2. Second Approach

A better approach to detect peers that lie in their feedbacks is to detect *suspicious transactions*. We define a *suspicious transaction* as one in which the appreciation is different from the one expected knowing the reputation of the sender. In other words, if $A^F_{i,j} = 1$ and $AB_j < 0$ or if $A^F_{i,j} = -1$ and $AB_j > 0$ then we consider this transaction as suspicious.

To detect peers that lie in their feedbacks, for each peer P_i we keep track of the following values:

1. N_i: The total number of downloads performed by peer P_i
2. N^*_i: The number of downloads by peer P_i where the sign of the appreciation sent by peer P_i is different from the sign of the sender's reputation, i.e. $A^F_{i,j} \times AB_j < 0$ (i.e. during a suspicious transaction)
3. TF_i: The total size of all the files uploaded by P_i.

Note that $N^*_i \leq N_i \ \forall i$

[8]We assume that the percentage of malicious peers, in a P2P system, is lower than the percentage of good peers. This assumption is realistic since this is the basis on which peer-to-peer systems can work.

When receiving the appreciation (i.e. $A^F_{i,j}$) of peer P_i, its supernode $Sup(i)$ will update the values of N_i and N^*_i as follows:

$$\begin{array}{l} N_i = N_i + 1 \\ \text{If } (A^F_{i,j} \times AB_j) < 0 \text{ then } N^*_i = N^*_i + 1 \end{array} \quad (5)$$

Let α_i be the ratio of N^*_i and N_i:

$$\alpha_i = \frac{N^*_i}{N_i} \quad (6)$$

Note that $0 \leq \alpha_i \leq 1 \ \forall i$

α_i is the ratio of the number of suspicious feedbacks[9] sent by peer P_i over the total number of feedbacks sent by peer P_i. α_i is a good indicator of the liar behavior of peer P_i. Indeed, if peer P_i lies in its feedbacks, the number of times $A^F_{i,j}$ and the sender's reputation having different signs, is high and hence the value of N^*_i. Liar peers will tend to have values of α_i near 1. Good peers will tend to have values of α_i near zero.

To minimize the effect of liar peers, we propose to use the following update strategy for the sender's appreciation; After receiving the appreciation $A^F_{i,j}$, the sender's supernode $Sup(j)$ will perform the following operations:

$$\begin{array}{l} \text{If } A^F_{i,j} = 1 \\ \quad \text{then } D^+_{*,j} = D^+_{*,j} + (1 - \alpha_i) \times Size(F) \\ \quad \text{else } D^-_{*,j} = D^-_{*,j} + (1 - \alpha_i) \times Size(F) \end{array}$$

$$TF_j = TF_j + Size(F)$$

Since liar peers (in categories $T3$ and $T4$) will have a high value of α_i, their effect on the reputation of the peer sending the file is minimized. This is because the higher the value of α_i, the lower the value of $(1 - \alpha_i)$. On the other hand, good peers will have a lower value of α_i and hence will keep having an impact of the reputation of other peers.

Note that AB_j is updated after each upload of peer P_j (equation 7) and α_i is updated after each download of peer P_i. This means that liar peers will be detected and punished even if they did not upload any file and inauthentic peers will

[9]A suspicious feedback is the feedback sent during a suspicious transaction

be detected and punished even if they did not perform any download.

$$\begin{array}{ll} AB_j = \frac{D^+_{*,j} - D^-_{*,j}}{TF_j} & \text{if } TF_j \neq 0 \\ AB_j = 0 & \text{otherwise} \end{array} \quad (7)$$

If peer P_i changes its behavior, α_i will also change, and hence its impact on the reputation of others. For example, if peer P_i changes its behavior from category $T3$ to $T1$ (c.f. table 2), the number of suspicious transactions N^*_i involving this peer (in comparison to the total number of transactions N_i) will be less and hence the value of α_i will decrease, making the impact of this peer more considerable.

Let the *Credibility Behavior* of peer P_i be: $CB_i = 1 - \alpha_i$.

