A Taxonomy of Reputation Systems on Decentralized Peer-to-Peer Networks

by

Wei Zeng

A thesis submitted to

the Faculty of Graduate Studies and Research

in partial fulfillment of

the requirements for the degree of

Master of Computer Science

Ottawa-Carleton Institute for Computer Science

School of Computer Science

Carleton University

Ottawa, Ontario

February 18, 2005

Abstract

Recently, the idea of reputation systems has been widely accepted among users who require assistance in measuring the trustworthiness of information providers across the peer-to-peer networks. In this thesis, we present a state-of-the-art taxonomy of reputation systems on the decentralized peer-to-peer networks. We have analyzed 15 different systems and have sorted them into a list of seven basic dimensions. These dimensions are then used to establish a taxonomy under which the systems analyzed are classified.

We develop a reputation system, Reptella, which expands an existing Gnutella peer implementation to demonstrate the advantage of our taxonomy. Moreover, based on the development of Reptella system, an analysis on how to incorporate reputation systems into existing peer-to-peer networks is carried out in terms of system design and system implementation. Such an analysis has important significance in that it relieves developers from the laborious job involved in developing a peer-to-peer reputation system from scratch.

Acknowledgements

My journey during graduate school was challenging and educational. I would like to acknowledge the efforts of the individuals who provided me with valuable guidance, encouragement and tools during all or a part of the process.

Most beneficial to my master research was the vision, direction and significant feedback from my advisor, Dr. Michael Weiss. He gave me support throughout my research in many ways. I just cannot find a way to thank him.

I am also thankful to my fellow graduate students who gave me selfless help in my study and research.

I received indispensable encouragement from my parents, my brother and sisters, many times despite the long distances that separated us. My dream is their dreams. I hope they can share my happiness.

The final word of thanks goes to my husband Dalong Xu. His love, understanding, moral support, care, and very existence are my keystone. I can honestly say that without him, I would not have finished this work.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Overview

The last half of the twentieth century has witnessed a revolution in information technology (IT). The Internet provides standards for worldwide connectivity, and has created vast new opportunities for people to interact with strangers. This is so in application areas such as e-commerce, knowledge sharing, and even game playing. However, due to this anonymity of Internet, there is a rapidly increasing difficulty for people to identify trustworthy parties or correspondents with whom he/she could interact with. Reputation systems are emerging as a popular approach that respects anonymity and builds trust on Internet scale.

A reputation system gathers, distributes, and aggregates feedback about participants' past behaviors, to give a "shadow of the future" for each interaction. The "shadow of the future" is a notion introduced by political scientist Robert Axelrod [1] to indicate that past experience with remote transaction partners can be projected into the future, giving a measure of their trustworthiness. Figure 1-1 shows a general model of reputation management, which is derived from V.Shmatikov's reputation model [47]. The Reputation module is to collect reputation information of the users based on their past behaviors. The Trust Model module aims to aggregate reputation information into predictions about the users' future behavior. Finally, the Interaction module analyzes the performance of those predictions to make a decision whether to interact

or not. The results of the interactions are then fed back into the reputation management module for the future use.

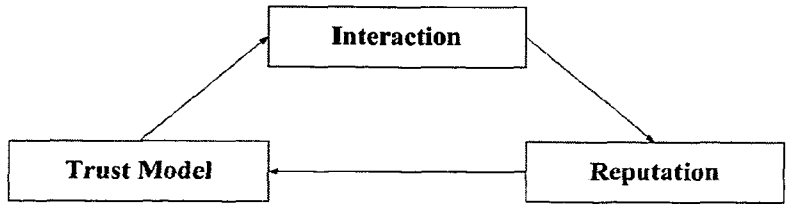

Figure 1-1. Reputation management

Reputation systems help in fostering trust amongst strangers and deter participation by those who are dishonest. Such systems have existed long before the Internet came into existence. Credit and employment agencies, for example, are indeed reputation-based systems. An individual with a good credit history (reputation) is able to receive credits or loans. In the era of Internet, reputation mechanisms have initially attracted attention as a mechanism for building trust and fostering cooperation in Electric Commerce, such as eBay (see section 2.2.4.1), Amazon (see section 2.2.4.1), Epinions, where more traditional quality assurance mechanisms (commercial law, government regulation, etc.) do not work as well.

Most of those commercial websites are based on a centralized architecture. That is, system information is solicited, and stored in a single repository, controlled by a central entity. In a centralized system, it is easier to monitor users' activities, which makes the storage and retrieval of reputation information easier. Generally, the

reputation for each user is published publicly to be visible by all system users. Empirical results show these centralized systems do encourage transactions between users. However, such centralized structure may incur a single point of failure, which means the failure of central entity may paralyze the whole system. Recognizing this issue, Peer-to-Peer (P2P) networks have emerged to provide an alternative to the centralized paradigm.

P2P networks have become a very popular medium for content distribution in the past few years. In such networks, computer resources and services are exchanged directly between systems. Beginning with the popular Napster [27] phenomenon in the late 1990s, the popularity of P2P has dramatically increased the volume of data transferred between Internet users. Recent studies concluded that the file sharing activity on P2P networks accounts for up to 60% of the Internet traffic on any given service provider network [26]. The earliest P2P networks such as Napster, are centralized (see section 2.1), still suffer from a single point of failure. For this reason, in the following discussion, whenever we refer to P2P networks, we mean decentralized (pure) P2P networks (see section 2.1), which do not rely on single central authorities.

1.2 Motivation

Many factors have fostered the recent explosive growth of P2P networks, for example, low cost attributed to the fact that P2P networks distribute resource to

different sites rather than a central server, and high availability of resources attributed to the fact that the same resource may be available at many different nodes simultaneously. The design objectives of decentralization and availability have been driving P2P systems to become a major paradigm in the era of distributed computing.

However, the open nature (that is, the fact that they allow anyone on the network to share any type and number of resources) of these P2P networks leads to a complete lack of accountability for the resource shared in these networks, opening the door to abuses of these networks by anonymous malicious peers, and the presence of these malicious peers can lead to the network becoming inundated with viruses, spam, and other inauthentic files. It has been suggested that the future development of P2P systems will depend largely on the availability of novel methods for ensuring that peers obtain reliable information on the quality of resources they are receiving [14]. In this context, attempting to identify malicious peers that provide inauthentic files is superior to attempting to identify inauthentic files themselves, since malicious peers can easily generate a virtually unlimited number of inauthentic files if they are not banned from participating in the network. Recognizing this problem, there must be a mechanism by which the network is able to distinguish trusted peers from the malicious peers. In other words, it is very essential to have a system by which peers are able to trust each other.

Reputation systems have been proposed by many researchers as effective ways of facilitating trust management in online systems. Recently, P2P networks have become one of the most compelling application areas for reputation systems. However, the decentralized nature of P2P networks makes it much more challenging to establish reputation systems in such networks than in centralized networks.

In the recent past, there has been some work on establishing reputation systems in P2P networks, as discussed in chapter 3. In those systems, there is no central server to monitor the agents' activities. Agents receive reputation information from a variety of sources, including direct experience, feedback from third parties, and implicitly extracted information.

However, to date, there are neither widely used P2P reputation systems, nor any universally accepted guidelines for designing effective P2P reputation systems. Most papers on such reputation systems provide an intuitive approach to reputation which appeals to common experiences without clarifying whether their use of reputation is similar or different from those used by others. It is therefore essential that we explore the existing P2P reputation systems, identify the existing different design possibilities and investigate their virtue and drawbacks. Such research job will help a newcomer learn about the current development level of P2P reputation systems, and as a starting point for an advanced study in this area, or benefit a developer in building a P2P reputation system satisfying the system's pre-defined requirements.

1.3 Proposal

In this paper, we carry out a comprehensive and systematic study of reputation systems on P2P networks. Through a survey of existing works on P2P reputation systems, we provide a taxonomy of decentralized P2P reputation systems.

In this taxonomy, based on the analysis regarding the functional aspect of P2P reputation systems, we identify a list of seven vital dimensions as well as the proposed variables for each dimension. Those dimensions are grouped into two blocks: dimensions regarding reputation information generation and maintenance, and dimensions about the reputation metrics. In decentralized P2P reputation systems, those dimensions will classify those reputation systems into different categories. We will explain the methods used by each dimension and describe their advantages and disadvantages.

