Reputation-based Trust Management in Peer-to-Peer File Sharing Systems

by

Loubna Mekouar

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2010

Abstract

Trust is required in file sharing peer-to-peer (P2P) systems to achieve better cooperation among peers and reduce malicious uploads. In reputation-based P2P systems, reputation is used to build trust among peers based on their past transactions and feedbacks from other peers. In these systems, reputable peers will usually be selected to upload requested files, decreasing significantly malicious uploads in the system.

This thesis surveys different reputation management systems with a focus on reputation based P2P systems. We breakdown a typical reputation system into functional components. We discuss each component and present proposed solutions from the literature. Different reputation-based systems are described and analyzed. Each proposed scheme presents a particular perspective in addressing peers' reputation.

This thesis also presents a novel trust management framework and associated schemes for partially decentralized file sharing P2P systems. We address trust according to three identified dimensions: *Authentic Behavior*, *Credibility Behavior* and *Contribution Behavior*. Within our trust management framework, we proposed several algorithms for reputation management. In particular, we proposed algorithms to detect malicious peers that send inauthentic files, and liar peers that send wrong feedbacks.

Reputable peers need to be motivated to upload authentic files by increasing the benefits received from the system. In addition, free riders need to contribute positively to the system. These peers are consuming resources without uploading to others. To provide the right incentives for peers, we develop a novel service differentiation scheme based on peers' contribution rather than peers' reputation. The proposed scheme protects the system against free-riders and malicious peers and reduces the service provided to them.

In this thesis, we also propose a novel recommender framework for partially decentralized file sharing P2P systems. We take advantage from the partial search process used in these systems to explore the relationships between peers. The proposed recommender system does not require any additional effort from the users since implicit rating is used. The recommender system also does not suffer from the problems that affect traditional collaborative filtering schemes like the *Cold start*, the *Data sparseness* and the *Popularity effect*.

Over all, our unified approach to trust management and recommendations allows for better system health and increased user satisfaction.

Acknowledgements

First of all, I thank Allah. Allah is the Almighty, Creator and Sustainer of the universe, who is similar to nothing and nothing is comparable to Him.

I present my sincere gratitude to my supervisor Professor Raouf Boutaba.

I also present my gratitude to Professor Ashraf Aboulnaga, Professor Ihab Ilyas and Professor Ahmed Karmouch.

I express my deepest gratitude to Professor Youssef Iraqi.

I would like to thank the faculty, staff and students at David R. Cheriton School of Computer Science. A special thank you for Margaret Towell and Jessica Miranda.

Dedication

I would like to dedicate my thesis to:

My father Driss, my mother Houriya, my grandmother Zineb, my sisters Aicha, Amal, Amina, Khadija and my brother Mohammed,

My husband Youssef and my daughters Imane, Ihsane and Nour,

My grandfather Alamine, my aunt Saadia and my uncle Mohammed,

My family and friends.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The most important challenge of online environments is to assure satisfactory transactions. People will usually back up from dealing with strangers. Therefore, people minimize their interactions and tend to remain in their comfort zone. Peer-to-Peer file sharing systems provide a large collection of files available for download. In traditional systems, little information is given to the user to help in the peer-selection and/or file-selection processes. For example, if a user wants to download a file, the user is given a list of peers that have the requested file. The process of selecting the right peer with no a priori information is frustrating and risky. To foster positive interactions and reduce the risk involved in P2P file sharing systems, peers need to reason about trust, and reputation systems are used to this end. Reputation systems are based on collecting information about peers' past transactions and computing a reputation value for these peers. The reputation values will be the basis for identifying trustworthy peers [1].

P2P file sharing systems can be centralized, completely decentralized or partially decentralized systems. Centralized P2P file sharing systems use a centralized directory for searching files, while downloading a file is achieved directly between peers. In completely decentralized P2P systems, all peers have equal role and responsibilities. Partially decentralized P2P systems occupy the middle ground between centralized and completely decentralized systems. In these systems, supernodes or superpeers are peers that have extra capabilities and assume more responsibilities than regular peers. A supernode act as a centralized server for the peers connected to it.

While centralized P2P systems suffer from the single point of failure, completely decentralized systems such as Gnutella are characterized by expensive search. As an example, for one query sent from the requester peer, more than 6 billion messages may be generated [2]. The high amount of traffic generated from sending a query and getting back results causes inefficiency in using network resources. Moreover, peers in completely decentralized systems have the same role although these peers have different capabilities in terms of processing power, bandwidth, memory and storage capacity, in addition to peers' uptime. Peers that have less capabilities and short uptime should not assume the same role, at least not at the same level. In contrast, the communication protocol used in partially decentralized systems reduces significantly the number of messages exchanged during the search phase. The supernode architecture offers an intermediate design, minimizes the weaknesses of both centralized and completely decentralized systems and combines the advantages of these systems.

Several reputation-based systems have been proposed for completely decentralized systems. However, all proposed research works in this field have completely focused on these systems. Almost no attention was directed toward partially decentralized systems.

Reputation is the most valuable information that helps reduce the offensive and deceptive behavior of online users. Reputation systems will motivate users to exhibit good behavior, identify malicious peers, increase users' confidence and satisfaction, and preserve network resources. However, reputable peers may be easily overwhelmed by download requests from other peers. Reputable peers need to be motivated to continue to display good behavior. Service differentiation is needed to grant services to peers that contribute positively to the system and reduce the service provided to free riders and malicious peers.

In this thesis, we propose a novel trust management framework and associated reputation management schemes for partially decentralized P2P file sharing systems. We also propose a complementary recommender system to help users in files' selection processes.

1.1 Trust Management Framework

The survey of reputation systems revealed the lack of clarity and fuzziness in the concept of reputation. The measurement and/or the computation of this important concept in some research works was not adequate. For example, the reputation of a peer in terms of sending authentic files has been used to measure its credibility and also for service differentiation which raises fairness issues. Our goal is to manage trust according to specific dimensions and clarify this ambiguity. In this thesis, we design a trust management framework and related algorithms for partially decentralized P2P systems. We propose to address trust according to the following dimensions: 1) *Authentic Behavior*, 2) *Credibility Behavior*, and 3) *Contribution Behavior.*

- *Authentic Behavior*: represents peer's reputation in sending authentic files in terms of accuracy and technical quality [3]. The *Inauthentic Detector Algorithm* is proposed to identify malicious peers and isolate them. As a result, malicious uploads are reduced.

- *Credibility Behavior*: represents peers' credibility in sending honest feedbacks [3]. The *Malicious Detector Algorithm* is proposed to distinguish between honest and liar peers in addition to good and malicious peers. As a result, the impact of liar peers is reduced.

- *Contribution Behavior*: represents peers' contribution to the system [4]. A *contribution based service differentiation* scheme is proposed to differentiate between peers that contribute positively to the system, and free riders and malicious peers.

Initially, we used the *Authentic Behavior* to capture the maliciousness of peers in terms of sending inauthentic files. However, liar peers had an impact on the reputation system and hence, there was a need to measure the *Credibility Behavior* in terms of sending honest feedbacks. Also, to differentiate among peers, a service differentiation scheme is required. To provide the right incentives for peers to exhibit good behavior, the *Contribution Behavior* dimension is used to measure the positive contribution of the peers to the system. The three dimensions are required to minimize the risk involved in the transactions and to maximize users' satisfaction.

In this thesis, we take advantage from the use of supernodes in the partially decentralized architecture. Since supernodes are already used to handle search requests on behalf of the peers connected to them, we use supernodes to handle trust data and compute peers' reputation. Supernodes are also used to compute peers' credibility and enforce service differentiation.

1.2 Recommender Framework

In P2P file sharing systems, peers have to choose the files of interest from a large collection of files. Peers spend a significant amount of time looking for relevant files. To alleviate the peers from this task, recommender systems can be used to make personalized recommendations to the peers according to their profile.

While reputation systems are used to rate peers in P2P systems, recommender systems are used to rate files. In reputation systems, peers are rated based on the accuracy and the technical quality of files, while in recommender systems, files are rated based on the content.

In this thesis, we propose a novel recommender framework for partially decentralized P2P file sharing systems [5]. Based on peers' profile, the proposed recommender system will suggest files that peers will most probably like. While these peers are downloading these files, they will upload files to others increasing their contribution. We investigate different similarity metrics that were proposed in information retrieval, exploratory data analysis and other fields and adapt them to our context. We analyze the effect of similarity metrics on the performance of the recommender system. Both weighted and non weighted approaches are studied. Weighted approaches achieve higher recommendation accuracy. Within the weighted approaches, similarity metrics that do not consider negative co-occurrence lead to better recommendation accuracy. To overcome the problems of traditional collaborative filtering recommender systems, an implicit rating approach is used.

1.3 Thesis Organization

The rest of the thesis is organized as follows:

Chapter 2: Reputation-based Management: Taxonomy and Anatomy
Section 2.1 presents an introduction to peer-to-peer systems. Section 2.2 describes traditional P2P systems versus reputation-based P2P systems. Section 2.3 provides a definition of trust, reputation and their properties. An analysis of the system model is presented in Section 2.4. In Section 2.5, an anatomy of reputation systems in file sharing environment is presented. A breakdown into functional components is realized and each component is described based on the proposed mechanisms from the literature. Section 2.6 explains the local trust, followed by the reputation query in Section 2.7 and reputation computation in Section 2.8. Section 2.9 describes the use of reputation. Credibility assessment is explained in Section 2.10 while Section 2.11 focuses on the incentives, rewards and punishment. Section 2.12 presents a survey of centralized reputation systems followed by a survey of completely decentralized reputation systems in Section 2.13. Section 2.14 highlights reputation schemes in partially decentralized systems. Section 2.15 presents the design requirements of reputation systems. Finally, Section 2.16 provides concluding remarks.

Chapter 3: Trust Management: Authentic Behavior
Section 3.1 describes the proposed trust management framework and explains each trust component. Section 3.2 presents the motivation behind the proposed solution. Section 3.3 presents the proposed reputation management scheme namely the *Inauthentic Detector Algorithm* followed by two Selection Advisor Algorithms that are described in Section 3.4. Section 3.5 presents a mathematical analysis of the *Inauthentic Detector Algorithm*. Section 3.6 presents the performance evaluation of the proposed schemes followed by concluding remarks in Section 3.7.

Chapter 4: Trust Management: Credibility Behavior
Chapter 4 describes the second dimension of trust. Section 4.1 presents an analysis of peers' behavior. Section 4.2 discusses the proposed approaches to detect malicious peers and introduces the concept of *Suspicious Transactions* in addition to the *Malicious Detector Algorithm.* Section 4.3 presents a mathematical analysis of the *Malicious Detector Algorithm.* Finally, performance evaluation is presented followed by a conclusion.

Chapter 5: Trust Management: Contribution Behavior
Chapter 5 describes the third dimension of trust. Section 5.1 presents the motivation behind this work. Section 5.2 presents the *Contribution Behavior* and describes how peers' contribution is computed. Section 5.3 explains the use of trust components in the trust management framework. Section 5.4 presents a service differentiation scheme based on peers' contribution. Section 5.5 presents the rational behavior adopted by peers. Section 5.6 shows the performance evaluation of the proposed scheme followed by a conclusion in Section 5.7.

Chapter 6: Personalized Recommendations
Section 6.2 discusses content-based versus collaborative filtering techniques and describes the user-based collaborative filtering technique. This section also highlights important recommender schemes used in e-Commerce and presents the challenges faced by the collaborative filtering algorithms. Section 6.3 discusses related works in P2P systems. This section also presents the goals that recommender systems strive to achieve and the features used for recommender system evaluation. Section 6.4 explains the proposed recommender framework. Section 6.5 describes the proposed *Popularity Based Recommendation* and *Asymmetric Peers' Similarity Based Recommendation with File Popularity* metrics. This section also presents the similarity metrics adopted in this work. Section 6.6 describes the performance evaluation conducted and presents an analysis of the achieved recommendation accuracy by the different similarity metrics. Finally, Section 6.7 concludes the chapter.

Chapter 7: Conclusion and Future Work
Section 7.1 highlights our contribution in this thesis including the proposed trust management framework and the recommender framework. Section 7.2 describes possible future research directions and finally, Section 7.3 presents concluding remarks.

Chapter 2

Reputation-based Trust Management: Taxonomy and Anatomy

In this chapter, we will survey reputation-based P2P systems. We will also breakdown a typical reputation system into functional components. For each component, we identify the important solutions proposed in the literature. We will describe the proposed schemes from the literature and show how they address peers' reputation.

2.1 Introduction and Motivation

In a P2P 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 [6]), use a centralized control server to manage the system while downloading a file is achieved directly between peers. These systems suffer from the single point of failure, scalability and censorship problems. Decentralized unstructured 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 [7]) have no hierarchical structure between the peers and all peers have equal role and responsibilities. Partially-decentralized P2P systems (e.g., Skype [8], KaZaA [9], Morpheus [10], and Gnutella2 [11]) occupy the middle ground between centralized and completely decentralized systems. These systems have been proposed to reduce the control overhead needed to run the P2P system by the use of "superpeers" or "supernodes". Supernodes are peers that have extra capabilities and assume more responsibilities than regular peers. A supernode act as a centralized server for the peers connected to it. Decentralized structured P2P systems like CAN [12] and CHORD [13] are based on Distributed Hash Tables (DHT) in that they use a hash function to deterministically map keys such as file names into points in a logical coordinate space. Peers are responsible for storing (key, value) pairs that are hashed into a point that is located within their region. Most P2P systems deployed on the Internet are unstructured.

In an open P2P system, peers often have to interact with unknown peers and need to manage the risks involved in these interactions. For example, if a user wants to download a file, 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 no help is given on how to choose the right peer. After the download has finished, the user has to check the received file for malicious content and that it actually corresponds to the requested file (i.e., the requested content). If the file is corrupted, 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. To solve this problem, peers need to be able to reason about trust in order to avoid untrustworthy peers and reduce risks. Trust management [14] is any mechanism that allows to establish mutual trust which allows peers to cooperate, and obtain in the long term an increased utility for the participating peers.

In [15], the authors classify trust management systems into three categories:

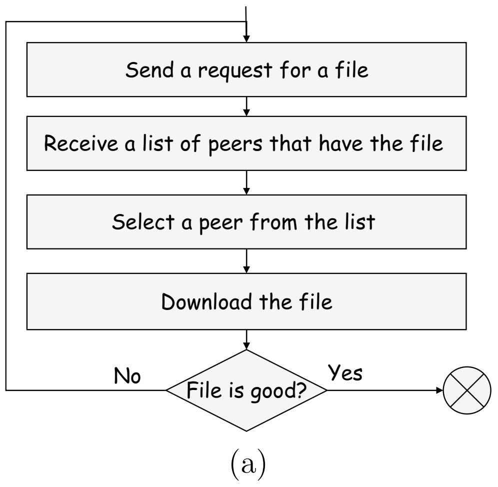
- Credential-based trust management systems: in these systems, service providers and the provided services are trusted, but service requesters are not. Service providers use credentials to evaluate the trustworthiness of service requesters and services may be granted or not.

- Reputation-based trust management systems: in these systems, service providers and provided services are not trusted. Service requesters select service providers based on their reputation values. Reputable service providers are selected to provide the service.

- Social network-based trust management systems: these systems are based on social networks. Reputation is computed based on social relationships.

The focus in this work is on reputation-based trust management in P2P file sharing systems. Reputation-based P2P systems [14, 16, 17, 18, 19, 20, 21, 22, 23] were introduced to build trust among peers. These systems try to evaluate the transactions performed by the peers and associate a reputation value to each peer. The reputation values will be used as selection criteria among peers.

In this chapter, we will investigate the existing research work proposed to address reputation in P2P systems. We will present existing reputation management schemes in centralized, completely decentralized and partially decentralized P2P systems. We will


Send a request for a file
Receive a list of peers that have the file
Select a peer from the list
Download the file
No
File is good?
Yes
(a)

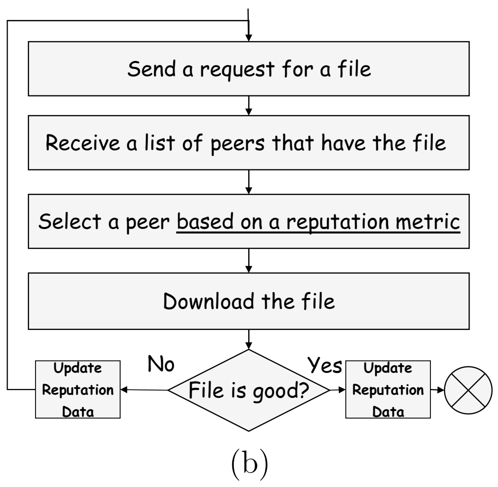

Send a request for a file
Receive a list of peers that have the file
Select a peer based on a reputation metric
Download the file
Update Reputation Data
No
File is good?
Yes
Update Reputation Data
(b)

Figure 2.1: (a) Life Cycle in a Traditional P2P System, (b) Life Cycle in a Reputation-based P2P System

describe these schemes and identify key properties for each reputation system to summarize the efforts of researchers in addressing peers' reputation.

2.2 Traditional Systems vs Reputation based Systems

The following is the life cycle of a transaction in a traditional P2P system (figure 2.1.a):

1. Send a file request (either to the supernode or to other peers depending on the P2P system)
2. Receive a list of peers that have the requested file
3. Select a peer
4. Download the file

The following is the life cycle in a reputation-based P2P system (figure 2.1.b):

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

Most reputation management schemes try to achieve the following goals:

1. Make informed decisions about transaction partners and choose appropriate peers to download files
2. Isolate malicious peers from the network by downloading files only from reputable peers, hence reducing malicious uploads
3. Increase users' confidence and satisfaction
4. Help in the bootstrapping process to select peers to connect to in the overlay network.
5. Motivate peers to exhibit good behavior
6. Use the network resources more efficiently

2.3 Trust & Reputation

2.3.1 Trust Definition

Trust is as old as the existence of human beings. People were grouped in tribes and within the same tribe, they trusted each other. The concept of trust has a significant role in the surviving of human beings. We experience and rely on trust on daily basis. However, trust is difficult to define clearly and precisely.

Researchers from different fields such as psychology, sociology, philosophy, history, law, business and economics have tackled the concept of trust from different views. According to the Oxford dictionary [24], *trust is a firm belief in the reliability, truth, ability, or strength of someone or something.*

In 1973, Deutsch [25] has specified that *trust is the confidence that an individual will find what is desired from another, rather than what is feared.*

In 1990, Diego Gambetta [26] defined trust as follows: *Trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent will perform a particular action, both before we can monitor such action (or independently of his capacity of ever to be able to monitor it) and in a context in which it affects our own action.*

In 1996, McKnight and Chervany defined trust as *Trust is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security even though negative consequences are possible.*

In 2000, Grandison and Sloman [27] defined trust as *The firm belief in the competence of an entity to act dependably, securely and reliably within a specified context.*

In this chapter, we adopt the following definition proposed in 2006 by Chang et al. [28]. Trust is defined as *The belief the trusting agent has in the trusted agent's willingness and capability to deliver a mutually agreed service in a given context and in a given time slot.*

Based on our experience in the physical world, we extract the necessary information that can help us build trust in the virtual world to increase users' confidence and reduce the risk. Marsh [29] is one of the first researchers to give a formal model of trust that can be used in computer science. This model is based on social properties of trust taken from sociology.

2.3.2 Reputation Definition

Reputation has been widely used in different disciplines such as psychology, sociology, business and economics. From the Oxford dictionary, *reputation is the beliefs or opinions that are generally held about someone or something.* Abdul Rahman et al. [30] define reputation as "*an expectation about an agent's behavior based on information about its past behavior*". Sabater et al. [31] define it as an "*opinion or view of one about something*". Mui et al. [32] define it as "*the perception that an agent creates through past actions about its intentions and norms*". In service-oriented environments, Chang et al. [28] define reputation as *"an aggregation of the recommendations from all of the third-party recommendations agents and their first, second and third hand opinions as well as the trustworthiness of the recommendation agent in giving correct recommendations to the trusting agent about the quality of the trusted agent*".

2.3.3 Trust Properties

Trust is personal, subjective and it is based on various factors. Trust is fuzzy since trust is imprecise and vague. Trust is dynamic since it is not stable and it changes as time goes by. Trust is also complex since different ways are possible for determining trust [28].

The trust relationship is usually asymmetric. The transaction between the trusting peer and the trusted peer results in a trust value assigned by the trusting peer to the trusted peer. This value shows the strength of the trust relationship. The trust relationship can be transitive.

2.4 System Model

2.4.1 Modeling The Network

Centralized P2P systems use a centralized index for the files shared by each peer. It simplifies the direct exchange and the sharing of files between peers. However, it represents a single point of failure which reduces the reliability of the system. In completely decentralized P2P systems, a central authority for storing data and handling all the queries is not available. Interconnected peers are able to participate in transactions by interacting with each other and make local autonomous decisions to achieve their objectives. Peers are responsible for storing, sharing information and handling the queries. Peers act as clients and request services from other peers as well as servers and provide services to other peers. These systems provide improved robustness and enhanced scalability compared to centralized systems. In [15], the authors indicate the interplay between the degree of autonomy and the requirement of trust. In these systems, peers are autonomous which leads to loosing control over peers. Since no global view of the system is available, peers will get a partial and limited perspective of the system by receiving information from other peers. Trust is more needed to facilitate the interaction among peers. In partially decentralized P2P systems, some peers (named "superpeers" or "supernodes") act as local central indexes for files shared by local peers. These systems provide lower discovery time as the discovery process involves only the supernodes.

In P2P systems, the following terms are used:

- Peers, users, nodes, agents: peers issue queries and reply to them
- Files, resources, items: the requested items from requester peers.
- Transactions: interactions and experiences between peers. In P2P systems, these interactions can be sharing a file, CPU cycles, or disk storage. A transaction in a P2P system involves:
  - The requester peer: the peer that requests the service.
  - The provider peer: the peer that has the requested service and is available to provide it to the requester peer.
  - Intermediary peer: the peer that is involved in a transaction. This peer participates to accomplish the transaction and to fulfill the request (e.g., forwarding messages). The intermediary peer is also referred to as a third party peer or a mediator or a queried peer.

  - Recommender peer: peer that sends a feedback to the requester peer.
  - Feedback, opinion, rating, experience, appreciation: a peer sends a feedback as a result of a transaction. A rating is one trust value, while an opinion is considered as an aggregation of all trust values based on previous transactions with a provider peer.

2.4.2 Modeling The Nodes

In a P2P file sharing system, peers are expected to exhibit good behavior. Ideally, peers are expected to be trusted, that they will share good quality files, that they will upload requested files, and that they will send honest feedbacks. Unfortunately, real life P2P systems have proved that a mechanism is needed to measure explicitly trust in order to deal only with trustworthy peers.
There are two types of users' behavior: Good peers and malicious peers.

Good Peers

Those are the well-behaved peers that will usually send an authentic file. The following is the expected behavior from good peers:

- Availability: available to share a file, to forward a query, to reply to a query.
- Contribution: a peer contributes positively to the system by uploading authentic files.
- Credibility/Honesty: Upon receiving a reputation query, a recommender peer sends an honest feedback.

Malicious Peers

Peers that misbehave to impact badly the system. The adversary is another term to refer to malicious peers as used in [33]. The maliciousness of peers can appear in different ways and malicious peers can act individually or by forming collectives. Malicious peers can be one or more of the following:

- Senders of inauthentic data: these peers upload inauthentic and corrupted data and send spurious information to pollute the system.
- Liar peers: these peers lie in their feedbacks.

- Free riders: take advantage from the system without contributing to it. Although these peers do not harm other peers, they do have an impact on the performance of the system. In the literature, they are also referred to as selfish peers, free-loaders [18] or rogue peers [34].

- Traitors: peers that induce the milking phenomenon. These peers provide authentic files for a while to get a high reputation value before starting uploading inauthentic files.

- Colluding Peers: these peers know each other and work together to form a malicious collective. These malicious peers can upload inauthentic files, decrease the reputation of good peers (i.e., defaming), or increase the reputation of other malicious peers (i.e., flattering).

- Whitewashers: these peers join the system, act maliciously or misbehave then leave and re-join with new identities to get a new reputation value and eliminate the bad reputation received previously. In general, it is difficult to trace these peers and prevent the system from considering them as regular new comers. Systems with weak identities are vulnerable to whitewashing. Examples of these systems include KaZaA, Gnutella, and EigenTrust.

- Sybil Attacks: the malicious attacker creates multiple identities allowing the attacker to control a whole portion of the network.

- Denial of Service (DoS) attackers: these peers consume large amount of resources from the system to subvert the whole system and bring it down. In P2P systems, detection, management, and the prevention from these attacks is still an open problem.

- Unreliable: they can promise availability of specific services and do not deliver them.

- Imposter: they can pose as other peers.

Malicious peers act maliciously for different reasons, including personal characteristics, psychological, social, and economic factors. For example, movie and music industry pollutes P2P networks to minimize the use of these systems and to push users to buy their products rather than sharing them for free using P2P file sharing applications. Polluting these systems is achieved by distributing files with legitimate titles but with silence or random noise.

Peer Identity

To distinguish between peers, identity is required. Each peer in the system is identified by its identity that must be kept the same during its lifetime. As an example, users in Nice choose a PGP style identifier. The identifier includes a plain text identification string and a public key. Desirable characteristics of peers' identity related to trust include:

- Longevity: peers need to keep the same identifiers during their lifetime [35]. It should be difficult for peers to change their identity easily (i.e., whitewashing) and start over by obliterating their past behavior.
- Anonymity: according to the application used, peers' identity can belong to different levels of anonymity. Most P2P applications use a simple, user-generated pseudonyms. In some applications (e.g., freenet), onion routing is used to provide the anonymity of peers. However, since the peer identity is protected, it will be difficult to track the actions done by this peer. In general, there is a tradeoff between anonymity and trust.
- Unforgeability: Unforgeable identities are generated by a central trusted entity or a set of trusted entities and are given to new comers when they join the system. Each user is given only one identity. In TrustMe, for example, users are assigned unforgeable identities. Unforgeable identity is used to protect from whitewashing (i.e., change of identity to eliminate peer's history) and sybil attacks (i.e., the use of multiple identities).
- Spoof-resistance: to prevent malicious peers from impersonating other peers identities.
- Sybil-attack resistance: peers identity should be resistant to sybil attacks. Some mechanisms have been proposed to slow the creation of new identities such as imposing an entry cost or a registration fee (e.g., Nice), requiring from users to solve a puzzle that can not be done by a computer, or read some text from a JPEG file.

New Comers Policy

When new comers join the system, these peers do not possess any local information regarding other peers in the system. In addition, all the peers in the system do not know yet the reputation of these new comers. Since new comers to the system are strangers and their behavior is not known yet, two policies can be adopted:

- The optimistic approach: new comers are trusted. However, other peers may be disappointed.

- The pessimistic approach: new comers are not trusted. However, in this case, new comers may not be able to build their reputation.

Reputation systems should welcome new comers and give them a chance to increase their reputation gradually. At the same time, these systems should protect peers from a new comer in case it turned out to be malicious.

New comers have to assume their responsibility in establishing trust with other peers by uploading authentic files. They should build and maintain good reputation to become reputable peers.

2.4.3 Modeling The Reputation Management Infrastructure

In centralized P2P systems, there is a central entity where information regarding peers' actions can be gathered and accumulated. Peers are controlled and malicious peers can be identified somewhat easily. Peers can be protected against malicious actions performed by malicious peers. For these systems, reputation management is also centralized since the central entity can handle the trust information in addition to the lookup mechanism. The use of a centralized reputation management infrastructure simplifies significantly the management of trust information.

In completely decentralized systems, malicious peers have more freedom to act maliciously and perform variety of attacks and it is more difficult to gather all the actions performed by the peers. In these systems, peers should protect themselves from malicious peers since they can not be identified easily and hence isolated from the system. Peers are required to gather information to assess the reliability of peers in providing the requested service. In these systems, a centralized reputation management entity is possible. [18] is an example where the Reputation Computation Agent RCA is used to collect reputation information and compute peers' reputation. However, the use of a centralized entity will exhibit a single point of failure and will be easy to attack. A natural possibility is to use a decentralized reputation management infrastructure to improve the robustness and the scalability of the system.

The followings are advantages and drawbacks of different types of trust and reputation management infrastructures:

- Centralized systems: use a trusted centralized server for managing reputation information. This server will handle and manage all the reputation data
    - Advantages:
        1. Easy to use and manage

    2. Simplifies the mechanism design
    3. Offers efficiency since all the reputation data is available locally
    4. Helps keeping the data consistent and coherent
  - Drawbacks:
    1. Difficult to achieve in case not all the peers can trust one entity.
    2. Represents a single point of failure and a bottleneck since millions of peers can send queries to this entity.
    3. Represents a single point of attacks by misbehaving peers such as DoS attacks, sabotage and subversion
    4. Expensive to achieve high performance and robustness
    5. Not scalable

- Completely decentralized systems: information regarding peers' reputation is stored at the peer's level and this information is scattered throughout the network.
  - Advantages:
    1. No single point of failure
    2. No need for a globally trusted entity
    3. Provides robustness and scalability
  - Drawbacks
    1. Message overhead as many messages are generated in order to maintain and manage reputation data
    2. Flooding is required to get the required information from peers in order to aggregate different local trust values
    3. The topology changes frequently due to the transient nature of peers that can join or leave the system at any time. Some reputation data may be lost or inaccessible.
    4. Reputation data is more vulnerable to tampering with.

2.5 The Anatomy of Reputation Systems

Sharing experiences between peers will ultimately lead to identifying good peers and isolating malicious ones. Reputation-based P2P systems create the appropriate environment for peers to rely on each other for downloading authentic files. Since we can not predict future peers actions, peers' past actions are used to measure their reputations.

Open systems need robust reputation management. A good reputation system should [36]:

- Identify malicious peers in order not to be selected as transactions partners: this is a precaution measure before starting the transaction.
- Spread information regarding a malicious peer in case that a negative transaction occurred: this is a retaliation measure after ending a transaction to help other peers in future interactions.

The survey of different reputation systems reveals the important mechanisms used to achieve good reputation management. The peer looking for a file is the requester peer, and peers that have the requested file are the provider peers. The components and issues involved in finding a file and downloading it in the case of a reputation-based P2P file sharing application are the followings:

1. The Local Trust: Here important issues include:
   (a) What kind of trust information is gathered?
   (b) Where to store trust information?
   (c) Is the local trust information sufficient?
2. The Reputation Query: The requester peer sends a reputation query regarding the potential provider peers. Related issues include:
   (a) To whom the requester peer sends a reputation query?
   (b) How many peers, should the requester peer contact?
   (c) What kind of information is sent to the requester peer?
3. The Reputation Computation
   (a) How to deal with liar recommender peers?
   (b) How to deal with fraudulent recommender peers?
   (c) How to compute the reputation?
4. The Use of Reputation
   (a) How to choose a peer based on the reputation value?
   (b) How to evaluate a transaction after downloading a file?
5. Credibility Assessment
6. Incentives, Rewards and Punishment

The following is a list of most important and representative reputation systems that have been proposed in the literature and will be used throughout the chapter: eBay [37], DistributedTrust [38], BinaryTrust [14], P2PBasic and P2PEnhanced [16], EigenTrust [17], Nice [39], DCRC and CORC [18], Travos [40], XRep [41], Regret [42], MDNT [28], MLE [21], LimitedReputation [19], CredibilityRecords [43], FGTrust [23], and PowerTrust [44].

2.6 The Local Trust

The requester peer sends a file search query and gets a search reply from the system containing a list of peers. The requester peer may or may not have previously interacted with some of the peers in the list. If the requester peer is not a new comer, and has already downloaded files from other peers, local trust information based on previous transactions can be used. The requester peer may:

- Eliminate from the list malicious peers that it has already interacted with.
- Accept to deal with good peers that have already provided him a good service.
- May not have any trust information for some of the peers if it did not interact with them in the past.

2.6.1 What kind of trust information is gathered?

. Some reputation management schemes use the number of negative and positive downloads (e.g., EigenTrust), other schemes use the negative downloads only (e.g., complaints in Binarytrust). For other schemes the size of the download is more important than the number of uploads. In debit-credit reputation computation DCRC and credit-only reputation computation CORC [18], peers' contribution is tracked and is considered as their reputation.

For each transaction, the following information can be gathered for each peer:

1. The type of trust information:
    - A qualitative value: positive, negative or both (e.g., satisfied and unsatisfied transactions in EigenTrust, complaints for negative transactions in BinaryTrust).
    - A quantitative value: the size of the files downloaded counted as positive for authentic files and negative otherwise.

Relying only on positive transactions will lead to dealing with malicious peers that conducted few successful transactions. While relying only on negative transactions may eliminate good peers. A combined approach is more efficient in identifying the real behavior of the peers.

2. A trustworthiness value which represents the outcome of the interaction. This value can be a binary value 0 or 1 (e.g., XREP, Travos, CredibilityRecords) which means that a requester peer is satisfied from the transaction or not, the provider peer is trusted or not trusted. The trustworthiness value can also be a discrete value (e.g., Excellent, Good, Fair and, Poor) as in Amazon and DistributedTrust, a scaled integer (e.g., 1 to 5) (e.g., MDNT) or a real value on a continuous scale from [0,1] (e.g., Nice, PeerTrust), a probability value (for EigenTrust and PowerTrust). While a binary value does not allow partial trust, a continuous value expresses better how much trust is given. However, assigning $\{0,1\}$ is much easier. The trustworthiness value can be the result of a transaction (e.g., a cookie which is a signed receipt of successful transactions in Nice) or computed based on either a qualitative or quantitative value or a combined approach. In some reputation systems, a value of 0 from [0,1] does not make a distinction between a malicious peer and a new comer to the system.

3. The time of the interaction: this can be used when computing reputation values (e.g., FuzzyTrust, Regret, MDNT, FGTrust).

4. The context of the interaction: this can differ from an application to another. An application may have different contexts. In file sharing applications, the focus is on one context which is providing authentic files.

Peers may keep track of all the records with every trusted peer (e.g., eBay) or only one record that summarizes all the transactions (e.g., TrustModel). Adopting this approach will reduce the storage cost. Keeping track of trust data will allow to benefit from the followings [28]:

- Recency: giving more weight to the recent values of the trust values. In this case, the use of *time of interaction* is important.
- Trend: it is preferable to know not only the trust value at a specific time slot but also the trend line over the last few time slots. Understanding the trend is important.
- Cyclical poor performance: keeping history of trust values according to their time slots may help in detecting such behavior.
- Variability in trustworthiness: it is important to know the variance value to detect the variability of the service provided.

It is important to note that the focus is more on the trustworthiness of peers rather than files since malicious peers can generate a large number of inauthentic files. Thus, identifying and isolating malicious peers will improve significantly the quality of the shared files. In [41], a reputation scheme is proposed for both peers and resources (e.g., files) to overcome the limitations of reputation schemes that tackle only the problem of peers' reputation. In the XRep protocol, each peer maintains two experiences repositories to keep track of authentic and inauthentic resources in addition to good and malicious peers, the peer had direct transactions with. *Credence* is another reputation scheme for identifying content that is not authentic (e.g., damaged, corrupt, missing contents, dangerous content, misleading metadata designed to confound user searches). However, computing reputations for files is usually time consuming when dealing with a large collection of files.