In this case, the reputation of peer P_i is the couple (AB_i, CB_i) which characterize the behavior of peer P_i in terms of *Authentic Behavior* (sending authentic or inauthentic files) and *Credibility Behavior* (lying or not in the feedback). Note that because the behavior of peers is characterized by two values, the supernode can still download a file from a peer with low value of CB_i as long as the value of AB_i is high. This means that the system can still take advantage of a peer that provides authentic files but lies in its feedbacks. We will refer to this algorithm as the *Malicious Detector Algorithm* (*MDA*).

The performance evaluation section presents results that confirm that the *Credibility Behavior* is a very good indicator of the liar behavior of peers, and hence can be used to differentiate among peers. This will in turn allow for better management of peers and hence, provide better performance.

6. Service Differentiation

In addition of being used as 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 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.

Moreover, the mechanism presented in 5.2 to detect malicious peers, will allow the supernode to enforce service differentiation according to peers' reputation. For example, when processing a search request, the supernode can give higher priority to good peers who do not send inauthentic files nor lie in their feedbacks. In addition, when a peer P_i receives a list of peers that have the requested file and chooses peer P_j to download from, peer P_i's reputation that is the couple (AB_i, CB_i) may be also sent to peer P_j who will decide whether to upload or not the requested file. In case that the value of CB_i is too low, the peer P_j can choose not to upload the file since its reputation can be affected. Usually, peer P_j will expect its AB_j to increase as a return of uploading a good-quality requested file to P_i but the low value of CB_i will not guarantee this increase. This service differentiation at peers level will motivate peers to protect their reputation by not lying in the feedbacks if they want to download files from other peers.

7. Performance Evaluation

In the performance evaluation section, we will focus on:

- Simulating the Difference-Based and the Inauthentic Detector algorithms: in this case these algorithms are compared with the KaZaA-Based and the Random Way algorithms. No liar peers are considered in this first set of simulations to prove the effectiveness of the proposed algorithms and their outstanding performance compared to other schemes.
- Simulating the Inauthentic Detector and

Algorithm	Acronym
Difference-Based algorithm	DB
Inauthentic Detector algorithm	IDA
KaZaA-Based algorithm	KB
Random Way algorithm	RW

Table 3
Simulated Algorithms

the Malicious Detector algorithms: in this case, these algorithms are compared to the KaZaA-Based and the Random Way algorithms. A high percentage of malicious peers is considered to show the effectiveness of both the IDA and MDA in detecting malicious peers who are sending malicious content. Moreover, this second set of simulations demonstrates the effectiveness of MDA in detecting liar peers and reducing the amount of malicious uploads hence allowing for better network resources utilization and higher peer satisfaction.

7.1. Simulation of the Difference-Based and the Inauthentic Detector Algorithms

In this first set of simulations, we will simulate the two selection advisor algorithms (cf. section 3.1 and 3.2) namely, the Difference-Based (DB) algorithm and the Inauthentic Detector algorithm (IDA). 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 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 3 presents the list of considered algorithms.

7.1.1. Simulation Parameters

We use the following simulation parameters:

- We simulate a system with 1000 peers.
- 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 proposed selection algorithms to isolate 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.
- Unless stated otherwise, the simulations were repeated 10 times over which the results are averaged.

7.1.2. Performance Parameters

In this first set of simulations we will mainly focus on the following performance parameters:

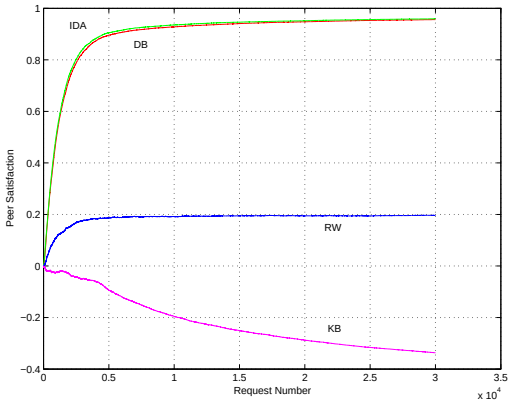

Figure 6. Peer Satisfaction

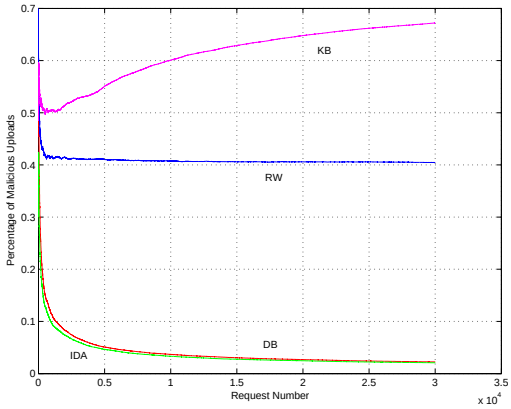

Figure 7. The percentage of malicious uploads

- 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. The peer satisfaction is averaged over all peers.
- The percentage of malicious uploads: computed as the sum of the size of all malicious uploads performed by all peers during the simulation over the total size of all uploads.
- Peer load share: we would like to know the impact of the selection advisor algorithm on the load distribution among 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 uploads in the system.

7.1.3. Simulation Results

Figure 6 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 is 1. Note also that the peer satisfaction can be negative. According to the figure, it is clear that the *DB* and *IDA* 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 *IDA*) make the distinction and do not choose 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 7 shows the percentage of malicious uploads, i.e. the percentage 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 and a fixed percentage of malicious uploads. On the other hand, our proposed schemes *DB* and *IDA* can quickly detect malicious peers and avoid choosing them for uploads. This isolates malicious peers and controls the size of malicious uploads. This, of course,

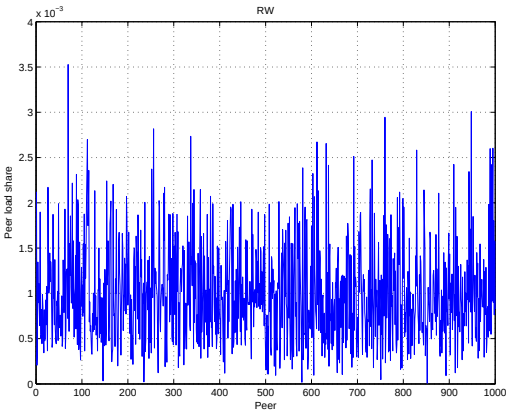

Figure 8. Peer load share for RW

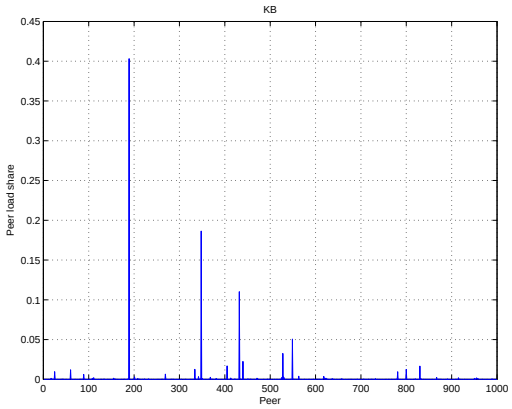

Figure 9. Peer load share for KB

results in using the network bandwidth more efficiently and higher peer satisfaction as shown in figure 6. 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 7 where *KB* has worse results than *RW*.

To investigate the distribution of loads among peers for the considered schemes, we plotted the normalized load supported by each peer during one simulation run. Figure 8, 9, 10 and 11 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)(c.f. figure 8). 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 180 has been chosen to perform most of the requested uploads (c.f. figure 9).

In figures 10 and 11 the results for the proposed schemes *IDA* and *DB* 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 malicious peers is involved in uploading the requested files[10], almost all the load (of the 30000 requests) is supported by the non malicious peers.