In addition to this taxonomy, we present a P2P file-sharing reputation system in our case study, and take advantage of our taxonomy to design this system. Furthermore, this system identifies the complete reputation management process, and gives the details for each aspect of reputation management. Finally, based on such system, we refine the principles about how to incorporate reputation systems into existing P2P networks.

The paper benefits two groups: academics studying reputation systems in E-commerce and implementers considering applying reputation systems in their networks. For academics, the taxonomy and case study provide a useful initial framework within which their research can be placed. The framework will undoubtedly be expanded to include future applications of reputation systems. For implementers, the paper provides a means of making choices among the available applications and technologies. An implementer can choose a goal, and pick an implementation technique that supports the goal. To summarize, the main purpose of this paper is to provide a starting point for researchers or implementer to construct their own P2P reputation systems.

1.4 Thesis Contribution

The direction of the research undertaken in this thesis is predicted by a belief that there are some common patterns among those reputation systems related to the decentralized P2P networks. Based on a taxonomy of decentralized P2P reputation systems, and the corresponding case study which shows the application of this taxonomy in developing a reputation system, we have gained an insight about creating, distributing, and computing reputation information in decentralized P2P networks. The thesis contributions are listed below:

1. This thesis presents a taxonomy of reputation systems for decentralized peer-to-peer networks. This taxonomy is based on seven basic dimensions categorized into two groups: reputation information generation and maintenance, and reputation metrics. Additionally, a cross-dimensional analysis is given at last to enhance the design for new decentralized P2P reputation systems. This taxonomy not only classifies the currently proposed techniques for developing reputation systems, but also identifies the core elements for constructing a P2P reputation system. In fact, it is a guide to construct a reputation system satisfying the pre-defined system requirements.

2. We develop a P2P file-sharing reputation system: Reptella, in our case study, and provide detailed information about how to design and implement a Gnutella-based reputation system. This system demonstrates the application of the above

taxonomy in the design of P2P reputation systems, that is, matching the system requirements with the available techniques provided by the taxonomy.

3. In the development of Reptella system, we identify the general reputation management activities, and illustrate the whole reputation management flow. We design and implement the reputation management process, and discuss the important design issues which might be not presented in our taxonomy, but were important for reputation management, e.g. trust chain.

4. We extend the original Gnutella messages with a RepuQuery message and Recommendation message. We describe in detail those message formats and implementation. At the same time, we design a set of reputation-based system protocols for resource rediscovery and resource download. We also implement those protocols in our reputation system.

5. This thesis also formulates rules that guide the process of integrating a reputation system into an existing P2P system. This work instructs how to reuse an existing P2P system in order to avoid developing such reputation system from scratch.

1.5 Thesis Outline

To meet the objectives outlined above, this thesis has been divided into five chapters. Chapter 2 covers the background information. It gives a broad range of description for all the relevant topics, including peer-to-peer networks, Gnutella network, reputation systems. The chapter is intended to familiarize the reader with concepts that are explored and developed in subsequent sections.

Chapter 3 presents a taxonomy of reputation systems for decentralized peer-to-peer networks. We have analyzed 15 different systems based on seven basic dimensions which are grouped into two basic blocks. These dimensions are then used to establish a taxonomy under which the systems analyzed are classified. Finally, we conclude this chapter with a cross-dimensional analysis with the aim of providing a starting point for researchers to develop new systems according to their requirements.

For our case study, we design and implement a Gnutella-based P2P reputation system, Reptella, in Chapter 4. This involves the development of the system concepts and ideas, for example, design considerations and solutions, system actors and scenarios, system reputation management, system architecture, protocol description, component-based system interactions, and the discussion regarding the way to incorporate reputation systems into existing P2P networks, etc. Chapter 5 summarizes the taxonomy and the Reptella system, with suggestions for future work and possible extensions for Reptella system.

Chapter 2 Background Literature

2.1 Peer-to-Peer Networks

Currently, the Internet users are rising rapidly. By year-end 2002, there are around 490 million Internet users. At the same time, each user in this massive network has a personal computer capable of more than 100 times that of a supercomputer in the early 1990s. However, some research reveals that over 95% of today's PC power is wasted. Recently, there has been a lot of discussion around peer-to-peer (P2P) networks as a way of tapping into this distributed computing resource.

In this section, we provide a brief review of P2P networks. We compare the different architectures of P2P networks and give a detailed introduction on the P2P network, which will be used as the underlying infrastructure for our P2P reputation system.

P2P is a class of applications that takes advantage of resources, e.g., storage, computing cycles, content, and users available at the edge of the Internet. In general, P2P networks connect individual users with each other, allowing them to share resource, such as files, services, etc. Such networks provide an alternative mechanism of computing to client/server architectures. In addition to the ability to pool together and harness large amounts of resources, the strengths of existing P2P systems [30, 29, 31, 27] include self-organization, load-balancing, cost-effective infrastructure, adaptation and fault tolerance. Because of these desirable qualities,

many research projects have been focused on understanding the issues surrounding these systems and improving their performance.

2.1.1 Architectures in P2P Networks

Based on the method by which search requests are sent into the network, peer-to-peer networks can be categorized as centralized, decentralized and hybrid networks [48]. We are going to introduce them one by one in the following sections.

2.1.1.1 Centralized Networks

Centralized networks are the most familiar form of topology, typically seen as the client/server pattern used by databases, web servers, and other simple distributed systems. Figure 2-1 shows the centralized network topology. All function and information is centralized in one server with many clients connecting directly to the server for sending and receiving information.

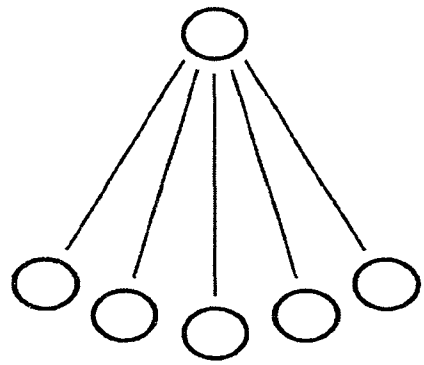

Figure 2-1. Centralized network topology

In centralized networks, the server coordinates and manages the communication between clients. Most Internet services are distributed using the traditional client/server (centralized) architecture (e.g. FTP) to obtain access to a specific resource. In P2P systems, centralized networks make use of a centralized indexing service to connect peers to one another. Figure 2-2 depicts the design of such a network. A central server stores information about the files that exist on each peer in the network. Each peer sends its location information to the central index server upon connecting to the network. For example, if Alice intends to download a resource, she sends a search request to the index server, which returns information about the peer (Bob) sharing the resource. Alice then connects directly to Bob and downloads the file.

This architecture offers very good performance in terms of the response time to search requests. Furthermore, this architecture requires less computational power on the client side. However, such architectures may not scale well, since the central server is a potential bottleneck. Also, due to the existence of a central server, these schemes are not robust to attacks – an attack on the server can bring down the entire network. Napster was a very popular P2P network that uses centralized indexing. Napster collapsed due to litigations over potential copyright infringements.