2.6.2 Where to store trust information?

While in centralized P2P systems, the central entity will be used to store trust information (e.g., eBay, amazon), in completely decentralized P2P systems, the trust information regarding a trusted peer can be stored at:

- The trusting peer's level: in this case, it is necessary to contact all the trusting peers that have interacted with the trusted peer to compute the trusted peer's reputation (e.g., CredibilityRecords). This task is very challenging in completely decentralized systems where trust information is scattered throughout the network.
- The trusted peer's level: in this case, trust information is gathered at the trusted peer's level (e.g., EigenTrust). This makes the reputation computation process easier, however, malicious peers may manipulate this information. In DCRC and CORC, each peer is responsible of storing its own reputation for fast retrieval. To prevent malicious peers from thwarting the reputation system, reputation scores are signed by the RCA private key. To avoid that trusting peers drop negative reputation scores resulting from downloading files, the RCA keeps the negative scores until these peers send credits for contributing to the system.
- Both the trusting peer's and the trusted peer's levels: since negative transactions may be deleted from the trusted peers records, the trusting peers keep this information (e.g., Nice).
- Third-Party peers: trust data is stored at another peer instead of the trusting or the trusted peers. If the trusted peer is responsible to store its reputation data, it can easily manipulate it. To prevent against such threat, another peer is selected to be responsible for storing this data. Another problem that may occur is that this

intermediary peer may report false trust information. To solve this issue, a set of peers is responsible to store trust information of each peer. This way, minimizing the impact of false trust reports. As an example, in secure EigenTrust, a distributed hash table is used to assign to each peer, score managers that are responsible to store and compute peers' global trust values. Also in BinaryTrust, the storage of the output of negative transactions is realized by other peers.

Since the local trust information stored at each peer's level in completely decentralized P2P systems is proportional to the size of the network, the storage cost increases linearly as the number of peers increases. Trust management schemes should be able to minimize the storage cost.

When storing reputation data at peer's level, reputation-based P2P systems need to provide mechanisms to ensure that reputation data is stored securely to prevent malicious peers from thwarting the reputation system and also to ensure the availability of trust information even with the transient nature of peers in P2P systems.

2.6.3 Is the local trust information sufficient?

At this stage, it is important to know if it is sufficient to choose the trusted peer according to local personal trust information or maybe it is necessary to get feedbacks form other peers in the system. In case where a few well-behaved peers know each other and always exchange files between them, local trust information is sufficient for making decisions. In systems where millions of peers are available to share files, local trust information is very limited. Therefore, relying on external advice from other peers is imperative. Reputation systems help in predicting peers' future behavior based on their past behavior. Since reputation is based on peers' opinions, the credibility of the peers must also be assessed.

2.7 The Reputation Query

To gather more information regarding peers that have the requested file, the requester peer needs to ask other peers about their opinions. The requester peer queries about the requested peers' reputations in a specific recommendation context.

In centralized reputation systems, there is no need to gather reputation data from other peers since the information is already collected and stored at the central entity. In [18], the RCA facilitates the collection of trust data and the computation of reputation scores. While in completely decentralized reputation systems, on demand processing of peers' reputation is needed by querying peers and collecting reputation data to compute peers' reputation.

In reputation-based systems, the reputation scheme collects, distributes and aggregates feedbacks about peers' past behavior. This scheme helps peers to identify trustworthy peers and decide whom to trust. However, for reputation systems to operate effectively is very challenging. These challenges will be detailed in the following sections.

2.7.1 To whom the requester peer sends a reputation query?

In the physical world, when recommendations are needed for a person for example, it is not practical to send a query to any person. It is preferable to target the people who know better this person and ask for their feedbacks (e.g., co-workers, friends). The same process can be applied to the virtual world, however, in completely decentralized P2P systems, it is difficult to find the peers who interacted previously with the provider peer.

The followings can be the receivers of the reputation query:

1. The neighbors of the requesting peer: the requester peer can use its neighbors in the overlay network to get reputation information. For example, in the basic polling (P2PBasic) and the enhanced polling (P2PEnhanced) algorithms, the requester peer broadcasts a reputation query, asking for the reputation of the provider peers. In LimitedReputation, the requester peer sends a reputation query to its neighbors (Neighbor-voting).

2. The peers that have already interacted with the requester peer and that are trustworthy. Another alternative in this approach is to ask the trustworthy peers to ask the trustworthy peers that they have interacted with (e.g., EigenTrust). Using the transitive trust, the requester peer gets more recommendations. In FuzzyTrust, recommender peers are chosen based on their number of performed transactions and local trust values. In LimitedReputation, the requester peer can also select recommender peers from its friends (Friend-voting), where friends are peers who have proved to be reputable.

3. The peers that have already provided the requester peer with accurate recommendations in previous queries (e.g., MDNT): the requester peer keeps track of past recommendations and uses metrics to identify good recommender peers. These recommender peers are knowledgeable peers and their opinions/ratings are accurate.

4. Specific selection of peers: as an example of social-based trust systems, Regret groups peers according to their relationships with the trusted agent. The most representative agent is selected as a recommender peer based on fuzzy rules.

5. Random selection of peers: the requester peer chooses a random set of peers as receivers of the reputation query.

2.7.2 How many peers should the requester peer contact?

The requester peer can send a reputation query asking for recommendations to the followings:

- A small number of peers: for example, a Time To Live (TTL) can be used to reduce the generated communication overhead. The forwarding of the query continues until the TTL is exhausted. However, with the TTL, the requester peer will not get a wide view of the peers' reputation, but the communication overhead will be reduced significantly. In case of Nice, a query is forwarded to only 5 users instead of following all the paths from the requester peer.
- A large number of peers: the more information is gathered, the more precise is the reputation of the provider peers. However, this approach incurs additional communication overhead. Each forwarded message consumes bandwidth and processing at each peer it visits. This approach may impact badly the performance and the scalability of the system.

In general, there is a tradeoff between accuracy and communication overhead. The larger the number of feedbacks received from recommender peers, the more accurate is peer's reputation. The number of recommender peers is determined according to the requirements of the P2P application. In addition, if providing a feedback is manually performed and a rewarding mechanism is not provided, the system may not be able to collect a sufficient number of feedbacks to compute reputation scores accurately.

The feedbacks sent to the requester peer include information regarding previous interactions with some of the provider peers. It is important to know what kind of information is conveyed in feedbacks. The content of feedbacks is needed for both credibility analysis and reputation computation.

2.7.3 What kind of information is sent to the requester peer?

For each provider peer, local trust information stored at the recommender's level can be sent to the requester peer in addition to the following information:

1. A confidence value: this value confirms how confident the recommender peer is in the trust value.
2. the time of recommendation: this time is provided when the feedback is generated (e.g., eBay, Nice, MDNT, EigenTrust, Travos, Regret).

The confidence value is usually based on the number of interactions that occurred between a recommender peer and a provider peer. A small number of interactions indicates uncertainty and does not reflect the real peer's trust value, while a large number of interactions increases the accuracy of the provided trust information. The confidence value can also be a subjective value given by the recommender peer based on its own experience. The concept of the confidence value is similar to reviewing a submitted paper and getting a feedback from a reviewer who indicates explicitly its expertise in the field. This represents how confident is the reviewer in the review. In Travos, the confidence value is a metric that represents the accuracy of the trust value based on the number of transactions. If the requester peer has a low confidence value in its assessment of trust for a provider peer, then seeking recommendations is necessary. In fact, it could be that the recommendations will add more misleading information than giving more accuracy to the peer's reputation.

At the queried peers, the trust information is filtered and only transactions corresponding to the context of recommendation are chosen to be sent to the requester peer. In case that recommendations pass through intermediary peers (e.g., PeerTrust, Nice, BinaryTrust, MLE) [45], additional information regarding these peers may be added to ensure the transparency of this process. In case of TrustModel, each intermediary peer adds some comments on the credibility of the recommendation received.

The identity of recommender peers can be revealed or not. In P2PBasic, the identity of these peers is not required. In P2PEnhanced, the identity is included in the feedbacks. This way, the requester peer is able to identify the peer sending the feedback and gives a weight to its feedback according to the credibility of this peer.

In some of the proposed reputation systems, the history of all trust transactions for a specific context are sent to the requester peer. In other systems (e.g., EigenTrust), only an aggregated trust value is sent to reduce the communication overhead and provide better scalability, but it lacks transparency. Aggregating feedbacks is a difficult task. The reputation score must represent the real behavior of a peer. The aggregation method of the trust values is not clearly described in most reputation systems. In Nice and TrustModel for example, any aggregation method can be used by each peer according to its requirements or by all the peers in the system to ensure homogeneity and avoid conflict of interests. Different approaches have been proposed for the feedback aggregation: in Regret, the weighted average of ratings is computed with the use of recency while in EigenTrust the difference between positive and negative transactions is normalized. In Travos, the feedback is an aggregation of the number of successful transactions and unsuccessful ones. The feedbacks received from recommender peers must provide sufficient information for credibility assessment. Incentive mechanisms are required to encourage peers to send recommendations.

2.8 Reputation Computation

Eliciting feedbacks from the peers that received the reputation query, and had already interacted with some of the provider peers, rises several issues. These peers may:

- Reply to the query honestly: sending the right feedback is the expected behavior from a well behaved recommender peer.
- Reply to the query dishonestly by lying: sending the wrong feedback in order to defame competitors or flatter conspirators. Reputation management schemes should be able to minimize the impact of this threat.
- Ignore the query: a free rider will ignore the need of the requester peer and will not take into account the reputation query. Ignoring the query by peers that have already interacted with the provider peers is an act of unreliability.

According to the reputation management scheme, the intermediary peers may transmit or not the feedback to other peers. Upon reception of feedbacks from other peers to be transmitted to the requester peer, the intermediary peers could omit the feedbacks or maliciously manipulate the content. Some security mechanisms are required to make sure that feedbacks are received intact by the requester peer. As an example, the recommender peer's digital signature can be added to ensure the recommendation integrity. Reputation management systems should provide mechanisms to enforce good contribution and cooperation from the peers that receive the reputation query.

The requester peer receives feedbacks from recommender peers that can be one-hop or multi-hop from the trusted peer or even all the peers in the system. Different types of opinions may result:

- First hand opinion: is a direct opinion from the queried peer to a provider peer. The queried peer has already interacted directly with the trusted peer.
- Second hand opinion: is an indirect opinion from a peer, that has been contacted by the queried peer, about a provider peer.
- Third hand opinion: is a public opinion

Different weights can be given to the received feedbacks according to their type. Typically, a first hand opinion is more trustworthy and then more important than the other ones.

This is one of the most important issues in reputation management systems and unfortunately, only few research works have focused on solving this problem. If the feedbacks received are not credible, the reputation can not be computed accurately. The majority of reputation schemes assume that peers are well-behaved, honest and do not provide any mechanism to check the accuracy of the gathered opinions which will reduce significantly the reliability of the reputation system. This problem is also referred to as misrepresentation. In eBay, for example, the credibility mechanism is left to members who can go through all the ratings and decide if a specific member is credible or not. Recently, the credibility assessment becomes an important issue since in e-commerce transactions where money plays an essential factor, accurate reputations are necessary in making the right decisions.

It is imperative to distinguish between two important issues when dealing with liar and fraudulent recommender peers:

- Liar recommender peers: the recommender peer had actually a transaction with the provider peer. The recommender peer may reply negatively even if the output of the transaction was positive. The recommender peer may lie and provide a dishonest feedback for different reasons (e.g., to increase the reputation of a colluding peer).
- Fraudulent recommender peers: the recommender peer did not interact with the provider peer before and provides the requester peer with a feedback. The feedback can be positive if the provider peer is a colluding peer or negative if the provider peer is a competitor (i.e., a good contributor peer).

Detecting malicious recommender peers is essential to the accuracy of the reputations and hence, the reliability of the reputation system. A credibility analysis is required to filter out recommendations, eliminate the suspicious feedbacks and select the accurate ones.

2.8.1 How to deal with liar recommender peers?

According to [36], there are two basic approaches that have been proposed: Endogenous and Exogenous methods. In Endogenous methods, the statistical properties of the reported feedbacks are used to detect unreliable feedbacks. In these methods, most of the proposed mechanisms assume that liar peers are the minority among peers and so the ratings that are different from the majority of peers are inaccurate. Exogenous methods rely on other information such as the reputation of the recommender or its feedback accuracy given its past recommendations or its relationship with the provider peer.

The followings are some mechanisms that have been proposed to minimize the impact of liar recommender peers:

- Keeping track of past recommendations (e.g., number of accurate and inaccurate recommendations) and weight the feedbacks according to the credibility of the recommender peers (e.g., MDNT, Travos, MLE, CredibilityRecords).
- The multivariate outlier detection technique: this technique is used to detect liar peers in FGTrust [23]. Outlier detection is an important task in data analysis. The outliers describe the abnormal data behavior which means data that is deviating from the natural data variability.
- The use of trust and reputation values as credibility metrics for the recommender peers (e.g., EigenTrust, Nice, FuzzyTrust). Trustworthy peers are considered as credible while untrustworthy peers are not. This is based on the fact that if peers are behaving correctly by sending authentic files, these peers will most probably be honest in their recommendations. In Regret, social relationships are also taken into account.
- Redundancy: the use of different score managers to compute the trust value and use a majority vote to eliminate the false reports by malicious score managers (e.g., EigenTrust)
- The use of Beta distribution based on previous recommendations as in Travos
- To let only one rating count from any single IP address
- Enforcing policies: in eBay, when a seller receives multiple feedbacks from the same buyer within the same week, the net effect on that seller's feedback score is based on the number of negatives, neutrals and positives received [37].

2.8.2 How to deal with fraudulent recommender peers?

The followings are some mechanisms that have been proposed to minimize the impact of fraudulent recommender peers:

- Only peers who were actually involved in a transaction are permitted to provide ratings (e.g., eBay).
- The use of proof of interactions: in TrustMe for example, both entities participating in a transaction sign a transaction certificate. This proof is needed to report the output of the interaction.

- The use of proof of processing: in DCRC and CORC, the RCA sends proof of processing for peers contribution to the system by processing, forwarding queries, staying online and serving files. To prevent malicious peers from getting credits without actually participating, the RCA maintains a transaction state where the credit_processed_list contains the list of peers who have already received the credit. The RCA ensures that a peer will collect credit only once for the same upload.

It is important to indicate that even with the presence of proof of transactions in some reputation management systems, it is difficult to prove that effectively there was a transaction between the two parties. In case of a file transfer for example, it is needed to prove that the file transfer has actually occurred. This proof must also indicate all information regarding the file transferred (e.g., size), the peer uploading the file, the peer downloading it and the time of interaction.

2.8.3 How to compute the reputation?

After filtering the feedbacks and getting rid of dishonest reports and fraudulent ones, the gathered data from different recommender peers is used for reputation computation in addition to the local trust data available at the requester peer.

Some of the proposed reputation schemes assign more importance to:

- Bad transactions versus good transactions: in some reputation schemes, it is proposed that the negative impact of bad behavior on reputation should outweigh the positive impact of good behavior.
- Local trust data versus the reputation information gathered from other peers: since the local trust data is more credible than the data collected from other peers. However, in P2P systems with millions of users, local trust data may not be very helpful in making decisions.
- Recent transactions versus old ones: to take into account the recency of interactions, the time of interaction is needed. This will help in identifying the traitors. In some reputation systems, old transactions are deleted and the focus is more on the recent transactions. In DCRC, a time stamp is used each time the RCA sends the reputation scores to peers, while, CORC time-stamps the reputation scores for expiration.

In some proposed reputation systems, a peer's reputation is equivalent to the group of peers it belongs to. A group's reputation can be, for example, the average of all the members reputations within this group. If the group has a high reputation, all its members are reputable. A peer can be rejected from the group in case it turned out to be acting maliciously.

Reputation systems should be exigent in terms of accuracy of the computed reputation according to the risk involved in the transactions. However, several factors affect the accuracy of reputation values such as: network overheads, congestion and loss of trust data during peers' communication. In some circumstances, this inaccuracy may be acceptable since the goal from reputation systems is to provide an approximation of the real peer's behavior.

Different approaches have been proposed to aggregate trust values received from recommender peers and synthesize them to generate a reputation value for a provider peer [28, 36]:

1. Deterministic approach: In this approach, peers' reputation is based on a simple summation or average of collected ratings. Even in the most popular e-Commerce applications that involve huge amounts of money, simple operations are used to compute reputation values. The reputation scheme used in eBay, for example, is based on the sum of the number of positive and negative ratings, while in amazon the reputation is computed based on the average of all the ratings. A weighted average of all the ratings is also possible based on different factors such as the credibility of recommender peers. This approach is easy to use and can be easily understood by users. Many proposed reputation systems use this approach for reputation computation. In MDNT, the reputation value is computed based on the weighting factor for the first, second and third opinions of third-party agents, the reputation opinion given by a recommender agent, the credibility of the recommender agent, and the time weighing factor that represents the importance of the opinion depending on the time of the last interaction between the recommender agent and the provider agent. In PeerTrust [46], different factors are taken into account such as transaction and community factors. In DCRC, reputation scores are computed based on Query-response Credit, the Upload Credit, the Download Debit and the Sharing Credit. In BinaryTrust, reputation is computed based on the number of negative complaints.

2. Probabilistic approach:

    (a) Bayesian Networks: The Bayesian approach uses a probabilistic approach to the determination of the reputation. It is based on the Bayes formula. Bayesian systems take binary ratings as input, and are based on computing reputation scores by statistical updating of beta probability density functions (PDF). Several works have used this approach including [32], [47], and [44]. In Travos, reputation is computed based on Beta Probability Distribution. However, the Bayesian approach suffers from strong assumptions of independence that are made.

    (b) Maximum Likelihood Estimation: MLE [21] uses a probabilistic approach to compute the reputation value based on the probability of recommender peers to provide inaccurate information. In MLE, the reputation value is the probability of a peer to cooperate and it is chosen to maximize the probability of the available ratings.

3. Fuzzy logic: Trust and reputation can be represented as linguistically fuzzy concepts. The information received from recommender peers is characterized by being imprecise and not accurately quantified. Fuzzy systems are used to deal with such situations by providing rules for reasoning with fuzzy measures. Different factors (e.g., the credibility of recommender peers) can be represented by fuzzy sets and membership functions are used. Several works have been proposed to compute reputation scores based on fuzzy logic systems including [42, 48]. Regret considers the following dimensions in reputation computation: the *individual dimension* which is the direct trust obtained by previous transactions, the *social dimension* which refers to a trust of an agent in relation with a group and finally the *ontological dimension* that includes the particularity of each agent. In FuzzyTrust, fuzzy inference is used to produce local trust values and aggregate them to global reputation values.

4. Flow models: Systems that compute trust or reputation based on transitive iteration through looped or arbitrarily long chains. As an example, EigenTrust, and FGTrust.

2.9 The Use of Reputation

2.9.1 How to choose a peer based on the reputation value?

After computing the reputation value, the following approaches can be used:

- The ranked-based approach: peers are ranked according to how reliable they are likely to be. The more reliable they are, the more trusted they are, the more the requester peer expects to get the required file. In this approach, the peer with the highest reputation value will be selected to upload the file. Highly reputable peers will handle almost all the uploads, yielding to an increase in their reputation. However, these peers will be overwhelmed by download requests. This approach suffers from unbalanced load share among peers. This approach has been adopted by EigenTrust and Travos.

- The threshold-based approach: The computed reputation scores may be compared against a trust threshold value. A random peer from the provider peers whose reputation values are greater than this trust threshold is selected. This approach spreads the load between these peers. This approach has been adopted by Nice, MDNT, MLE, and BinaryTrust.

- The Probabilistic-based approach: chooses the peer that will upload the file probabilistically based on its trust value. This probability is proportional to its normalized trust value and with a probability of $p\%$ selects a new comer (e.g., 10% for EigenTrust). This approach distributes better the load share among reputable peers, giving them a chance to increase their reputation in addition to increasing the reputation of new comers.

Once the requester peer finds a provider peer and it is willing to download from it the requested file, the requester peer becomes the trusting peer and the provider peer becomes the trusted peer. Both the trusting and the trusted peers will participate in a transaction.

2.9.2 How to evaluate a transaction after downloading a file?

It is important at this stage that a peer is able to ascertain when a transaction is successful. In case of failure, it could be due to network congestion or network failure, and sometimes it is due to the maliciousness of the transaction partner. Assigning a trust value based on the quality of the transaction is subjective. The subjectivity of trust values is inherent in most trust systems.

According to a detailed study on eBay, recommender peers may not send negative feedbacks by fear of reprisals or may also tend to reciprocate in both a positive and a negative way. To deal with the problem of reprisals, some reputation systems provide mechanisms such as:

- A peer can have an identity as a regular peer and another identity as a recommender peer. Recommender peers will not be easily recognized.
- Recommender peers can remain anonymous as in BinaryTrust. Recommender peers fill complaints and their identity is not provided. In this case, keeping records for assessing the credibility of recommender peers may not be possible. However, the identity of the mediators storing the recommendations are known.
- The privacy of recommender peers is protected by using symmetric-key cryptographic functions as in FGTrust.

2.10 Credibility Assessment

After assessing the transaction output, it is imperative to update the credibility of recommender peers. In MDNT, the credibility of the recommender peer is computed based on the distance between the transaction result and the trust value provided by this recommender peer. Updating the credibility of recommender peers will be beneficial for the assessment of feedbacks in future transactions. In Travos, the credibility of recommender peers in terms of accuracy of the provided recommendation is updated according to the transaction output.

In completely decentralized P2P systems, credibility values could be stored at the trusting peer's level for easy access in case this information is needed.

2.11 Incentives, Rewards and Punishment

In P2P systems, if all peers receive the same service regardless of their behavior, peers will not be motivated to strive for high reputation values since they will be always asked to upload files without receiving any special benefit or reward. This is why service differentiation is needed.

The system may reward reputable peers with increased connectivity to other peers, greater bandwidth, and/or higher priority/probability of performed requests. Peers can also be rewarded for providing feedbacks and sharing trust data with other peers. While good peers are rewarded for exhibiting a good behavior, malicious peers can be punished by sending them lists of peers with low reputation values to download from them. Several incentive mechanisms have been proposed to motivate peers to contribute to the system [49, 50, 51, 52, 53, 54, 55].

In [54], the authors introduce a reputation-based mechanism that assigns better service to higher performing peers. The reputation-based policies are classified into two dimensions: *Provider Selection* and *Contention Resolution*. In *Provider Selection* policies, a peer among peers providing a service is chosen to provide the service. Three schemes have been proposed for *Provider Selection*: *Highest Reputation* where peer with the highest reputation is selected, *Comparable Reputation* where peers can request services only from peers with reputation values comparable to their reputation and *Black List* where peers with low reputation are not providing any service. However, *Comparable Reputation* policy uses the concept of "Layered Communities" and provides the requesting peer with a list of peers having similar reputation values. This approach will incur an important increase of malicious uploads. Indeed, if a peer receives a service from a lower reputable peer, it will most probably receive a bad service (e.g., malicious file) and hence does not help the peer in providing good service to others. *Contention Resolution* policies help in selecting a peer among all peers requesting a service from the same provider peer. Two policies are presented: *Highest Reputation* policy where the peer with the highest reputation is selected to get the service and Probabilistically Fair w.r.t Reputation policy where a peer is selected with a probability according to its reputation.

In [53], the authors analyze the effectiveness of different incentives mechanisms to motivate peers to share files. The authors present a *reputation-based peer-approved* scheme. The scheme uses a reputation mechanism based on rating peers according to the number of files they are advertising. Peers are allowed to download files only from peers with lower or equal rating. The results show that the scheme can be used to counter the selfish behavior. However, this scheme will allow malicious peers to advertise a high number of corrupted files. According to this scheme, these peers will still receive good service. Even non malicious peers may advertise a large number of non popular or useless files and still benefit from the system.

In [55], the authors propose a reputation scheme that combines trust and incentive mechanisms. The proposed scheme uses explicit and implicit evaluations such as files' vote and retention time, download volume and users' rank to construct direct trust relationships. Based on the reputations, service differentiation is used to motivate users to share, vote on files, rank users and remove fake files. However, performance evaluation is needed to assess the performance of the proposed scheme.

In [56], the authors propose a service differentiation based on the services each node has provided to a P2P community. A resource distribution mechanism is proposed to increase the utility of the whole network and provide incentive for nodes to share information. A generalized mechanism that provides incentives for nodes having heterogeneous utility functions is also described.

The next sections survey centralized and decentralized reputation management systems.

2.12 Centralized Reputation Systems

2.12.1 e-Commerce Applications

The Internet opened up new opportunities for millions of people to interact with each other through applications such as electronic markets. However, in e-Commerce sites, customers do not have enough information about the sellers and the products/services offered. Trust systems create a platform between different parties to help each other and build trust [35, 57].

eBay [37] uses the feedback profile for rating sellers and computing the sellers' reputation. Buyers rate their transaction partners with a positive or negative feedback, and explain why. The reputation is computed by assigning 1 point for each positive comment and -1 point for each negative comment. The reputation score for a seller is the sum of the received ratings.

The reputation system used in eBay has contributed significantly to increase eBay's revenues. eBay recognizes that the collected ratings to compute their members' reputations are one of the company's main assets. eBay relies on a central trusted server to maintain the reputation system that has been improved recently. For members identity, eBay members use pseudonyms and personal information is used to identify members and kept confidential by the system. No special benefits are given to new users to increase their reputation.

The eBay feedback system suffers from the following issues:

- Easy to attack
- A seller may have a good reputation value by satisfying many small transactions even if he did not satisfy a large transaction.
- No mechanism is used to detect liar members that send wrong feedbacks. This task is left to users to detect such behavior.

eBay's reputation concept has been widely used to increase the trust level of online users. Similar reputation systems have been proposed in e-Commerce applications [28]:

- BizRate.com: is a shopping search engine which lists stores and products. It has an index of over 30 million products provided by more than 40,000 stores. BizRate.com claims that they use feedbacks collected from more than one million online users each month. BizRate.com uses ShopRank which is a proprietary shopping search and rating algorithm. This algorithm weights up price, popularity and availability of products against the reputation of the sellers. BizRate.com provides:
  - Rating of products: a rating scale of 1-5 stars (awful, poor, average, very good, excellent). In addition a breakdown of the overall product rating is also available (e.g., ease of use, portability, sound quality) and the reviews provided by users. Based on this information, a user can decide to buy or not the product.
  - Rating of merchants: for each product, a list of stores that sell the product, along with the price, the availability of the product and the rating of the store. This rating is based on a smiley scale (poor, satisfactory, good, outstanding) in addition to a "Customer Certified" logo for stores that satisfy specific criteria. This logo increases the users' trust in these stores. Only collected data during the last 90 days is considered for computing the merchants' reputation.

- Amazon.com: started as online bookstore and expended to sell different kind of products. Amazon.com uses ratings for products and for merchants. In merchant rating, sellers are rated based on the quality of the service provided on a scale from 1 to 5 stars in addition to comments left by buyers. In addition, the “Safe Buying Guarantee” increases the buyers confidence and encourage them to trust the sellers. Products in Amazon.com are also rated by users. Electronic products are only rated while other products are rated and reviews are provided. Users check reviews and vote whether the reviews are helpful or not.

- MoneyControl.com: is used to track the stock market. People can read different posted opinions and rate them. These opinions inform the users about the stock market and help them to make informed decisions regarding the stock they want to invest in. Opinions given by users are rated on a five stars scale. The average of all the ratings provided by users for a specific opinion is its overall rating and is expressed by a number between 0 and 5. Rating of reviewers is also available. A user can track the users who always provide good opinions. These users form the trusted agents network. A reviewer’s reputation is based on the number of the trusted agents who track this reviewer.

- Yahoo.com: provides a variety of products to customers. Both rating of products and merchants are available. In product rating, users can rate the products in addition to ratings provided by experts. Customers prefer more this latter rating since the expert rating is more trusted. Merchants are also rated based on the feedbacks collected from customers. Customers who made the purchase and rated the merchant are given more weight than other users.

- CNET.com: provides an up-to-date information regarding technology products. Ratings of products is realized by editors. These editors are experts and their opinion is more trusted by customers. Editors use the products and rate them according to different criteria (e.g., set up, design, features, performance, service and support) according to the type of the product (e.g., camera, computer). To compute the quality of a product, a specific weight is given to each criterion. The rating generated is a numerical score between 1 (Abysmal) and 10 (Perfect). Users can also provide their opinion regarding a specific product to exchange information between each other. Ratings for merchants is also provided based on four criteria: site functionality, store standards, order fulfillment, and customer feedback.

The feedbacks received from online users represent a rich set of collected information (e.g., ratings, comments, opinions, feedbacks) that help the business managers in discovering and identifying the needs of customers and improving the services provided to them and hence, increasing the customers’ trust in the business. As a consequence, the profit generated is increased in addition to the business reputation and value. The feedbacks received strengthen the relationship between customers and the business. In addition, they provide to online users the information needed to reduce the risk involved in virtual environments. While in physical environments, customers can check the products before buying them, exchanging information by sending feedbacks allows customers to make informed decisions.

2.12.2 P2P Systems

Although P2P systems use a distributed overlay for search and control messages, a centralized reputation system is used.

Reputation Management using DCRC and CORC

In [18, 22], the reputation system uses objective criteria to track peers' contribution to the system. Each peer stores the reputation value locally for a fast retrieval. Two mechanisms are proposed to compute the reputation:

- The Debit Credit Reputation Computation (DCRC)
- The Credit Only Reputation Computation (CORC)

By using a Reputation Computation Agent (RCA), peers' reputations are updated periodically in a secure and distributed manner. The DCRC mechanism credits peers for serving content and debits for downloading. The second mechanism (CORC) credits peers for serving content with no debits for downloading. The two mechanisms credit peers for query processing, query forwarding and for staying online. RCA is responsible for computing peers' reputation and certain amount of accuracy is lost due to the periodic update of reputation scores and the loss of data during the communication with RCA. Peers can be differentiated according to their *behaviors* and their *capabilities*. The behavior of a peer depends on its contribution to forwarding, processing requests and uploading files. The capability of a peer depends on the processing power, memory, bandwidth and storage capacity.

In DCRC, a peer's reputation score is computed using the following components:

- Query-Response Credit: peers are credited for being online and processing query-response messages.
- Upload Credit: each peer uploading a file gets a credit for serving the file and contributing to the good functioning of the system.

- Download Debit: each peer downloading a file, will get a debit.
- Sharing Credit: a peer gets credit for sharing hard-to-find files.

The CORC mechanism is similar to the previous scheme except that the download will not be debited. In order to prevent peers from acting maliciously once they get a high CORC score, CORC time stamps the reputation scores for expiration.

Drawbacks:

- RCA is contacted periodically by peers for updating peers' reputation scores. This is potentially a central point of failure.
- RCA needs to be replicated to ensure system robustness. However, no replication schemes have been proposed.
- Malicious downloads are not taken into consideration in peers' reputation.

In [22], two methods for tracking reputation are proposed: *Strong Reputations* and *Weak Reputations*. In *Strong Reputations*, the RCA is expected to have a copy of all the content served by peers to ensure content reliability. In addition, the RCA crawls periodically the P2P topology to maintain snapshots of the topology. This assumption is not practical in real P2P systems even with the fact that RCA is not required to serve the content. This proposed scheme for reputation tracking incurs higher overhead compared to *Weak Reputations*.

In [49], a Service Differentiation Protocol (SDP) for service differentiation in completely decentralized unstructured P2P networks is proposed. This protocol works as follows:

- During the *search* phase, a peer sends its reputation score along with the Query message. Each peer that receives this query extracts the reputation score and maps this value to a Level of Service (LoS). This peer will provide this LoS to the requester peer.
- During the *content download* phase, the peer requesting the file sends its reputation score to the peer uploading the requested file. This peer will send the file with a rate of transfer according to the reputation score of the requester peer.

A Fine-Grained Reputation System: FGTrust

In [23], the authors propose a reputation system built upon the multivariate Bayesian inference theory. [23] defines the reputation of a server (i.e., file provider, peer) as the probability that it is expected to demonstrate a certain behavior, as assessed by a client based on self experiences (i.e., the output of direct transactions) and users' feedbacks.

For reputation management, a centralized server is used as:

1. Account Manager: crediting/rewarding users and maintaining social groups.
2. Query Processor: for reputation queries
3. Feedback Collector: collects feedbacks
4. Reputation Engine: computing reputation scores

FGTrust has the following characteristics:

- Reputation data: After each transaction with a server, the user will increment the corresponding Quality of Service (QoS) level by one. This way, each user keeps only one vector for each server he interacted with.
- Reputation data storage: QoS information is stored across system users either in a random fashion or through a distributed hash table. In this latter case, the central server let λ users store the reputation feedback to improve system tolerance in case users are not available. Higher values of λ leads to higher fault tolerance, larger communication overhead and storage cost.
- Reputation computation: Users keep QoS experiences with servers after each transaction. They return QoS experiences after receiving a query from the centralized server. Users may inquire the centralized server about servers' reputations. Collecting all QoS information by the centralized server simplifies significantly the computation of peers' reputations. Old experiences are not as important as recent ones.

Drawbacks:
The centralized server is a single point of failure and is easy to attack (cf. Section 2.4.3).

2.13 Decentralized Reputation Systems

Since decentralized P2P applications are characterized by the absence of a central authority that coordinates the reputation management. Peers store information about past experiences with other peers and may be required to get other peers' opinions for reputation computation. Several decentralized reputation management schemes have been proposed for completely decentralized systems [14, 16, 17, 19, 20, 21].

2.13.1 The Distributed Trust Model: DistributedTrust

Abdul-Rahman et al. [38] proposed a model for trust based on distributed recommendations. This work is one of the first works on distributed trust models and that can be used in P2P systems. The proposed approach is based on four goals:

- Decentralization: each agent is responsible for managing trust information.
- Generalization of the notion of trust by using trust categories and trust values for different levels of trust in each category.
- The use of explicit trust statements in order to reduce ambiguity.
- Recommendations are used to get trust information regarding other agents in the system.

In this trust model, a trust relationship is between two entities, is non symmetrical and is conditionally transitive. The model has two types of trust relationships:

- Direct trust relationship: when an agent has trust in another agent.
- Recommender trust relationship: when an agent trust another agent to give recommendations about another agent's trustworthiness.

In this approach, each agent stores its trust data regarding other agents. Key-based encryption is used to protect recommendation messages from malicious agents. Network traffic is generated only for recommendations in case that the agent has not a direct trust value for another agent in the system.

Drawbacks:

- Each agent needs to store all history of past experiences and received recommendations. Storing this information will provide for the user a kind of global view of the whole network, however, a large storage capacity is needed.

- Updating this information can be time consuming and difficult.
- Increase of network traffic due to message exchange between agents in order to get reputation information.

2.13.2 The Binary Distributed Trust Model: BinaryTrust

In [14], the trust model is based on binary trust. An agent can be trustworthy or not. Each transaction between two agents can be either performed correctly or not. When an agent cheats, this agent becomes untrustworthy and a complaint is sent to other peers. In [14], only dishonest transactions are considered, based on the fact that malicious behavior is the exception. Reputation of a peer is based on the global knowledge of the complaints. PGrid that is a data storage, is used to store complaints. This trust model works as follows:

- A peer files a complaint about another peer by sending the complaint to other peers using *insert* messages
- When a peer wants to inquire about the trustworthiness of another peer, it will search for the complaints on that peer. In order to avoid additional traffic overhead, once this peer receives similar trust information from a certain number of peers, there is no need for further search.