According to figures 10 and 11, we can observe that even if the two proposed schemes *DB* and *IDA* 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 *IDA* scheme distributes the load more uniformly among the non malicious peers than the *DB* scheme (c.f. figure 10). The *DB* scheme tends to concentrate the load on a small number of peers (c.f. figure 11). This can be explained by the way each scheme computes the reputation of peers. As explained in sections 3.1

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

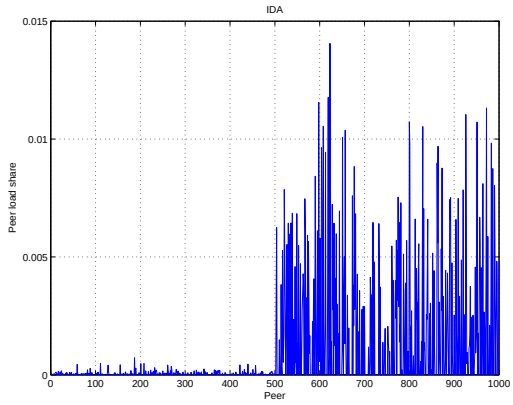

Figure 10. Peer load share for IDA

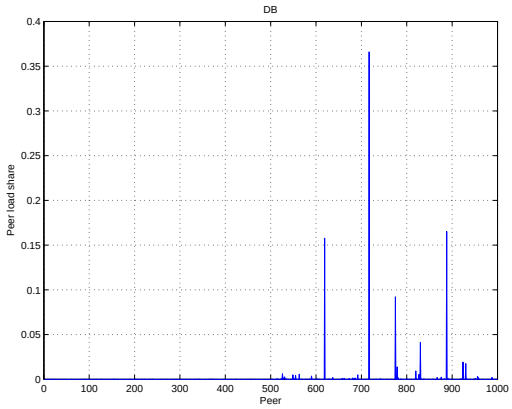

Figure 11. Peer load share for DB

and 3.2, the *DB* scheme computes the reputation of peer P_j as shown in equation 2 based on a difference between non malicious uploads and malicious ones. The *IDA* 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 11, the load achieved by *DB* is not distributed uniformly.

The *IDA* 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 among peers as depicted by figure 10. If free riders are considered, reputation mechanisms will not be affected since reputation data is based on the uploads of peers. Obviously, the load distribution will be different.

7.2. Simulation of the Inauthentic Detector and the Malicious Detector Algorithms

In this second set of simulations, we will simulate the Inauthentic Detector *(IDA)* and the Malicious Detector *(MDA)* algorithms. Their performance will be compared to KB and RW. In these simulations, liar peers will be considered as opposed to the first set of simulations in section 7.1.

7.2.1. Simulation Parameters

We use the same simulation parameters as in 7.1 with the following modifications:

- At the beginning of the simulation, each peer has 30 randomly chosen files and each file has at least one owner.
- Peers behavior and distribution are as depicted in Table 4.
- Only 40% of all peers with the requested file are found in each request. We have considered a situation where we have a lower percentage of peers with the requested file to assess the proposed algorithms in a highly dynamic network.

Taking into consideration Table 4, peers with indices from 1 to 300 belong to category $M2$, peers with indices from 301 to 600 belong to category $M1$ and peers with indices from 601 to 1000 belong to category G. We have considered a situation where we have a high percentage of malicious peers to show the effectiveness of our proposed scheme.

Category	Percentage of peers	Percentage of sending inauthentic files	Percentage of sending wrong feedback
G	40%	1%	1%
$M1$	30%	50%	50%
$M2$	30%	90%	90%

Table 4
Peer Behavior and Distribution

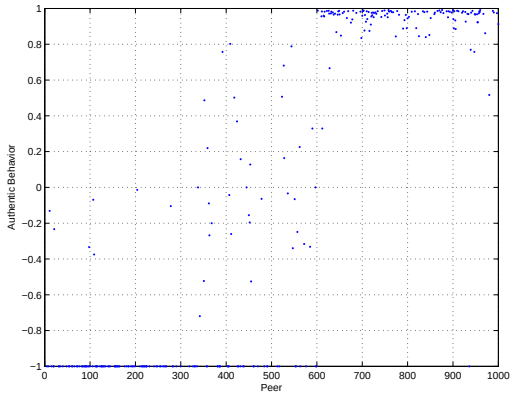

Figure 12. Authentic Behavior with *IDA* (with no liar peers)