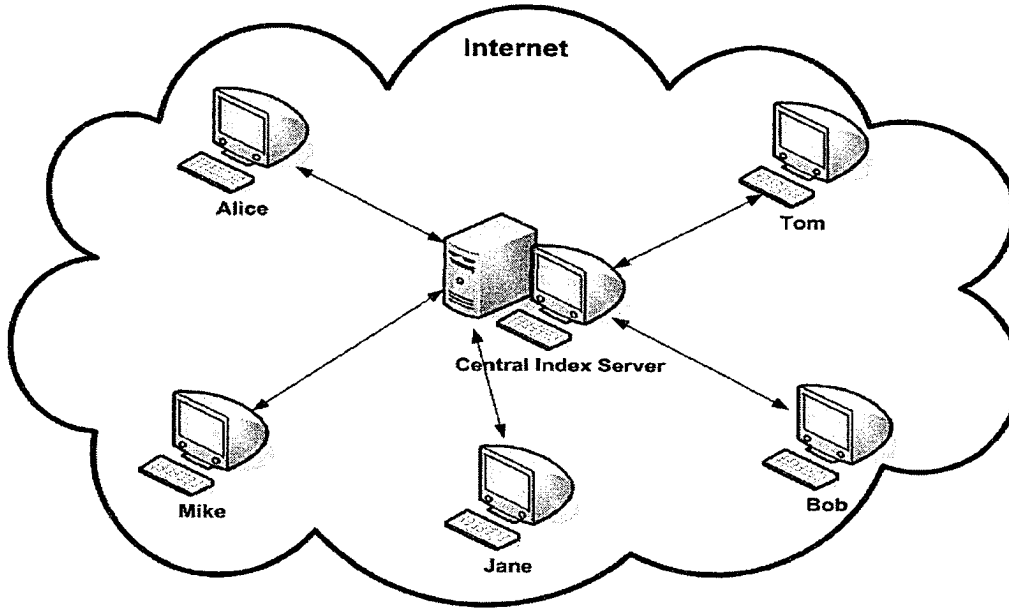

Figure 2-2. Centralized network architecture

2.1.1.2 Decentralized (Pure P2P) Networks

Another topology we consider in P2P networks is a decentralized network architecture, where all peers communicate symmetrically and have equal roles (i.e., both are clients as well as servers). Peers in a decentralized P2P network are therefore also referred to as "servents". Figure 2-3 depicts a decentralized network topology. Decentralized networks are not new; the routing architecture of the Internet itself is largely decentralized, with the Border Gateway Protocol [44] used to coordinate the peering links[1] between various autonomous systems.

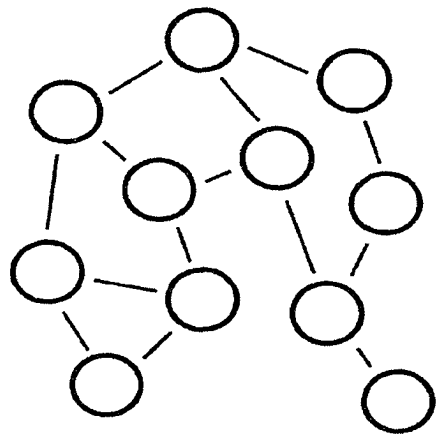

Figure 2-3. Decentralized network topology

Decentralized networks adopt a distributed architecture for searching content shared by peers. Each peer discovers and establishes connections with a variable number of

[1] A peering link is a link between two Internet Service Providers of equal status. Running peering link between two ISPs means both of them agree to forward each other's packets directly across this link instead of using the standard Internet backbone [41][42][43].

servents to exchange content as shown in figure 2-4. A peer discovers new peers in the network using broadcast messages. A search request is broadcast from a source peer to the peers directly connected to the source. Each of these peers in turn broadcasts the request to its neighbors, until the message reaches a peer that possesses the content, or the message has traversed a maximum number of hops. Gnutella is an example of a pure decentralized P2P network system used in practice today, with only a small centralized function[2] to bootstrap a new host.

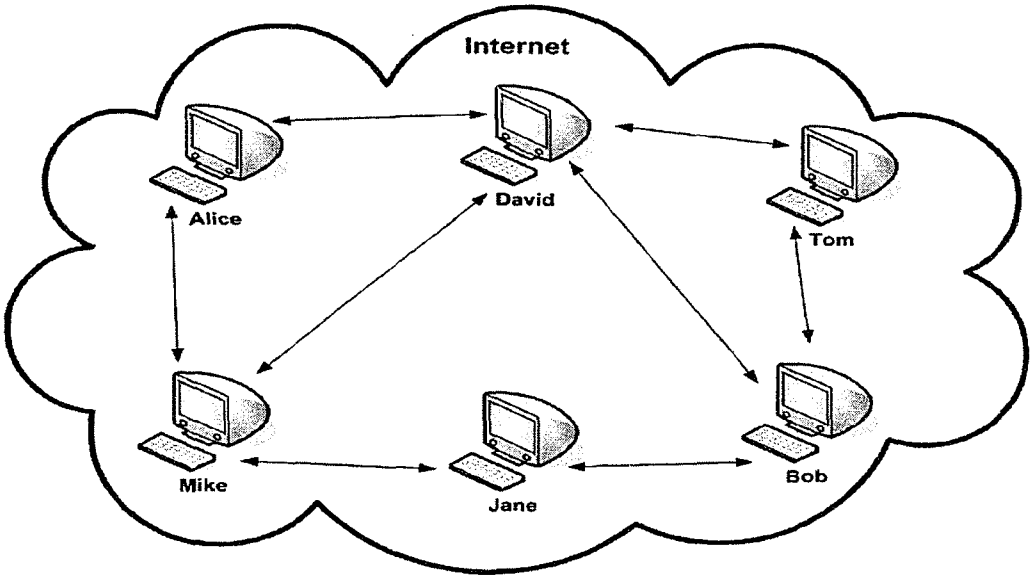

Figure 2-4. Decentralized network architecture

[2] Gnutella relies on a list of pre-existing hosts to bootstrap a newcomer's presence, that is, a newcomer should connect to one of the pre-existing hosts in order to join this system. Also see 2.1.2.1.

2.1.1.3 Hybrid Networks

Real-world distributed systems often have a more complex organization than any one simple topology. They have a hybrid topology. Peers typically play multiple roles in such a system. For example, a node might have a centralized interaction with one part of the system, while being part of a hierarchy in another part. In this section, we introduce two types of hybrid P2P networks, centralized + centralized and centralized + decentralized.

The server in a centralized network is itself often a client of one or more other servers. For example, a web server can respond requests from a web browser by formatting results into HTML for presentation in the web browser, and at the same time the server itself calls upon servers hosting business logic or data for necessary information. This forms a new type of P2P network known as centralized + centralized networks. Figure 2-5 shows the topology of such a network. Web service intermediaries such as Grand Central Networks [45] also create several layers of centralized systems. Centralized systems are often stacked as a way to compose function.

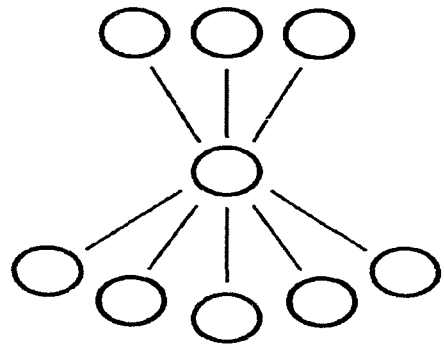

Figure 2-5. Centralized + Centralized network topology

The latest wave of P2P systems exhibits an architecture of centralized networks embedded within decentralized networks. Figure 2-6 is the topology of such a network. In this figure, most peers have a centralized relationship to a super node, forwarding all file queries to this server; much like a Napster client sends queries to the Napster server. But instead of super nodes being standalone servers, they are themselves connected in a Gnutella-like decentralized network, through which queries are propagated. This hybrid topology is realized with hundreds of thousands of peers in systems like the FastTrack file-sharing system used in KaZaa or Morpheus. Internet email also exhibits this kind of hybrid topology. Mail clients have a centralized relationship with a specific mail server, but mail servers themselves share email in a decentralized fashion.

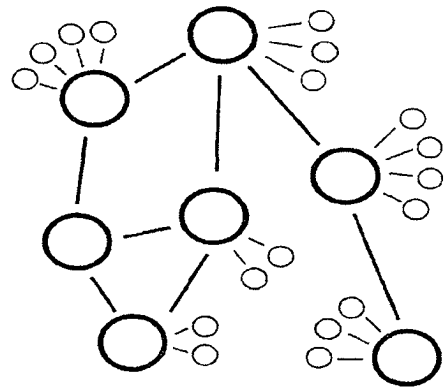

Figure 2-6. Centralized + Decentralized network topology

In hybrid networks, peers are classified into leaf nodes and supernodes, based on their computational power and their bandwidth. Supernodes are special servents responsible for indexing the network content and for routing request and response messages, and leaf nodes only provide content to the network, thus a two-tier hierarchy is established among the servents. Figures 2-7 and 2-8 illustrate the centralized + centralized and centralized + decentralized architectures, respectively. As shown in both figures, a request originating at a leaf node or supernode is processed only by intermediate supernodes. Kazaa is an instance of such a network. However, many of the Gnutella servents have also adopted this hybrid model because it inherits benefits from both centralized and decentralized models.