Drawbacks:

- Some peers may receive complaints about themselves. Conflict of interest may occur and peers can delete this information to drop the complaints.
- No mechanism is provided to prevent from inserting arbitrary complaints about peers
- Maintenance of PGrid is required.

2.13.3 Reputation Management: P2PBasic and P2PEnhanced

In [16], the distributed polling algorithm *P2PRep* is used to allow a servant p (i.e., the resource requester) looking for a resource to ask about the reputation of offerers (i.e resource providers) by polling peers. After receiving a response from all the resource providers available to provide peer p with the requested resource, peer p selects a set of peers from the offerers and broadcasts a message asking other peers to give their opinion about the reputation of the offerers. Two variants of the algorithm are provided: in *the basic polling*, peers send their opinion and peer p uses the vote to determine the best offerer. In *the* *enhanced polling*, the peers provide their own opinion about the reputation of the offerers in addition to their identities. This latter will be used by peer p to weight the vote received.

Credibility Management is considered in *the enhanced polling* algorithm. Trust data of the peers sending their opinion to the resource requester will be considered in computing the reputation of the resource providers. Credible peers are given more weights in the reputation computation. In *the basic polling* algorithm, credibility of peers is not taken into consideration.

Drawbacks:

- The proposed schemes incur considerable overhead by polling peers for their votes. The *basic polling* algorithm checks whether the voters have provided the vote by sending *TrueVote* and *TrueVoteReply*. 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.

2.13.4 Reputation Management by the XRep Protocol: XRep

In this paper [41], the authors propose an approach that uses combined reputations of servents and resources. The authors describe the XRep protocol used for maintaining and exchanging reputations and its advantages against security attacks in P2P systems. In XRep, each servent maintains information based on its own experience with resources and servents and can share this information with other peers. Each peer maintains two experience repositories:

- A resource repository: for storing resources' identifiers and a binary value to describe each resource if it is good or bad
- A servent repository: for storing the servent's identifier for each peer it interacted with, in addition to the number of successful and unsuccessful downloads.

The XRep protocol has the following phases:

- *Resource searching* phase: searches for the peers that have the requested resource.
- *Resource selection and Vote polling* phase: selects the resource from the received list. The requester peer broadcasts *Poll* messages to enquire about the reputation of the resource and the peers providing this resource.

- *Vote evaluation* phase: evaluates the reputation of the requested resource based on the received votes.
- *Best servent check* phase: checks that the best servent provides really this resource and that the resource digest is in fact reliable.
- *Resource downloading* phase: decides from which servent to download the resource and contact this servent directly for the download operation.

Drawbacks:

- Inquiring about the reputation of resources and the resource providers incurs considerable traffic overhead.
- No performance results are provided to prove that the reputation management based on both the resources and the servents outperforms the schemes that are based only on the reputation of the servents.

2.13.5 Reputation Management using EigenTrust

In [17], the EigenTrust algorithm assigns to each peer in the system a global trust value based on peer's history of uploads. This trust value reflects the experiences of all peers with the peer. The authors propose a distributed and secure method to compute global trust value based on power iteration.

EigenTrust has the following characteristics:

- Reputation data: local trust values that represents the number of both satisfied and unsatisfied transactions. Local trust values are normalized and are between 0 and 1.
- Reputation computation: EigenTrust is based on transitive trust. The global reputation of a peer i is given by the local trust values assigned to peer i by other peers, weighted by the global reputations of the assigning peers. The use of transitive trust leads to a system where global trust values correspond to the left principal eigenvector of a matrix of normalized local trust values. Since each peer i computes and reports its own trust value, malicious peers can easily report false trust values. In secure EigenTrust, a distributed hash table is used to assign score managers that are responsible to compute global trust values. Each score manager is responsible for a set of peers. For each peer, the score manager learns about the peers that downloaded files from this peer and gets trust assessments from them.

- Credibility mechanism: different score managers are used to compute the global trust value for a peer. A majority vote is used on the trust values to reduce the impact of malicious score managers that report false trust values.
- Malicious peers policy: EigenTrust is robust to malicious collectives of peers who know each other and try collectively to subvert the system. In secure EigenTrust, using the one-way hash function, it is not possible for a score manager to know from whom the global trust value is computed. Malicious peers can not increase the reputation of each other. In addition, peers can not know their location in the hash space, thus, peers are unable to manipulate their own trust value. The system breaks up malicious collectives through the presence of pre-trusted peers.

Drawbacks:

- Normalizing local trust values will not make the distinction between peers who the requester peer did not interact with and peers that performed more unsatisfied transactions than satisfied ones.
- The scheme requires reputations for each provider peer to be computed *on-demand* which requires cooperation and collaboration from a large number of peers in performing computations.
- The scheme introduces additional latency and requires long periods of time to collect data and compute a global trust value for each provider peer.
- The use of a distributed hash table and score managers for each peer in the system in order to collect trust local information and compute global trust values increases significantly the communication overhead.

2.13.6 Limited Reputation Sharing: LimitedReputation

In [19], the proposed algorithms use only limited reputation sharing between peers. Each peer records statistics and ratings regarding other peers. As the peer receives and verifies files from peers, it updates the stored data. In the proposed voting reputation system, a requester peer receives ratings from other peers and weights them according to the ratings that the requester peer has for these peers. 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 proved to be reputable. Note that a peer can be reputable, but not credible. No mechanism is given to detect liar peers.

2.13.7 Trust and Credibility Records: CredibilityRecords

In [43], a reputation-based distributed trust system is proposed. In this system, the outcomes of past transactions are stored in trust vectors. These vectors are maintained by peers that perform the downloads. Trust vectors are constant length, binary vectors of m bits. Each bit represents the result of a transaction: 1 if successful, 0 if not. A number is associated with each vector to indicate the number of significant bits. When the requester peer receives a list of provider peers, the peer will compute the reputation values for these peers based on local information. If this information is not available, a trust query is issued in order to inquire about the reputation of the provider peers. The responses are weighted by the credibility ratings of recommender peers. The credibility vectors are similar to the trust vectors. Each bit represents the result of a previous judgment: 1 if the judgment was right, 0 if not.

2.13.8 Reputation Management using CCCI: MDNT

In [28], the authors presented a conceptual framework for measurement of quality and trust. They presented quality assessment through CCCI metrics (Correlation of delivered quality against defined quality, quality Commitment to each of the defined Quality Assessment Criteria, Clarity of each criterion from both parties' views and Influence of each criterion on the overall quality assessment).

The CCCI methodology for a trustworthiness measure provides four metrics, and defines the maximum possible correlation value for a business service interaction, $Corr_{qualities}$ and the maximum possible values of $Commit_{criterionc}$, $Clear_{criterionc}$, $Inf_{criterionc}$. Using these metrics, the trustworthiness value is computed.

2.13.9 Cooperative Peer Groups in Nice

Nice system is a platform for implementing cooperative applications over the Internet. A cooperative application allocates a subset of its resources (e.g., processing, bandwidth and storage) to be used by peers. In Nice, the resource provider is trusted. To access a remote resource, the trustworthiness of the resource requester is inferred. A trust value represents how likely a user considers another user to be cooperative.

In a successful transaction between two users Alice and Bob where Alice consumes a set of resources from Bob, Alice signs a cookie. This cookie states that she has successfully completed a transaction with Bob. A cookie can be stored by Bob to prove his trustworthiness to others. However a negative cookie may be destroyed by Bob. In this case, Alice may store it.

In Nice, each user stores a set of signed cookies. In case Alice wants to get resources from Bob, there are two possibilities:

- Alice has cookies from Bob: Alice gives these cookies to Bob and Bob computes a trust value for Alice
- Alice has no cookies from Bob: Alice uses the cookies that she has. The users who signed these cookies are contacted to check if they have cookies from Bob. For example, Alice has a cookie from Carol, and Carol has a cookie from Bob. Alice gets a copy of the cookie that Carol has and presents the two cookies to Bob. This means that Bob trusts Carol who trusts Alice. Based on these cookies, Bob infers a trust value for Alice. Based on this trust value, the requested resource will be granted or not to Alice.

2.13.10 BitTorrent

BitTorrent, is a widely used second generation P2P protocol. In BitTorrent, files are split up into chunks (e.g., 256 Kbytes) and the *tit-for-tat* strategy is adopted. Using this strategy, peers are able to optimize their download and upload rates. Peers typically upload to the k peers that recently provided it with the best downloading rate. The process of temporarily refusing to upload to some peers is called *choking*. BitTorrent also include a mechanism called *optimistic unchoking* by letting peers to upload chunks of files to randomly selected peers. This gives the new comers a chance to download pieces of data. Recent studies [58, 59, 60] have shown that the *tit-for-tat* strategy does not effectively reward good peers and punish free riders. Selfish peers can get more bandwidth while honest peers can receive low download rates. In [34], trust has been incorporated to the BitTorrent protocol, however, in that work trust is defined in terms of uploads compared to the downloads not in terms of maliciousness of the provider peer.

Tables 2.1 and 2.2 present a summary of some of the proposed reputation management systems according to the anatomy presented in Section 2.5.

	DCRC	CORC	FGTrust	DistributedTrust	BinaryTrust	P2PBasic
Reputation management	centralized	centralized	centralized	decentralized	decentralized	decentralized
Trust information	peer contribution is based on uploads, downloads, processing requests and sharing hard to find files	peer contribution is based on uploads, processing requests and hard to find files	QoS experiences	divided into trust categories and trust values for different levels of trust in each category	negative transaction: complaint	successful, unsuccessful downloads
Trust information storage	trusted peer	trusted peer	different peers	trusting peer	different peers	trusting peer
Sending a reputation query	no	no	no	yes	yes	yes
Number of recommender peers	-	-	-	large	few	large
Checking recommender peers credibility	receipt of the transaction	receipt of the transaction	the use of social groups	the use of trusted recommender peers	no mechanism	no mechanism
Information used to compute reputation	credit for uploading, debit for downloading, credit for query processing and sharing hard to find files	credit for uploading, credit for query processing and sharing hard to find files	output of direct transactions and received feedbacks	past transactions and received recommendations	number of complaints	the difference between positive and negative votes

Table 2.1: Reputation Management Systems

Components	P2PEnhanced	XRep	EigenTrust	LimitedReputation	CredibilityRecords
Reputation management	decentralized	decentralized	decentralized	decentralized	decentralized
Trust information	successful, unsuccessful downloads	resource repository: good or bad, peer repository: the number of successful and unsuccessful transactions	satisfactory and unsatisfactory transactions	number of authentic files	honest and dishonest transactions
Trust information storage	trusting peer	trusting peer	score managers	trusting peer	trusting peer
Sending a reputation query	yes	yes	yes	yes	yes
Number of recommender peers	large	large	large	neighbors, friends	few
Checking recommender peers credibility	the number of agreements and disagreements	no mechanism	reputation of peers	reputation of recommender peers	the number of right and wrong judgments
Information used to compute reputation	trust information weighted by the credibility of recommender peers	received votes	local trust values weighted by the global reputations of the peers	received ratings weighted by the reputation of the recommender peers	recommendations weighted by the credibility of recommender peers

Table 2.2: Reputation Management Systems (cond.)

2.14 Partially Decentralized Systems

KaZaA Media Desktop (KMD) a proprietary partially-decentralized P2P system has introduced *Participation Level* for rating files and peers. This *participation level* is considered as peers' reputation in the literature. Priority is given to peers with high *participation level*, however the exact process of how this priority is given is not known. In KaZaA, malicious peers will still have a high value of *participation level* even if their participation is affecting badly other peers if they are uploading corrupted content. KaZaA has no mechanism to detect malicious peers.

In [61], the authors propose SupP2PRep which is a protocol for reputation management via polling in P2P networks with superpeers. In [61], when a servent is looking for a resource, it broadcasts a *Query* message and receives a list of provider peers. The requester peer polls its peers by broadcasting a message *PollRequest* requesting their opinion about the selected provider peers. These peers reply with a *PollReply*. The superpeers collect the *PollReply* messages into one message *CumulativePollReply* and then send it to the requester peer. This protocol incurs additional messages overhead for direct communication between peers. No performance evaluation is presented.

2.15 Design Requirements

Reputation systems should have minimal overhead in terms of infrastructure, computation, storage and message complexity. The design of reputation management schemes must consider the followings issues [36]:

- Enforcing Local Control: as trust data can be stored by peers in the system, it is important not to allow these peers to arbitrarily manipulate trust data. Local control mechanisms are needed to protect trust data.
- Minimizing Storage Cost: this can be achieved by minimizing the trust data that need to be stored at peers level.
- Minimizing Bandwidth Cost: this can be realized by minimizing the messages exchanged between peers.
- Fault Tolerance: the topology of P2P networks is changing frequently due to the transient nature of peers that join and leave at any time. Leaving the system without any notice, may result in unavailability of trust information stored at peers level. Trust information need to be replicated to assure having the required information when needed.

- Scalability: the ability of the reputation scheme to scale with an increase in the number of peers. Peers will need to handle more trust data by collecting, storing, generating more computations and replying to queries. To investigate the scalability of a reputation system, it is needed to address the load placed on peers, especially on highly reputable ones, due to high demand for upload transactions and on the network as a result of exchanging messages between peers.

- Reliability: reputation systems should provide mechanisms to protect peers from malicious threats. Based on peers' reputation, peers can be identified as good or malicious. Peers need to rely on reputation systems to efficiently identify malicious peers and isolate them from the system. The more reliable reputation systems are, the more trust is given by peers to these systems.

2.16 Concluding Remarks

Reputation is used to build trust among peers, minimize the risk involved in the transactions, and increase users' confidence and satisfaction. In this chapter, we surveyed reputation-based trust management in P2P systems. We investigated research works proposed to handle peers' reputation. To understand better the reputation systems, we identified typical components. A description of the role of each component and their relationships is presented.

Several reputation management systems have been proposed for centralized and completely decentralized systems. While centralized systems suffer from the single point of failure, the major challenge in completely decentralized systems is the expensive search. Partially decentralized P2P systems overcome the weaknesses of centralized and completely decentralized systems. Partially decentralized P2P systems require efficient reputation management systems.

The next chapter introduces the trust management framework. We describe the first dimension of trust, named the *Authentic Behavior*, and the proposed reputation management schemes.

Chapter 3

Trust Management: Authentic Behavior

3.1 Trust Management Framework

We propose to address trust in a peer-to-peer system according to the following dimensions: 1) *Authentic Behavior*, 2) *Credibility Behavior*, and 3) *Contribution Behavior*

Authentic Behavior (AB): this is the reliability of a peer in providing accurate and good quality files. This value represents the reputation of a peer. It allows to differentiate between good and malicious peers.

Credibility Behavior (CB): this represents the sincerity of a peer in providing a honest feedback. The *Credibility Behavior* is an important indicator that allows to identify liar peers and reduce their effect on the reputation system.

Contribution Behavior (CTB): this represents the participation of a peer in terms of sharing files and positively contributing to the system. This value allows to distinguish between peers that contribute positively[1] to the system (i.e., altruistic) and the free riders (i.e., egoistic).

The trust given to a peer is based on its behavior in terms of *Authentic Behavior* (sending authentic or inauthentic files), *Credibility Behavior* (lying or not in the feedback) and *Contribution Behavior* (availability and involvement) (cf. figure 3.1). The trust given to peer P_i is characterized by the triplet (AB_i, CB_i, CTB_i). Peers with a good behavior are peers that send authentic files, honest feedback, and are available to share files in addition of being effectively involved in uploading files. Good peers will have high values along the three defined dimensions.

[1]We do not consider uploading malicious content as a contribution.

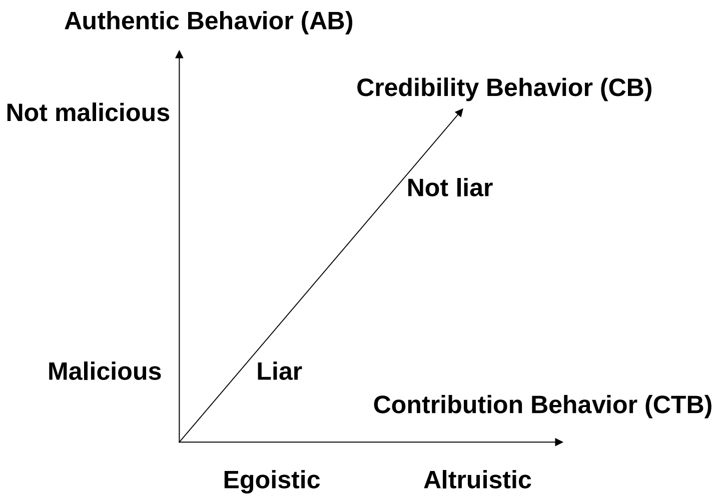


Figure 3.1: The Three Dimensions of Trust

In this chapter, we focus on the first dimension of the proposed trust management framework. In all other works, what researchers call *reputation* is in fact the *Authentic Behavior* of the peer which represents only one dimension of the trust associated with the peer.

3.2 Partially Decentralized Systems

In partially decentralized P2P file sharing systems, powerful computers are selected to become supernodes and act as a proxy for peers that are connected to them. Supernodes are involved only during the search process, while in the download process, peers are connected directly to each other. Partially decentralized P2P systems have been proposed to reduce the control overhead needed to run the P2P system. FastTrack-based systems have grown very quickly with no slowdowns that characterized Napster and Gnutella in their growth [62].

Several reputation management systems have been proposed in the literature. However, they have focused on completely decentralized P2P systems. The proposed reputation management schemes for completely decentralized P2P systems are not compatible with partially decentralized systems. These systems rely on supernodes for control message exchange which means there is no direct management messages that 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. Our goal 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 protocol), the proposed approach achieves more accuracy and reduces message overhead significantly. Once a supernode sends a search request on behalf of 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 data will represent all past experiences from all the peers that had interacted with these peers. A supernode may be also used to provide service differentiation based on reputation data of the peers it is responsible for.

We presented the advantages and drawbacks of centralized and completely decentralized reputation management systems in Section 2.4.3, we propose partially decentralized reputation management system that collect reputation data and compute reputation values at supernodes level:

- Advantages:
    1. Efficient search for reputation data of the provider peers compared to completely decentralized systems
    2. Lower discovery time for reputation data
    3. Easier to manage
    4. Scalable
    5. Small number of supernodes to manage reputation data
    6. Easy access to reputation data by other supernodes
    7. Alleviating the peer from the burden of replying to unnecessary queries as it is the case in completely decentralized P2P systems. Peers are contacted by their supernodes only if they have the requested file.
    8. Efficient enforcement of service differentiation (as we will see in chapter 5)
- Possible issues:
    1. Supernodes should be trusted to handle reputation data
    2. Additional load for supernodes
    3. Affected by supernodes join and leave

In addition of selecting peers based on their uptime and their capabilities (e.g CPU power, storage capacity, memory, bandwidth) to become supernodes, reputation should also be used. Highly reputable peers will be selected for this task to handle search requests on behalf of other peers.

In this chapter, we propose a reputation management system for partially decentralized P2P systems. This reputation system allows more clear sighted management of peers and files. Good reputation is obtained by having consistent good behavior through several transactions. This reputation management system detects malicious peers that are sending inauthentic files and isolates them from the system.

3.3 Reputation Management System

In this section, we describe the proposed reputation management system. The following notations will be used.

3.3.1 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[2] 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 derived from the trust given to these supernodes for control messages exchange.

[2]appreciation and feedback will be used interchangeably

3.3.2 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 to 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. $D^+_{i,*}$: Satisfactory downloads of peer P_i from other peers,
2. $D^-_{i,*}$: Unsatisfactory downloads of peer P_i from other peers,
3. $D^+_{*,i}$: Satisfactory uploads from peer P_i to other peers,
4. $D^-_{*,i}$: Unsatisfactory 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:

$$\begin{aligned} D^+_{i,*} + D^-_{i,*} = D_{i,*} \quad \forall i \\ D^+_{*,i} + D^-_{*,i} = D_{*,i} \quad \forall i \end{aligned} \tag{3.1}$$

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

Figure 3.2 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 sections 3.3.3 and 3.3.4.

When peer P_i joins the system for the first time, all values of its *reputation data* REP_{P_i} are initialized to zero[3]. Based on peer's transactions of uploads and downloads, these values are updated. The supernode of peer P_i sends (REP_{P_i}) periodically to the peer. This period of time is not too long to preserve accuracy and not too short to avoid extra overhead.

[3]This is a neutral reputation value.

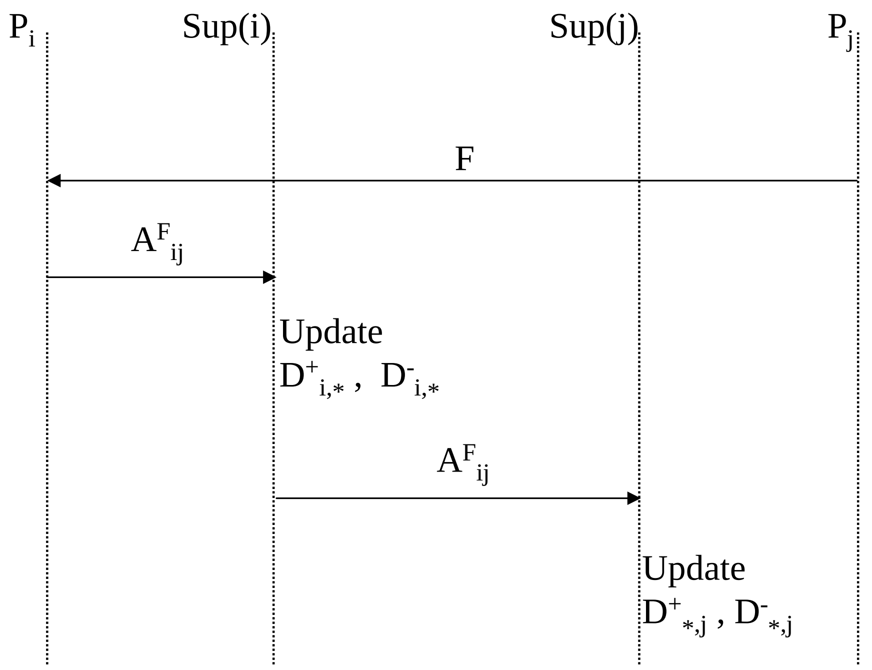


Figure 3.2: Reputation Update Steps

The peer will keep a copy of (REP_{P_i}) to be used the next time it joins the system or if its supernode changes. To keep this data protected, guaranteed to be authentic and since peers are not trusted, the supernode digitally signs (REP_{P_i}).

3.3.3 The Number Based Appreciation

In this first scheme, we use the number of downloads as an indication of the amount downloaded. This means that $D_{*,j}$ indicates 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, algorithm 1 is executed.

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 authentic files.

Algorithm 1: The Number-based Appreciation Scheme

```
$Sup(i)$ performs the following operation:
if $A^F_{i,j} = 1$ then
    $D^+_{i,*} = D^+_{i,*} + 1$
else
    $D^-_{i,*} = D^-_{i,*} + 1$
end if
$Sup(j)$ performs the following operation:
if $A^F_{i,j} = 1$ then
    $D^+_{*,j} = D^+_{*,j} + 1$
else
    $D^-_{*,j} = D^-_{*,j} + 1$
end if
```

3.3.4 The Size Based Appreciation

A better approach may be 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 peer P_i from peer P_j) is also sent. 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, algorithm 2 is executed.

Algorithm 2: The Size-based Appreciation Scheme

```
$Sup(i)$ performs the following operation:
if $A^F_{i,j} = 1$ then
    $D^+_{i,*} = D^+_{i,*} + Size(F)$
else
    $D^-_{i,*} = D^-_{i,*} + Size(F)$
end if
$Sup(j)$ performs the following operation:
if $A^F_{i,j} = 1$ then
    $D^+_{*,j} = D^+_{*,j} + Size(F)$
else
    $D^-_{*,j} = D^-_{*,j} + Size(F)$
end if
```

If we want to include the type of 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.).

3.4 The Selection Advisor

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 transaction of 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 sections: 3.4.1 and 3.4.2).
4. Download the file F
5. Send feedback $A_{i,j}^F$. $Sup(i)$ and $Sup(j)$ will update the reputation data REP_{P_i} and REP_{P_j} respectively.

The following subsections describe two proposed selection algorithms. Each of these algorithms can be based on one of the appreciation schemes presented in Section 3.3.3 and 3.3.4. Note that only the satisfactory and unsatisfactory units of uploads (i.e., $D_{*,j}^+$, $D_{*,j}^-$) are considered for computing the reputation of a peer. The satisfactory and unsatisfactory units of downloads (i.e., $D_{i,*}^+$, $D_{i,*}^-$) will be taken into consideration for service differentiation as it will be presented in chapter 5.

3.4.1 The Difference Based Algorithm

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

$$DB_j = D^+_{*,j} - D^-_{*,j} \tag{3.2}$$

This value gives an idea about the aggregate behavior of the peer. Note that the reputation as defined in equation 3.2 can be negative. This reputation value gives preference to peers who performed more good uploads than bad ones.

3.4.2 The Inauthentic Detector Algorithm (IDA)

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 such as $D^+_{*,1} = 40$, $D^-_{*,1} = 20$, $D^+_{*,2} = 20$, and $D^-_{*,2} = 0$ and assume we are using the Size Based Appreciation scheme, then according to the Difference-based algorithm (i.e., equation 3.2) 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:

$$\begin{array}{ll} AB_j = \frac{D^+_{*,j} - D^-_{*,j}}{D^+_{*,j} + D^-_{*,j}} = \frac{D^+_{*,j} - D^-_{*,j}}{D_{*,j}} & \text{IF } D_{*,j} \neq 0 \\ AB_j = 0 & \text{OTHERWISE} \end{array} \tag{3.3}$$

Note that the reputation as defined in equation 3.3 is a real number between -1 (if $D^+_{*,j} = 0$) and 1 (if $D^-_{*,j} = 0$).

If we go back to the previous example, 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 can

be 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.

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.

3.5 Mathematical Analysis of Size-based IDA

Let's assume that a peer P_i downloads a file F from a peer P_j. We focus on the *Authentic Behavior* (sending authentic or inauthentic files) of peer P_j since it is sending the file. The goal is to assess if the *Authentic Behavior* can capture the probability of P_j sending inauthentic files.

- let X_n^j be the value of $D_{*,j}^{+}$ after the n^{th} upload of peer P_j
- let Y_n^j be the value of $D_{*,j}^{-}$ after the n^{th} upload of peer P_j
- let F_n be the size of the file sent during the n^{th} upload of peer P_j
- let AB_j^n be the Authentic Behavior value after the n^{th} upload of peer P_j

This means that:

$$AB_j^n \ = \frac{X_n^j - Y_n^j}{X_n^j + Y_n^j} \tag{3.4}$$

The values of X_n^j and Y_n^j will be updated according to the authenticity of the file sent by peer P_j.

We assume that peers do not lie and send the appropriate feedback:

- Peer P_j will send an inauthentic file with probability p_j
- The value of Y_n^j will increase by the size of the uploaded file F_n with probability p_j
- The value of X_n^j will increase by the size of the uploaded file F_n with probability $(1 - p_j)$

The new values of X_n^j and Y_n^j are:
$X_{n+1}^j = X_n^j + (1-p_j)F_{n+1}$
$Y_{n+1}^j = Y_n^j + p_j F_{n+1}$

Let's find a closed formula for X_n^j.
$X_n^j = X_{n-1}^j + (1-p_j)F_n$
$X_{n-1}^j = X_{n-2}^j + (1-p_j)F_{n-1}$
...
$X_2^j = X_1^j + (1-p_j)F_2$
$X_1^j = X_0^j + (1-p_j)F_1$
$X_0^j = 0$

This means that we have:
$X_n^j = (1-p_j)\sum_{k=1}^{k=n} F_k$

Using the same approach we have:
$Y_n^j = p_j \sum_{k=1}^{k=n} F_k$

The *Authentic Behavior* value of peer P_j is:

$$\begin{aligned} AB_j &= \frac{X_n^j - Y_n^j}{X_n^j + Y_n^j} \\ &= \frac{[(1-p_j)\sum_{k=1}^{k=n} F_k] - [p_j \sum_{k=1}^{k=n} F_k]}{[(1-p_j)\sum_{k=1}^{k=n} F_k] + [p_j \sum_{k=1}^{k=n} F_k]} \\ &= \frac{(1-p_j-p_j)\sum_{k=1}^{k=n} F_k}{(1-p_j+p_j)\sum_{k=1}^{k=n} F_k} \\ &= 1-2p_j \end{aligned} \tag{3.5}$$

The reputation of peer P_j depends only on its probability of sending inauthentic files. The closest this probability to zero, the higher the reputation is. If $p_j = 0$, AB_j equals 1. Otherwise, if $p_j = 1$, AB_j will be equal to -1

It is important to note that our reputation computing technique is more general and can capture more elaborated P_j behaviors. We consider that the behavior of P_j is totally captured by the probability p_j independently from other factors. However, a P_j may have different probabilities of uploading the file depending on the file's size. In this case, we consider the behavior of P_j to be captured by a probability distribution. The proposed reputation computing technique can also capture this.

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

Table 3.1: Simulated Algorithms

3.6 Performance Evaluation

In this set of simulations, we evaluate the two selection advisor algorithms (cf. Sections 3.4.1 and 3.4.2) namely, the *Difference-based* (*DB*) Algorithm and the *Inauthentic Detector Algorithm* (*IDA*). Both schemes use the Size-Based Appreciation Scheme proposed in Section 3.3.4. We will compare the performance of these two algorithms with the following two schemes.

In KaZaA [9], the peer participation level is computed as follows: $(uploaded/downloaded) \times 100$, i.e., using our notation (cf. Section 3.3.1) 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 3.1 presents the list of considered algorithms.

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 10 MB and 150 MB.
- As observed by [63], each peer can ask for a file with a Zipf distribution over all the files that the peer does not already have. Initially, we have simulated the proposed schemes with a random selection of files. However, we wanted to incorporate a realistic parameter in our simulations. As discussed in [63], the file distribution follows Zipf with a parameter close to 1. We have chosen this parameter to be equal to 0.9.

- The percentage of malicious peers is 50%. Our goal is to assess the performance of the proposed selection algorithms even with the presence of a high percentage of malicious peers. Studies show that KaZaA suffers from a high percentage of inauthentic files[4].
- The probability of a malicious peer to upload an inauthentic file is 0.8.
- We simulate 30,000 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 several times over which the results are averaged.

Performance Metrics

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

- 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.

Simulation Results

Figure 3.3 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

[4]Simulations with lower percentages of malicious peers have been contacted and have led to similar results.

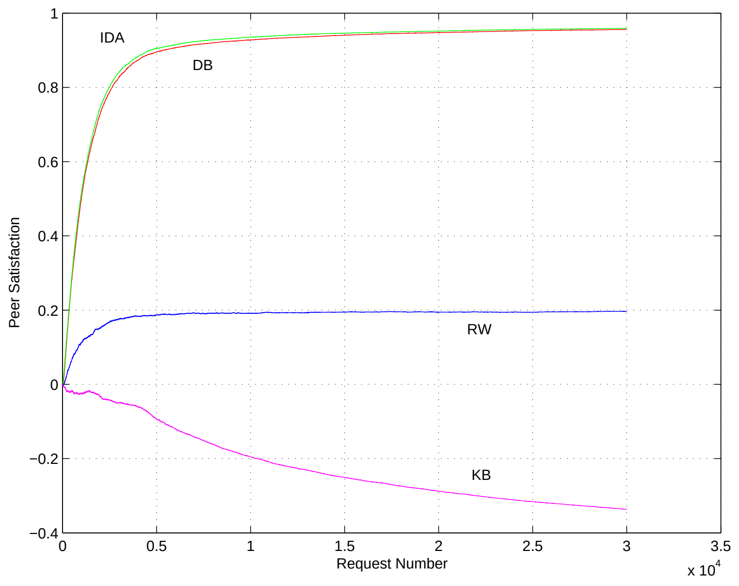


Figure 3.3: Peer Satisfaction

and hence the results observed from the simulations (i.e., 0.2 satisfaction) can be explained as follows; With 50% malicious peers and 0.8 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 $(60 - 40)/(60 + 40) = 0.2$.

Figure 3.4 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, results in using the network bandwidth more efficiently and higher peer satisfaction as shown in figure 3.3. 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 3.4 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. Figures

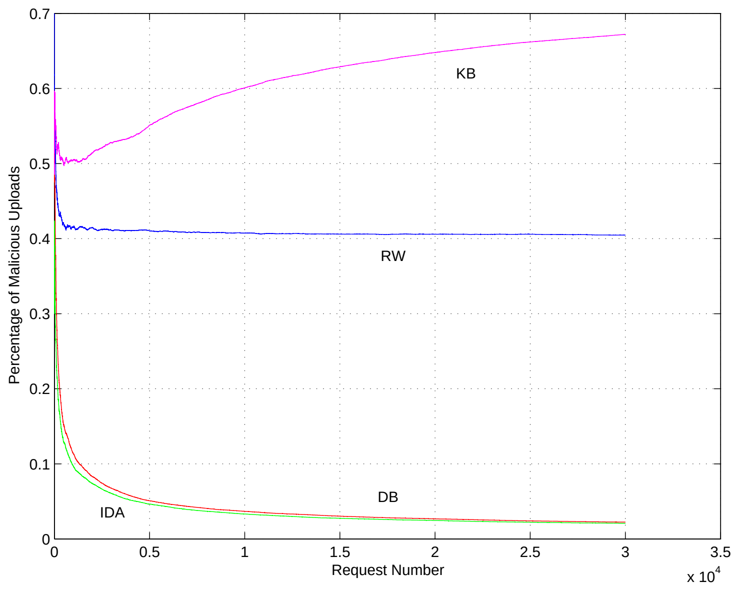


Figure 3.4: The Percentage of Malicious Uploads

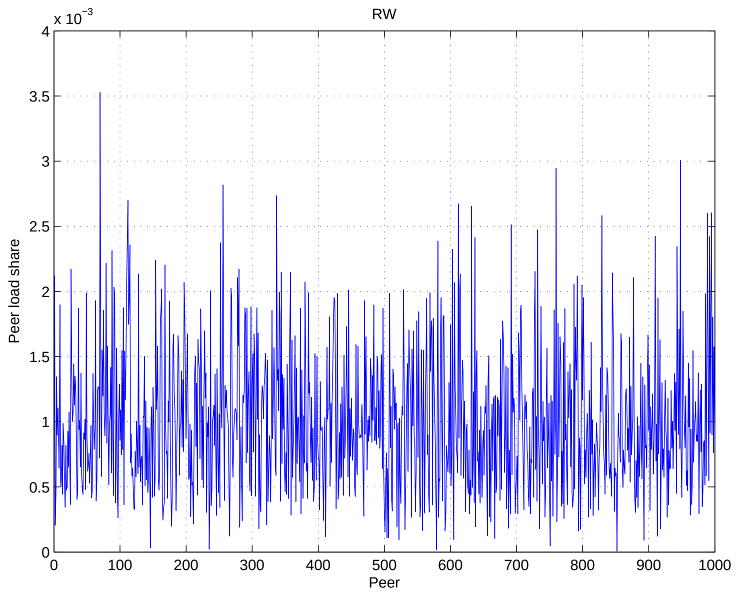


Figure 3.5: Peer Load Share for RW

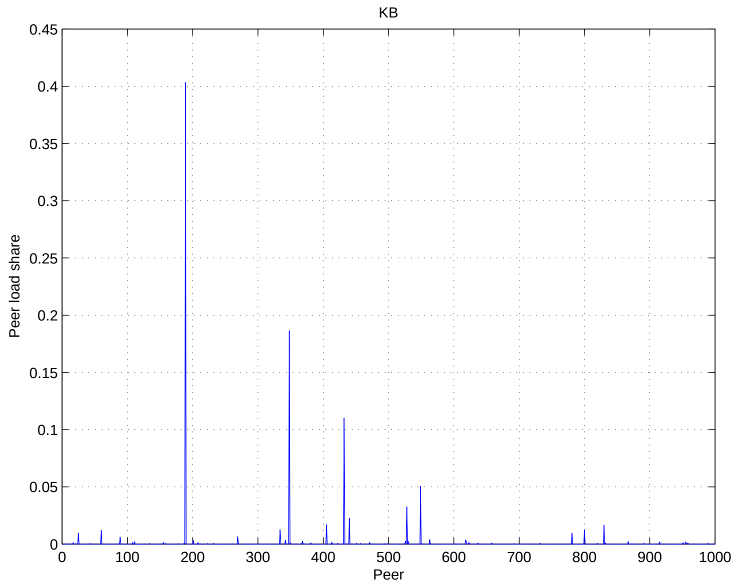


Figure 3.6: Peer Load Share for KB

3.5, 3.6, 3.7 and 3.8 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)(see figure 3.5). 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 case, the malicious peer number 180 has been chosen to perform most of the requested uploads (see figure 3.6).