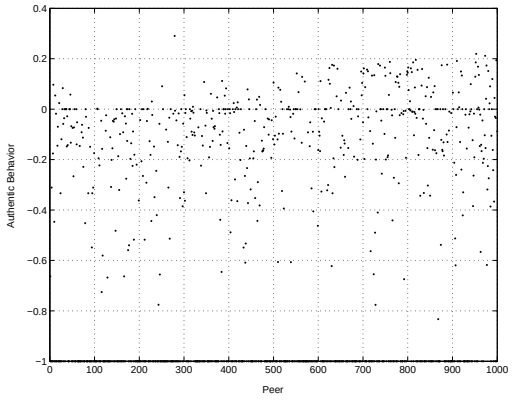

Figure 13. Authentic Behavior with *IDA* (with liar peers)

7.2.2. Performance Parameters

In this second set of simulations we will mainly focus on the following performance parameters:

- 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. The peer satisfaction is averaged over all peers.
- The percentage of malicious uploads: computed as the sum of the size of all malicious uploads performed by all peers during the simulation over the total size of all uploads.

7.2.3. Simulation Results

Figures 12 and 13 show the *Authentic Behavior* values for peers when using *IDA*. Figure 12 presents the results in a situation where no peer lies in its feedbacks, while figure 13 shows the results where there are liar peers in the system. The distribution of peers' behavior in the case where no liar peers exist is the same as in table 4 with the fourth column set to zero in all categories.

It is clear from figure 12 that *IDA* is able to differentiate among peers and detect those that send inauthentic files. Good peers (with indices from 601 to 1000) have high AB values while malicious peers (from 1 to 600) have low AB values (most of peers with indices between 1 and 300 have a value of -1). However, if liar peers exist, those peers affect badly the system and makes it difficult to differentiate among peers (c.f. figure 13). This affects greatly the performance of the system as will be shown in figure 17.

Figure 14 depicts the *Credibility Behavior* of peers when using *MDA*. The figure shows that CB is a very good indicator of the liar behavior of peers. Indeed, good peers (with indices from 601 to 1000) have a very high value of credibility

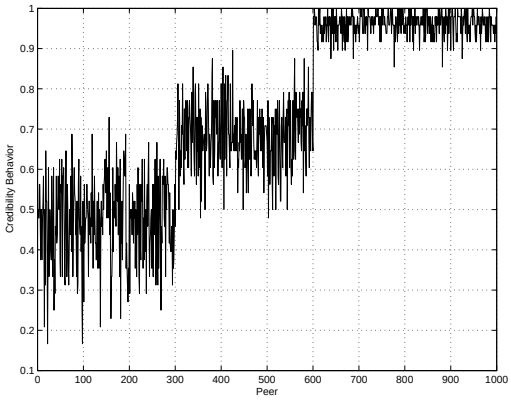

Figure 14. *Credibility Behavior*

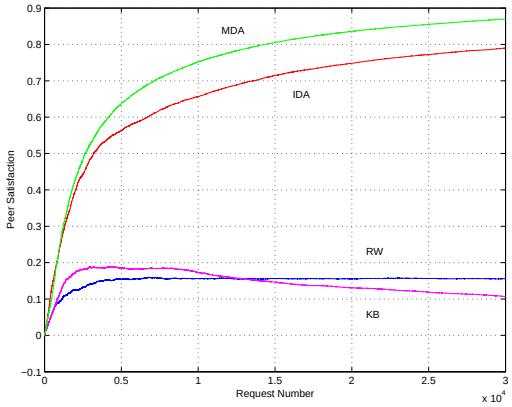

Figure 15. Peer Satisfaction

while liar peers (from 1 to 600) have lower values. This indicator is also able to differentiate among degrees of liar behavior; peers with lower probability of lying (indices from 301 to 600) have higher credibility than those with higher probability of lying (indices 1 to 300).