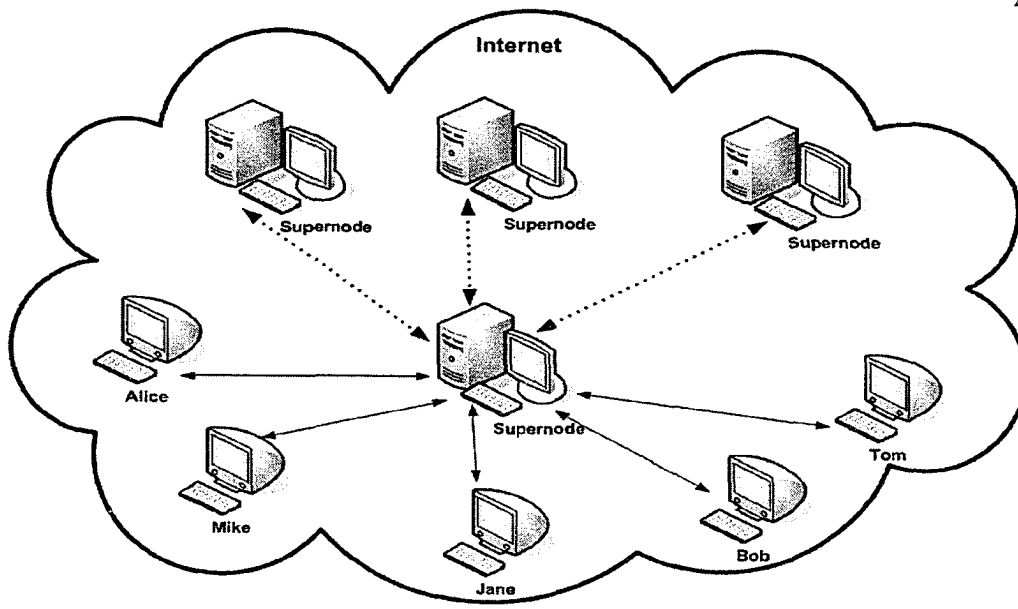

Figure 2-7. Centralized + Centralized network architecture

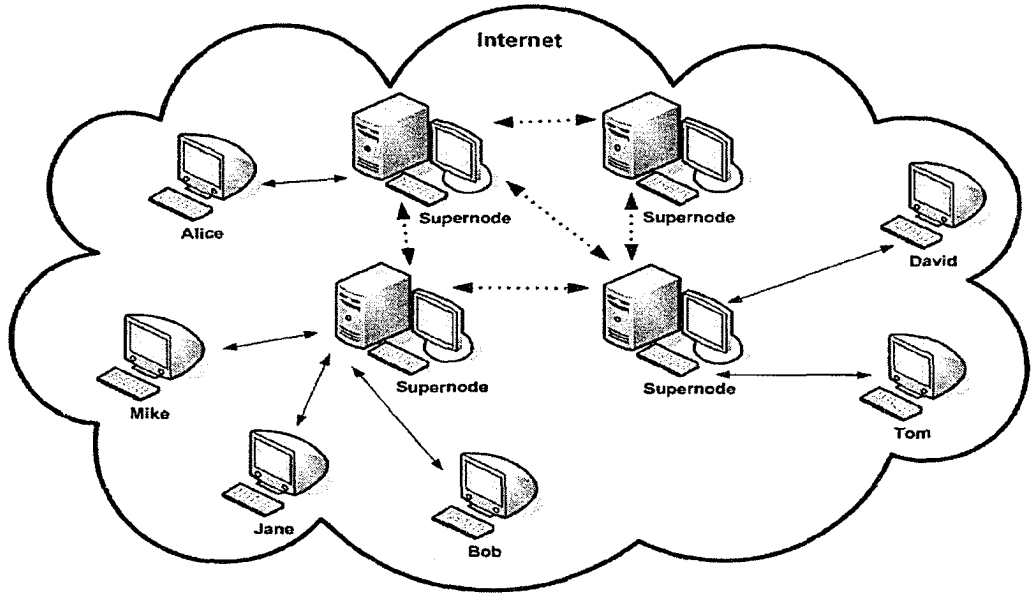

Figure 2-8. Centralized + Decentralized network architecture

2.1.2 Review of the Gnutella Design and Implementation

Since we will develop a reputation system which extends an existing Gnutella implementation in our case study, a detailed description about Gnutella is necessary. Gnutella is a popular P2P system and could be labeled as one of the first truly P2P applications. Initially, it was developed by programmers at NullSoft. It orients itself toward the file-sharing community. NullSoft was acquired by AOL because of its popular MP3 application WinAmp. Because of the litigation problems between Recording Industry Association of America (RIAA) and Napster, AOL pulled the plug on any further formal development of Gnutella. However, that in fact provided a significant incentive for grassroot effort and innovation, as many individuals and organizations have started to spearhead efforts to keep Gnutella alive and make it work better.

Gnutella operates on a set of network protocols, being called by a joint name: Gnutella protocol. A detailed description of those protocols can be found in the Gnutella specification [26]. We will leave out details of the specification and describe only sections relevant to this thesis. The Gnutella protocol is an open, decentralized group membership and search protocol, primarily used for file-sharing. Gnutella protocol has been one of the most popular file-sharing protocols. People who wish to use the Gnutella network will download [30] or develop [32] an application that adheres to the Gnutella protocol. We will first talk about the operation of a Gnutella system.

2.1.2.1 Operation of Gnutella

Since there are no central servers in the Gnutella network, in order to join the system a peer initially connects to one of several known hosts (e.g., those listed on gnutellahosts.com) that are almost always available (generally, such hosts do not themselves provide shared files). Such hosts are known as host caches. A host cache then forwards IP addresses and port numbers of other peers that are currently online. This connection is performed over a TCP session.

Once attached to the network (having one or more open connections with nodes already in the network), nodes send messages to interact with each other. Messages can be either request messages or responses to some previously received request messages. Messages can be broadcast (i.e., sent to all nodes with which the sender has an open TCP connection) or simply back-propagated (i.e., sent over an existing connection in the reverse direction, along the path taken by the corresponding broadcast message). Several features of the Gnutella protocol facilitate this broadcast/back-propagation mechanism. First, each message has a randomly generated unique identifier which uniquely identifies the message on the network. Second, each node keeps a short memory of recently routed messages, used to prevent re-broadcasting and to implement back-propagation. Third, message packets contain fields for time-to-live (TTL) and the number of hops (i.e., peers visited on the path).

2.1.2.2 Description of the Gnutella Protocol

A Gnutella servent connects itself to the network by establishing a connection with another servent that is already connected to the network. To search for a particular file, for example, a servent *p* broadcasts a Query message to every node linked directly to it (see figure 2-9). The fact that the message is broadcast through the P2P network implies that a node not directly connected to *p* may receive this message via intermediaries. Servents that receive the query and have the requested file in their repository, answer with a QueryHit packet that contains a ResultSet plus the IP address and port number of the servent from which the files can be downloaded using the HTTP protocol. Servents can gain a complete vision of the network within their horizon[3] by broadcasting Ping messages. Servents within the horizon reply with a Pong message containing the number and size of the files they share. Finally, to enable communication with servents located behind firewalls, the Gnutella protocol provides Push messages. A Push message behaves more or less like passive communication in traditional protocols such as FTP, inasmuch the requesting servent can not set up connection with the serving servent until the latter initiates the connection for downloading.

[3] Horizon is the set of peers reachable by broadcast, i.e., all peers within a number of hops less than the Time To Live (TTL) of the system.