In figures 3.7 and 3.8, 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 no malicious peer is involved in uploading the requested files[5], almost all the load (of the 30,000 requests) is supported by the non malicious peers.

According to figures 3.7 and 3.8, 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*

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

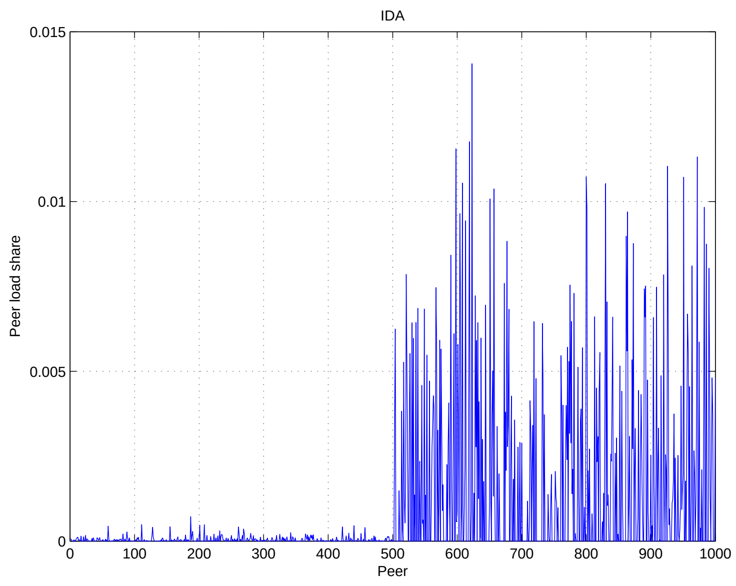


Figure 3.7: Peer Load Share for IDA

scheme distributes the load more uniformly among the non malicious peers than the *DB* scheme (see figure 3.7). The *DB* scheme tends to concentrate the load on a small number of peers (see figure 3.8). This can be explained by the way each scheme computes the reputation of peers. As explained in sections 3.4.1 and 3.4.2, the *DB* scheme computes the reputation of peer P_j as shown in equation 3.2 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 3.3. 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 3.8, 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. 3.3). 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 3.7. 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.

We opt to use *IDA* rather than *DB* scheme since most of the peers get a chance to upload files and build their reputation. In the *DB* scheme, only a small number of peers are involved in uploading files to others.

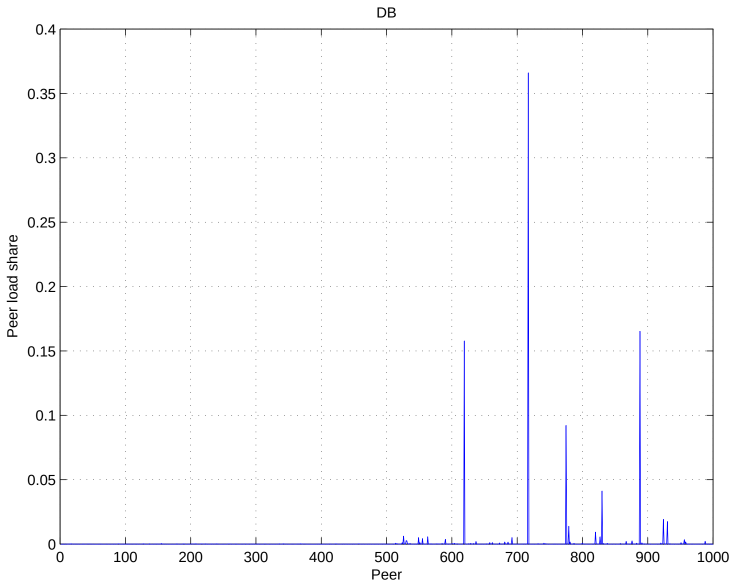


Figure 3.8: Peer Load Share for DB

3.7 Concluding Remarks

In this chapter, we proposed a trust management framework for partially decentralized peer-to-peer systems. Reputation data is stored at the supernode level to take advantage of the use of supernodes. Two selection advisor algorithms were proposed for assisting peers in selecting the most trustworthy peers to download from.

In the next chapter, *Credibility Behavior* is introduced as another dimension of trust. We propose *Malicious Detector Algorithm* to detect liar peers and the concept of *Suspicious Transactions* introduced to detect these peers.

Chapter 4

Trust Management: Credibility Behavior

In this chapter, we focus on the *Credibility Behavior* which is the second dimension in the proposed trust management framework. We propose a novel scheme, that in addition to detecting peers that send inauthentic files, detects liar peers. This scheme reduces considerably the amount of malicious uploads.

4.1 Peer Behavior Understanding

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.

In all proposed feedback-based reputation management schemes for P2P systems, the emphasize was more on detecting peers who are sending inauthentic files than detecting liar peers. Some of the proposed feedback-based reputation schemes (e.g., EigenTrust, fuzzyTrust) rely on peers' reputation for their peer-selection process. These schemes use the reputation value for a recommender peer as a credibility value which is not appropriate. Indeed some peers may behave well while sending authentic files and provide at the same time inaccurate recommendations. However, it is not practical to collect feedbacks regarding the credibility of the recommender peers as suggested in some works (e.g., P2PEnhanced). Not only broadcasting is used to receive the reputation of the provider peers, but also, to get the credibility of peers that have replied. This approach will incur

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 4.1: Peer Behavior

a significant amount of overhead and waste of network resources.

4.1.1 Peer Behavior Categorization

In a P2P system, good peers are those that send authentic files and do not lie in their feedbacks (Type $T1$ in Table 4.1). 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 authentic files and lie in their feedbacks (Type $T3$), and 3) peers that send inauthentic files and 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 behavior 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 send inauthentic files by mistake. Note also that peers can change their behavior over time and hence can jump from one category to another.

4.1.2 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.

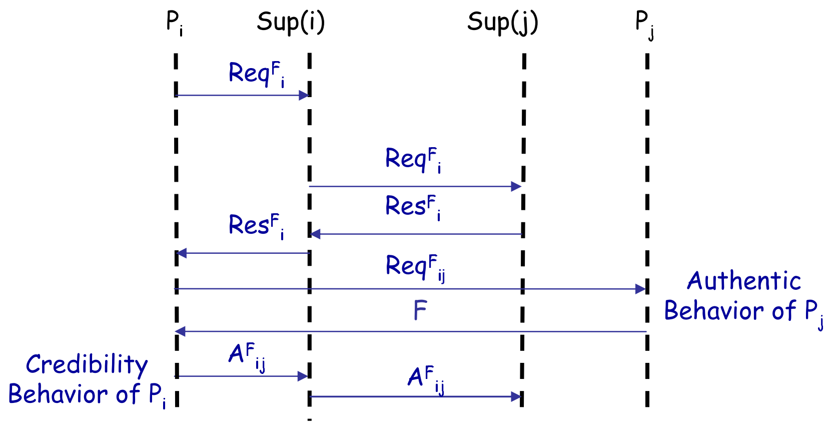


Figure 4.1: A Typical File Request-Download Procedure

4.2 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 $T4$.

Figure 4.1 shows a typical file request-download procedure involving the requester and provider peers and their supernodes. Peer P_i requests a search service Req_i^F from its supernode $Sup(i)$. When peer P_i receives the result of the search request Res_i^F, peer P_i will choose peer P_j according to the *Authentic Behavior* of P_j. Peer P_i sends a request Req_{ij}^F to download file F from peer P_j. After downloading this file, peer P_i sends feedback $A_{i,j}^F$. The *Credibility Behavior* of peer P_i will have a significant impact on the *Authentic Behavior* of peer P_j.

IDA (section 3.4.2) 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.

4.2.1 Reputation-based 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 3.3.4, to:

$$\begin{array}{ll} \text{If } A_{i,j}^F = 1 & \text{THEN } D_{*,j}^+ = D_{*,j}^+ + \frac{1+AB_i}{2} Size(F) \\ & \text{ELSE } D_{*,j}^- = D_{*,j}^- + \frac{1+AB_i}{2} Size(F) \end{array} \tag{4.1}$$

Using this approach[1], the impact of peer P_i on the *Authentic Behavior* of peer P_j is related to the *Authentic Behavior* of peer P_i (i.e., 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, only a small 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 it 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 4.1, those peers will have a high reputation since they always send authentic files and hence will receive good feedbacks[2]. Those peers will be trusted by the system and will affect badly the reputations of other peers and may eventually subvert the system. The performance evaluation section assesses the effect of liar peers on the reputation of other peers.

4.2.2 Malicious Detector Algorithm (MDA)

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 $A_{i,j}^F = 1$ and $AB_j < 0$ or if $A_{i,j}^F = -1$ and $AB_j > 0$ then we consider this transaction as suspicious.

To detect peers that lie in their feedbacks, the supernode $Sup(i)$ keeps track of the following values for each peer P_i:

[1]In operation (4.1), $\frac{1+AB_i}{2}$ can be replaced by any function of AB_i that is strictly increasing from 0 to 1.

[2]We assume that the percentage of malicious peers, in a P2P system, is lower than the percentage of good peers [36]. This assumption is realistic since this is the basis on which peer-to-peer systems can work.

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., $A_{i,j}^F \times AB_j < 0$ (i.e., during a suspicious transaction)
3. TF_i: The total size of all the files uploaded by P_i.

Note that $N_i^* \leq N_i \; \forall i$

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

$$\begin{array}{l} N_i = N_i + 1 \\ \text{IF } (A_{i,j}^F \times AB_j) < 0 \text{ THEN } N_i^* = N_i^* + 1 \end{array} \tag{4.2}$$

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

$$\alpha_i = \frac{N_i^*}{N_i} \tag{4.3}$$

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

α_i is the ratio of the number of suspicious feedbacks[3] 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 $A_{i,j}^F$ 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 $A_{i,j}^F$, the sender's supernode $Sup(j)$ performs algorithm 3.

Algorithm 3: The Update Strategy

```
**if** $A_{i,j}^F = 1$ **then**
    $D_{*,j}^+ = D_{*,j}^+ + (1 - \alpha_i)Size(F)$
**else**
    $D_{*,j}^- = D_{*,j}^- + (1 - \alpha_i)Size(F)$
**end if**
$TF_j = TF_j + Size(F)$
```

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

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. On the other hand, good peers will have a lower value of α_i and hence will keep having an impact on the reputation of other peers.

In this approach, the *Authentic Behavior* is computed as follows:

$$\begin{array}{ll} AB_j = \frac{D^+_{*,j} - D^-_{*,j}}{TF_j} & \text{IF } TF_j \neq 0 \\ AB_j = 0 & \text{OTHERWISE} \end{array} \tag{4.4}$$

This equation is more general. Indeed if all peers are honest ($\alpha_i = 0$), equation 4.4 becomes equation 3.3. Note that AB_j is updated after each upload of peer P_j (equation 4.4) and α_i is updated after each download of peer P_i. This means that liar peers will be detected even if they did not upload any file and inauthentic peers will be detected even if they did not perform any download.

If peer P_i changes its behavior, α_i will also change, and hence its impact on the reputation of other peers. For example, if peer P_i changes its behavior from category $T3$ to $T1$ (cf. table 4.1), 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.

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

In this case, the reputation of peer P_i is the couple (AB_i, CB_i) which characterizes 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 a peer 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.

4.3 Mathematical Analysis of MDA

In this section, we provide a mathematical analysis for computing peers' reputation with the existence of liar peers.

Let's assume that a peer P_i downloads a file F from a peer P_j. We focus on the *Authentic Behavior* (sending authentic or inauthentic files) of peer P_j since he is sending the file, and the *Credibility Behavior* (lying or not in the feedback) of peer P_i since he is sending the appreciation that will affect the reputation of peer P_j.

Let's assume that peer P_i and peer P_j are characterized by the following probabilities:

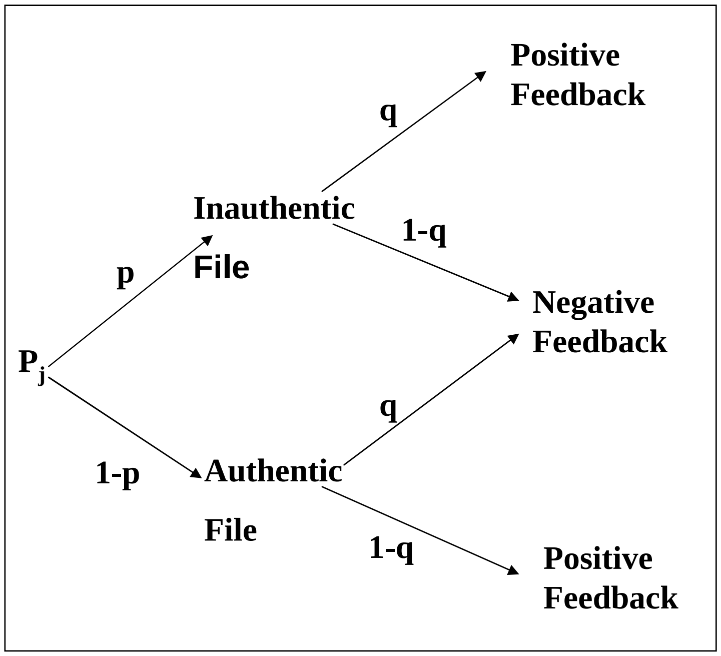


Figure 4.2: Authentic and Credibility Probabilities

- P_i: (p_i, q_i)
- P_j: (p_j, q_j)

Where p and q represent the probabilities of sending inauthentic files and wrong feedbacks respectively. We use the notations introduced in Section 3.5.

We have $AB_j^n = \frac{X_n^j - Y_n^j}{X_n^j + Y_n^j}$ (cf. eq. 3.4)

The values of X_n^j and Y_n^j will be updated according to the authenticity of the file sent by peer P_j and the credibility of peer P_i.

4.3.1 Case 1: Same Liar Behavior

Let's consider a scenario where all peers send wrong feedbacks with the same probability. All peers have the same value of q.

According to figure 4.2, we have the following:

- Y_n^j will increase by the size of the uploaded file F_n with probability $Q = (1 - p_j)q + p_j(1 - q)$
- The value of X_n^j will increase by the size of the uploaded file F_n with probability $(1 - Q)$

Using the same approach as in Section 3.5, we obtain:

$$\begin{aligned} AB_j &= 1 - 2Q \\ &= 1 - 2[(1 - p_j)q + p_j(1 - q)] \\ &= 1 - 2(q - qp_j + p_j - qp_j) \\ &= 1 - 2q + 4qp_j - 2p_j \\ &= 1 - 2p_j - 2q(1 - 2p_j) \\ &= (1 - 2p_j)(1 - 2q) \end{aligned} \tag{4.5}$$

This means that the reputation of peer AB_j depends on the probability of sending inauthentic files p_j and the probability of lying in the feedback (i.e., q) of all peers that downloaded a file from peer P_j.

In case that the probability of sending inauthentic files of peer P_j is almost null, its reputation has to be close to 1. However, due to the lying behavior of peers who downloaded files from peer P_j, its reputation can become -1 if q is close to 1.

4.3.2 Case 2: Different Liar Behavior

In the following case, let's consider that peers have different values of q. Let q_n be the probability of lying for the peer P_n who downloaded the file F_n from peer P_j at the n^{th} upload of peer P_j.

Since peer P_j will send an inauthentic file with probability p_j, we obtain the followings:

- The value of Y_n^j will increase by the size of the uploaded file F_n with probability $(p_j(1 - q_n) + (1 - p_j)q_n)$
- The value of X_n^j will increase by the size of the uploaded file F_n with probability $((1 - p_j)(1 - q_n) + p_jq_n)$

According to figure 4.2, the new values of X_n^j and Y_n^j are:

$X_n^j = X_{n-1}^j + ((1 - p_j)(1 - q_n) + p_jq_n)F_n$

$X_{n-1}^j = X_{n-2}^j + ((1 - p_j)(1 - q_{n-1}) + p_jq_{n-1})F_{n-1}$

...
$X_2^j = X_1^j + ((1-p_j)(1-q_2) + p_j q_2)F_2$
$X_1^j = X_0^j + ((1-p_j)(1-q_1) + p_j q_1)F_1$
$X_0^j = 0$

This means that we have:
$X_n^j = (1-p_j)\sum_{k=1}^{k=n}(1-q_k)F_k + p_j \sum_{k=1}^{k=n} q_k F_k$

Using the same approach, we obtain:
$Y_n^j = (1-p_j)\sum_{k=1}^{k=n} q_k F_k + p_j \sum_{k=1}^{k=n}(1-q_k)F_k$

To compute the reputation of peer P_j using equation 3.3, we get:

$$\begin{aligned}
AB_j^n &= \frac{X_n^j - Y_n^j}{X_n^j + Y_n^j} \\
&= \frac{(1-p_j)\sum_{k=1}^{k=n}(1-q_k)F_k + p_j\sum_{k=1}^{k=n} q_k F_k - (1-p_j)\sum_{k=1}^{k=n} q_k F_k - p_j \sum_{k=1}^{k=n}(1-q_k)F_k}{(1-p_j)\sum_{k=1}^{k=n}(1-q_k)F_k + p_j\sum_{k=1}^{k=n} q_k F_k + (1-p_j)\sum_{k=1}^{k=n} q_k F_k + p_j \sum_{k=1}^{k=n}(1-q_k)F_k} \\
&= \frac{\sum_{k=1}^{k=n} F_k[(1-p_j)(1-q_k)+p_j q_k] - [(1-p_j)q_k + p_j(1-q_k)]}{\sum_{k=1}^{k=n} F_k[(1-p_j)(1-q_k)+p_j q_k] + [(1-p_j)q_k + p_j(1-q_k)]} \\
&= \frac{\sum_{k=1}^{k=n} F_k[1-q_k-p_j+p_j q_k+p_j q_k-q_k+p_j q_k-p_j+p_j q_k]}{\sum_{k=1}^{k=n} F_k[1-q_k-p_j+p_j q_k+p_j q_k+q_k-p_j q_k+p_j-p_j q_k]} \\
&= \frac{\sum_{k=1}^{k=n} F_k[1-2q_k-2p_j+4p_j q_k]}{\sum_{k=1}^{k=n} F_k} \\
&= \frac{\sum_{k=1}^{k=n} F_k[1-2q_k-2p_j(1-2q_k)]}{\sum_{k=1}^{k=n} F_k} \\
&= \frac{\sum_{k=1}^{k=n} F_k[(1-2q_k)(1-2p_j)]}{\sum_{k=1}^{k=n} F_k} \\
&= (1-2p_j)\frac{\sum_{k=1}^{k=n}(1-2q_k)F_k}{\sum_{k=1}^{k=n} F_k} \\
&= (1-2p_j)\frac{\sum_{k=1}^{k=n} F_k - 2\sum_{k=1}^{k=n} q_k F_k}{\sum_{k=1}^{k=n} F_k} \\
&= (1-2p_j)(1-2\frac{\sum_{k=1}^{k=n} q_k F_k}{\sum_{k=1}^{k=n} F_k})
\end{aligned} \tag{4.6}$$

In this case, the reputation of peer P_j depends also on the probability q_k of all peers P_k that downloaded files from P_j. In case that the probability of lying is the same for all peers, we obtain equation 4.5 by replacing q_k in eq. 4.6 by q.

We have:

Category	Percentage of peers	Probability of sending inauthentic files	Probability of sending wrong feedback
$T1$	40%	0.01	0.01
$T4.1$	30%	0.5	0.5
$T4.2$	30%	0.9	0.9

Table 4.2: Peer Behavior and Distribution

$$\begin{aligned} AB_j &= (1-2p_j)(1-2\frac{\sum_{k=1}^{k=n} qF_k}{\sum_{k=1}^{k=n} F_k}) \\ &= (1-2p_j)(1-2q\frac{\sum_{k=1}^{k=n} F_k}{\sum_{k=1}^{k=n} F_k}) \\ &= (1-2p_j)(1-2q) \end{aligned} \quad (4.7)$$

4.4 Performance Evaluation

In these simulations, we simulate the Inauthentic Detector *(IDA)* and the Malicious Detector *(MDA)* algorithms. Their performance will be compared to the KaZaA-based (KB) and the Random Way (RW) algorithms. In these simulations, liar peers will be considered.

Simulation Parameters

We use the same simulation parameters as in Section 3.6 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 4.2.

Table 4.2 presents the peers distribution and their probabilities of sending inauthentic files and wrong feedbacks. Taking into consideration this table, peers with index from 1 to 300 belong to category $T4.2$, peers with index from 301 to 600 belong to category $T4.1$ and peers with index from 601 to 1000 belong to category $T1$. We have considered a situation where we have a high percentage of malicious peers to show the effectiveness of our proposed scheme.

Performance Metrics

In this set of simulations, we focus on the following performance metrics:

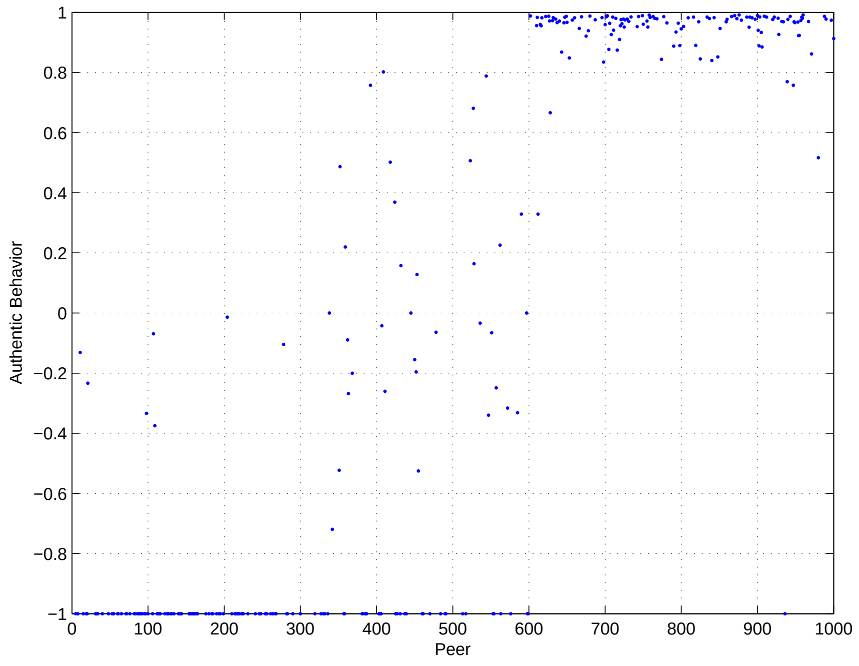


Figure 4.3: Authentic Behavior with *IDA* (with no liar peers)

- 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.

Simulation Results

Figures 4.3 and 4.4 show the *Authentic Behavior* values for peers when using *IDA*. Figure 4.3 presents the results in a situation where no peer lies in its feedbacks, while figure 4.4 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 4.2 with the fourth column set to zero in all categories.

It is clear from figure 4.3 that *IDA* is able to differentiate among peers and detect those that send inauthentic files. Good peers (with index from 601 to 1000) have high *AB* values while malicious peers (from 1 to 600) have low *AB* values (most of peers with index between 1 and 300 have a value of -1). However, if liar peers exist, those peers affect badly

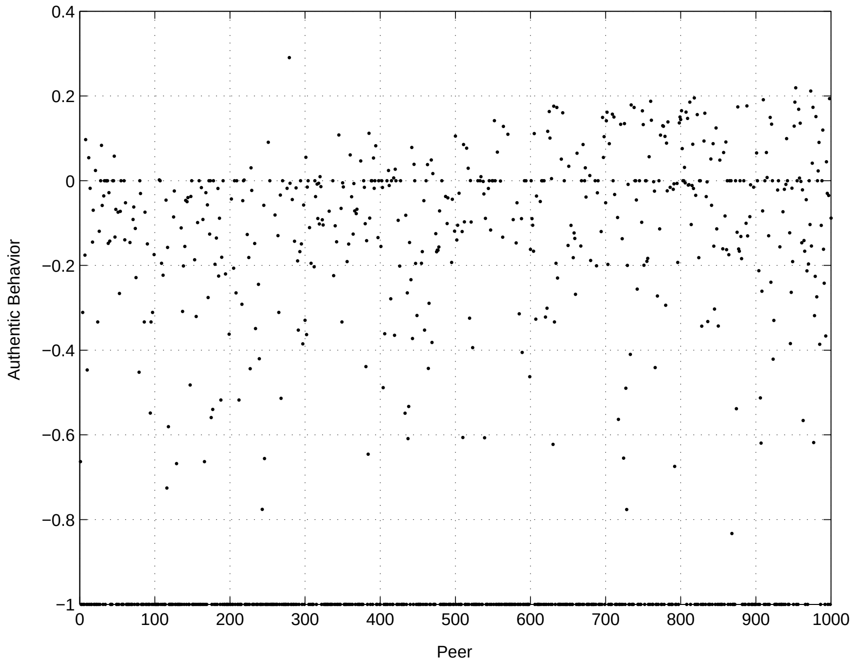


Figure 4.4: Authentic Behavior with *IDA* (with liar peers)

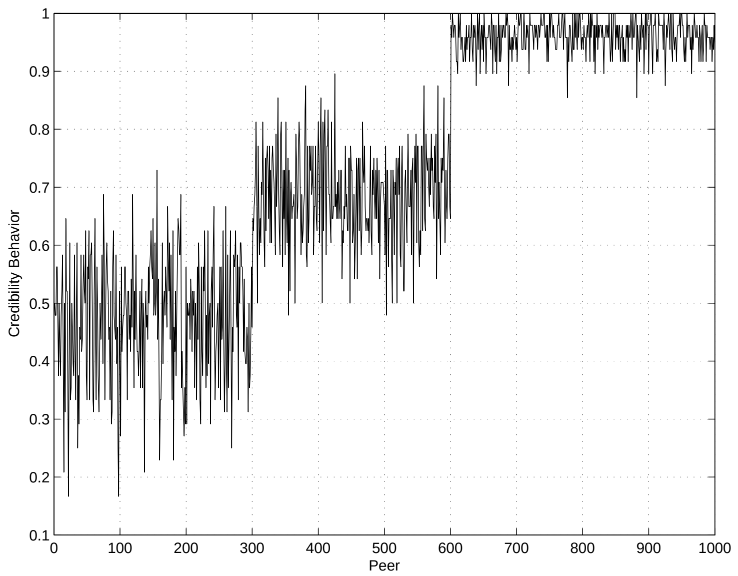


Figure 4.5: *Credibility Behavior*

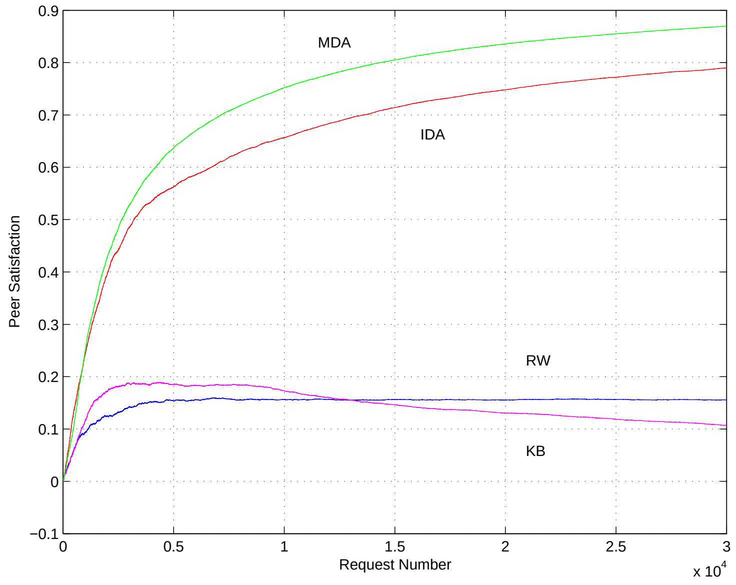


Figure 4.6: Peer Satisfaction

the system and makes it difficult to differentiate among peers (cf. figure 4.4). This affects greatly the performance of the system as will be shown in figure 4.8.

Figure 4.5 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 index from 601 to 1000) have a very high value of credibility 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 (index from 301 to 600) have higher credibility than those with higher probability of lying (index 1 to 300).

Figure 4.6 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. 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., 0.15 satisfaction) can be explained as follows; With the values of table 4.2, we can expect to have $(0.99 \times 40\% + 0.5 \times 30\% + 0.1 \times 30\% =)$ 57.6% of authentic uploads and $(0.01 \times 40\% + 0.5 \times 30\% + 0.9 \times 30\% =)$ 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 $(57.6 - 42.4)/(57.6 + 42.4) = 0.15$.

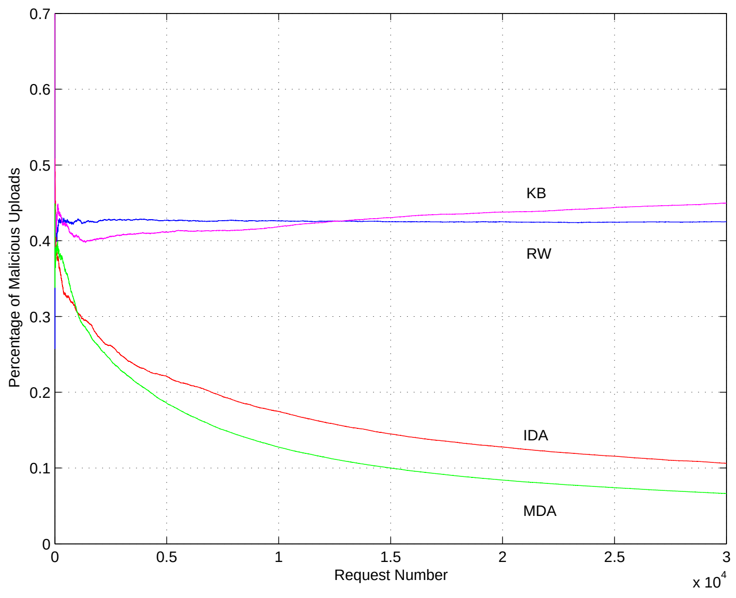


Figure 4.7: Percentage of Malicious Uploads

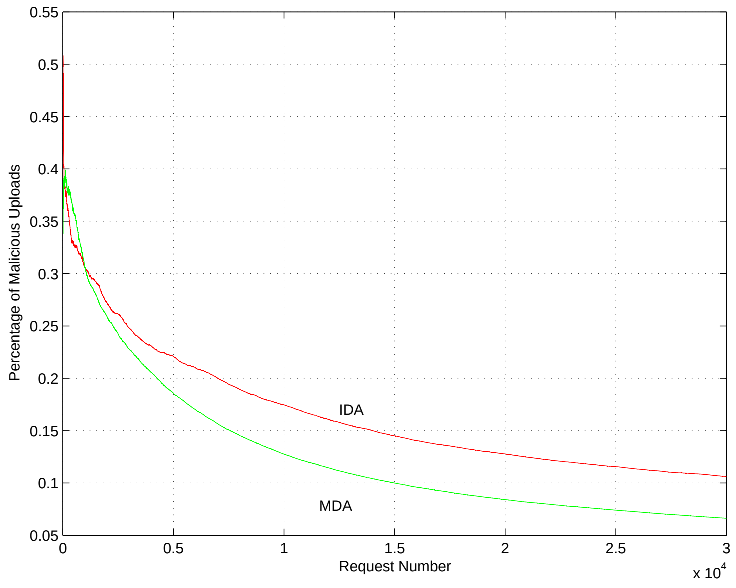


Figure 4.8: Percentage of Malicious Uploads for MDA and IDA

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 4.7 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 4.4. 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 4.6. Figure 4.8 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%.

4.5 Concluding Remarks

In this chapter, we introduced the *Credibility Behavior* as the second dimension of the trust management framework. We proposed the *Malicious Detector Algorithm* that is able to detect liar peers. The new concept of *suspicious transactions* is introduced to detect these liar peers. Performance evaluations show that the proposed scheme is able to detect and isolate malicious peers from the system, hence, providing higher peer satisfaction and better network resource utilization.

In the next chapter, the *Contribution Behavior*, the third dimension of trust is introduced. Peers' contribution is used as a guideline for service differentiation.

Chapter 5

Trust Management: Contribution Behavior

In chapter 3 and chapter 4, we introduced the *Authentic Behavior* and the *Credibility Behavior* of peers respectively. In this chapter, we introduce another important trust dimension which is the *Contribution Behavior*. We propose a new service differentiation scheme based on peers' contribution.

5.1 Motivation

Although most reputation management schemes try to achieve the goals described in Section 2.2, mechanisms for providing incentives and service differentiation are needed to achieve the following goals:

1. Motivate peers to share files and contribute to the system.
2. Reward the reputable peers by providing them with better service, and punish malicious peers.

Peers need to be motivated to display good behavior because it will have an impact on their future interactions. Political scientist Robert Axelrod refers to this phenomenon as *the shadow of the future* [64]. For example, in the case of the eBay reputation system, members have interest to get a high reputation value and maintain a good history of transactions as members with high reputation values are more trusted and selected for commercial transactions. The higher is the reputation of a seller, the higher is the chance that buyers will trust to deal with him.

In a P2P file sharing system, the situation is different. What is the benefit that a peer can gain from having a high reputation value? This peer will be more and more requested for uploads which is not a gain for this peer, but more for the peers that download from it (cf. figure 3.7). In P2P systems, if all peers receive the same service regardless of their behavior, peers will not be motivated to strive for high reputation values since they will be always asked to upload files without receiving any special benefit or reward. This is why service differentiation is needed.

Some of the proposed reputation-based P2P systems use peers' reputation as a guideline for service differentiation. This means that a peer with a high reputation, will receive better service than a peer with a lower reputation. This however does not address the problem of free riders. For example, a free rider may upload few authentic files and get a high reputation. Then, the free rider starts taking advantage of the system thanks to its high reputation. This is called the milking phenomenon.

For these reasons, we argue that a good scheme for service differentiation should be able to detect free riders and malicious peers and lower the service provided to them. For example, the system may reject search requests generated from free riders. A more elaborated discussion will be presented in Section 5.4. This will have a double effect. On one hand, this will encourage free riders and malicious peers to change their behavior if they want to receive a better service. And, on the other hand, good peers will receive a better service and will be motivated to continue providing good service.