Figure 15 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. According to the figure, it is clear that the *MDA* and *IDA* 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. The *RW* scheme chooses peers randomly and hence the results observed from the simulations (i.e. 15% satisfaction) can be explained as follows. With the values of table 4, we can expect to have $(99\% \times 40\% + 50\% \times 30\% + 10\% \times 30\% =)$ 57.6% of authentic uploads and $(1\% \times 40\% + 50\% \times 30\% + 90\% \times 30\% =)$ 42.4% 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 $(57.6 - 42.4)/(57.6 + 42.4) = 15.2\%$.

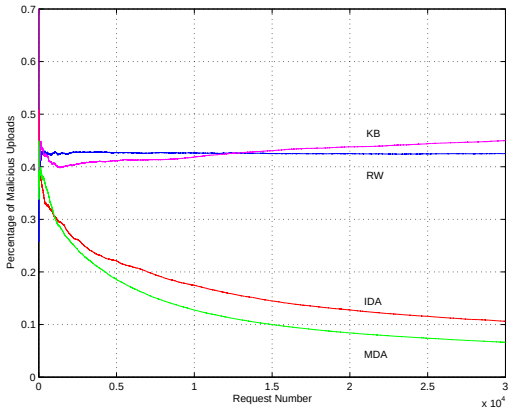

Figure 16. Percentage of malicious uploads

Our schemes (*MDA* and *IDA*) make the distinction and do not choose a peer if it is detected as malicious. Since *MDA* is able to detect liar peers, it can protect the system from them and hence is able to take the right decision when choosing a peer to download from. In *IDA*, liar peers affect the *Authentic Behavior* values of other peers and hence lower the achieved peer satisfaction.

Figure 16 shows the percentage of inauthentic file uploads. *KB* has the worst results compared to other schemes as explained earlier. *IDA* can quickly detect inauthentic peers and avoid choosing them for uploads. This isolates the inauthentic peers and controls the size of malicious uploads. However, since *IDA* does not detect liar peers, the reputation of peers is affected as shown in figure 13. This will sometimes result in bad decisions. *MDA* on the other hand takes into consideration liar behavior and thanks to the *Credibility Behavior* parameter, is able to reduce the effect of liar peers on the system. This allows the system to take more clearsighted decisions. This, of course, results in using the network bandwidth more efficiently and higher peer satisfaction as shown in figure 15. Figure 17 shows that the new scheme achieves about 40% improvement in comparison to *IDA*. This gain will continue to increase with the number of requests as *MDA* makes more and more good decisions.

Note that our scheme achieves good performance even if we have a high number of malicious peers. As stated earlier, without any reputation management scheme we can expect 42.4% of inauthentic uploads. After the 30000 requests considered, our scheme reduces this to about 6% with a peer satisfaction of almost 88%.

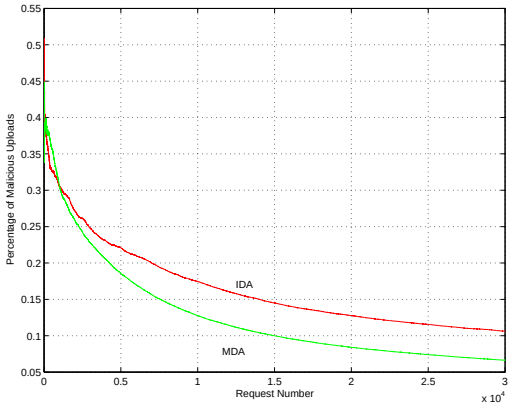

Figure 17. Percentage of malicious uploads for MDA and IDA

8. Open Issues

If providing an appreciation is manually performed and a rewarding mechanism is not provided, the system may not be able to collect sufficient amount of appreciations to compute reputation scores accurately. Some of the incentives for peers to provide appreciations are high priority and extensive search requests from their supernodes. Usually, a peer sends search requests to download a file. If the peer is sending several search requests without sending appreciations, the supernode can suspect that the peer is not providing appreciations and assigns a lower priority value for search requests from this peer.