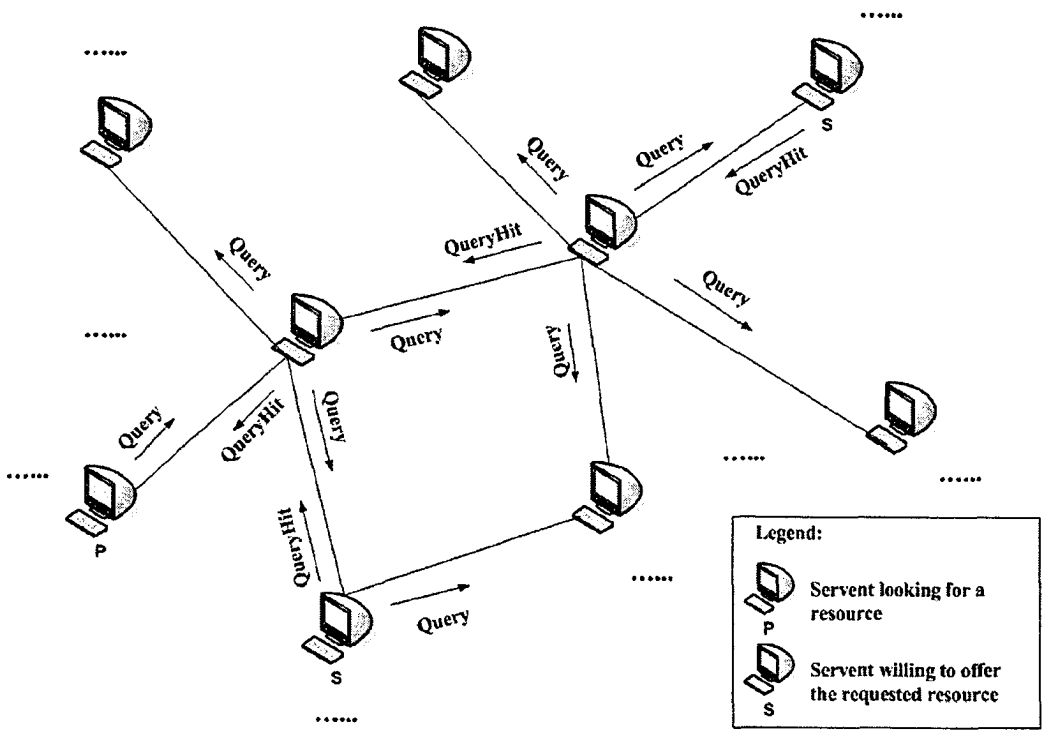

Figure 2-9. Locating Resources in Gnutella P2P Environment

2.1.2.3 Protocol Messages

After being connected successfully, a peer communicates with other servents by sending and receiving Gnutella protocol messages. Peers will create and initiate a broadcast of messages as well as re-broadcasting others (receiving and transmitting to neighbors). The messages allowed in the network are: Ping, Pong ...

Ping - Used to actively discover hosts on the network. A servent receiving a Ping message is expected to respond with one or more Pong messages. Essentially, it is an "are you there?" message directed at a host.

Pong - The response to a Ping. Includes the address of a connected Gnutella servent and information regarding the amount of data it is making available to the network. The meaning of a Pong is "yes, I'm here". The Pong message contains information about the peer such as their IP address and port as well as the number of files shared and the total size of those files.

Query - The primary mechanism for searching a Gnutella network. A servent receiving a Query descriptor will respond with a Query Response Message if a match is found against its local data set. This message can be interpreted as "I am looking for x", and can get forwarded throughout the entire network (at least in theory). Query messages have unique identifiers, but their source is unknown.

QueryHit - These are replies to Query messages, and they include the information necessary to download the file (IP address, port, and other information). QueryHit messages also contain a unique client ID associated with the replying peer. These messages are propagated backwards along the path that the query message originally comes from.

Get/Push - Get messages are used to request a file returned in response to a query from the servent that has the file. The requesting peer connects to the serving peer directly and requests the file. Certain hosts, usually if they are located behind a firewall, are unable to respond directly to requests for files. For this reason the Gnutella protocol includes Push messages. Push messages request the serving client to initiate the connection to the requesting peer and upload the file. However, if both peers are located behind a firewall a connection between the two will be impossible.

Several features of the Gnutella protocol avoid that messages are re-broadcast indefinitely through the network in the presence of cycles in the network. One such feature includes a short memory of messages that have been routed through a peer (thus preventing re-broadcasting). Additionally, messages contain a time-to-live (TTL) field. At each hop (when a message is re-broadcasted) the TTL is decremented. As soon as a peer sees a message with a TTL of zero, the message is dropped (i.e. it is not re-broadcast).

2.2 Reputation Management

Large distributed system such as the Internet has created vast new opportunities to interact with strangers. The interactions can be fun, informative, and even profitable. However, such online interactions involve a lot of uncertainty. Is the information or files from a system participant reliable, will the service provided by a provider satisfy our expectation, or will the participants be cooperative? While some users intentionally seek out negative interactions, how to effectively evaluate the reliability or trustworthiness of system users has become a crucial concern in online communities.

In real life, we handle such problem through the social mechanisms of trust and word-of-mouth, or reputation. When we talk about how much we trust someone, we often consider that person's reputation. We usually are willing to put great trust in someone whom we have personally observed to be highly capable and who has a high level of integrity. In the absence of personal observation, the recommendation of a trusted friend can lead to trust in someone. Hence, in real life, trust is often increased by establishing positive reputations and networks for conveying these reputations. Similarly, we can deal with the complexities and uncertainties of online interactions through reputation management.

Online reputation systems provide a mechanism which minimizes the unpredictability and risk in online transactions through reputation management.

Reputation systems collect, distribute and aggregate feedback about system participants' past behavior. They are looked to as possible solutions to those problems in distributed systems, motivating co-operation, providing recommendations and as a distributed authorization mechanism [25]. We will describe reputation systems in more details in section 2.2.4.

2.2.1 Definition of Reputation

The notion of reputation is commonly used in social life and economy, and there exists some common sense regarding its meaning. In [3], a reputation is defined as an expectation about an agent's behavior based on information about or observations of its past behaviors. Gertz [37] defines reputation as the memory and summary of participant behavior from past transactions, as well as the aggregated feedback on participants. There is an implication that past behavior is indicative of future behavior in those definitions. In general, the reputation for a service provider can be formed by means of a collection of ratings by different users; each such rating is taken as a measure of user satisfaction. In this thesis, we use the following definition for reputation.

Reputation is a measure that is derived from direct or indirect knowledge on earlier interactions of agents and is used to assess the level of trust an agent puts into another agent [13]. In P2P networks, reputation is a peer's belief in another peer's

capabilities, honesty and reliability based on the direct interactions among peers and the recommendations received from other peers [24].

2.2.2 Reputation Properties

Below are the most essential properties of reputation according to [3, 21, 24]:

- Reputation is context-dependent. A peer has different reputation values within different interest groups. For file-sharing systems, it means a peer may have different reputations in different file categories; and for game systems, it means the same player may have varying ranks in different games.
- Reputation is based on prior experience. Peers are able to identify repeated experiences within a similar context and with the same peers.
- Reputation is dynamic. Reputation should be updated in response to changing behavioral patterns.
- Reputation is non-symmetric. A peer may rate another peer higher than it itself is rated by the other peer.

2.2.3 Reputation and Trust Relationship

The notion of reputation is very relevant to systems where there is information asymmetry about trust, due to the large number of players involved and their anonymity[4]/pseudonymity[5]. Reputation can give evidence about the missing trust information.

Trust is defined as a subjective expectation an agent has about another agent's future behavior based on the history of their encounters [10]. It is a measure of the willingness to proceed with an action (decision) that places parties (entities) at risk of harm and is based on an assessment of the risks, the rewards and the reputations associated with all the parties involved in a given situation. Trust is an important factor in communities that want to maintain serious and long-lasting relationships, networking, and collaboration between peers in P2P networks.

Reputation, as an expectation about a peer's behaviors based on past transactions, is therefore an important component in evaluating the trustworthiness and reliability of the peer. The reputation of a peer clearly affects the amount of trust that others have toward this peer; thus, the aspect of reputation can be an incentive to make peers act in a more trustworthy way. In other words, reputation management may establish and reinforce trust in online communities. In the e-business domain, there already exist

[4] A security service that prevents the disclosure of information that leads to the identification of the end users.
[5] Pseudonymity is a property of some protocols and communications channels. A pseudonymous entity is one which is using a persistent identity which does not necessarily disclose other aspects of his or her identity [46].

means for trust enhancement in the form of reputation management systems, e.g., eBay's feedback forum, which will be described in section 2.2.4.1.