5.2 Contribution Behavior

In this chapter, we propose a contribution management scheme. Peers will have to contribute to the system to receive services. The *shadow of the future* is maintained because peers are forced to contribute to be served. The higher the contribution value, the greater the services available to the peer. The contribution of peers rather than their reputation is used as a guideline for service differentiation.

The proposed scheme will allow to achieve the following objectives:

- Stopping the egoistical behavior of free riders that want only to take advantage of the system. This is achieved by providing the right incentives for free riders to change their behavior from free riding to positively contributing to the system and punishing them otherwise.
- Creating a competitive environment that will push peers to continuously being available to upload files.

- Allowing new comers or formerly free riders to build their reputation and increase their contribution.

The proposed contribution system along with the reputation and the credibility management systems already proposed in chapters 3 and 4 form a coherent and complete framework that addresses major issues related to peers behavior in partially decentralized P2P systems.

5.2.1 Contribution Components

In a reputation-based system with millions of users, the competition to upload requested files is very high. Peers with higher reputation values are always chosen. If we consider that the *Contribution Behavior* of a peer is computed based only on its uploads and downloads, highly reputable peers will have higher contribution values and will receive better services. New comers and peers that are still in the process of building their reputation will not be selected to perform the upload. These peers will receive lower services and may have no or not enough files to upload to other peers. They will not be able to increase their contribution and reputation values. This may lead to peers' starvation. Therefore, we need to recognize peers that are available to upload files and reward them. With the recognition of peers' availability, peers will have a chance to receive services, and build their reputation. These peers will, slowly, but surely, have their requests handled by the system. They will be able to download files, have more chances to share with others, and increase their reputation and contribution values gradually.

The *Contribution Behavior* of a peer represents its participation in terms of sharing files and positively contributing to the system. The *Contribution Behavior* should be based on:

- Peer's *Availability*: being available for uploading requested files.
- Peer's *Involvement*: non-malicious uploads performed versus downloads received by a peer.

Several other features may be taken into consideration to assess peers' contribution in addition to the *Availability* and *Involvement*. These features could be peer's upload rate, peer's uptime, the diversity of files that a peer is providing, or sharing rare and hard to find files, etc. However, in our opinion the most important concepts that prove the peer's contribution to the system are its *Availability* in terms of being available and ready to upload the requested files and its *Involvement* in terms of what the peer has positively uploaded to the system compared to what it has downloaded from it.

5.2.2 Peer Availability

When a peer requests a file and receives a list of peers providing this file, all these peers are available for an eventual upload. These peers can be considered as contributor peers since they are willing to upload the requested file. Since only after an upload is actually performed that it can be assessed as good or malicious, all these peers have to be rewarded for being available irrespective of being good or malicious.

Let the value of $Available_i$, be the number of times, the peer P_i was available for an upload. This value is incremented by the supernode of P_i each time peer P_i is available to provide a requested file after a search request is received.

The availability of peer P_i ($Availability_i$) is computed as the ratio between $Available_i$ and the average $Available_k$ for all peers P_k attached to the same supernode. The average of $Available_k$ can be computed easily by each supernode since $Available_k$ is stored at the supernode level for each peer P_k that is connected to this supernode. $Availability_i$ is computed as follows:

```
$Average_{sup} = \frac{\sum_k Available_k}{NbrPeers_{sup}}$
**if** $Average_{sup} > 0$ **then**
  $Availability_i = Min(\frac{Available_i}{Average_{sup}}, 1)$
**end if**
```

Where $NbrPeers_{sup}$ is the number of peers attached to the supernode $Sup(i)$. Note that $Availability_i$ of peer P_i is computed based on the average availability of all peers attached to the same supernode. In case that $Availability_i \geq 1$, $Availability_i$ is set to 1 which means that the peer is available more than the average availability of all peers that belong to its supernode. The $Availability_i$ value for peer P_i will be high if P_i is a contributor peer, otherwise, this value will be low. This is because the $Average_{sup}$ value is continuously increased by contributor peers. The goal is to create a competitive environment that will push peers to continuously being available for providing files. The value of $Average_{sup}$ could also be the average value among several supernodes. This value can be easily exchanged between supernodes.

If a peer P_i is available yet never solicited, this peer will not be deprived from benefiting from the system since its $Availability_i$ value will not be null. This peer will get a chance to receive service. It is important to note that the supernode $Sup(i)$ updates the value $Availability_j$ for all its peers periodically. The frequency of this update should not be high (e.g., after each search request) to avoid extra overhead and not too low to preserve accuracy.

In [65], it has been found that most of the shared content in Gnutella is provided by only 30% of peers which means that 70% of peers are free riders. Assuming that peers are uniformly distributed among supernodes. We can expect to have almost the same distribution for each supernode. This means that 70% of peers connected to a supernode are free riders and only 30% are contributor peers. Because of the high availability of contributor peers, free riders will have to be available in order to receive services from this supernode. A peer can achieve a high *Availability* value by accepting to share files with others, and being available for uploads during long periods of time. The greater the number of popular files this peer is offering, the greater its *Availability* value will be.

5.2.3 Peer Involvement

The $Involvement_i$ of peer P_i is defined as:

```
  **if** $(D^+_{i,*} + D^-_{i,*} \neq 0)$ **then**
    $Involvement_i = \frac{D^+_{*,i} - D^-_{*,i}}{D^+_{i,*} + D^-_{i,*}}$
  **else**
    $Involvement_i = D^+_{*,i} - D^-_{*,i}$
  **end if**
  $Involvement_i = Min(Involvement_i, 1)$
```

While the reputation value is based only on the uploads of a peer to reflect its *Authentic Behavior*, the involvement should be based on both the uploads and the downloads of the peer to express how much the peer gave to the system compared to how much the peer took from the system. The $Involvement_i$ of peer P_i is the ratio between what the peer has positively uploaded to the system and what it has downloaded from it. The term $(D^+_{*,i} - D^-_{*,i})$ means that the $Involvement_i$ value is sensitive to peer's maliciousness. This term affects both free riders and malicious peers since it will be very low for free riders and may be negative for malicious peers. Peers that download much more than they upload to other peers will get a low *Involvement* value. Thus, peers have to continuously upload files if they want to receive files from others. In case that $Involvement_i \geq 1$, $Involvement_i$ is set to 1 which means that the peer is contributing to the system more than what it is downloading from it.

Ideally, a peer should be charged only for its authentic downloads since it is not responsible for the malicious content that it received from other peers. However, some malicious peers may rate all their downloads as inauthentic so that these downloads will not be counted in the *Involvement* value. To avoid this situation, the total downloads is used for computing the *Involvement* value. This will also motivate peers to deal only with highly reputable peers.

5.2.4 Peer Contribution

The *Contribution Behavior* CTB_i of peer P_i is computed using algorithm 4.

Algorithm 4: Peer Contribution Algorithm

```
$a = Availability_i$
**if** $Involvement_i < 0$ **then**
    $b = -1$
**else**
    $b = Involvement_i$
**end if**
$c = Max(\alpha a + \beta b, 0)$
$CTB_i = Min(c, 1)$
```

The value of CTB_i is computed based on a weighted sum of a and b: $CTB_i = \alpha a + \beta b$, (with $\alpha \geq 0$ and $\beta \geq 0$). α and β are application dependent and represent the weights given to *Availability* and *Involvement* of peer P_i. The value of CTB_i is based on the maximum between $\alpha a + \beta b$ and 0 and the minimum between the obtained result and 1. This guarantees that CTB_i value will be between 0 and 1. An in depth analysis can be conducted for parameter settings to achieve the best performance for the system. In general, more weight can be given to β compared to α since *Involvement* is more important than *Availability*.

The use of *Credibility Behavior* in computing peers' *Involvement* will reduce significantly the impact of colluding peers that report fake transactions among themselves to increase their *Contribution Behavior* value (cf. equation. 4.4).

5.3 Trust Management Framework Components

5.3.1 The Use of Trust Components

Figure 5.1 shows the three dimensions of trust along with different aspects of behavior that they characterize. The *Authentic Behavior* allows to distinguish between good peers that send authentic files, and malicious peers. The *Credibility Behavior* makes possible to distinguish between liar and honest peers while the *Contribution Behavior* allows to distinguish between good contributor peers, and free riders and malicious peers.

Figure 5.2 shows a typical file request-download procedure involving the sender and receiver peers and their supernodes. The figure shows steps affected by the values of trust

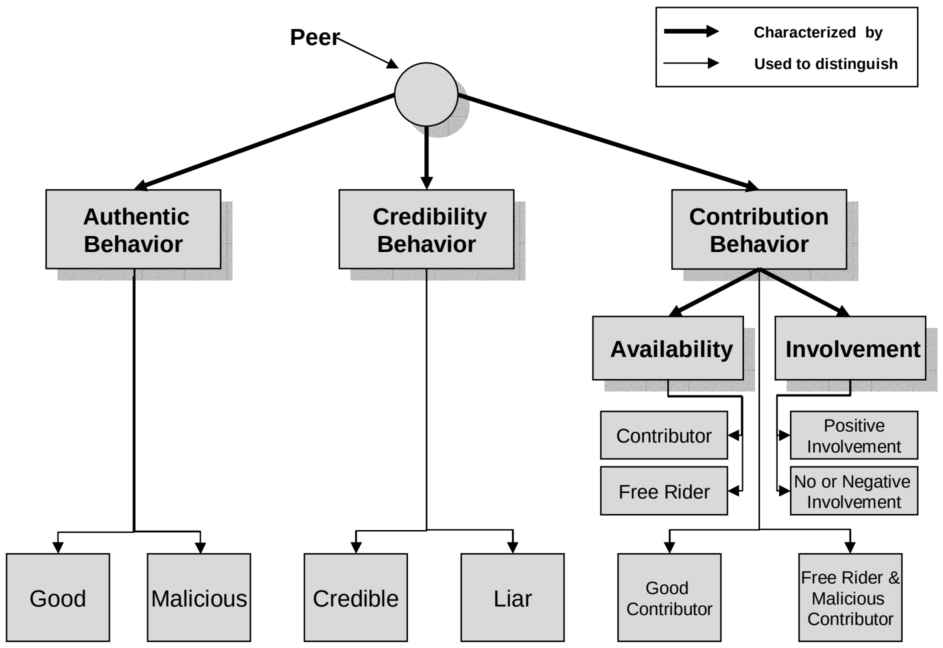
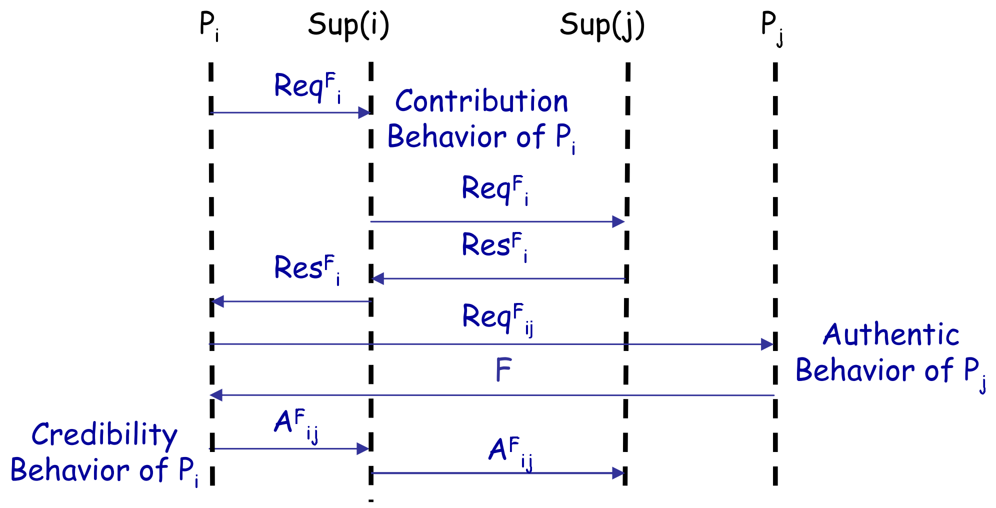


Figure 5.1: Peer Trust and Behavior Analysis

triplet. When peer P_i is requesting a search service Req_i^F from its supernode $Sup(i)$, this latter will perform the request only after considering the *Contribution Behavior* of peer P_i. According to peer's *Availability* and *Involvement*, the request can be processed or rejected. When peer P_i is given a list of peers providing the requested file Res_i^F which represents the result of the search request, peer P_i will choose peer P_j according to the *Authentic Behavior* of P_j. Peer P_i is not interested to know other characteristics of peer P_j since the most important issue for peer P_i is to receive the exact requested file. Peer P_i sends a request Req_{ij}^F to download file F from peer P_j. After downloading this file, peer P_i sends feedback $A_{i,j}^F$. The credibility of peer P_i will have a significant impact on the feedback and the reputation of peer P_j. Indeed, if peer P_i has a low credibility, this peer may send a wrong feedback and hence, affects the reputation of peer P_j. Hence, a peer may decide not to upload a file to a peer with a low credibility value (along the *Credibility Behavior* dimension), since the latter peer may wrongfully send negative feedback and affect badly the reputation of the peer performing the upload.



P_i
Sup(i)
Sup(j)
P_j
Req^F_i
Contribution Behavior of P_i
Req^F_i
Res^F_i
Res^F_i
Req^F_ij
Authentic Behavior of P_j
F
A^F_ij
A^F_ij
Credibility Behavior of P_i

Figure 5.2: A Typical Exchange between Peers

5.3.2 Reputation Data

Each peer P_i in the system has *Reputation data* (REP_{P_i}), stored by its supernode $Sup(i)$ that includes the following:

1. $D^+_{i,*}$: satisfactory downloads of peer P_i from other peers,
2. $D^-_{i,*}$: unsatisfactory downloads of peer P_i from other peers,
3. $D^+_{*,i}$: satisfactory uploads from peer P_i to other peers,
4. $D^-_{*,i}$: unsatisfactory uploads from peer P_i to other peers
5. N_i: The total number of downloads performed by peer P_i
6. N^*_i: The number of suspicious transactions
7. TF_i: The total size of all the files uploaded by P_i
8. $Available_i$: The number of times P_i was available to share files

When peer P_i joins the system for the first time, all values of its *Reputation data* REP_{P_i} are initialized to zero. Protecting the integrity of peers' *Reputation data* is imperative to prevent malicious peers from increasing their *Authentic Behavior* and *Contribution Behavior* values and take advantage from the system.

5.4 Service Differentiation

We divide service differentiation into two categories: *implicit* and *explicit* service differentiation.

Implicit service differentiation, is the service differentiation that results from the normal evolution of the system. For example, when a peer has a low reputation (e.g., its *Authentic Behavior* value), this peer will have a low probability of being selected for uploads, which will not allow it to increase its contribution value.

Explicit service differentiation, is the one that results from the explicit decision of system entities. For example, a supernode may decide to enforce service differentiation policies on the peers it manages. *Explicit* service differentiation can also be enforced at the level of peers. For example, a peer may decide not to upload to a peer with a low contribution value (along the *Contribution Behavior* dimension), since the peer requesting the upload may be a free rider.

In the proposed contribution management system, we focus on enforcing service differentiation policies at the supernode level. When a peer P_i sends a request to its supernode $Sup(i)$, this latter will associate to the request a probability $prob_i$ according to the contribution level of peer P_i. This is the probability of performing the requested service by $Sup(i)$. The higher the contribution value is, the more chances the supernode will execute the requests for this peer[1].

When supernode $Sup(i)$ receives a request for searching a file on behalf of peer P_i, $Sup(i)$ will compute the probability $prob_i$ for peer P_i, and will execute the request according to this probability as follows:

```
if $(D^{+}_{i,*} + D^{-}_{i,*}) \leq MinDownload$ then
  $prob_i = 1$
else
  $prob_i = CTB_i$
end if
```

The greater CTB_i value is, the more likely peer P_i will receive service from the system. Even if a peer did not get a chance to upload a file, it can still have its requests handled by the system based on its $Availability_i$. If peer P_i is contributing negatively by uploading malicious files, this peer will get a negative $Involvement_i$ value which will reduce its contribution value, and hence its probability to benefit from the system although this peer

[1]To prevent peers from repeatedly sending the same request to the supernode over and over until the request is handled, a minimum time period can be enforced between consecutive requests. This will motivate peers to contribute if they want their requests to be processed by the system.

may have a high $Availability_i$ value. The proposed mechanism allows to reduce significantly services provided to malicious peers that harm the system.

5.4.1 New Comers Policy

New comers to the system are entitled to download up to a maximum amount set to *MinDownload*. The probability $prob_i$, used by the supernode in this case, is equal to 1 to allow new comers (i.e., with no involvement) to download files. After exceeding this maximum amount of downloads, the probability used by the supernode will be computed according to the *Contribution Behavior* CTB_i.

5.4.2 White-washing Policy

White-washing is the situation when a peer changes its identity and rejoins the network with a new one. This may be beneficial for malicious peers. Indeed, once these peers are identified as malicious, they will not be able to upload inauthentic files. To start over again, malicious peers can rejoin the system with a new identity and their past history will simply be forgotten. However, a new comer has a null *Authentic Behavior* value (AB); hence, it has a neutral trust. To be able (to be chosen by others) to upload, this peer has to increase its AB. This peer will need to contribute positively by uploading authentic files before being able to upload inauthentic ones. This will not be very attractive to malicious peers whom their main goal is to harm the system as quickly as possible.

White-washing may also be beneficial for free riders. These peers could take advantage from the *MinDownload* value to download files without uploading to others. The value of *MinDownload* should be carefully chosen not to encourage peers to change identities and benefit from free downloads. A small *MinDownload* value will not allow newcomers to quickly benefit from the system. On the other hand, a high *MinDownload* value may encourage white-washing.

5.5 Rational Behavior

Rational Behavior for peers has been introduced for completely decentralized P2P systems in [54]. The algorithm makes rational peers change their behavior according to the output of their actions. This algorithm assumes a periodical update of peer behavior in terms of probability of sharing files (*ProbShare*). After each evaluation period, a peer will decide if a change in its behavior is needed or not.

In this chapter, we improved this algorithm to fit partially decentralized P2P systems. The following values are stored by each peer P_i:

1. $SuccessfulRequest$: Number of requests successfully performed by Sup_i for P_i during the current evaluation period.

2. $Request$: Number of requests sent to supernode Sup_i by peer P_i during current evaluation period (i.e., period of time),

3. $ProbShare$: The probability of peer P_i to share files with other peers during current evaluation period. Typically, free riders will have lower values of $ProbShare$ than contributor peers,

4. $increment$: Represents the unit of increasing or decreasing $ProbShare$,

5. $OldBenefit$: Benefit obtained during previous evaluation period,

6. $LastAction$: The action performed on $ProbShare$ during previous evaluation period. The value of $ProbShare$ will increase or decrease and the value of $LastAction$ will be 1 or -1 respectively.

The algorithm has been modified from its original version to become Algorithm 5.

Rational behavior involves comparing benefits before and after the evaluation period. If the new strategy (the new value of $ProbShare$) is better than the old strategy, the same action as in $LastAction$ will be performed. Otherwise, the opposite of $LastAction$ will be executed.

In our algorithm, if the old benefit and the new one have low values (almost null), the peer will increase its probability of sharing files $ProbShare$. In the original version, this case was not addressed and peers cannot receive any benefits when their $ProbShare$ is equal to zero. The original version leads to a deadlock and peers cannot change their behavior to receive better services. Moreover, in our algorithm, peers can evaluate their benefits from the system at different periods of time instead of making this evaluation in a synchronous way as it is the case in [54].

The majority of peers in the network are selfish (e.g., free riders) and they want to maximize their own utility. According to the proposed algorithm, this utility is the number of requests sent by a peer and handled successfully by its supernode. For free riders, the only way to achieve this goal is by increasing the probability of sharing files. As a result, the availability of these peers will increase and also their involvement. Thus, their contribution value will also increase. For malicious peers, they will have to upload authentic files. As a result, their involvement will become positive, increasing their contribution.

Algorithm 5: Adapted Rational Behavior Algorithm

```
At the end of each evaluation period, do
if $Request \geq 0$ then
  $NewBenefit = SuccessfulRequest/Request$
  if $NewBenefit > OldBenefit$ then
    if $LastAction == 1$ then
      $ProbShare = ProbShare + increment$
      $ProbShare = min(ProbShare, 1)$
    else
      $ProbShare = ProbShare - increment$
      $ProbShare = max(ProbShare, 0)$
    end if
  end if
  if $OldBenefit > NewBenefit$ then
    if $LastAction == -1$ then
      $ProbShare = ProbShare + increment$
      $ProbShare = min(ProbShare, 1)$
      $LastAction = 1$
    else
      $ProbShare = ProbShare - increment$
      $ProbShare = max(ProbShare, 0)$
      $LastAction = -1$
    end if
  end if
  if $(OldBenefit == NewBenefit)$ & $(NewBenefit <= 0.1)$ then
    $ProbShare = ProbShare + increment$
    $ProbShare = min(ProbShare, 1)$
    $LastAction = 1$
  end if
  $OldBenefit = NewBenefit$
  $SuccessfulRequest = 0$
  $Request = 0$
end if
```

5.6 Performance Evaluation

In this section, we simulate a system under two scenarios: service differentiation with static peer behavior and service differentiation with rational peer behavior. The goal from these simulations is to show that:

- Service differentiation is important
- Service differentiation based on *Contribution Behavior* instead of peers' reputation identifies better free riders and reduces the services provided to these peers
- Service differentiation based on *Contribution Behavior* will motivate free riders to change their behavior from free riding to contributing to the system.

5.6.1 Simulation Parameters

We use the following simulation parameters:

- We simulate a system with 500 peers and 500 files. We have chosen 500 peers since a supernode typically supports between 300 to 500 peers, depending on availability of resources [11].
- At the beginning of the simulation, each peer has at most 15 randomly chosen files and each file has at least one owner.
- 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
- Peers are divided into two categories: Contributors and Free Riders. Free riders constitute 70% of the peers. From each category, 30% of peers are malicious peers that send inauthentic content. Peers' behavior and distribution are summarized in table 5.1.
- $MinDownload$ is set to the average file size.
- We simulate $150,000$ requests.

We do not consider liar peers. This issue was already addressed in chapter 4.

Table 5.1 presents peers' distribution and their corresponding probability of sending inauthentic files. Following this table, peers with index from 1 to 350 belong to the category of free riders (FR), peers with index from 351 to 500 belong to the category of contributor

		Probability to send inauthentic files	
Category	Percentage	Malicious (30%)	Not malicious (70%)
Contributors	30%	0.9	0.01
Free Riders	70%	0.9	0.01

Table 5.1: Peers' Behavior and Distribution

peers (CP). Accordingly, peers with index from 1 to 245 are good free riders (GFR) and peers with index from 246 to 350 are malicious peers in addition of being free riders (MFR). Peers with index from 351 to 395 are malicious contributor peers (MCP) that provide malicious content but still participate in uploading files to other peers. Peers with index from 396 to 500 are good contributor peers (GCP). We have considered a situation where we have a high percentage of free riders as observed by [65] to show the effectiveness of our proposed scheme in identifying and handling free riders both good and malicious.

5.6.2 Static Behavior

In this first set of simulations, we consider static peer behavior. This means that peers do not change their behavior over time. We will compare the following schemes:

1. The reputation management scheme with no service differentiation ($NOSD$). This is to show the importance of service differentiation among the peers.

2. The reputation management scheme with the reputation value as a guideline for service differentiation. We call this scheme the Reputation-Based Service Differentiation ($RBSD$). Since the reputation values (i.e., AB_i) are between -1 and 1, in this scheme, the probability $prob_i$ is computed as follows: $prob_i = (1 + AB_i)/2$.

3. The reputation management scheme with the *Contribution Behavior* as a guideline for service differentiation. We will call this scheme the Contribution-Based Service Differentiation ($CBSD$).

Free riders share files with a probability of 5%. In addition, 100 of the non malicious free rider peers will accept uploading the first file to get a high reputation.

In these simulations, we will focus on the following performance metrics:

- Percentage of successful requests: computed as the total number of requests that have been performed for the peer during the simulation over the total number of all submitted requests by this peer.

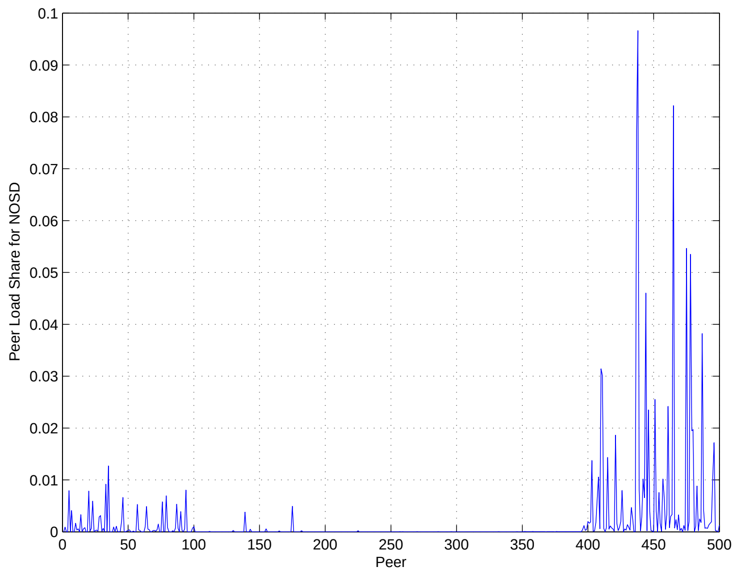


Figure 5.3: Peer Load Share for $NOSD$

- Peer contribution level: shows the contribution behavior of each peer.
- Peer load share: this parameter is computed as the normalized load supported by the peer. This is computed as the sum of the uploads performed by the peer over the total uploads in the system.

No Service Differentiation case

In case that there is no service differentiation, all peers categories (i.e, GCP, MCP, GFR, and MFR) will receive the same level of service. Obviously, it is unfair that free riders (GFR and MFR) and malicious contributor peers (MGP) benefit from the system even if they are not supporting the same load as good contributor peers (GCP) do.

Figure 5.3 depicts the normalized load supported by different peers after $150,000$ requests sent to the system in the case of the $NOSD$ scheme. The X axis represents peers' id while the Y axis represents the normalized peer load share. From the figure, it is clear that the proposed reputation management scheme (*Authentic Behavior*) is able to detect, identify and isolate malicious peers (i.e., peer id 246 to 395), as they are not requested to upload files, preventing the peers from receiving malicious content. Since the probability of sharing for GCP is equal to 1, all the load is almost supported by non malicious contributor peers (i.e., peer id 396 to 500). Free riders do not contribute significantly to the system since they do not share any files as their probability of sharing is only 0.05. Good free

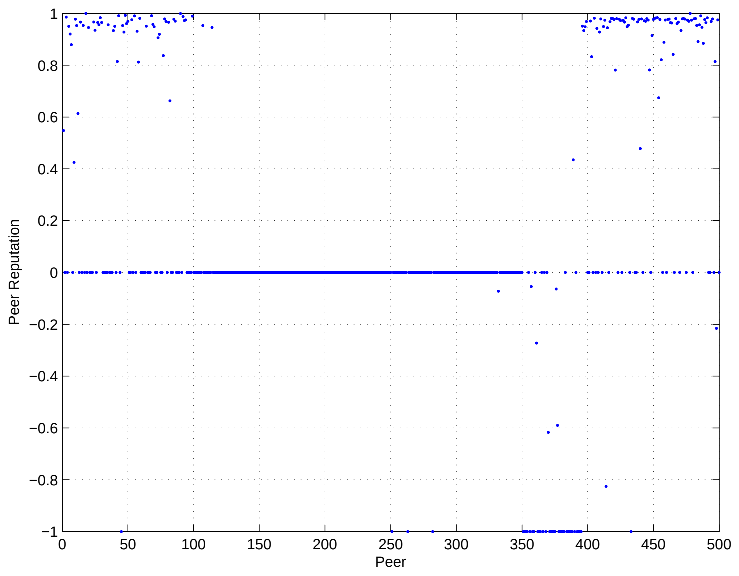


Figure 5.4: Peers Reputation in $RBSD$

riders that are using the milking strategy (i.e., peer id 1 to 100) have been participating in uploading some files to get a high reputation value.

Since there is no service differentiation, all the requests sent to the supernode will be performed regardless of the contribution of the peers. This is obviously unfair to the peers that contribute significantly to the system.

Service Differentiation case

Figure 5.4 depicts the reputation values of the peers (i.e., the *Authentic Behavior*) in the case of the Reputation Based Service Differentiation ($RBSD$) scheme. It is clear that the scheme is able to identify malicious peers. However, the scheme is not able to differentiate between free riders and contributor peers. Reputation is not a good indicator of the contribution of the peer as we can see from comparing figure 5.3 and figure 5.4.

Figure 5.5 depicts the *Contribution Behavior* value in the case of the Contribution Based Service Differentiation ($CBSD$) scheme. Comparing this figure with figure 5.3, the Contribution Behavior value is a good indicator of the peer load share. In other words, a peer with a high contribution level is supporting more load than a peer with a low contribution level. Note that the Contribution Behavior values of malicious peers (i.e., peer id 246 to 395) are null. This is because malicious peers are harming the system by uploading malicious files. This means that the Contribution Behavior value can be used

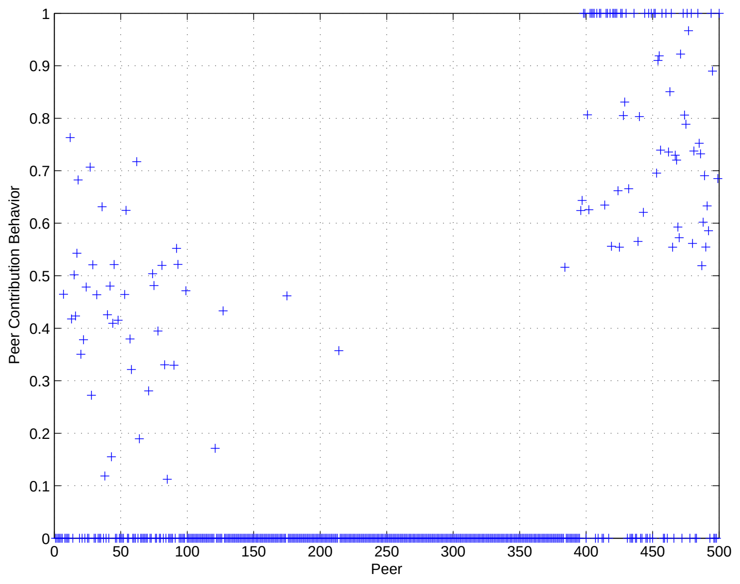


Figure 5.5: Peers Contribution Behavior in *CBSD*

for service differentiation which will effectively reward good peers and punish both free riders and malicious peers.

Figures 5.6 and 5.7 show the percentage of successful requests for *RBSD* and *CBSD* respectively (i.e., accepted requests by the supernode). From figure 5.6, we can notice that free riders have about 50% chance to have their request processed by the supernode. Free riders with high reputation values (i.e., peer id 1 to 100) have almost the same percentage of successful requests as non malicious contributor peers. However, free riders did not contribute at the same level. In figure 5.7, free riders with id from 1 to 100, have a lower percentage of successful requests since they uploaded only few files compared to non malicious contributor peers GCP. The latter peers are rewarded with a high level of service since they have supported almost all the load. They contributed significantly and positively to the system. The supernode processed their requests with a high probability. Some of the malicious peers uploaded more malicious content than good one, hence their percentage of successful requests is very low. This is because their contribution is null as shown in figure 5.5. Also, free riders with id from 101 to 350 receive a very low level of service since their contribution values are very low. Indeed, these peers did not upload files nor were they available to share files, and hence, their *Availability* and *Involvement* values are very low.

Note that in these simulations, we assumed a static peer behavior. This is to assess the capability of the proposed scheme in detecting malicious and free rider peers and

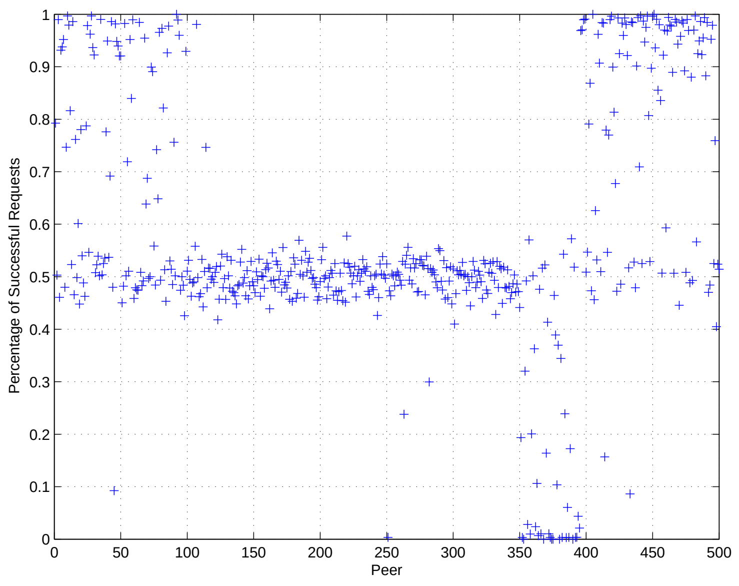


Figure 5.6: Percentage of Successful Requests for $RBSD$

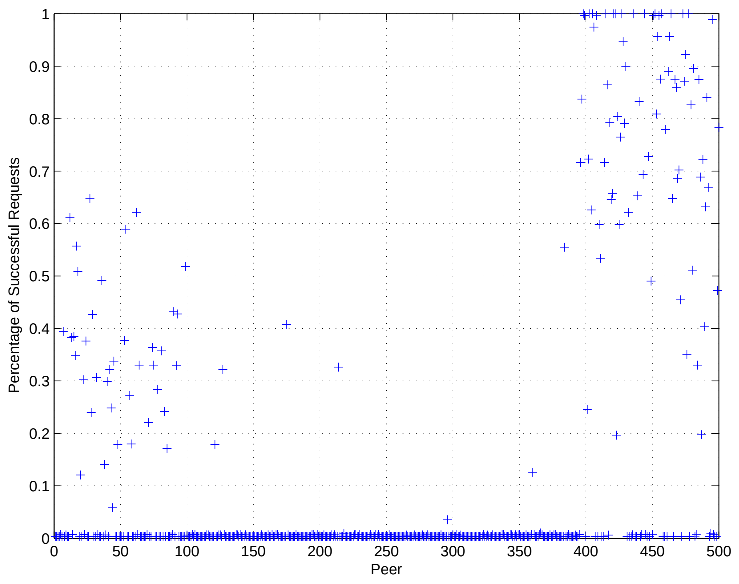


Figure 5.7: Percentage of Successful Requests for $CBSD$

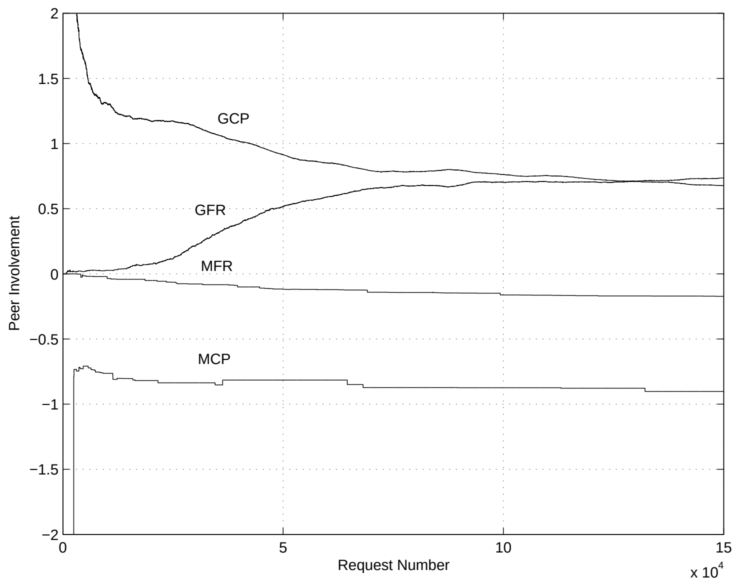


Figure 5.8: Peer Involvement

preventing them from obtaining good service. In real life, however, peers will tend to change their behavior. Free rider peers with a rational behavior will change from free riding to contributing to the system.