Some countermeasures have also to be set to protect from peers that report a value for $A^F_{i,j}$, although no file download has happened. Some peers could fake additional identities and use this scheme to increase their own reputation. One suggested approach to solve this issue is to use a token given by the supernode of the sender peer that has to be sent with the feedback. This way, the supernode receiving the feedback can check whether this is a valid feedback for a download that has indeed occurred. These issues are under investigation and left for future research.

9. Related Works

9.1. Reputation in Centralized P2P Systems

Reputation Management in eBay

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. The reputation is calculated by assigning 1 point for each positive com-

ment, 0 points for each neutral comment and -1 point for each negative comment.

The eBay feedback system suffers from the following issues:

- Single point of failure
- It is not an easy task to look at all feedbacks for a member in the way the feedback profile is presented on eBay.
- A seller may have a good reputation by satisfying many small transactions even if he did not satisfy a transaction with a higher amount.
- No special mechanism is provided to detect members that lie in their feedbacks.

9.2. Reputation in Completely Decentralized Systems

Several reputation management schemes have been proposed for completely decentralized P2P systems. As these schemes do not apply for partially decentralized systems which is our target in this paper, we will present only the following approaches.

9.2.1. Reputation Management by Choosing Reputable Servents

In [10], the distributed polling algorithm P2PRep is used to allow a servant p looking for a resource to enquire about the reputation of offerers by polling its peers. After receiving a response from all offerers available to provide p with the requested resource, p selects a set of servants from offerers and polls its peers by broadcasting a message asking them to give their opinion about the reputation of the servants. Two variants of the algorithm are provided. In the first variant (the basic polling), peers send their opinion and p uses the vote to determine the best offerer. In the second variant of the algorithm (the enhanced polling), peers provide their own opinion about the reputation of the offerers in addition to their identities. This latter will be used by p to weight the vote received.

This scheme incurs considerable overhead by polling peers for their votes. In addition, the basic polling algorithm checks whether the voters have provided the vote by exchanging *TrueVote* and *TrueVoteReply* messages. In the enhanced polling algorithm, *AreYou* and *AreYouReply* messages are used to check the identity of the voters. This will further increase the network overhead. Moreover, each peer has to keep track of past experiences with all other peers. In addition, the reputation of the peers is used to weight their opinions, however, as we have shown earlier the reputation score (i.e. *Authentic Behavior*) is not enough to reflect the credibility of a peer.

9.2.2. Reputation Management using EigenTrust Algorithm

In [11], the EigenTrust algorithm assigns to each peer in the system a global trust value. This value is based on its past uploads and reflects the experiences of all peers with the peer. This scheme is a reactive one, it requires reputation 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 as it requires long periods of time to collect statistics and compute a global rating. Most of the proposed reputation management schemes for completely decentralized P2P systems are reactive and hence suffer from this drawback. Moreover, they tend to consider the reputation (i.e. *Authentic Behavior*) as a score for the credibility of a peer which was shown in this paper to be ineffective.

9.2.3. Limited Reputation Sharing in P2P Systems

In [17], the proposed algorithms use only limited reputation sharing between nodes. Each node records statistics and ratings regarding other peers. As the node receives and verifies files from peers, it updates the stored data. Nodes can share their opinions and incorporate them in their ratings. In the proposed voting reputation system, the querying node receives ratings from peers and weights them accordingly to the ratings that the peer has for these peers to compute a quorum rating. The peers can be selected from the neighbor list (Neighbor-voting) or from

the friend list (Friend-voting). In the latter case, friends are chosen from peers who have proven to be reputable. Note that a peer can be reputable (i.e. with high *Authentic Behavior*), but not credible (i.e. with low *Credibility Behavior*). In [17], no mechanism is given to detect liar peers.