Reputation management has important significance for P2P systems where the vast majority of interactions among peers are between complete strangers who do not have any prior knowledge about each other. It is thus lacking trust among the peers. To help build up trust in such online systems, reputation-based trust management is an appropriate solution. An ideal reputation-based trust management is based on reporting, sharing and using reputation information in a society of agents in which cooperation among the agents is the dominant strategy. Reputation systems provide a mechanism to realize and enhance the trust among the peers in P2P networks. Reputation systems play two different roles in the interactions in P2P networks involving trust. The first role is informational. It makes a person trust more when given favorable information about the business partner. The second role reputation systems play is as a tool for disciplining or restraining, in order to control dishonest behavior. Therefore, in P2P systems, trust is the basis of system interactions, and reputation is fuel for trust or distrust.

2.2.4 Reputation Systems

As we mentioned before, reputation is the memory and summary of participant behavior from past transactions. However, when little or no information about reputation of a participant can be obtained, the others are left in the dark when

making decisions about the transactions. To help the participants make a decision, a reputation mechanism needs to be designed and introduced. The goal of a reputation system is to provide such a mechanism by gathering and aggregating feedback about participants' behaviors in order to derive measures of reputation.

2.2.4.1 Centralized Reputation Systems

Reputation systems have been implemented adopting either a centralized structure or a decentralized structure. In a centralized system, feedback is received and stored by a feedback collection centre [4]. Figure 2-10 shows a typical centralized reputation framework where X and Y denote transaction partners.

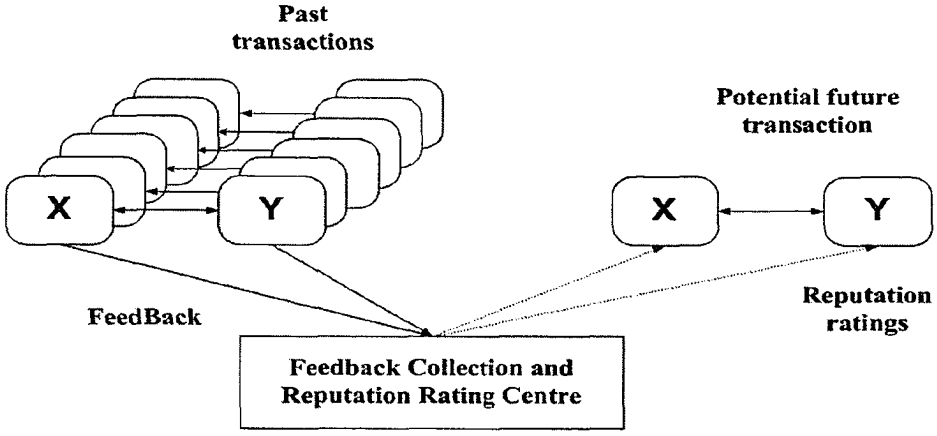

Figure 2-10. General framework for a centralized reputation system

In the figure, after a transaction is completed, the agents provide feedback about each other's performance during the transaction. The reputation centre collects feedback

from all the agents and continuously updates each agent's reputation rating as a function of the received feedback. Updated reputation ratings are provided online for all the agents to view, and can be used by the agents to decide whether or not to transact with a particular agent.

In some famous centralized reputation systems, such as eBay or Amazon, reputation mechanisms have been developed and used. For example, eBay's system of user feedback is one of the best known reputation systems in current use. eBay manages members' reputations in a centralized fashion with a system called "Feedback Forum". Buyers and sellers are able to leave comments about transactions, and both the number of successful transactions and the amount of feedback (passive or negative) strongly affect buyer and seller perceptions. Figure 2-11 is an example of auction items on eBay. As shown, the running total of feedback points is attached visibly to each participant's screen name, so everybody is able to readily check a potential transaction partner's accumulated "reputation". And figure 2-12 is an example of feedback from eBay's buyers.

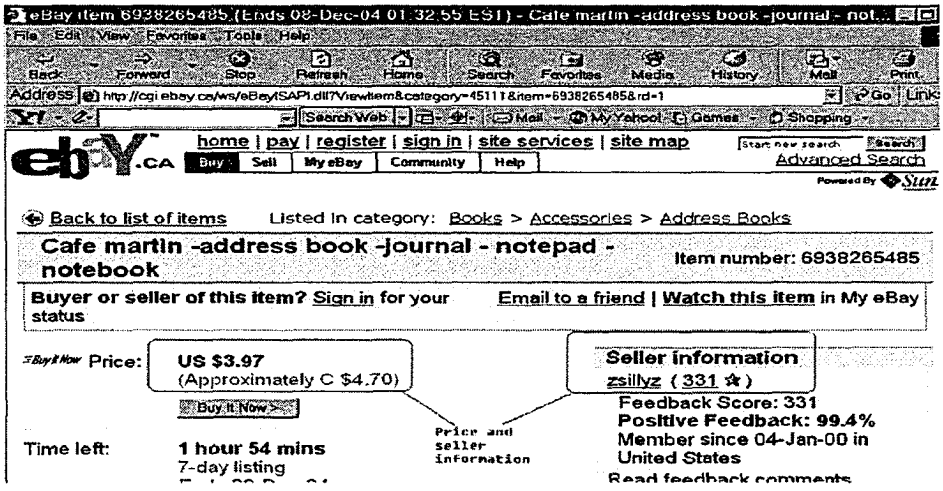

Figure 2-11. An example of auction items in eBay

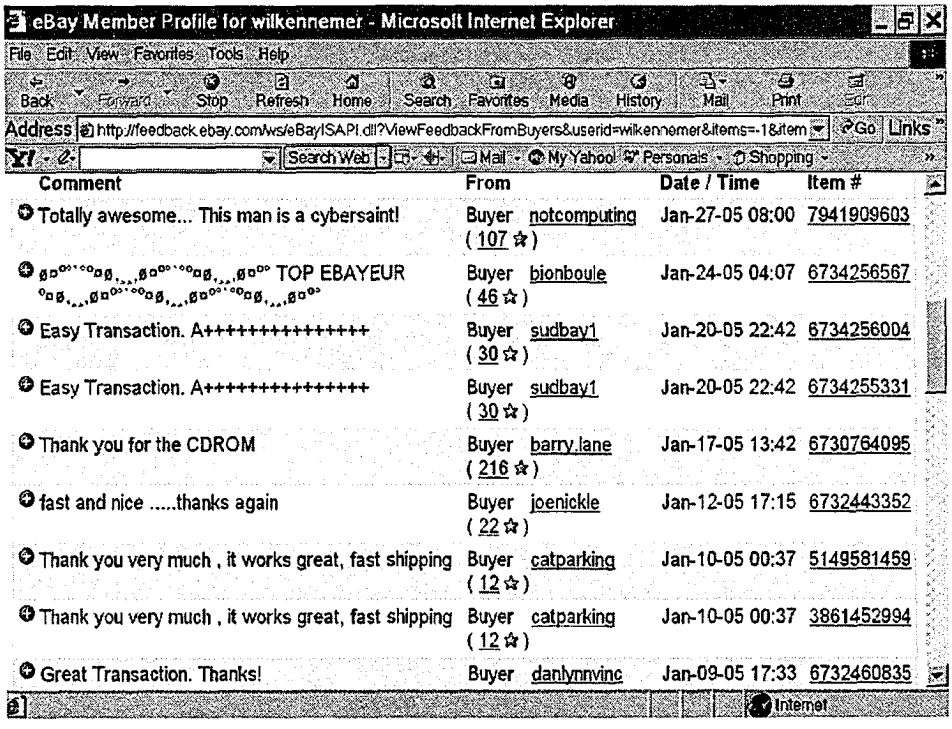

Figure 2-12. An example of feedback from eBay's buyers

Another famous centralized reputation system is Amazon's used books market platform for independent dealers (brick-and-mortar bookstores as well as private individuals). It provides a simple but illustrative example of how these systems work. Sellers post the price and a description of the book's condition on Amazon's site. Buyers pay through Amazon, but sellers ship directly to buyers. The moral hazard problems inherent in the seller's side of the deal – stipulating the book's condition and the shipping – is addressed through a feedback system in which buyers are invited to comment on the transaction, and which future buyers can view before deciding whether to make a purchase. Figure 2-13 is an example of the overview rating for a seller and figure 2-14 is an example of the feedback on this seller.