5.6.3 Rational Behavior

In the following set of simulations, we assume that peers use rational behavior as presented in Section 5.5. The goal is to show that under the rational behavior assumption, free riders will change their behavior from free riding to sharing and uploading files. As in real life, peers will tend to change their behavior to maximize the benefit obtained from the system.

Initially, free riders share files with a null probability and contributor peers with a probability equal to 1. The probability of sharing ($ProbShare$) is increased or decreased by a parameter set to 0.2.

Figure 5.8 shows the average peer involvement for different categories of peers. The X axis represents the number of requests while the Y axis represents the average peer involvement. At the beginning of the simulation, the involvement of free riders is very low since they are not sharing any files. As their probability of sharing increases, good free riders (GFR) get more involved in the system by uploading files until they reach a similar value as good contributor peers (GCP). The average peer involvement for good contributor

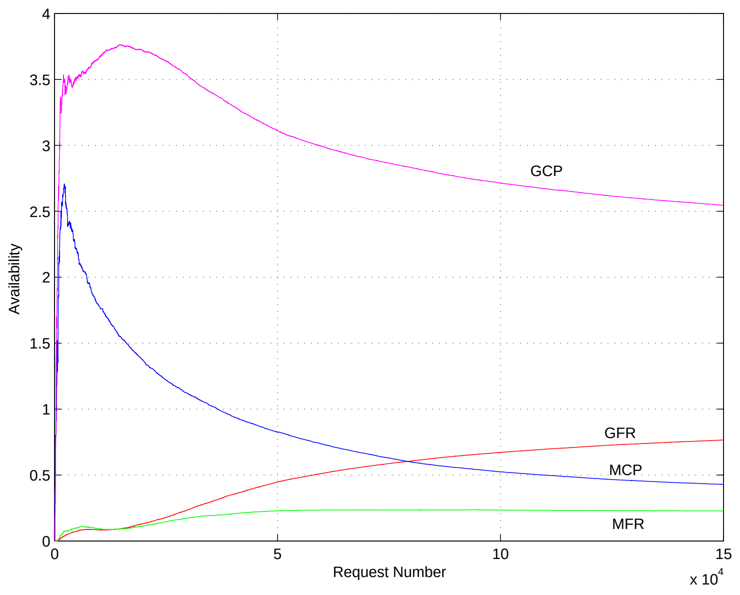


Figure 5.9: Peer Availability

peers decreases gradually since they are uploading less than before due to the fact that good free riders are becoming more involved in the system. As a consequence, GCP are relieved from supporting a high load, reducing the amount of resources dedicated by those peers to the system. This is considered an additional benefit received by GCP in addition to receiving higher services as will be shown in figure 5.10. However, malicious peers (both MFR and MCP) have negative involvement values since they are uploading more malicious content.

Figure 5.9 shows the average peer availability for different categories of peers. At the beginning of the simulations, the availability of free riders is null since their probability of sharing files is null. As this probability increases for good free riders, their availability also increases. Hence, their contribution increases and also the amount of received services. During the beginning of the simulation, the availability of good contributor peers (GCP) increases as they are the only ones available to upload files. However, the availability of malicious contributor peers (MCP) decreases. Using the contribution behavior as a guideline for service differentiation, these peers get less services, hence they will not be able to download as many files as good contributor peers.

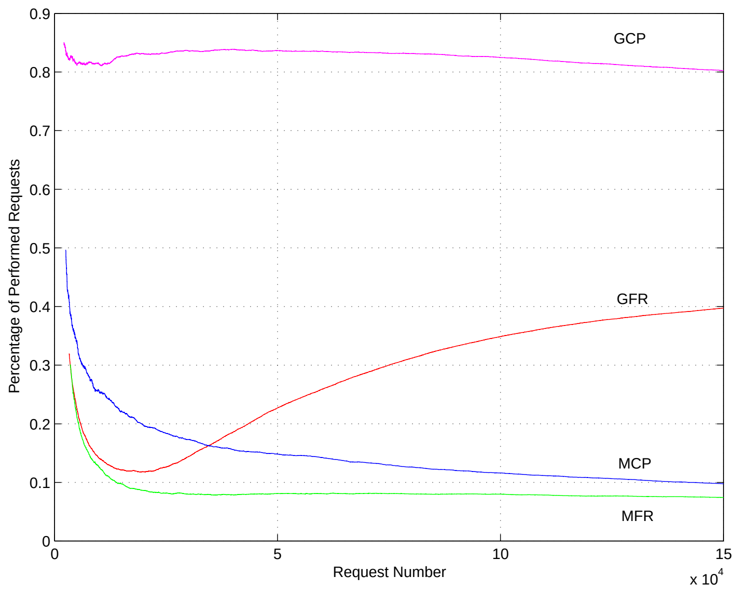


Figure 5.10: Percentage of Performed Requests with Contribution Behavior based on Availability and Involvement

Impact of Availability

We want to investigate the impact of Availability on the percentage of performed requests (i.e., accepted requests by the supernode) for free riders and contributor peers. Figure 5.10 shows the results obtained in the case where the Contribution Behavior is based on both peers' Availability and Involvement. Figure 5.11 shows the results in the case where the Contribution Behavior is computed based on peers' Involvement only.

Figure 5.10 shows that at the beginning of the simulation, only 30% of free riders' requests are performed by the system. This is thanks to the minimum amount of downloads $MinDownload$ they are authorized to have. This percentage will decrease gradually until free riders do not receive any significant benefit from the system due to their low contribution as explained earlier. This will push these peers to change their behavior and start sharing files with others. As GFR probability of sharing increases, so does the benefit they receive from the system. Malicious contributor peers (MCP) and malicious free riders (MFR) have a very low percentage since their involvement is negative. Good contributor peers (GCP) get a high percentage of accepted requests since they have a high contribution value due to their high availability and high positive involvement.

Figure 5.11 shows the results in the case where the Contribution Behavior is computed based on peers' Involvement only. This figure shows that good free riders (GFR) receive a

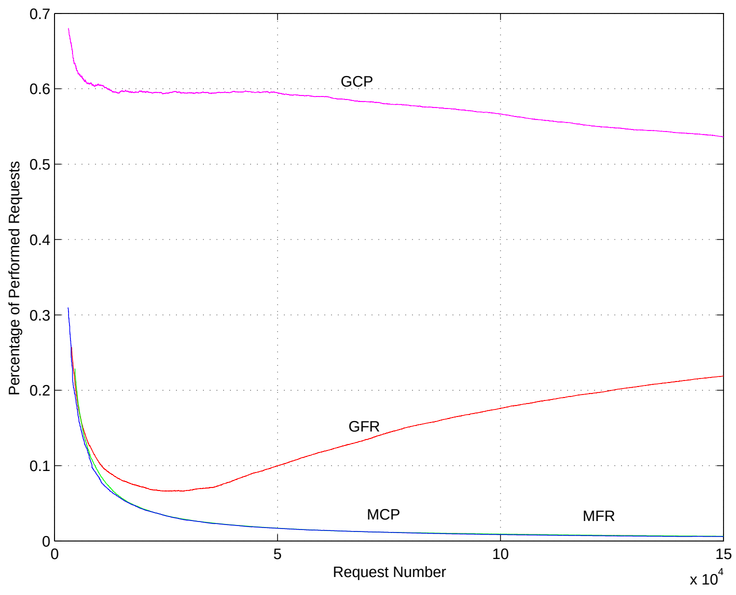


Figure 5.11: Percentage of Performed Requests with Contribution Behavior based only on Involvement

lower level of service compared to the previous case (see figure 5.10). Also, good contributor peers (GCP) receive a lower percentage of performed requests in figure 5.11 compared to the percentage received by these peers in figure 5.10. Using peers Availability and Involvement to compute the contribution behavior will reward better GCP and GFR. Note that in this case, both MCP and MFR also receive a slightly better service as shown in figure 5.10. Although, these peers do not deserve any benefit from the system, our new scheme provides them with an opportunity to receive services and change their behavior. Using the new scheme, these peers can slowly download good quality files and be able to upload them increasing their contribution and hence, their reputation.

We also want to investigate the impact of Availability on the normalized load supported by different categories of peers. Figure 5.12 shows the normalized load in the case where the Contribution Behavior is computed based on both peers Availability and Involvement. Figure 5.13 shows the load in the case where the Contribution Behavior is computed using only peers Involvement.

Figure 5.12 shows that at the beginning of the simulation, since the probability of sharing for free riders is null, they were not participating in uploading files and all the load was exclusively supported by good contributor peers (GCP). Note that malicious contributor peers (MCP) are detected very quickly by the system and are isolated (i.e., not requested for uploads). Using our proposed scheme for service differentiation and

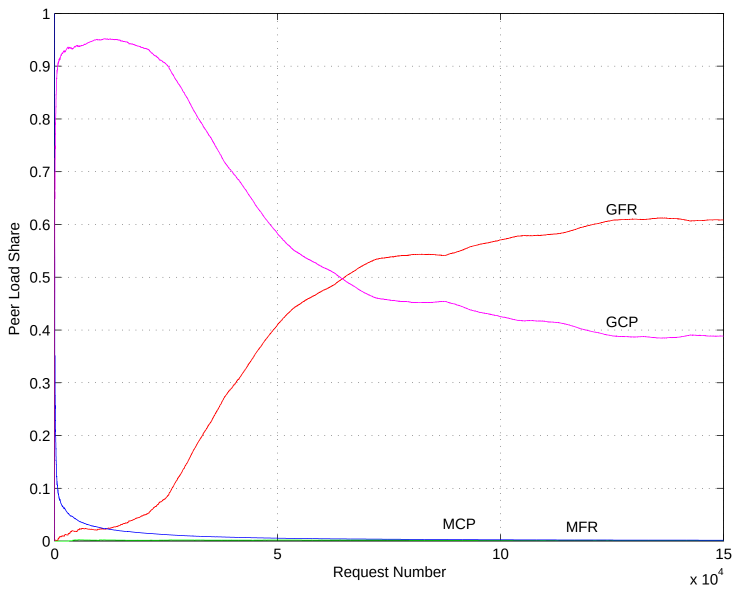


Figure 5.12: Peer Load Share with Contribution Behavior based on Availability and Involvement

with rational behavior, good free riders (GFR) are forced to share and upload files to get high level of service. Good contributor peers (GCP) are rewarded by the reduction of the supported load since good free riders are now uploading files. As to malicious peers (both MFR and MCP), they are not participating in uploading files since our proposed scheme is able to identify and isolate them.

As shown in figure 5.13, using the Contribution Behavior based only on peers *Involvement* does not motivate free riders to share files in the same manner as shown in figure 5.12. In figure 5.13, GFR will need more time (150, 000 requests) to support equally the load with GCP. In figure 5.12, GFR will start supporting the load equally with GCP only after 60, 000 requests.

In summary, using service differentiation free riders change their behavior and start participating positively to the system. The new scheme provides the right incentives to motivate free riders to start sharing. Free riders and malicious peers are punished. The new scheme successfully achieves the objectives described in Section 5.1.

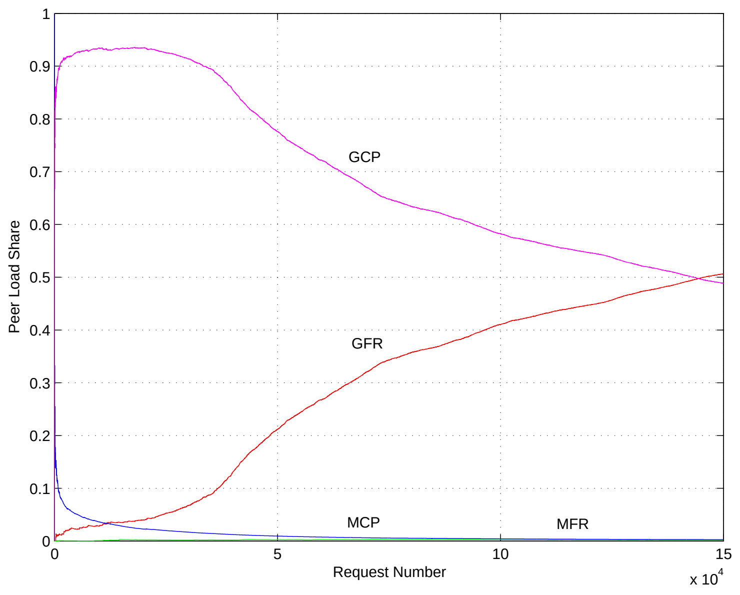


Figure 5.13: Peer Load Share with Contribution Behavior based only on Involvement

5.7 Concluding Remarks

In this chapter, we introduced the *Contribution Behavior* which is the third dimension of trust. The *Contribution Behavior* of a peer represents its participation in terms of sharing files and positively contributing to the system. This value is computed based on peer's *Availability* and its *Involvement*. *Availability* shows if a peer is available for uploading requested files, while its *Involvement* shows non-malicious uploads performed versus received downloads.

We also introduced a novel service differentiation scheme for partially decentralized P2P systems. The *Contribution Behavior* is used as a guideline for service differentiation rather than peer's *Authentic Behavior*. While, peer's *Authentic Behavior* reflects the authenticity of the files this peer is uploading, the peer's *Contribution Behavior* reflects its availability to sharing files taking into account its uploads compared to its downloads. The proposed scheme provides the right incentives for free riders to share files. Performance evaluations confirm the ability of the proposed scheme to effectively identify both free riders and malicious peers and reduce the level of service provided to them. On the other hand, good peers receive better service. Assuming a rational behavior, free riders tend to increase their contribution to get better service and indirectly reducing the load supported by good contributor peers. Moreover, the proposed scheme generates a competitive environment where peers are forced to continuously participate to benefit from the system, this way, reducing significantly the milking phenomenon.

In P2P file sharing systems, peers have to choose the files of interest from a large collection of files. This task is difficult and time consuming. To alleviate the peers from the burden of manually looking for relevant files, recommender systems are used to make personalized recommendations. In the next chapter, we propose a novel recommender system.

Chapter 6

Personalized Recommendations

6.1 Introduction

While reputation systems are used to enforce appropriate behavior by rating users as explained in chapter 3, recommender systems are used to allow satisfactory transactions by rating the quality of items (e.g., products, services).

In P2P file sharing systems, users are overwhelmed by a large collection of files available for download. Unfortunately, finding files of interest is time consuming. Recommender systems suggest to users files based on their profile. These users will be motivated to download the recommended files and hence, will remain active members. While they are downloading the files, they will upload files to others increasing their contribution to the system (cf. Section 5.2).

Recommender systems are widely used in e-Commerce applications (e.g., amazon.com, BizRate.com, Epinions.com, Yahoo.com) [66, 67, 68]. Recommender systems take advantage of the collected data that represents customers' experiences to predict their future needs. These systems suggest products and services that most likely will be of interest to the customers. The collaborative filtering recommender techniques are achieving widespread success on the web [67, 69, 68].

Although, e-Commerce applications have been using recommender systems for at least a decade, this research field is still a fertile area in P2P systems. Only few research works have addressed recommender schemes in P2P systems [70, 71, 72].

In P2P file sharing systems, peers spend a significant amount of time looking for relevant and interesting files. However, the files available for download represent on one hand a rich collection for different needs and preferences and on the other hand a struggle for the peers to find files that they like. The search process in partially decentralized P2P file sharing systems can be divided into the following steps:

1. Providing keywords.
2. Sending the request to the supernode.
3. The supernode will search for the file by contacting local peers if the file is available locally or sending a request to other supernodes.

In this chapter, we propose a novel recommender framework for partially decentralized file sharing P2P systems. This recommender framework will help peers in the first step by finding relevant filenames for files of interest based on peers' profile. The profile reflects their past choices, experiences and preferences. The proposed recommender system is based on *collaborative filtering*. Peers collaborate to filter out irrelevant files and find interesting ones. Relationships between peers are explored by taking advantage from the partial search process used in partially decentralized systems. In order to make personalized recommendations, the implicit rating approach is used and hence, no additional effort is required from the users. In addition, the implicit rating helps in overcoming the problems that traditional collaborative filtering schemes suffer from like the *Cold start*, the *Data sparseness* and the *Popularity effect*.

Similarity has been used in many fields like natural and social sciences as well as engineering and statistics. Several metrics have been proposed to compute similarity. In this chapter, we investigate several similarity metrics in the context of P2P recommender systems. We adapt these metrics to the context of recommender systems and particularly to file sharing P2P systems. We also propose a new similarity metric. We investigate these similarity metrics in both the weighted and non weighted techniques. The impact of each similarity metric on the accuracy of the recommendations is analyzed.

6.2 Recommender Systems in e-Commerce

6.2.1 Content-based versus Collaborative Filtering

Most recommender systems in e-commerce [67, 69, 68] are based on one of the two following techniques: content-based and collaborative filtering.

In the content-based approach, items are described in terms of their characteristics and recommendations are based on items already selected by the active user (i.e., the user to whom a recommendation is made). This approach is time consuming and expensive for subjective files such as songs and movies. It is not recommended for highly dynamic environments with millions of users and files [69].

Collaborative filtering is the most widely used technique for recommender systems. This approach is based on collecting users' ratings. It suggests items based on similarities between the active user's profile and other users or similarities between items. In this approach, it is required that a large number of users rate items to ensure recommendation accuracy [69]. This technique has proved to be one of the most successful techniques in recommender systems in recent years.

Collaborative filtering can be divided into two main categories:

- User-based collaborative filtering algorithms: relationships between users are explored first to find similar users to the active user. These users are like-minded as the active user and based on their ratings of the item in question, a rating value is predicted. This value represents an estimation of the likeliness of the item in question by the active user.
- Item-based collaborative filtering algorithms: relationships are explored between items first rather than users. Items that are similar to the item in question are identified. Based on this similarity, a predicted rating is provided. The item-to-item recommender scheme used in amazon.com is an example of this approach.

6.2.2 User-based Collaborative Filtering

User-based collaborative filtering has the following steps:

- Consider the user-item matrix where each row represents the profile of a user and each column represents the users that have the item (e.g., purchased, rented, ...etc).
- Compute the similarity metric: the similarity metric is computed for each pair of users. This is used to predict the ratings for the active user A. Similar peers have almost similar tastes and preferences, and so, they will have similar ratings for the same items. For N users, a user similarity matrix $(N \times N)$ is computed.
- Choose the k most similar users to the active user A. These users are called neighbors of user A.
- Compute the predicted rating for the active user A for items i. These items are not yet purchased by the active user A.
- Recommend the items that have a high predicted rating value to the active user A.

The similarity matrix is usually computed using the *Pearson correlation* or the *Cosine measure* [67, 69, 68].

6.2.3 Using The Pearson Correlation

To compute the similarity of peers, the most used technique is Pearson correlation coefficient (PC) [67, 69, 68, 71]:

$$PC_{A,B} = \frac{\sum_{l=1}^{m}(R_{Al} - \bar{R}_A)(R_{Bl} - \bar{R}_B)}{\sqrt{\sum_{l=1}^{m}(R_{Al} - \bar{R}_A)^2 \sum_{l=1}^{m}(R_{Bl} - \bar{R}_B)^2}} \tag{6.1}$$

Where:

- A is the active user for whom a recommendation will be proposed.
- B is a user.
- $\bar{R}_B$ is the average rating by the user B.
- $\bar{R}_A$ is the average rating by the user A.
- m is the number of items that they both rated.
- R_{Al} is the rating given by the user A to the item l.
- R_{Bl} is the rating given by the user B to the item l.

The Pearson Correlation coefficient is significant if users share two or more ratings. Positive correlation shows similarities between users, while a negative value shows that these users are not similar and their interests are different.

In the similarity matrix, each row represents the similarity between a user and other users in terms of ratings. This matrix is used to predict the ratings for the current user. This is based on the assumption that if two users have similar preferences and interests, they will have similar ratings.

6.2.4 Using The Cosine Measure

Another way to compute similarity between users is to use the cosine measure [67, 68]. The active user A and a user B are represented by two vectors and the similarity between them is measured by computing the cosine of the angle between the two vectors.

$$\cos(\overrightarrow{A}, \overrightarrow{B}) = \frac{\overrightarrow{A} \cdot \overrightarrow{B}}{\| \overrightarrow{A} \|_2 \times \| \overrightarrow{B} \|_2} = \frac{\sum_{l=1}^{n} R_{Al} R_{Bl}}{\sqrt{\sum_{l=1}^{n} R_{Al}^2 \sum_{l=1}^{n} R_{Bl}^2}} \tag{6.2}$$

$\overrightarrow{A} \cdot \overrightarrow{B}$ denotes the dot product between the vectors $\overrightarrow{A}$ and $\overrightarrow{B}$, $\| \overrightarrow{A} \|_2$, $\| \overrightarrow{B} \|_2$ represent the Euclidean norm for the two vectors and R_{Al}, R_{Bl} represent their respective ratings for the n items.

6.2.5 Computing The Predicted Rating

The predicted rating value measures the likeliness of the active user A for an item l. The predicted rating value is computed as follows [67, 69, 68]:

$$PR_{A,l} = \bar{R}_a + \frac{\sum_{j=1}^{k} PC_{A,j}(R_{jl} - \bar{R}_j)}{\sum_{j=1}^{k} PC_{A,j}} \tag{6.3}$$

k is the number of users that are the neighbors of the active user A. These users are the most similar to the active user A.

Another alternative for computing the predicted rating is to use the cosine measure instead of the Pearson Correlation coefficient $PC_{A,B}$.

The items that have a high value of $PR_{A,l}$ are recommended to the active user A.

6.2.6 Challenges of Collaborative Filtering Algorithms

The main known problems of collaborative filtering are the followings [67, 69, 68]:

- *Cold start*: This problem occurs for a new user or at the start of the system. It is difficult to make recommendations for a new user based on users' similarities since no rating is provided yet or the user's profile is not known yet.
- *Popularity effect*: This problem occurs when the given recommendations are obvious and evident from the user's point of view.
- *Data sparseness*: This problem occurs when only few users have rated few items. It is difficult to predict the user's interests and make accurate recommendations.
- *Trust*: This problem occurs when untrustworthy users provide false ratings. The system should be able to choose only highly reputable users while making recommendations. This will reduce the impact of untrustworthy users that influence badly the recommendation accuracy and hence, will increase the trust given by the peers to the recommender system.

6.3 Recommender Schemes in P2P Systems

6.3.1 Related Works

In [70], the authors propose a decentralized recommendation system that takes advantage of the high clustering coefficient of Preference Networks. The nodes of these networks are users of a file sharing system and the links are connections between pairs of nodes that share one or more identical files. The authors experimentally prove that the preference networks are small worlds. They propose a recommendation scheme based on the fact that nodes can be naturally gathered together on the basis of common interests. The top-N ranked items are recommended to the user and the location information in the buddy tables can be used to locate the recommended items.

In [71], the authors propose a distributed collaborative filtering method that is self-organizing and operates in a distributed way. Similarity ranks between items are computed and are stored locally in buddy tables. To perform a recommendation for a given user, the buddy tables for all the items in user's profile are downloaded and the relevance ranks are computed based on a user-content relevance model. Based on this work, the authors in [72], introduce personalization on Tribbler, a P2P television system. In this work, buddyCast which is a distributed profile exchanger, generates a semantic overlay by clustering peers into social networks according to their profile. Periodically, a user connects either to one of his buddies to exchange social networks and current profile list (exploitation) or to a new randomly chosen user from the random cache to exchange this information (exploration). A ranked list is created based on the similarity of their profile with the profile of the active user. The buddy list of the selected user is merged and the top-N best ranked users are kept.

These recommender schemes are suitable for decentralized P2P systems but not for partially decentralized systems. In addition, theses schemes generate a significant amount of overhead to make files' recommendations. As an example, in [72], it is required to maintain the following lists by each peer in the system: the top-N most similar users, the top-N most fresh random IP addresses and the K most recently visited users. The periodic exchange and update of information between peers is costly.

6.3.2 Advantages of Recommender Systems

In P2P file sharing systems, the goal from using a recommender system is to achieve the following advantages:

- Attract more users by making the search process easier, and more efficient.

- Increase peers' satisfaction by informing them about files of interest.
- Increase peers' contribution as presented in Section 5.2, since peers will be motivated to stay connected to download the recommended files and upload files to other peers. Free riders may be motivated to share their files to get a profile that reflects their preferences in order to receive accurate recommendations.
- Preserve network resources since peers will not have to download a large number of files that they do not like and will just discard.

6.3.3 Evaluation of Recommender Systems

Several features can be taken into consideration for recommender schemes evaluation [67, 68]:

- The additional effort that users are required to make: Explicit rating requires users to participate by providing ratings. However, there is no guarantee that users will make such commitment. Therefore, taking information implicitly from users' behavior is more preferable.
- Ease of understanding by users: when the recommender scheme is not understood by the users, they will not trust the recommendations. Users need to understand how recommendations are made. The simplicity of the recommender scheme plays an important role for its success.
- The accuracy of the recommendations: providing accurate recommendations will increase peers' satisfaction.
- Ease of designing and maintaining the proposed system.
- Performance issues: the scheme should not suffer from scalability, *Cold start* and *Data sparseness* as it is the case in traditional collaborative filtering schemes.

6.4 The proposed Recommender Framework

In e-Commerce applications, the collaborative filtering technique is based on the ratings of the products provided by the customers. In P2P file sharing systems, the collaborative filtering technique can be used based on the ratings of the files provided by the users.

6.4.1 Implicit Rating versus Explicit Rating

After downloading a file, two rating approaches can be considered: explicit rating and implicit rating. In the *explicit rating* approach, the user has to explicitly provide a rating for each file she/he downloads according to its content (i.e., matches the user's preferences or not). This approach necessitates an additional effort from the users. A rating scheme from 1 (not interesting at all) to 5 (very interesting) can be useful to assure recommendation accuracy. Users have to provide their ratings for different files to enrich the system with different opinions and experiences. Since *explicit rating* solicits an additional effort from users, it is difficult to enforce, especially in systems where 70% of peers are free riders [65]. This approach will likely suffer from the *Cold start* and *Data sparseness*. Also, *explicit rating* provides malicious peers with a way to influence the rating system which may lead to the *Trust* issue described in Section 6.2.6.

The *implicit rating* approach does not require the users to rate the files. It assigns ratings implicitly. The fact that ratings are generated automatically without involving users, alleviate them from the burden of explicitly providing ratings for each file they have downloaded. We propose to assign a rating of 1 (*I like it*) to the files owned by the user. All other files are assigned a rating of 0 (*I do not know*). Note that a rating of 0 does not mean that the user does not like the file.

We adopt the *implicit rating* approach in the proposed framework since it solves the problems of collaborative filtering in e-Commerce. The *implicit rating* approach has the following advantages:

- It solves the *Cold start* problem: Indeed, even at the start of the system or when a new user joins the P2P system, a ratings of 0 or 1 is always automatically available for every file.
- It avoids the *Data sparseness* problem as ratings are available.
- The subjective rating of files opens the door to malicious peers to manipulate files' recommendation. The *implicit rating* approach avoids tampering with the ratings of the files by malicious peers. This reduces their impact on the recommender system and thereby minimizing the *Trust* problem.

6.4.2 User-based versus Item-based Collaborative Filtering

Figure 6.1 depicts the steps required in the proposed framework to make recommendations to the peers. During the life cycle of a transaction in a P2P system, the following steps are performed:

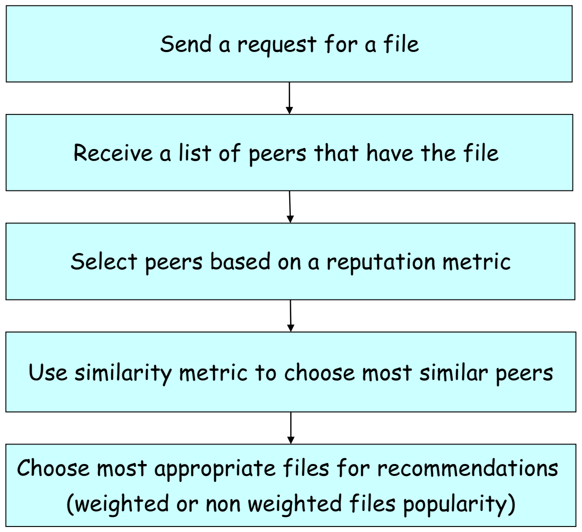
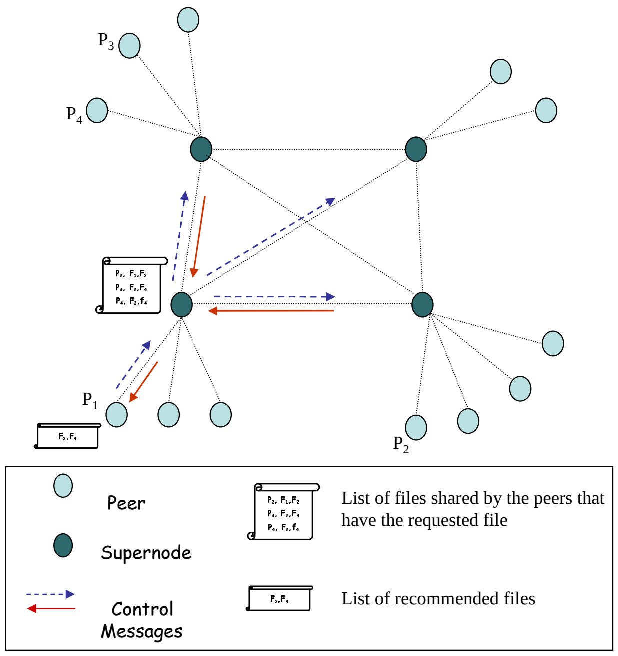

Send a request for a file
Receive a list of peers that have the file
Select peers based on a reputation metric
Use similarity metric to choose most similar peers
Choose most appropriate files for recommendations (weighted or non weighted files popularity)

Figure 6.1: Recommendation Transaction Life Cycle

1. Send a file request
2. Receive a list of peers that have the requested file
3. Use similarity metric to choose most similar peers to the peer requesting the file (the active peer)
4. Use the weighted or non weighted files' popularity to choose most appropriate files for recommendations. The non weighted file popularity approach selects the most popular files among the selected similar peers independently from how similar the peers are to the active peer. The weighted approach uses the similarity metric to compute a weighted file popularity before suggesting files for recommendation.

The similarity metrics considered in this work are used during step 3, and the weighted and non weighted approaches have been enforced in step 4.

Figure 6.2 depicts an example of the information flow between the peer P_1 requesting a file and its supernode. After receiving a request from peer P_1, and assuming the file is not found locally, its supernode sends a request to other supernodes. These supernodes will send back the search result which is a list of peers that have the requested file and the files that these peers are sharing. Based on this information, the supernode of P_1 will use the proposed recommender scheme to generate a list of recommended files.



P3
P4
P1
P2
Peer
Supernode
Control Messages
List of files shared by the peers that have the requested file
List of recommended files

Figure 6.2: The Proposed Recommender Framework

Since recommendations are given to peers in real time, it is preferable to explore relationships between peers rather than between files. We take advantage from the partial search process used in partially decentralized systems. The partial search performed by supernodes limits the number of peers in the search result. This number is much less than the number of files shared by all the peers in the system. In addition, finding relationships between all files is time consuming and is usually done offline. For these reasons, adopting user-based collaborative filtering in P2P systems is more practical than using item-based collaborative filtering algorithms.

6.5 The Similarity Metrics

6.5.1 Formal Notations

In the remaining of the chapter, we will use the following formal notations:

Let P be the set of all peers in the system.

Let F be the set of all files shared by the peers.

Let p_i be the requester peer looking for a file f_x. p_i is the user to whom the recommendation will be made.

Let P_{f_x} be the set of peers that possess the file f_x.

Let $F_{P_{f_x}}$ be the set of files that these peers possess in addition to f_x. This is the set of files that these peers are sharing.

Let $f : P \rightarrow \Omega(F)$, such that $f(p_j)$ is the set of files held by peer p_j for every j and $\Omega(F)$ is the power set of F. Then we have:

$$F_{P_{f_x}} = \bigcup_{p_k \in P_{f_x}} f(p_k)$$

6.5.2 Files' Popularity Based Recommendation (FP)

This technique will allow a peer to discover the files that are more popular within the peers that have the requested file.

Let $G_{p_i} = F_{P_{f_x}} - f(p_i) - \{f_x\}$ be the set of files that p_i does not have from the set $F_{P_{f_x}}$ not including the file f_x. These are all the files owned by the peers in P_{f_x} that the peer p_i does not have.

For every file $f_k \in G_{p_i}$, we define its popularity as:

$$Pop(f_k) = \frac{|P_{f_k} \cap P_{f_x}|}{|P_{f_x}|} \tag{6.4}$$

where $|P|$ is the cardinality of the set P.

The value of $Pop(f_k)$ is a numerical score that shows the popularity of the file f_k among the peers in P_{f_x}.

In this technique, files f_k that are more popular will be recommended such that $Pop(f_k) \geq t_1$, where t_1 is a threshold. This recommendation list is sorted according to the popularity of the files $Pop(f_k)$ with the files that are most popular at the top of the list. The supernode of peer p_i may keep track of these files for future recommendations. This technique will accelerate significantly the spread of popular files which will increase peers' satisfaction.

6.5.3 Asymmetric Peers Similarity Based Recommendation (AS)

Peers' similarity is an important factor in this technique. To be able to make accurate recommendations, we compare the active user's files against those of other users. The goal of this process is to find peers with similar preferences as the active peer p_i and make recommendations based on the files that they have. In fact, we apply the files' popularity approach within these peers.

For every p_j in P_{f_x} we define the similarity relationship as:

$$ASim_{p_i}(p_j) = \frac{|f(p_i) \cap f(p_j)|}{|f(p_i)|} \tag{6.5}$$

We assume that $|f(p_i)|$ is not null, which means that the peer p_i owns at least one file. If the peer does not own any file, the *FP* scheme is used. The value of $ASim_{p_i}(p_j)$ is a numerical score that shows how similar the peer p_j is to the peer p_i. Note that this similarity relationship is not symmetric, i.e., $ASim_{p_i}(p_j)$ may not be equal to $ASim_{p_j}(p_i)$

This scheme will choose only peers that have $ASim_{p_i}(p_j) \geq t_2$. Where t_2 is a threshold.