9.3. Reputation in Partially Decentralized Systems

Reputation Management in KaZaA

KaZaA Media Desktop (KMD) [3] uses *Integrity Rating Files* for rating files and *Participation Level* for rating peers. Users can rate the files they share based on technical quality and accuracy of their descriptive file data. A file can be given the following ratings: *excellent*, *average*, *poor*, or *delete file*. In KaZaA a *Participation Level* is assigned to each user in the network based on the files that are uploaded from the user and the files the user downloads from other users. The participation Level is: (Uploads in MB/Downloads in MB)*100. In KaZaA, malicious peers can still upload inauthentic files as shown in the performance evaluation section (cf. section 7). This will reduce the peers' satisfaction and waste network resources.

10. Conclusion

In this paper, we propose a reputation management scheme for partially decentralized peer-to-peer systems. Reputation data is stored at the supernode level to take advantage of the use of supernodes in partially decentralized P2P systems. Two selection advisor algorithms were proposed for assisting peers in selecting the most trustworthy peer to download from. The Malicious Detector algorithm is able to detect liar peers who are sending wrong feedback and the new concept of *suspicious transactions* is introduced to detect these liar peers. 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 among peers. To our knowledge, we are the first to represent the reputation of peers using two values, one for the *Authentic Behavior* and one for the *Credibility Behavior*, which characterizes more effectively the real behavior of peers. Performance evaluations show that our schemes are able to detect and isolate malicious peers from the system hence providing higher peer satisfaction and better network resource utilization.

REFERENCES

1. A.Oram, Peer-to-Peer:Harnessing the Power of Disruptive Technologies, O'Reilly Books, 2001, Ch. 2, pp. 21–37.
2. Gnutella Protocol Specification v0.4, `http://www9.limewire.com/`.
3. KaZaA, `http://www.kazaa.com/`.
4. Morphus, `http://www.morphus.com/`.
5. Gnutella2 Specification, `http://www.gnutella2.com/`.
6. S. Androutsellis-Theotokis, A Survey of Peer-to-Peer File Sharing Technologies, Tech. rep., ELTRUN (2002).
7. VBS/GWV.A, `www.commandsoftware.com/`.
8. E. Adar, B. A. Huberman, Free Riding on Gnutella, Tech. rep., HP (2000).
9. K. Aberer, Z. Despotovic, Managing Trust in a Peer-2-Peer Information System, in: The 9th International Conference on Information and Knowledge Management, Atlanta, USA, 2001, pp. 310–317.
10. F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, P. Samarati, Choosing Reputable Servents in a P2P Network, in: The 11th International World Wide Web Conference, Honolulu, USA, 2002, pp. 376–386.
11. S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, The EigenTrust Algorithm for Reputation Management in P2P Networks, in: The 12th International World Wide Web Conference, Budapest, Hungary, 2003, pp. 640–651.
12. M. Gupta, P. Judge, M. Ammar, 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, USA, 2003, pp. 144–152.
13. L. Xiong, L. Liu, PeerTrust:Supporting Reputation-Based Trust for Peer-to-Peer

Electronic Communities, IEEE Transactions on Knowledge and Data Engineering 16 (7) (2004) 843–857.
14. S. Rafaeli, D. Hutchison, A survey of key management for secure group communication, ACM Computing Surveys 35 (3) (2003) 309–329.
15. K. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, J. Zahorjan, Measurement, Modeling, and analysis of a Peer-to-Peer File Sharing Workload, in: The 19th ACM Symposium on Operating Systems Principles, New York, USA, 2003, pp. 314–329.
16. eBay Feedback, `http://www.ebay.com`.
17. S. Marti, H. Garcia-Molina, Limited Reputation Sharing in P2P Systems, in: ACM Conference on Electronic Commerce, New York, USA, 2004, pp. 91–101.