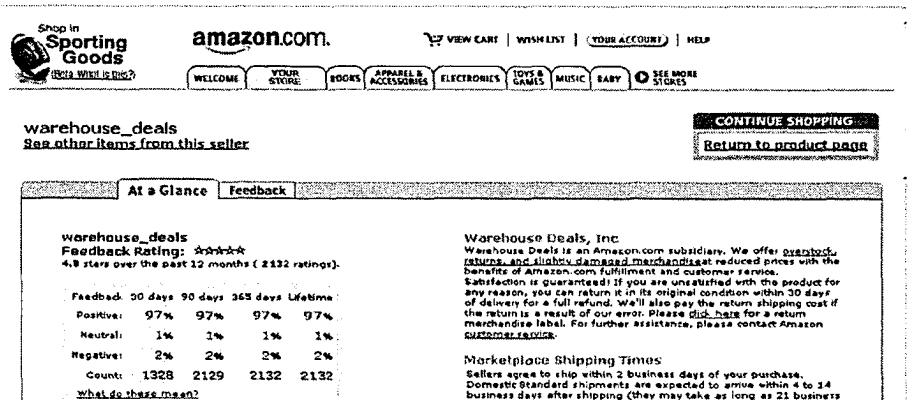

Figure 2-13. An overview rating for an Amazon used book seller

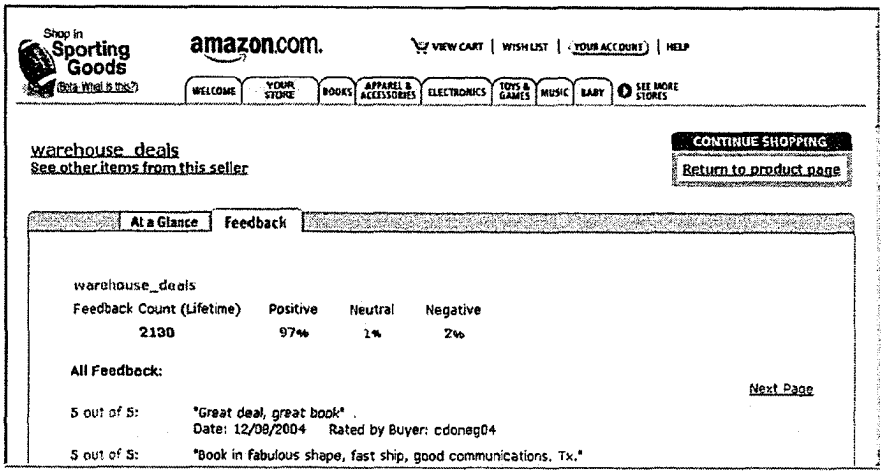

Figure 2-14. An example of feedback for an Amazon used book seller

There are some common characteristics in these centralized systems as below,

- A centralized server acts as the system manager responsible for collecting ratings involved in an interaction.
- Agents' reputations are public and global. The reputation of an agent is visible to all the other agents and the same from different agents' views.
- Agents' reputations are built by the system.

Centralized reputation systems are easy to monitor, and empirical results show that these systems do encourage transactions between sellers and buyers. However, there are some drawbacks in the centralized systems. For example, due to their centralized nature, these systems are vulnerable to a centralized attack, that is, if the centralized node (system manager) is attacked and broken, the whole system may crash.

2.2.4.2 Decentralized Reputation Systems

Decentralized reputation systems, on the other hand, do not make use of a central repository to collect and report reputation ratings. In this type of system, participants help one another with the provision of reputation ratings in order to evaluate the trustworthiness of potential transaction partners. Each participant is responsible for his own local repository of reputation and the collection and propagation of feedback as needed. The absence of a central server avoids a central point of failure.

In a decentralized system, agent X can get agent Y's reputation based on its own knowledge of the truthfulness of agents that make recommendations for agent Y. So it is difficult for agent Y to increase its reputation artificially. Since only agent X can see the recommendations, the references can express their feelings truthfully, and do not have to be concerned about potential revenge. But the tradeoff is that agents have to conduct more communication and computation. Figure 2-15 below shows a typical decentralized reputation framework where X and Y denote transaction partners.

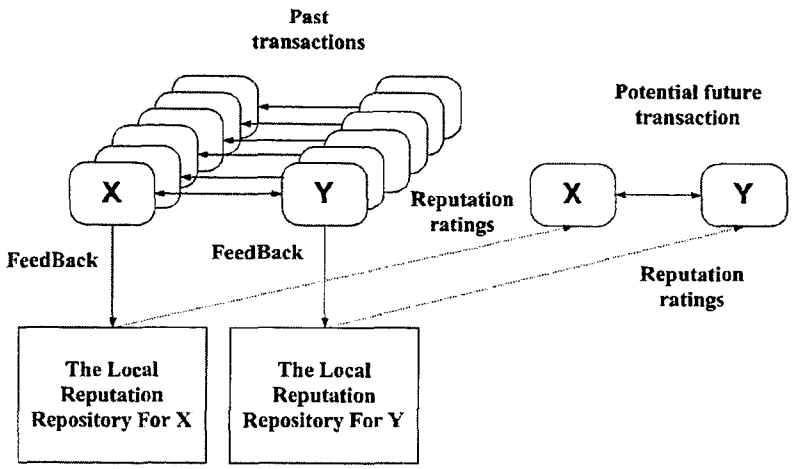

Figure 2-15. General framework for a decentralized reputation system

In figure 2-15, after a transaction is completed, the agents provide feedback about each other's performance during the transaction. Each agent collects feedback on other agents and updates those agents' reputation value in the local repository. Updated reputation values are not published publicly, but can only be used by the local agent to decide whether or not to transact with a particular agent.

Some decentralized reputation systems have been developed, such as EigenRep [6, 7], PeerTrust [15], P2PRep [14] and RCert [18]. EigenRep is a P2P file-sharing network and uses a distributed and secure method to assign a unique global reputation value to each peer. PeerTrust is a simple yet effective trust mechanism for quantifying and comparing the trustworthiness of peers in a decentralized P2P electronic marketplace. P2PRep uses a distributed broadcast polling mechanism to search and distribute

reputation information in the P2P network. RCert is another reputation-based system to provide efficient retrieval of reputation information by storing the content locally at the owner's peer. In chapter 3, we will provide a more detailed description and comparison of existing decentralized P2P reputation systems.

On the other hand, decentralized reputation mechanisms are more complex than centralized reputation systems. They have the following characteristics [38, 39, 40]:

- No centralized system manager is needed to govern trust and reputation.
- Subjective trust information is explicitly collected by each agent. Each agent is responsible for collecting its trust information on other agents based on their direct interaction with those agents.
- No reputation is published publicly. If agent A wants to get other agents' opinions about agent B's reputation, it has to actively ask other agents for their evaluations of B, then aggregates these ratings for a measure of agent B's reputation. Agent A can decide how to combine the collected evaluations. For example, it can weight differently the evaluations coming from different agents: trusted agents, unknown agents, or even untrustworthy agents.
- Significant communication is required between the agents to exchange their evaluations.

2.3 Related Taxonomy

Mui, et al. [23] proposed a taxonomy for different notions of reputation that have been studied by various researchers and implemented in real world systems (see Figure 2-16). At the topmost level, Mui, et al. has described reputation as individual and group reputation. The former has been applied by most of existing reputation systems such as those in eBay, Amazon, Free Haven, etc. Group reputation is studied by economists from the perspective of firm.

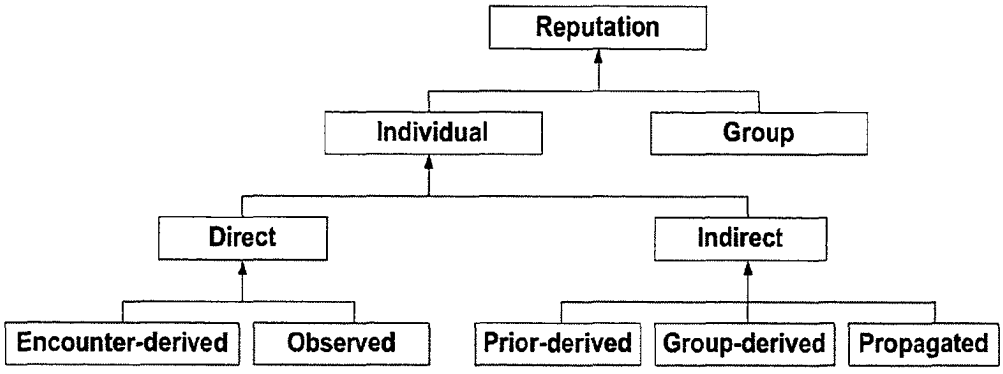

Figure 2-16. Reputation typology

The individual reputation is considered derived from either *Direct Reputation* or *Indirect Reputation.* Direct reputation refers to reputation estimated according to direct experience, while indirect reputation refers to reputation inferred from information gathered indirectly (e.g. using other observers' opinions).