Let $S_{p_i}^{t_2} = \{p_j, p_j \in P_{f_x} \text{ } and \text{ } ASim_{p_i}(p_j) \geq t_2\}$

Asymmetric Peers' Similarity with File Popularity (ASFP)

We apply the *FP* within the set $S_{p_i}^{t_2}$ of peers most similar to peer p_i. For every file, we compute:

$$Pop_{ASim}(f_k) = \frac{|P_{f_k} \cap P_{f_x} \cap S_{p_i}^{t_2}|}{|P_{f_x} \cap S_{p_i}^{t_2}|} \tag{6.6}$$

Note that if $t_2 = 0$ then $Pop_{ASim}(f_k) = Pop(f_k)$.

This scheme will recommend only files f_k such that $Pop_{ASim}(f_k) \geq t_1$, where t_1 is a threshold. This recommendation list is sorted according to the popularity of the files $Pop_{ASim}(f_k)$ with the files that are most popular at the top of the list. Both t_1 and t_2 are application dependent values.

Asymmetric Peers' Similarity with Weighted File Popularity (ASWFP)

We apply the *Weighted File Popularity* technique within the set $S_{p_i}^{t_2}$ of peers most similar to peer p_i.

In this technique, we weight the files owned by the peers within the set $S_{p_i}^{t_2}$ of peers most similar to peer p_i according to peers' similarity. For every file, we add the similarity value for each peer P_j that owns this file and then we divide by the sum of all peers' similarities for peers that belong to the set $S_{p_i}^{t_2}$.

For every file, we compute:

$$WPop_{ASim}(f_k) = \frac{\sum_{P_j \in S_{p_i}^{t2} \cap P_{f_k}} ASim_{p_i}(p_j)}{\sum_{P_j \in S_{p_i}^{t2}} ASim_{p_i}(p_j)} \tag{6.7}$$

The recommendation list is sorted according to the weighted popularity of the files $WPop_{ASim}(f_k)$ with the files that have a higher weight at the top of the list.

6.5.4 Symmetric Peers' Similarity Based Recommendation (SS)

Here, we define another similarity metric. For every peer p_j in P_{f_x} we define the similarity relationship as:

$$SSim_{p_i}(p_j) = \frac{|f(p_i) \cap f(p_j)|}{|f(p_i) \cup f(p_j)|} \tag{6.8}$$

Note that the denominator $|f(p_i) \cup f(p_j)|$ can not be null.

The value of $SSim_{p_i}(p_j)$ is a numerical score that shows how similar the peer p_j is to the peer p_i. Note that this similarity relationship is symmetric, i.e., $SSim_{p_i}(p_j) = SSim_{p_j}(p_i)$

This scheme will choose only peers that have $SSim_{p_i}(p_j) \geq t_3$. Where t_3 is a threshold.

Let $SS_{p_i}^{t_3} = \{p_j, p_j \in P_{f_x} \ and \ SSim_{p_i}(p_j) \geq t_3\}$.

Symmetric Peers' Similarity with File Popularity (SSFP)

We apply the FP scheme within the set $SS_{p_i}^{t_3}$ of peers most similar to peer p_i. For every file, we compute:

$$Pop_{SSim}(f_k) = \frac{|P_{f_k} \cap P_{f_x} \cap SS_{p_i}^{t_3}|}{|P_{f_x} \cap SS_{p_i}^{t_3}|} \tag{6.9}$$

Note that if $t_3 = 0$ then $Pop_{SSim}(f_k) = Pop(f_k)$.

This scheme will recommend only files f_k such that $Pop_{SSim}(f_k) \geq t_1$, where t_1 is a threshold. This recommendation list is sorted according to the popularity of the files $Pop_{SSim}(f_k)$ with the files that are most popular at the top of the list.

Symmetric Peers' Similarity with Weighted File Popularity (SSWFP)

We apply the *Weighted File Popularity* technique within the set $SS_{p_i}^{t_3}$ of peers most similar to peer p_i. For every file, we compute:

$$WPop_{SSim}(f_k) = \frac{\sum_{P_j \in SS_{p_i}^{t_3} \cap P_{f_k}} SSim_{p_i}(p_j)}{\sum_{P_j \in SS_{p_i}^{t_3}} SSim_{p_i}(p_j)} \tag{6.10}$$

The recommendation list is sorted according to the weighted popularity of the files $WPop_{SSim}(f_k)$ with the files that have a higher weight at the top of the list.

6.5.5 Similarity Metrics and Binary Ratings

Similarity has been used in data mining, pattern recognition, information retrieval, information theory, data clustering and artificial intelligence.

The most used similarity techniques for recommender systems are the Pearson correlation and the Cosine measure [67, 69, 68]. However, a thorough investigation of similarity metrics based on binary ratings reveals the existence of a number of other potentially better similarity metrics.

Adopting an implicit rating approach, implicates a binary value (i.e., 1 if the peer has the file, 0 otherwise) and hence promotes the use of similarity measures for binary data.

Over the last century, similarity metrics were proposed and applied to various fields such as biology, ethnology, taxonomy, image retrieval, geology, chemistry and biometrics. Several similarity metrics have been used in exploratory data analysis [73], and in genetics and molecular biology [74]. [75] provides an extensive survey on these measures.

We adopt the following notations:

Let p_i be the active peer (i.e., the peer requesting the file).

Let p_j be the peer for which we want to compute the similarity with the active peer p_i.

For a particular file f, let C be the observation that the active peer p_i has the file f. And let D be the observation that a peer p_j has this file.

Let a, b, c, and d as follows:

- a: number of times $C = 1$ and $D = 1$. This represents the number of files common to both p_i and p_j.
- b: number of times $C = 1$ and $D = 0$. This represents the number of files owned by p_i but not p_j.

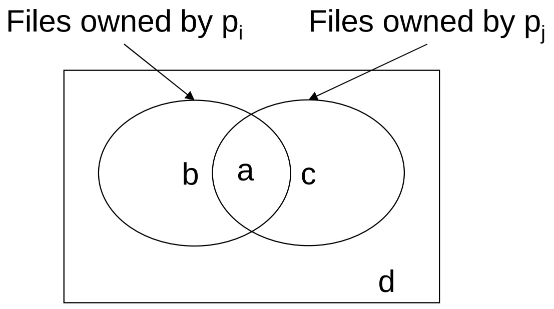


Figure 6.3: a, b, c and d

Similarity metric	Equation	Scheme number
Rogers and Tanimoto [77]	$\frac{a+d}{a+d+2(b+c)}$	4
Simple Matching [78]	$\frac{a+d}{a+b+c+d}$	5
Ochiai II [79]	$\frac{ad}{\sqrt{(a+b)(a+c)(d+b)(d+c)}}$	6
Sokal and Sneath [80]	$\frac{2(a+d)}{2(a+d)+b+c}$	7
Russel and Rao [81]	$\frac{a}{a+b+c+d}$	11

Table 6.1: Similarity Metrics with Negative Co-occurrence

- c: number of times $C = 0$ and $D = 1$. This represents the number of files owned by p_j but not p_i.
- d: number of times $C = 0$ and $D = 0$. This represents the number of files neither owned by p_i nor p_j.

Figure 6.3 depicts a graphical representation of the considered notations.

The similarity metrics may be grouped into two classes according to how they deal with the negative co-occurrence (i.e., d value)[73]. These are the metrics that use the d value in their equation. Table 6.1 shows the similarity metrics that consider the negative co-occurrence, while table 6.2 shows the similarity metrics that do not consider this co-occurrence. In [76], the similarity metrics in the former table are named type 2 similarity metrics, while those in the latter table are named type 1 similarity metrics.

Each similarity metric has its own characteristics and properties. In this chapter, we explore all these similarity metrics by applying them to find the most similar peers in order to make appropriate recommendations. We also investigate both the weighted approach and the non-weighted approach in computing the recommendations. We want to analyze

Similarity metric	Equation	Scheme number
Ochiai I [79]	$\frac{a}{\sqrt{(a+b)(a+c)}}$	1
Jaccard [82]	$\frac{a}{a+b+c}$	2
Anderberg [83]	$\frac{a}{a+2(b+c)}$	8
CzekanowskySorensen-Dice [84]	$\frac{2a}{2a+b+c}$	9
Kulczynski II [85]	$\frac{a}{2}(\frac{1}{a+b}+\frac{1}{a+c})$	10

Table 6.2: Similarity Metrics without Negative Co-occurrence

the impact of the similarity metrics on the recommender system. Furthermore, we study these similarity metrics under different scenarios to evaluate their performance and their ability to make accurate recommendations.

It is important to note that since implicit rating is used for files' recommendations, the use of Pearson correlation is not applicable since the average rating given by a peer p to its files is always 1. In this case, the Pearson correlation measure is not well defined.

6.6 Performance Evaluation

6.6.1 Simulated Schemes

In this chapter, we simulate the following techniques using the non weighted and weighted rating approaches:

- Scheme 3: *Asymmetric Peers' Similarity with File Popularity (ASFP)*. As stated in equation 6.5, this metric uses the following equation: $\frac{a}{a+b}$
- and the following schemes: *Ochiai I* (OcI), *Jaccard* (Jac), *Simple Matching* (SM), *Rogers and Tanimoto* (RT), *Ochiai II* (OcII), *Sokal and Sneath* (SS), *Anderberg* (And), *CzekanowskySorensen-Dice* (CSD), *Kulczynski II* (KII) and *Russel Rao* (RR) presented in tables 6.1 and 6.2.

In the *Cosine measure* technique, the active peer and any other peer are represented by two vectors (generated from the list of files they own) and the similarity between them is measured by computing the cosine of the angle between the two vectors. In Binary rating, the *Cosine measure* and *Ochiai I* are equivalent. We simulate *Ochiai I*.

The *Jaccard* similarity metric is also equivalent to the previously proposed *Symmetric Peers' Similarity*. We simulate the *Jaccard* metric.

The goal from these simulations is to compare the performance of the presented schemes in terms of providing accurate files' recommendations.

6.6.2 Simulation Parameters

The simulation parameters are the following:

- We simulate a system with 1,000 peers and 1,000 files.
- At the beginning of the simulation, each peer has several files and each file has at least one owner.
- Peers are divided into four interest categories (C1: Action, C2: Romance, C3: Drama and, C4: Comedy) and files are also divided into the same four categories.
- The percentage of peers in each category is 25% and the percentage of files in each category is 25%.
- Each peer belongs to one category. Peers prefer to have most of the files from their category and only few files from other categories. We investigate the different schemes using different probabilities termed *Initial Profile* (0.5, 0.6, 0.7, 0.8, 0.9, and 1) leading to 6 scenarios. In the case of 0.9 for example, initially, each peer will have files from the category that she/he prefers with a probability of 0.9 and files from other categories with a probability of 0.1.
- If no file is recommended, file requests follow the real life distribution observed in [63].
- The threshold for each similarity metric is set to 0.1. This means that the similarity of a peer should be greater than 10% for the peer to be considered.
- We simulate 50,000 requests for each simulation.

Our simulations were implemented using the peer-to-peer simulator PeerSim [86]. The simulations were repeated several times for each scheme and for each *Initial Profile* probability. The results presented are the average values. Each scheme has been simulated using the weighted and non weighted rating techniques.

The performance metrics used in the literature are called *Recall* and *Precision*. While *Precision* represents the probability that a recommended item is relevant, *Recall* represents the probability that a relevant item will be recommended. In [87], *Recall* is measured by taking into account the number of hits. A hit is considered when an item from the top N recommended files is in the test set. Usually, $N = 10$ is the number of items returned to users. The greater is N, the greater is the value of *Recall*.

In some research works [88], both the *Recall* and the *average reciprocal hit-rank* ($ARHR$) are computed to assess the performance of the recommender systems. The *Recall* treats all

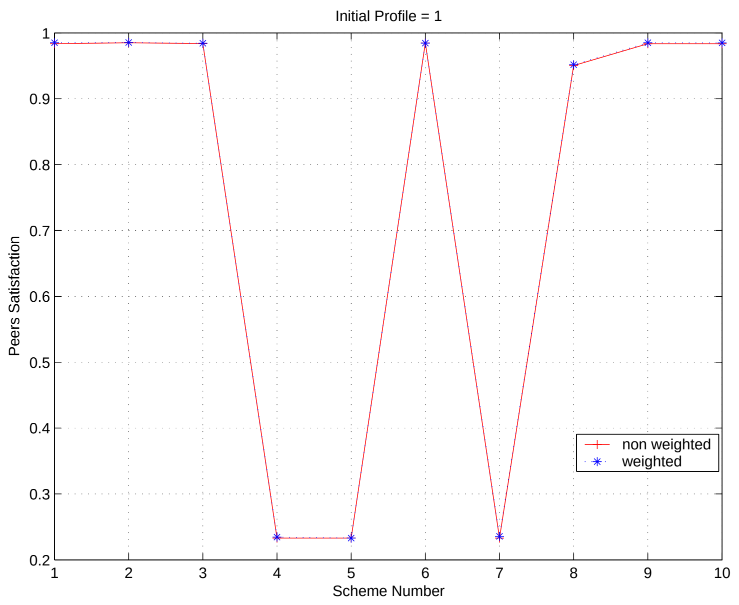


Figure 6.4: Peers Satisfaction (first scenario)

the hits equally regardless of their position in the top N recommended items. In contrast, the *ARHR* takes into account the position of the hits by giving more weight to the hits that occur in the first positions.

In these simulations, we limit the N value to only 1 item. This will make it hard to get a hit. For each scheme, we compute the *Peers Satisfaction*. This value is computed for all peers' categories and it represents the average value of the ratio between the number of recommended files that match peer's category over all the files recommended to the peer. Our goal is to assess accurately the effectiveness of the proposed recommender system and the used similarity metrics.

6.6.3 Simulation Results

We simulated all the schemes under the same conditions and we compared the performance of these schemes. The simulations were conducted in six different scenarios based on the *Initial Profile* probability.

First Scenario

At the beginning of the simulations, peers get files from the category that they prefer with a probability of 1 and no file from other categories is selected. Figure 6.4 depicts the peers' satisfaction for all the schemes with the non weighted and weighted rating approaches. By comparing the results obtained, there is no significant difference between these approaches. Peers' satisfaction almost reaches 100% for the following schemes: Ochiai I (1), Jaccard (2), ASFP (3), Ochiai II (6), CzekanowskySorensen-Dice (9) and Kulczynski II (10). For the Anderberg (8) scheme, peers' satisfaction is relatively lower (95%). However, this value decreases significantly for the following schemes: Rogers and Tanimoto (4), Simple Matching (5), Sokal and Sneath (7). This peers' satisfaction is settling around 23%. In these schemes, 98% of files that have been downloaded by the peers were recommended to them. The bad performance of these schemes can be explained by the fact that their corresponding similarity metrics take into account the negative co-occurrence as explained in table 6.1. However, the fact that two peers do not have a specific file, does not mean that they do not like it. Also, it does not mean that they have the same interests.

Second Scenario

In this scenario, peers get files from the category that they prefer with a probability of 0.9 and only a probability of 0.1 for files from other categories. Figure 6.5 depicts the peers' satisfaction for all the schemes. The results are similar for both the weighted and non weighted approaches. The files that are recommended to peers match the peers' preferences for the following schemes: Ochiai I (1), Jaccard (2), ASFP (3), Ochiai II (6), CzekanowskySorensen-Dice (9) and Kulczynski II (10). Peers' satisfaction reaches 98%. A slightly decrease in peers' satisfaction is noticed for the Anderberg (8) scheme (92%). In this scheme, an average of 84% of files downloaded by peers, were recommended to them.

Decreasing the value of *Initial Profile* probability will necessarily decrease peers' satisfaction. This can be explained by the fact that peers have files from several categories, recommender schemes can not easily identify peer's category and recommend files that match its interests.

Third Scenario

Peers start with files that match their preferences with a probability of 0.8 and a probability of 0.2 for files from other categories. Figure 6.6 shows the results obtained for all the schemes. In this scenario, peers' satisfaction is almost 98% for the following schemes: Ochiai I (1), Jaccard (2), ASFP (3), Ochiai II (6), CzekanowskySorensen-Dice (9) and Kulczynski II (10). Most of the files that are recommended to peers are of interest to

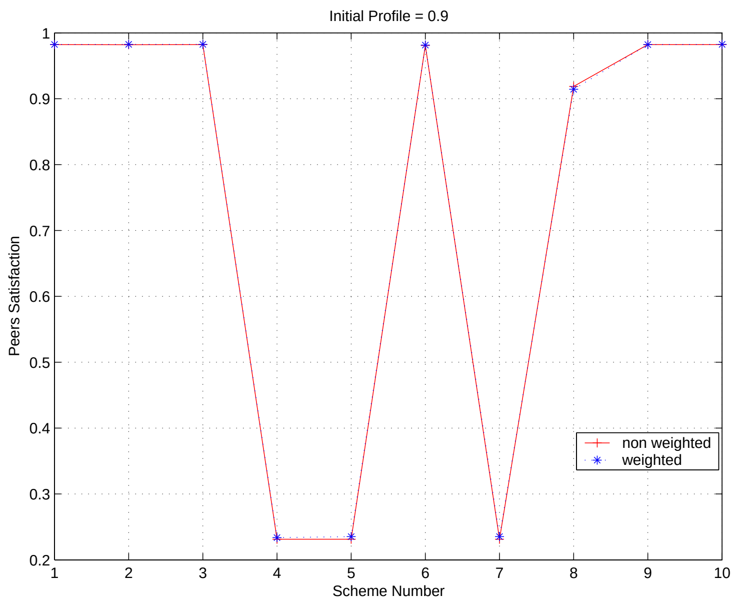


Figure 6.5: Peers Satisfaction (second scenario)

them. The Anderberg scheme (8) is less accurate in making recommendations. In this scheme, peers' satisfaction is 88%. The achieved performance of the other schemes is lower, settling around 23%.

Fourth Scenario

To show the effectiveness of the proposed schemes, we performed another set of simulations. In this scenario, peers start with files that match their category with an *Initial Profile* probability equals to 0.7. Figure 6.7 presents the results. Peers satisfaction is still higher for the following schemes: Ochiai I (1), ASFP (3), Ochiai II (6), CzekanowskySorensen-Dice (9) and Kulczynski II (10) compared to other schemes. Peers' satisfaction is decreased when using the Jaccard scheme (2) to achieve only 91% in the non weighted rating. A significant decrease in the performance of the Anderberg scheme (8) is also noticed in this scenario. The peers' satisfaction is only 82% in the non weighted rating approach which is slightly higher than the weighted rating approach. As mentioned in the previous scenarios, the following schemes: Rogers and Tanimoto (4), Simple Matching (5), Sokal and Sneath (7) do not provide good recommendations to the peers.

A decrease of the *Initial Profile* value to 0.7 will not lead to a significant decrease in *Peers Satisfaction* while using the weighted and non weighted rating approaches.

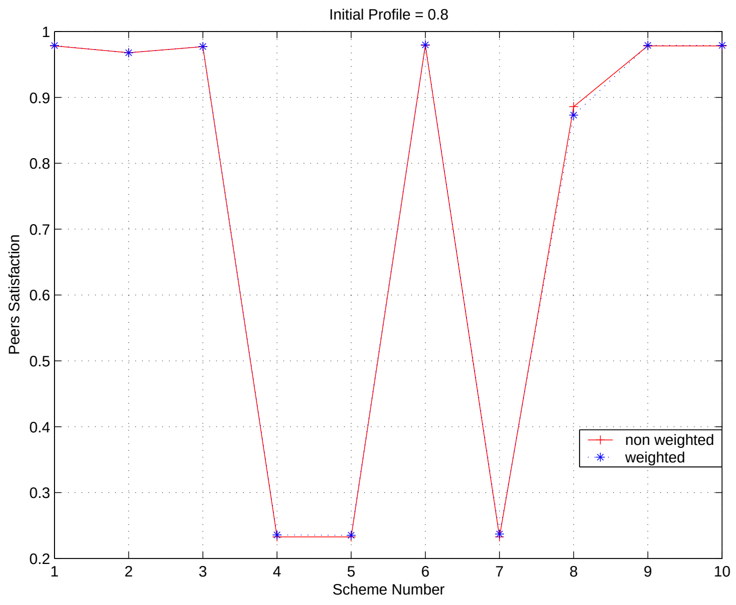


Figure 6.6: Peers Satisfaction (third scenario)

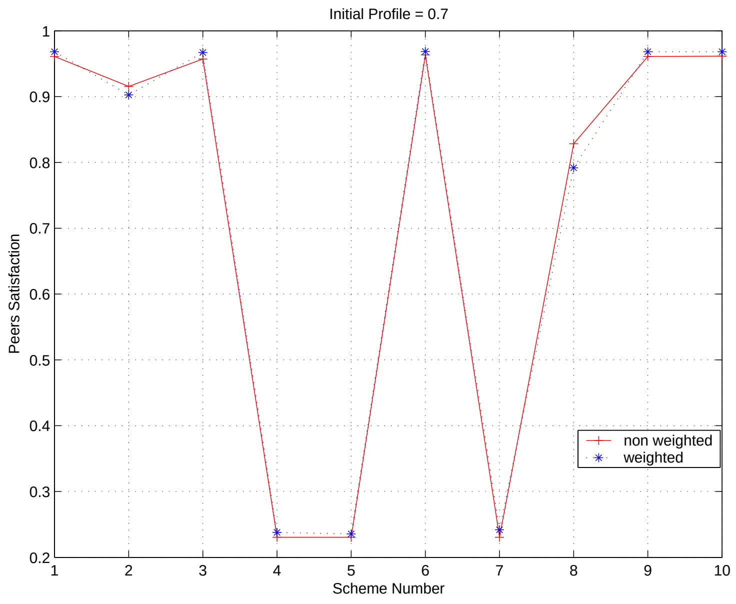


Figure 6.7: Peers Satisfaction (fourth scenario)

Fifth Scenario

Decreasing the value of *Initial Profile* allows to distinguish among the schemes that provide better recommendations to the peers. Figure 6.8 depicts the peers' satisfaction for all the schemes. The results are not as good as in the previous set of simulations. Peers' satisfaction is approximatively 70% by using the non weighted rating for the following schemes: Ochiai I (1), Jaccard (2), ASFP (3), CzekanowskySorensen-Dice (9) and Kulczynski II (10). However, despite of the low value of *Initial Profile* probability, the peers' satisfaction value is still acceptable. The recommender scheme Ochiai II (6) shows a significant increase in peers' satisfaction compared to the previously mentioned schemes. In the Ochiai II (6) scheme, peers satisfaction achieves a high score equals to 79%. The performance of this scheme in this scenario surpasses all other schemes. The Anderberg scheme (8) is less accurate in making recommendations. As discussed in the previous scenario, the following schemes: Rogers and Tanimoto (4), Simple Matching (5), Sokal and Sneath (7), are the worst schemes in making recommendations.

Figure 6.8 shows the good performance of the following recommender schemes using the weighted rating approach: Ochiai I (1), ASFP (3), CzekanowskySorensen-Dice (9) and Kulczynski II (10). These schemes surpass the other schemes in providing appropriate and accurate recommendations. Although the value of *Initial Profile* probability is relatively lower, the use of the weighted rating technique allows these schemes to make a good distinction between files' categories and recommend the appropriate files based on peers' profiles. Peers' satisfaction reaches 87% in contrast to 79% in the non weighted approaches. In general, the weighted rating techniques provide better recommendations' accuracy compared to the non weighted rating techniques.

Sixth Scenario

Figure 6.9 shows the results for the *Initial Profile* probability of 0.5. In this case, all schemes achieved lower results than in the other two scenarios. Again, the weighted approaches outperformed the non weighted ones.

The Russel Rao Scenario

Using the *Russel Rao* metric under the same conditions does not provide any recommendations. This similarity metric is different from the other ones. The Russel and Rao similarity metric presents results different from the other similarity metrics because it excludes the negative co-occurrences in the numerator and includes it in the denominator. To show the performance of this metric, we conducted a new set of simulations using the same parameters as presented in Section 6.6.2 with the threshold $t_2 = 0$. This means that we consider

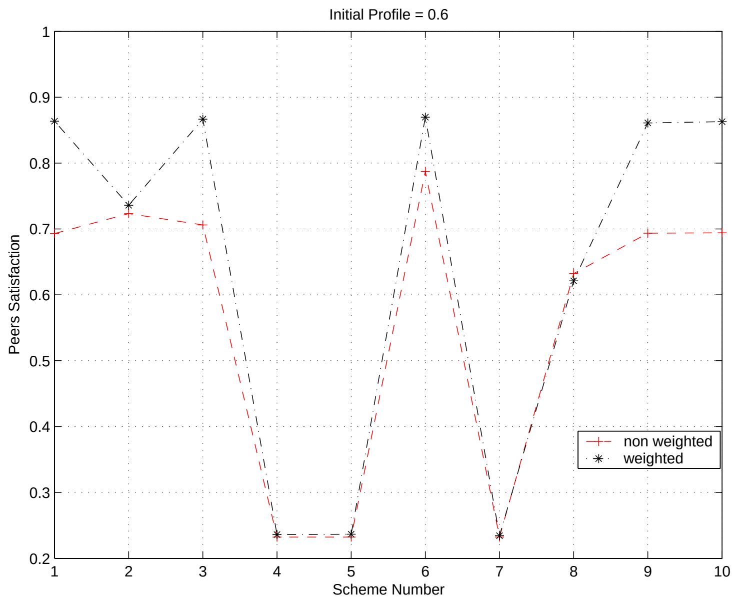


Figure 6.8: Peers Satisfaction (fifth scenario)

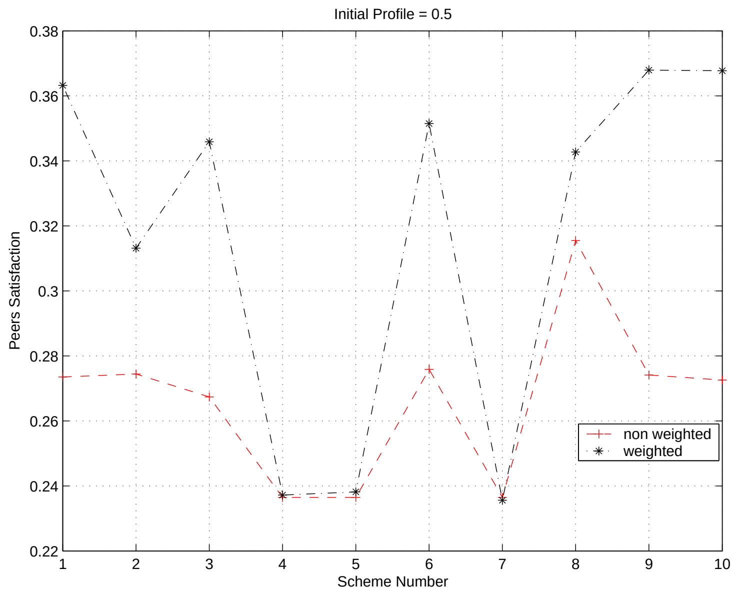


Figure 6.9: Peers Satisfaction (sixth scenario)

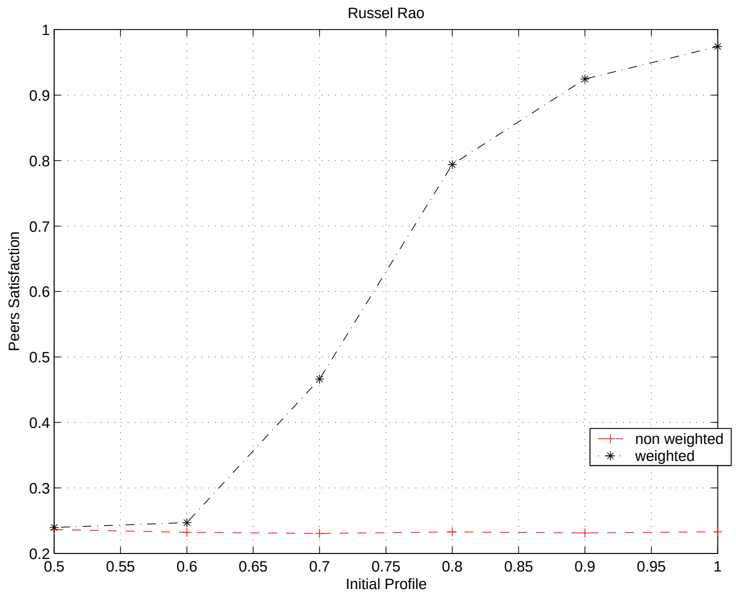


Figure 6.10: Peers Satisfaction of the Russel Rao

all the peers that have the requested file as neighbors. However, the degree of similarity between the active peer and the peers from the neighborhood set can be different from one peer to another. We repeated the simulations 10 times for each of the following *Initial Profile* probability: 0.5, 0.6, 0.7, 0.8, 0.9, 1.

Figure 6.10 presents the obtained results. The X axis represents the *Initial Profile* probability while the Y axis represents the peers' satisfaction using the non weighted and the weighted approaches. In the former approach, peers satisfaction achieves a low score (23%). In the second approach, peers satisfaction increases according to the value of *Initial Profile*. For example, an *Initial Profile* value of 0.8 will lead to 79% of peers satisfaction. The more precise is the *Initial Profile*, the higher is the peers' satisfaction.

Probability	Oc I	Oc I W	Jaccard	Jaccard W	ASFP	ASWFP	RT	RT W	SM	SM W
1	98.34	98.50	98.49	98.53	98.37	98.42	23.30	23.46	23.30	23.30
0.9	98.22	98.22	98.18	98.25	98.21	98.25	23.13	23.38	23.13	23.53
0.8	97.81	97.88	96.77	96.77	97.69	97.72	23.28	23.60	23.28	23.53
0.7	96.08	96.81	91.54	90.25	95.72	96.69	23.05	23.80	23.05	23.56
0.6	69.30	86.36	72.33	73.59	70.61	86.66	23.22	23.60	23.22	23.66
0.5	27.35	36.32	27.45	31.31	26.74	34.58	23.65	23.72	23.65	23.82

Table 6.3: Summary of Peers' Satisfaction

Probability	Oc II	Oc II W	SS	SS W	And	And W	CSD	CSD W	K II	K II W
1	98.46	98.44	23.30	23.58	95.03	95.16	98.34	98.50	98.34	98.48
0.9	98.08	98.13	23.13	23.54	91.87	91.44	98.22	98.21	98.22	98.24
0.8	97.90	97.96	23.28	23.71	88.60	87.31	97.81	97.89	97.81	97.90
0.7	96.33	96.84	23.05	24.19	82.84	79.19	96.09	96.82	96.15	96.81
0.6	78.72	86.97	23.22	23.44	63.22	62.13	69.35	86.08	69.42	86.28
0.5	27.59	35.15	23.65	23.56	31.55	34.27	27.41	36.79	27.26	36.77

Table 6.4: Summary of Peers' Satisfaction (cond.)

6.6.4 Analysis of Recommender Schemes

Similarity metrics aim at quantifying the extent to which objects resemble each other [76]. Similarity metrics make possible to determine if the compared peers can be assigned to the same class or not.

Similarity metrics express the proportion of matches between two peers in a different way. While *Anderberg* and *Rogers and Tanimoto* similarity metrics give twice weight to disagreement, the similarity metrics *Sokal and Sneath* and *CzekanowskySorensen-Dice* give more weight to agreement. Also, the *Sokal and Sneath* metric is similar to the *Simple Matching* metric but gives double weight to matches. Similarly, *CzekanowskySorensen-Dice* metric is similar to the *Jaccard* similarity metric but gives twice the weight to matches. In table 6.2, most of the similarity metrics are increasing functions of a and decreasing functions of b and c. Similarity is higher when the compared peers share more common files and have few distinctive files. While, similarity metrics in table 6.1 take into consideration the intersection, the differences and also the intersection of the complementary sets of the compared peers. For these metrics, the common files and the absence of same files have the same role. In addition to the common files, the absence of same files increases the similarity between the compared peers. However, the *Russel and Rao* metric is more severe in attesting the resemblance between peers, since the absence of same files is added only in the denominator. In this metric, similarity is based only on the common files owned by the compared peers over all the files.

In many applications such as image retrieval, the user is interested in the list of objects most similar to its request (ordered-based approach) rather than the values of the similarity scores (value-based approach) [76]. The similarity scores are not as important as the order of similar objects. However, in the performed simulations, we were interested to know the order of similar peers to the requester peer (i.e., to whom the recommendation is made) in addition to the similarity scores. The similarity of a peer should be greater than 10% to be considered. We considered both the ordered-based and the value-based comparisons in the simulations to assess the performance of the similarity metrics.

Tables 6.3 and 6.4 present a summary of the results of the simulations. By comparing all the schemes using the weighted and non weighted rating techniques in terms of peers' satisfaction, we found that the following schemes: Ochiai I (1), ASFP (3), Ochiai II (6), CzekanowskySorensen-Dice (9) and Kulczynski II (10), provide better performance in terms of recommendations' accuracy. The Jaccard (2) and the Anderberg (8) schemes are less accurate. However, a low performance in providing appropriate recommendations is observed for the following schemes: Rogers and Tanimoto (4), Simple Matching (5), and Sokal and Sneath (7). From the tables 6.3 and 6.4, it is also clear that as the initial distribution of files becomes fuzzy, the schemes are not able to clearly find the exact peers' profile and hence will lead to poor peers' satisfaction. Moreover, the weighted rating technique improves the performance of the schemes since the weight of similarity measures is taken into account while computing peers recommendations compared to the non weighted approach. The low performance of the schemes: Rogers and Tanimoto (4), Simple Matching (5), Sokal and Sneath (7) can be explained by the fact that these schemes take into consideration negative co-occurrence as explained in table 6.1. If two peers do not have a file, a rating of 0 is assigned to this file. Considering that these two peers are similar if there are files that they both do not have, does not make them necessarily similar. It does not mean that they have the same interests. Indeed, the negative co-occurrence does not mean necessarily any resemblance or similarity in our context. However, an acceptable performance of the OchiaiII (6) scheme has been shown in the simulations although this scheme belongs to this category. On the other hand, similarity coefficients with no negative co-occurrence as described in table 6.2, lead to better recommendations' accuracy and higher peers' satisfaction.

Probability	Oc I	Oc I W	Jaccard	Jaccard W	ASFP	ASWFP	RT	RT W	SM	SM W
1	97.61	97.61	97.42	97.42	97.61	97.61	98	98	98	98
0.9	97.72	97.71	97.17	97.16	97.72	97.71	98	98	98	98
0.8	97.75	97.75	96.20	96.22	97.75	97.72	98	98	98	98
0.7	97.75	97.75	93.93	93.91	97.75	97.75	98	98	98	98
0.6	97.72	97.72	91.26	91.19	97.73	97.73	98	98	98	98
0.5	97.68	97.68	89.51	89.39	97.70	97.71	98	98	98	98

Table 6.5: Summary of Percentage of Recommended Files

Probability	Oc II	Oc II W	SS	SS W	And	And W	CSD	CSD W	K II	K II W
1	97.61	97.61	98	98	90.40	90.42	97.61	97.61	97.61	97.61
0.9	97.71	97.71	98	98	84.21	84.11	97.72	97.71	97.72	97.71
0.8	97.71	97.72	98	98	78.89	78.77	97.75	97.75	97.75	97.75
0.7	97.61	97.61	98	98	74.34	74.27	97.75	97.75	97.75	97.75
0.6	97.41	97.42	98	98	70.88	70.70	97.72	97.72	97.72	97.72
0.5	97.14	97.14	98	98	68.88	68.78	97.68	97.68	97.68	97.68

Table 6.6: Summary of Percentage of Recommended Files (cond.)

In the performed simulations, we want to investigate other important issues as the *Cold start* and the *Data sparseness* as discussed in Section 6.2.6. In the ASFP scheme with weighted rating approach and with *Initial Profile* probability equals to 0.8, 97.72% of files downloaded by the peers were recommended. This means that the ASFP scheme does not suffer from the *Cold start* and the *Data sparseness* as explained in Section 6.5 since recommendations are made to the users. Simulations results also show that the recommendations are provided to the peers as soon as the system starts. Although no explicit rating was provided, the scheme is able to make recommendations and more precisely accurate ones. Tables 6.5 and 6.6 present a summary of the results of the simulations. By comparing all the schemes using the weighted and non weighted rating techniques in terms of percentage of recommended files, we found that the following schemes: Ochiai I (1), ASFP (3), Rogers and Tanimoto (4), Simple Matching (5), Sokal and Sneath (7), Ochiai II (6), CzekanowskySorensen-Dice (9) and Kulczynski II (10), provide recommendations with a higher percentage value compared to the Jaccard (2) and the Anderberg (8) schemes.

Another important issue to discuss is the *Trust* problem. Malicious peers that send inauthentic files may also share useless files or provide high ratings for irrelevant files to impact badly files' recommendations. To solve this problem, we suggest to make recommendations based on files shared by the reputable peers. The recommender system can choose files only from trustworthy provider peers that have the requested file. These peers may be selected based on a reputation value (e.g., *Authentic Behavior* in Section 3.4.2). This will reduce the impact of malicious peers on the recommendations.

6.7 Concluding Remarks

In this chapter, we proposed a novel recommender framework for partially decentralized file sharing P2P systems. We investigated similarity metrics, that were proposed in other fields, and adapted them to file sharing P2P systems. We analyzed the impact of each similarity metric on the accuracy of the recommendations. Both weighted and non weighted approaches were investigated. In general, the weighted approaches achieve higher recommendation accuracy. Within the weighted approaches, similarity metrics that do not consider negative co-occurrence lead to better recommendation performance. Files' recommendations will, on one hand, increase users' satisfaction since they will receive recommendations on files that they prefer. On the other hand, they will help peers stay connected to the system to serve other peers in addition to increasing the peers' loyalty to the system.

In the next chapter, a summary of the proposed trust management and recommender frameworks is presented along with future research directions.

Chapter 7

Conclusion and Future Work

In this chapter, we conclude our thesis by highlighting our contributions. We also propose future research directions to extend our work.

7.1 Contributions

The thesis addresses the complementing issues of trust management and recommender systems.

7.1.1 Trust Management Framework

The thesis starts with a survey of reputation-based trust management in P2P systems. We investigated research works proposed to deal with peers' reputation. To understand better the reputation systems, we identified their typical components. A description of the role of each component and their relationships was presented.

The survey of reputation systems revealed the lack of clarity and fuzziness among the concepts of reputation, credibility and contribution. The measurement and/or the computation of these important aspects in some research works was not adequate. For example, the reputation of a peer in terms of sending authentic files has been used to measure its credibility and also for service differentiation which raises fairness issues. Peers belong to users and hence, may exhibit complex human-like behavior. In this context, trust is complex and multi dimensional. Our goal was to manage trust according to specific dimensions and clarify this ambiguity. We collected from each transaction the relevant information about the peers involved (i.e., the peer requesting the file and the peer uploading it) without overwhelming the system with unnecessary overhead. We have captured the peer's behavior according to three dimensions and designed a trust management framework around them. Trust is addressed according to the following dimensions: 1) *Authentic Behavior*, 2) *Credibility Behavior*, and 3) *Contribution Behavior*.

The *Authentic Behavior* of a peer represents the peer's reliability in sending authentic files. To measure the *Authentic Behavior*, two schemes were proposed. In the *Number Based Appreciation* scheme, the number of satisfactory and unsatisfactory uploads is considered, while in the *Size Based Appreciation* scheme, the size of the upload is used. Two selection advisor schemes were proposed: The *Difference Based Algorithm* and the *Inauthentic Detector Algorithm*. Performance evaluation confirmed that the *Authentic Behavior* captures peer's reliability in sending authentic files. Also, the proposed scheme handles the fact that malicious peers could cheat on large files' downloads and not on small ones.

The *Credibility Behavior* of a peer represents the peer's sincerity in providing honest feedbacks. Since peers can send authentic files and send wrong feedbacks, the concept of *Suspicious Transactions* was proposed to compute the credibility of a peer. To handle liar peers and minimize their impact, peers' reputation is computed taking into account the credibility of peers downloading the requested files. The *Malicious Detector Algorithm* was proposed to achieve this goal.

The *Contribution Behavior* of a peer represents the peer's contribution to the system. A peer's contribution is based on its *Availability* in sharing files and its *Involvement* in uploading authentic files. Performance evaluation confirmed that using the *Contribution Behavior* as a guideline for service differentiation provides better service to peers that contribute positively, and reduces the level of service provided to free riders and malicious peers.

We designed the proposed trust management framework for partially decentralized systems. These systems combine the advantages of centralized and completely decentralized systems. We proposed to take advantage of the use of supernodes for trust management. In addition to being used for control message exchange, a supernode is used to collect relevant information and compute peers' reputation. The reputation computation takes into account the three aspects of sending authentic files, sending honest feedbacks, and being involved in terms of sharing and uploading files. However, the concepts of our proposed trust management framework can be adapted to a wide variety of similar problems in centralized systems (e.g., e-Commerce applications) and completely decentralized systems.

7.1.2 Recommender Framework

In addition to the proposed trust management system that we developed to enforce appropriate behavior by rating peers, we proposed a novel recommender system to rate files' content.

The proposed recommender system is based on *user-based collaborative filtering*. We took advantage from the partial search process used in partially decentralized systems to explore the relationships between peers. No additional effort is required from the users since implicit rating is used. The recommender system also does not suffer from the problems of traditional collaborative filtering schemes like the *Cold start*, the *Data sparseness* and the *Popularity effect*. To measure the similarity between peers, we proposed *Files' Popularity Based Recommendation (FP)* and *Asymmetric Peers' Similarity Based Recommendation with File Popularity (ASFP)*. We also investigated other similarity metrics, that were proposed in other fields, and adapted them to our context. We analyzed the effect of similarity metrics on the performance of the recommender system. Both weighted and non weighted approaches were investigated.

The analysis of different similarity metrics gives new insights in the context of recommendations. Our proposed recommender system fits perfectly within our trust management framework.

7.2 Future Work

7.2.1 Simulation Parameters

As a future work, we propose to evaluate different parameters used in our simulations. We propose an in depth study of the followings:

- The distribution of behavior (free riders, malicious, liar) among peers.
- The weight of both *Availability* and *Involvement* in peer's contribution computation.
- The *MinDownload* value: as stated in Section 5.4.2, this value should be carefully chosen not to encourage white-washing and at the same time to allow new comers to benefit from the system.
- The *Initial Profile* value in the recommender framework.
- The fact that users may have multiple interests instead of a single interest defined in terms of file category. This scenario is more expected in practice.
- Other malicious behaviors like collusion, sybil attacks.

In addition, combining the trust management and the recommender frameworks in one unified simulator environment, will help to investigate the role of trust in recommendations' accuracy.

Another issue to consider is moving beyond binary ratings to accept explicit rating in different ranges for both the trust management and the recommender systems. Instead of considering a binary value feedback and a binary rating for recommendations, a rating scale can be used to express better these values. Making appropriate modifications is required in the trust management and the recommender frameworks.

7.2.2 Security Threats

In the proposed trust management framework, we take advantage of the use of supernodes. Supernodes are assumed to be trusted to manipulate reputation data.

Two issues that need to be addressed are the followings:

1. making sure to receive a feedback for each transfer.

2. making sure not to receive a feedback for a non-occurring transfer.

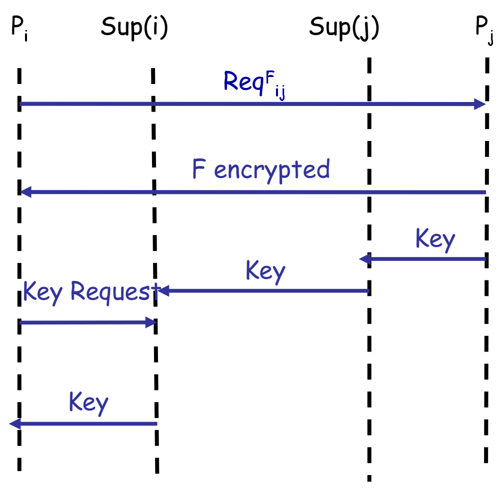
For the first issue, our goal is to make sure that the supernodes involved in a transaction are aware of this transfer since updating the reputation data will be based on the transaction. Let P_i be the peer requesting the file and P_j the peer uploading it. $Sup(i)$ and $Sup(j)$ are their corresponding supernodes. Two approaches are possible.

Figure 7.1 shows the required steps to make sure that the supernodes know about the transfer. Figure 7.1.a presents a first approach. Peer P_i sends a request Req_{ij}^{F} to download file F from peer P_j. P_j sends the file encrypted. After finishing from uploading the file, peer P_j sends the key to its supernode. Then, $Sup(j)$ sends the key to $Sup(i)$. Once the file is downloaded, P_i requests the key from its supernode and gets it. Figure 7.1.b presents a second approach. Once the file is downloaded, P_i requests the key from its supernode. $Sup(i)$ sends the request to $Sup(j)$. After receiving the key, $Sup(i)$ sends it to P_i.

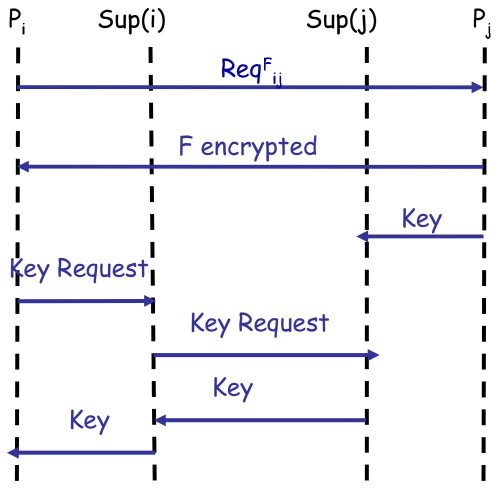
Handling the second issue is much more challenging. In this case, some mechanism has to be set to protect from peers that report a feedback although no file download has occurred. Colluding peers may use the proposed schemes to increase their reputation. As a future work, security threats will be investigated.

7.2.3 Incorporating Trust in Network Virtualization

Network virtualization is the technology that allows the simultaneous operation of multiple logical networks on a single common physical platform. By using this technology, dis-



(a)



(b)

Figure 7.1: (a) First Approach, (b) Second Approach

tributed participants are able to create their own network with application-specific naming, routing, and resource management mechanisms. A virtual network (VN) in the network virtualization environment is managed by one service provider that may require physical resources from many infrastructure providers. A VN is composed of virtual nodes connected by a set of virtual links. Distinct VNs coexist within a common physical network [89].

Recently, network virtualization has received tremendous attention since it is expected to be one of the major paradigms for the future Internet as adopted by numerous international projects on future networks [90]: 4WARD [91], CABO [92], GENI [93], UCLP [94], Clean Slate [95], Trilogy [96], VINI [97], AKARI [98] and PlanetLab [99].

In the different network virtualization projects, trust if addressed, is always addressed from the security and privacy point of view only. Authentication, authorization, access control, ensuring integrity of information and protecting the source of information are used to provide a secure virtual network. However, there are other trust aspects that need to be taken into consideration. For example, we should be able to trust that an infrastructure provider will fulfill its part of a Service Level Agreement.

In virtual network embedding works [100, 101, 102, 103], the location and the CPU of the nodes, and the bandwidth of the links were taking into consideration. In addition to these factors, we propose to take into account the reputation of the infrastructure providers in embedding a VN request. Service providers are dealing with several infrastructure providers for VN mapping requests. The goal is to distinguish among the infrastructure providers based on their past transactions and feedbacks received from service providers.

In the believe that the proposed trust management framework can be successfully applied to other environments, we propose to tailor the proposed trust management framework and incorporate trust in network virtualization environments [104]. We assume that a service provider can assess the quality of service of an infrastructure provider involved in a virtual network in terms of availability of resources, reliability, confidentiality and integrity, and adaptability to network conditions. We define the concept of *Degree of Involvement* of an infrastructure provider in terms of nodes and links. The *Degree of Involvement* represents the contribution of an infrastructure provider in the mapping of the virtual network. The reputation of an infrastructure provider is based on its *Degree of Involvement* and the lifetime of the virtual network.

Incorporating these aspects of trust in the context of network virtualization environments is novel and relevant. This will increase the service providers' satisfaction by reducing the selection of infrastructure providers that do not fulfill their Service Level Agreement.

7.2.4 Trust in Web Services

A Web service (WS) is a self-describing software application that can be advertised, located and used across the Web using a set of standards (WSDL, UDDI and SOAP) [105]. Service consumers are facing the challenge to select the most appropriate Web services among those offering the same functionality. Web services may provide low quality, time-consuming, unreliable and/or expensive services. Recent research papers on Web services emphasize on the importance of integrating trust in the WS environments. There is a need for trust management systems since service consumers are interested to know in advance the expected service that they will get and hence, select the appropriate Web services to fulfill their requests [106].

We adapt our trust management framework to the context of Web services. We propose to address the reputation of a WS in terms of fulfillment of the required service as advertised. Several QoS metrics can be used to capture the behavior of the WS and evaluate the received functionality. The quality of a Web service refers to its performance (e.g., processing time), dependability (e.g., availability, reliability), security (e.g., authentication, confidentiality) and other application-specific metrics (e.g., metrics related to a specific domain) [107].

Feedbacks are collected from users and trust data is managed by a trust management service called *Trust Service*. A user may send a reputation query to the *Trust Service* regarding potential Web services. The *Trust Service* is used as a feedback collector rather than a monitor of the services provided to users. Monitoring increases the burden of the centralized entity by checking on a regular basis the performance of the web services. Moreover, this approach will not reflect the experience as perceived by the users.

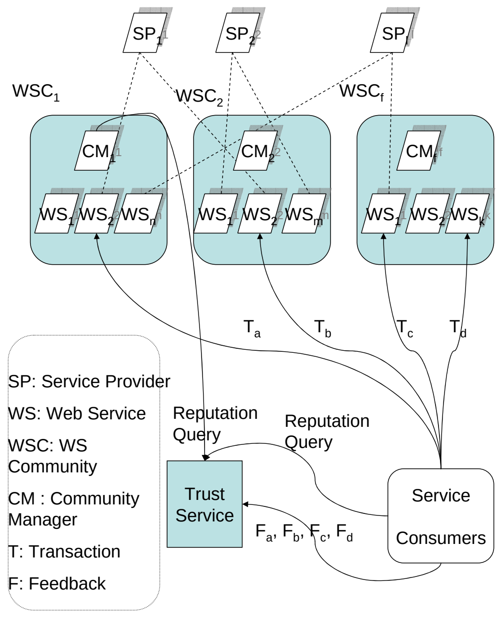


Figure 7.2: Trust Framework for Web Services Communities

Web services communities are virtual clusters that agglomerate Web services with the same functionality [108]. In [108], the authors argued that selecting the best community to deal with is challenging for both the service providers and the users.

Our proposed work can be adapted to rate the communities to facilitate the selection process for both the service providers and service consumers. Integrating trust in these communities may be achieved as follows:

- A service provider may choose to publish its WS within the community that has a high reputation value.
- A service consumer selects a highly reputable community first and then invokes a Web service within this community.
- A WS community manager will use the trust data to decide the integration of a WS in the community or its dismissal from it due to its unsatisfactory performance.

We propose to compute a WS community reputation (WSC) as follows:

$$WSC = \frac{\sum_i n_i \times R_i}{\sum_i n_i} \tag{7.1}$$

Where R_i represents the reputation of a WS_i that belongs to the community WSC and n_i represents the number of times WS_i was invoked.

Figure 7.2 presents the proposed trust framework for Web services communities. A service provider may publish its WS within different communities. After conducting a transaction with a WS community, a service consumer sends a feedback to the *Trust Service*. The feedbacks received are based on the quality of service as perceived by the users and based on the agreement with the WS communities. The collected feedbacks will be used to update the reputation of Web services and WS communities. *Reputation queries* to inquire about an entity's reputation are sent to the *Trust Service*.

As a future work, we plan to investigate the other dimensions of trust in Web services environments.

7.3 Concluding Remarks

Trust management is a challenging task especially in P2P file sharing systems where humans are in control of the peers' behavior.

In this thesis, we have identified three important dimensions of trust in a uniform framework making a step forward into understanding and making use of trust in open environments. We have also addressed the complementary problem of recommendations, one of the challenging problems in collaborative environments. We believe that a unified approach in addressing these problems leads to better system performance and increased users'satisfaction.

Bibliography

[1] L. Mekouar, Y. Iraqi, and R. Boutaba, *Handbook of Peer-to-Peer Networking*, chapter Reputation Management in Peer-to-Peer Systems: Taxonomy and Anatomy, Springer, 2009. 1

[2] P. Garbacki, "Applying the Super-Peer Concept to Existing Peer-to-Peer Networks," Tech. Rep. PDS-2003-010, Delft University of Technology, Netherlands, 2003. 1

[3] L. Mekouar, Y. Iraqi, and R. Boutaba, "Peer-to-Peer Most Wanted: Malicious Peers," *Computer Networks Journal, Special Issue on Management in Peer-to-Peer Systems: Trust, Reputation and Security*, vol. 50, no. 4, pp. 545–562, 2006. 2, 3

[4] L. Mekouar, Y. Iraqi, and R. Boutaba, "A Contribution-Based Service Differentiation Scheme for Peer-to-Peer Systems," *International Journal on Peer-to-Peer Networking and Applications*, vol. 2, no. 2, pp. 146–163, 2009. 3

[5] L. Mekouar, Y. Iraqi, and R. Boutaba, "Impact of Peers' Similarity on Recommendations in P2P Systems," in *IEEE International Conference on Computer and Information Technology*, 2010. 3

[6] A.Oram, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, pp. 21–37, O'Reilly Books, 2001. 6

[7] "Gnutella Protocol Specification v0.4," http://www9.limewire.com/. 6

[8] "Skype," http://www.skype.com/. 6

[9] "KaZaA," http://www.kazaa.com/. 6, 61

[10] "Morpheus," http://www.morphus.com/. 6

[11] "Gnutella2 Specification," http://www.gnutella2.com/. 6, 95

[12] S. Ratnasamy, *A Scalable Content-Addressable Network*, Ph.D. thesis, University of California, Berkeley, 2002. 7

[13] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *SIGCOMM*, 2001, pp. 149–160. 7

[14] K. Aberer and Z. Despotovic, "Managing Trust in a Peer-2-Peer Information System," in *International Conference on Information and Knowledge Management*, 2001, pp. 310–317. 7, 18, 39, 40

[15] G. Suryanarayana and R. N. Taylor, "A Survey of Trust Management and Resource Discovery Technologies in Peer-to-Peer Applications," Tech. Rep., ISR, 2004. 7, 11

[16] F. Cornelli, E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati, "Choosing Reputable Servents in a P2P Network," in *International Conference on World Wide Web*, 2002, pp. 376–386. 7, 18, 39, 40

[17] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks," in *International Conference on World Wide Web*, 2003, pp. 640–651. 7, 18, 39, 42

[18] M. Gupta, P. Judge, and M. Ammar, "A Reputation System for Peer-to-Peer Networks," in *ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2003, pp. 144–152. 7, 13, 15, 18, 21, 36

[19] S. Marti and H. Garcia-Molina, "Limited Reputation Sharing in P2P Systems," in *ACM Conference on Electronic Commerce*, 2004, pp. 91–101. 7, 18, 39, 43

[20] T. G. Papaioannou and G. D. Stamoulis, "Effective Use of Reputation in Peer-to-Peer Environments," in *IEEE/ACM CCGrid: International Symposium on Cluster Computing and the Grid*, 2004, pp. 259–268. 7, 39

[21] Z. Despotovic and K. Aberer, "P2P Reputation Management: Probabilistic Estimation vs. Social Networks," *Computer Networks*, vol. 50, no. 4, pp. 485–500, 2006. 7, 18, 30, 39

[22] M. Gupta, M.H. Ammar, and M. Ahamad, "Trade-offs between Reliability and Overheads in Peer-to-Peer Reputation Tracking," *Computer Networks*, vol. 50, no. 4, pp. 501–522, 2006. 7, 36, 37

[23] Y. Zhang and Y. Fang, "A Fine-Grained Reputation System for Reliable service Selection in Peer-to-Peer Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 8, pp. 1134–1145, 2007. 7, 18, 27, 38

[24] "Oxford Dictionary," http://www.askoxford.com/. 9

[25] M. Deutsh, "The resolution of Conflict:Constructive and Destructive," Tech. Rep., New Haven, Yale University Press, 1973. 9

[26] D. Gambetta, "Can We Trust Trust?," in *Trust: Making and Breaking Cooperative Relations*, chapter 13, pp. 213–237. Published Online, 2000. 9

[27] T. Grandisan and M. Sloman, "A survey of Trust in Internet Applications," vol. 3, no. 4, pp. 2–16, 2000. 10

[28] E. Chang, T. Dillon, and F. K. Hussain, *Trust and Reputation for Service-Oriented Environments*, Wiley, 2006. 10, 18, 19, 29, 34, 44

[29] S. Marsh, *Formalising Trust as a Computational Concept*, Ph.D. thesis, University of Stirling, UK, 1994. 10

[30] A. Abdul-Rahman and S. Hailes, "Supporting Trust in Virtual Communities," in *International Conference on System Sciences*, 2000, pp. 6007–6007. 10

[31] J. Sabater and C. Sierra, "REGRET: Reputation in Gregarious Societies," in *International Conference on Autonomous Agents*, 2001, pp. 194–195. 10

[32] L. Mui, M. Mohtashemi, and A. Halberstadt, "A Computational Model of Trust and Reputation for E-businesses," in *International Conference on System Sciences*, 2002, pp. 2431–2439. 10, 29

[33] S. Marti and H. Garcia-Molina, "Taxonomy of trust: categorizing P2P reputation systems," *Computer Networks Journal, Special Issue on Management in Peer-to-Peer Systems: Trust, Reputation and Security*, vol. 50, no. 4, pp. 472–484, 2006. 12

[34] P. Shah and J.F. Paris, "Incorporating Trust in the BitTorrent Protocol," in *International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, 2007, pp. 586–593. 13, 45

[35] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation Systems," *ACM Communications*, vol. 43, no. 12, pp. 45–48, 2000. 14, 33

[36] A. Josang, R. Ismail, and C. Boyd, "A Survey of Trust and Reputation Systems for Online Service Provision," *Decision Support Systems*, vol. 43, no. 2, pp. 618–644, 2007. 16, 26, 29, 48, 71

[37] "ebay," http://www.ebay.com/. 18, 27, 34

[38] A. Abdul-Rahman and S. Hailes, "A distributed trust model," in *Workshop on New security paradigms*, 1997, pp. 48–60. 18, 39

[39] S. Lee, R. Sherwood, and B. Bhattacharjee, "Cooperative Peer Groups in NICE," in *IEEE Conference on Computer Communications*, 2003, pp. 1272–1282. 18

[40] W. T. Teacy, Jigar Patel, Nicholas R. Jennings, and Michael Luck, "TRAVOS: Trust and Reputation in the Context of Inaccurate Information Sources," *Autonomous Agents and Multi-Agent Systems*, vol. 12, no. 2, pp. 183–198, 2006. 18

[41] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," in *ACM conference on Computer and communications security*, 2002, pp. 207–216. 18, 20, 41

[42] J. Sabater and C. Sierra, "Reputation and Social Network Analysis in Multi-agent Systems," in *International Conference on Autonomous Agents and Multi-agent Systems*, 2002, pp. 475–482. 18, 30

[43] A.A. Selcuk, E. Uzun, and M.R. Pariente, "A Reputation-based Trust Management System for P2P Networks," in *IEEE International Symposium on Cluster Computing and the Grid*, 2004, pp. 251–258. 18, 44

[44] R. Zhou and K. Hwang, "PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing," *IEEE Transactions on Parallel Distributed Systems*, vol. 18, no. 4, pp. 460–473, 2007. 18, 29

[45] S. Ruohomaa and L. Kutvonen and E. Koutrouli, "Reputation Management Survey," in *International Conference on Availability, Reliability and Security*, 2007, pp. 103–111. 24

[46] L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, 2004. 29

[47] Y. Wang and J. Vassileva, "Trust and Reputation Model in Peer-to-Peer Networks," in *International Conference on Peer-to-Peer Computing*, 2003, pp. 150–150. 29

[48] S. Song, K. Hwang, R. Zhou, and Y. Kwok, "Trusted P2P Transactions with Fuzzy Reputation Aggregation," *IEEE Internet Computing*, vol. 9, no. 6, pp. 24–34, 2005. 30

[49] M. Gupta and M. Ammar, "Service Differentiation in Peer-to-Peer Networks Utilizing Reputations," in *ACM International Workshop on Networked Group Communications*, 2003, pp. 70–82. 32, 37

[50] K. Ranganathan, M. Ripeanu, A. Sarin, and I. Foster, "To Share or not to Share: An Analysis of Incentives to Contribute in File Sharing Environments," in *International Workshop on Economics of Peer-to-Peer Systems*, 2003. 32

[51] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "Robust Incentive Techniques for Peer-to-Peer Networks," in *ACM conference on Electronic commerce*, 2004, pp. 102–111. 32

[52] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica, "Free-riding and whitewashing in Peer-to-Peer systems," in *ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, 2004, pp. 228–236. 32

[53] K. Ranganathan, M. Ripeanu, A. Sarin, and I. Foster, "Incentive Mechanisms for Large Collaborative Resource Sharing," in *International Symposium on Cluster Computing and the Grid*, 2004, pp. 1–8. 32, 33

[54] T. G. Papaioannou and G. D. Stamoulis, "Reputation-based Policies that Provide the Right Incentives in Peer-to-Peer Environments," *Computer Networks Journal: Special Issue on Management in Peer-to-Peer Systems: Trust, Reputation and Security*, pp. 563–578, 2006. 32, 92, 93

[55] M. Yang, Q. Feng, Y. Dai, and Z. Zhang, "A Multi-dimensional Reputation System Combined with Trust and Incentive Mechanisms in P2P File Sharing Systems," in *Distributed Computing Systems Workshops*, 2007, pp. 29–29. 32, 33

[56] R. T. B. Ma, S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau, "Incentive and Service Differentiation in P2P Networks: a Game Theoretic Approach," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 978–991, 2006. 33

[57] C. Dellarocas, "The Digitization of Word of Mouth: Promise and Challenges of Online Feedback Mechanisms," *Management Science*, vol. 49, no. 10, pp. 1407–1424, 2003. 33

[58] S. Jun and M. Ahamad, "Incentives in BitTorrent Induce Free Riding," in *Workshop on Economics of Peer-to-Peer Systems*, 2005, pp. 116–121. 45

[59] R. Thommes and M. J. Coates, "BitTorrent Faireness: Analysis and Improvements," in *Workshop of the Internet Telecommunications and signal Processing*, 2005. 45

[60] C. Herley A. Bharambe and V. Padmanabhan, "Analyzing and Improving a BitTorrent Network's Performance Mechanisms," in *IEEE Conference on Computer Communications*, 2006, pp. 1–12. 45

[61] S. Chhabra, E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati, “A Protocol for Reputation Management in Super-Peer Networks.,” in *DEXA Workshops*, 2004, pp. 979–983. 48

[62] “N. Minar,” http://www.openp2p.com/pub/a/p2p/2002/01/08/p2p-topologies-t2.html/. 51

[63] K. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, “Measurement, Modeling, and analysis of a Peer-to-Peer File Sharing Workload,” in *ACM Symposium on Operating Systems Principles*, 2003, pp. 314–329. 61, 124

[64] R. Axelrod, *The Evolution of Cooperation*, Basic Books, 1984. 83

[65] E. Adar and B.A. Huberman, “Free Riding on Gnutella,” Tech. Rep., HP, 2000. 86, 96, 115

[66] G. Linden, B. Smith, and J. York, “Amazon.com Recommendations: Item-to-Item Collaborative Filtering,” in *IEEE Internet Computing*, 2003, pp. 76–80. 108

[67] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, “Analysis of Recommendation Algorithms for E-commerce,” in *EC*, 2000, pp. 158–167. 108, 109, 110, 111, 112, 114, 121

[68] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, “Item-Based Collaborative Filtering Recommendation Algorithms,” in *International Conference on World Wide Web*, 2001, pp. 285–295. 108, 109, 110, 111, 112, 114, 121

[69] P. Massa and P. avesani, “Trust-aware Collaborative Filtering for Recommender Systems,” in *International Conference on Cooperative Information Systems*, 2004. 108, 109, 110, 111, 112, 121

[70] G. Ruffo, R. Schifanella, and E. Ghiringhello, “A Decentralized Recommendation System based on Self-Organizing Partnerships,” in *IFIP Networking*, 2006, pp. 618–629. 108, 113

[71] J. Wang, J. Pouwelse, R. L. Lagendijk, and M. J. T. Reinders, “Distributive Collaborative Filtering for Peer-to-Peer File Sharing Systems,” in *ACM Symposium on Applied Computing*, 2006, pp. 1026–1030. 108, 111, 113

[72] J. Wang, J.A.Pouwelse, J.E. Fokker, A.P. de Vries, and M.J.T. Reinders, “Personalization on a Peer-to-Peer Television System,” *Multimedia Tools and Applications*, vol. 36, pp. 89–113, 2008. 108, 113

[73] F. Lourenço, V. Lobo, and F. Baçao, “Binary-based Similarity Measures for Categorical Data and their Application in Self-Organizing Maps,” in *JOCLAD*, 2004. 121, 122

[74] J.M. Duarte, J.B. dos Santos, and L.C. Melo, “Comparison of similarity coefficients based on RAPD markers in the common bean,” *Genetics and Molecular Biology*, vol. 22, no. 3, pp. 427–432, 1999. 121

[75] S. Choi, S. Cha, and C.C. Tappert, “A Survey of Binary Similarity and Distance Measures,” *Journal of Systemics, Cybernetics and Informatics*, vol. 8, no. 1, pp. 43–48, 2010. 121

[76] M.J Lesot, M. Rifqi, and H.Benhadda, “Similarity Measures for Binary Numerical Data: a Survey,” *International Journal on Knowledge Engineering and Soft Data Paradigms*, vol. 1, no. 1, pp. 63–84, 2009. 122, 133

[77] J.S. Rogers and T.T. Tanimoto, “A Computer Program for Classifying Plants,” *Science*, vol. 132, pp. 1115–1118, 1960. 122

[78] R.R. Sokal and C.D. Michener, “A statistical Method for Evaluating Systematic Relationships,” *Bulletin of the Society of University of Kansas*, vol. 38, pp. 1409–1438, 1958. 122

[79] A. Ochiai, “Zoogeographic Studies on the Soleoid Fishes found in Japan and its Neighbouring Regions,” *Bulletin of the Japanese Society for Fish Science*, vol. 22, pp. 526–530, 1957. 122, 123

[80] R.R. Sokal and P.H. Sneath, “Principles of Numeric Taxonomy,” *W.H. Freeman*, 1963. 122

[81] P.F. Russel and T.R. Rao, “On Habitat and Association of Species of Anopheline larvae in South-Eastern Madras,” vol. 3, pp. 153–178, 1940. 122

[82] P. Jaccard, “Etude Comparative de la Distribuition Florale dans une Portion des Alpes et de Jura,” *Bulletin de la Societe Voudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901. 123

[83] M. Anderberg, “Cluster Analysis for Applications,” *Academic Press*, 1973. 123

[84] L.R. Dice, “Measures of the Amount of Ecological Association between Species,” *Ecology*, vol. 26, pp. 297–302, 1945. 123

[85] S. Kulczynski, “Classe des Sciences Mathmatiques et Naturelles,” *Bulletin International de l’Acadamie Polonaise des Sciences et des Lettres Srie B (Sciences Naturelles)*, vol. Supplement II, pp. 57–203, 1927. 123

[86] "PeerSim," http://peersim.sourceforge.net/. 124

[87] G. Karypis, "Evaluation of Item-Based Top-N Recommendation Algorithms," in *International Conference on Information and knowledge Management*, 2001, pp. 247–254. 124

[88] M. Deshpande and G. Karypis, "Item-based top-N recommendation algorithms," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 143–177, 2004. 124

[89] M. Chowdhury and R. Boutaba, "Network Virtualization: State of the Art and Research Challenges," *IEEE Communications Magazine*, vol. 47, no. 7, pp. 20–26, 2009. 141

[90] "Future-internet," http://www.future-internet.eu/activities/fp7-projects.html/. 141

[91] "4ward," http://www.4ward-project.eu/. 141

[92] "Cabo," www.cs.princeton.edu/jrex/virtual.html/. 141

[93] "Geni," http://www.geni.net/. 141

[94] "Uclp," http://www.uclp.ca/. 141

[95] "Cleanslate," http://cleanslate.stanford.edu/. 141

[96] "Trilogy," http://www.trilogy-project.org/. 141

[97] "Vini," http://www.vini-veritas.net/. 141

[98] "Akari," http://akari-project.nict.go.jp/. 141

[99] "PlanetLab: An Open Platform for Developing, Deploying, and Accessing Planetary-scale Services," http://www.planet-lab.org/. 141

[100] Y. Zhu and M. Ammar, "Algorithms for Assigning Substrate Network Resources to Virtual Network Components," in *IEEE Conference on Computer Communications*, 2006, pp. 1–12. 141

[101] J. Lu and J. Turner, "Efficient Mapping of Virtual Networks onto a Shared Substrate," Tech. Rep., Washington University, USA, 2006. 141

[102] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking Virtual Network Embedding: Substrate support for Path Splitting and Migration," *ACM SIGCOMM Computer Communications Review*, vol. 38, no. 2, pp. 17–29, 2008. 141

[103] M. Chowdhury, M.R. Rahman, and R. Boutaba, "Virtual Network Embedding with Coordinated Node and Link Mapping," in *IEEE Conference on Computer Communications*, 2009, pp. 783–791. 141

[104] L. Mekouar, Y. Iraqi, and R. Boutaba, "Incorporating Trust in Network Virtualization," in *IEEE International Symposium on Trust, Security and Privacy for Emerging Applications*, 2010. 142

[105] Z. Malik and A. Bouguettaya, "RATEWeb: Reputation Assessment for Trust Establishment among Web Services," *VLDB Journal*, vol. 18, no. 4, pp. 885–911, 2009. 142

[106] N. Limam and R. Boutaba, "Assessing Software Service Quality and Trustworthiness at Selection Time," *IEEE Transactions on Software Engineering*, vol. 99, no. PrePrints, 2010. 142

[107] Y. Wang and J. Vassileva, "Toward Trust and Reputation Based Web Service Selection: A Survey," *International Transactions on Systems Science and Applications Journal, Special Issue on New tendencies on Web Services and Multi-agent Systems*, vol. 3, no. 2, pp. 118–132, 2007. 142

[108] H. Yahyaoui, Z. Maamar, J. Bentahar, N. Sahli, S. Elnaffar, and P. Thiran, "On the Reputation of Communities of Web Services," in *International Conference on New Technologies in Distributed Systems*, 2008, pp. 1–8. 143