Direct reputation is further subdivided into *Observed Reputation and Encounter-derived Reputation.* The former is based on the observations made about an agent's encounters with other agents, and the latter refers to the reputation derived from direct experience interacting with the other agents.

On the other hand, indirect reputation is further subdivided into *prior-derived reputation, group-derived reputation* and *propagated reputation.* Prior-derived reputation is based on prior beliefs of users. For example, most systems would rather assign lowest possible reputation value to new agents so that there is no incentive to discard an online identity when an agent's reputation falls below a starting point. Group-derived reputation means that an agent's reputation can be inferred from the reputation of the group to which the agent belongs. Propagated reputation is to evaluate agent's reputation based on information gathered from others in the environment.

The typology serves a useful function in unifying the diverse literature on reputation. Based on this typology, Mui's paper has studied the relative strengths of different notions of reputation in a set of evolutionary games. We will identify the relationship between Mui's typology and our taxonomy in the next chapter.

Chapter 3

A Taxonomy of Reputation Systems on Decentralized Peer-to-Peer Networks

This taxonomy is aiming to present some basic dimensions for classifying decentralized P2P reputation systems, and the corresponding set of methods used in these systems. We will explain the methods used by systems in each dimension and describe their advantages and disadvantages. Thus, the main purpose is to provide a starting point for researchers to construct their own P2P reputation systems.

This section is organized as follows. First, we give an outline of the taxonomy, then we take a look at the existing P2P reputation systems, next, we present the dimensions that constitute the taxonomy, which we group in two blocks: dimensions regarding reputation information generation and maintenance, and dimensions about the reputation metrics. In section 3.3 and 3.4, we provide the classification of existing systems according to the dimensions of reputation information generation and maintenance, and reputation metrics, respectively. We continue by performing a cross-dimensional analysis in section 3.5 and end with section 3.6, in which a comparison with a related taxonomy and several conclusions are presented.

3.1 Outline of the Taxonomy

In recent years, increasing numbers of P2P reputation systems have been developed and there is a wide variety of such systems, all of which take advantage of a P2P network infrastructure. We have followed two main approaches in this study of reputation systems: infrastructure-oriented and function-oriented.

The infrastructure-oriented approach produces a classification of reputation systems according to the system's underlying infrastructure (see Table 1 for the infrastructures of the various systems analyzed). Here, we categorize infrastructures in the context of decentralized P2P networks, and do not discuss centralized reputation systems. Further, most reputation systems do not limit the underlying infrastructures on which they can be applied. However, some reputation models may have preferred infrastructures. For example, the EigenRep model can work better in Distributed Hashtable (DHT)[6] based networks. Therefore, we classify such systems according to their intended infrastructures.

The function-oriented approach produces a classification based on the different task-achievement techniques used in the system, including the creation, maintenance and computation of reputation information. The latter approach allows us to study the

[6] A **Distributed Hash Table** (DHT) is a distributed and often decentralized mechanism for associating hash values (keys) with some kind of content. Participants in the DHT each store a small section of the contents of the hashtable [49]. A lot of decentralized peer-to-peer systems are based on DHT mechanism, e.g. CAN, Chord, Pastry [49].

systems in the aspect of system design[7]; consequently, we have devoted more space to it.

Table 3-1 Underlying infrastructure of the analyzed systems

NAME	REFERENCES	Infrastructure
Rahman &Hailes	A Distributed Trust Model, 1997	Decentralized P2P networks
DCRC &CORC	A Reputation System for Peer-to-Peer Networks, 2003	Gnutella
EigenRep	EigenRep: Reputation Management in P2P Networks, 2003	DHT-based P2P networks, such as CAN, Chord.
Kinateder &Rothermel	Architecture and Algorithms for a distributed Reputation System, 2003	Decentralized P2P networks
Managing Trust	Managing Trust in a Peer-2-Peer Information System, 2001	P-Grid
Mui &Mohtashemi	A Computational Model of Trust and Reputation, 2002	Decentralized P2P networks
OpenPrivacy	Design of the OpenPrivacy Distributed Reputation System, 2002	Decentralized P2P networks
P2PRep	Choosing Reputable Servents in a P2P Network, 2002	Gnutella

[7] That is, a lot of system functions we addressed in this approach, are the major concerns for designing a P2P reputation system. That is why our taxonomy can benefit the development of a P2P reputation system.

PeerTrust[8]	Building Trust in Decentralized Peer-to-Peer Electronic Communities, 2002	P-Grid
Pierre&Steen	A Trust Model for Peer-to-Peer Content Distribution Networks, 2001	Decentralized P2P networks
Poblano	Poblano: A Distributed Trust Model for Peer-to-Peer Networks, 2002	JXTA
RCert	Managing Trust in Peer-to-Peer Systems Using Reputation-Based Techniques, 2003	Gnutella
Selcuk&Uzun	A Reputation-based Trust Management System for P2P Networks, 2004	Gnutella
XRep	A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks, 2002	Gnutella
Yu&Singh	An Evidential Model of Distributed Reputation Management, 2002	Decentralized P2P networks

To analyze the reliability of information and assess information providers' trustworthiness, the key issue is reputation information, which is based on past interaction history. Reputation information generation and maintenance requires four design decisions, which constitute the first four dimensions of our taxonomy:

[8] Although PeerTrust declared that it could be applied to all kinds of P2P networks, at this time, it is just implemented on a specific one: P-Grid. Thus, we categorize PeerTrust according to its existing application.

reputation representation, reputation information generation, reputation information storage, and reputation information update.

Reputation representation provides the notation for a user's reputation. *Reputation information generation* indicates the techniques or ways of generating the initial reputation information. *Reputation information storage* addresses the structure and method of storage of reputation information. Finally, *Reputation information update* will adapt the reputation information as time advances.

With the creation and maintenance of reputation information, a new challenge is to develop algorithms or metrics through which these sources of information are combined to compute the reputation value of peers. With respect to reputation metrics, three main dimensions will be discussed: local reputation calculation algorithm, indirect reputation calculation algorithm and global reputation calculation algorithm. *Local reputation calculation algorithms* calculate peers' reputation value based on reputation seekers' direct experience with the target peers in the past. *Indirect reputation calculation algorithms* identify the ways to calculate peers' reputation value based on the others' opinions from the rest of the network. *Global reputation Calculation algorithms* specify the principle to compute peers' global reputation value. Global reputation value is the aggregation of local reputation value and indirect reputation value.

All in all, we have identified, from a functional viewpoint, seven classification dimensions for decentralized P2P reputation systems, four in terms of reputation information generation and maintenance and three in terms of reputation metrics (see table 3-2).

Table 3-2 Dimensions of the taxonomy

TAXONOMY OF REPUTATION SYSTEMS	
Reputation Information Generation and Maintenance	Reputation Metrics
Reputation Representation	Local Reputation Calculation Algorithm
Reputation Information Generation	Indirect Reputation Calculation Algorithm
Reputation Information Storage	Global Reputation Calculation Algorithm
Reputation Information Update	

Figure 3-1 shows the relationships between these mentioned dimensions. Generally, a peer estimates another peer (target peer)'s reputation based on the latter's local reputation value and indirect reputation value. The local *reputation value* is based on the peer's direct experience with the target peer, and is calculated through *Local Reputation Calculation Algorithm.* The indirect *reputation value* is based on the other peers' opinions about the target peer, and calculated through *indirect reputation calculation algorithm.* The local and indirect reputation value will be aggregated through *global reputation calculation algorithm* to generate the global *reputation*

value of the target peer. According to the global reputation value, the peer may interact with the target peer. Next, *feedback* may be provided by resource consumer upon the interaction and collected by the corresponding peer for *reputation information generation*, which in turn motivates the *reputation information update* and *storage*.

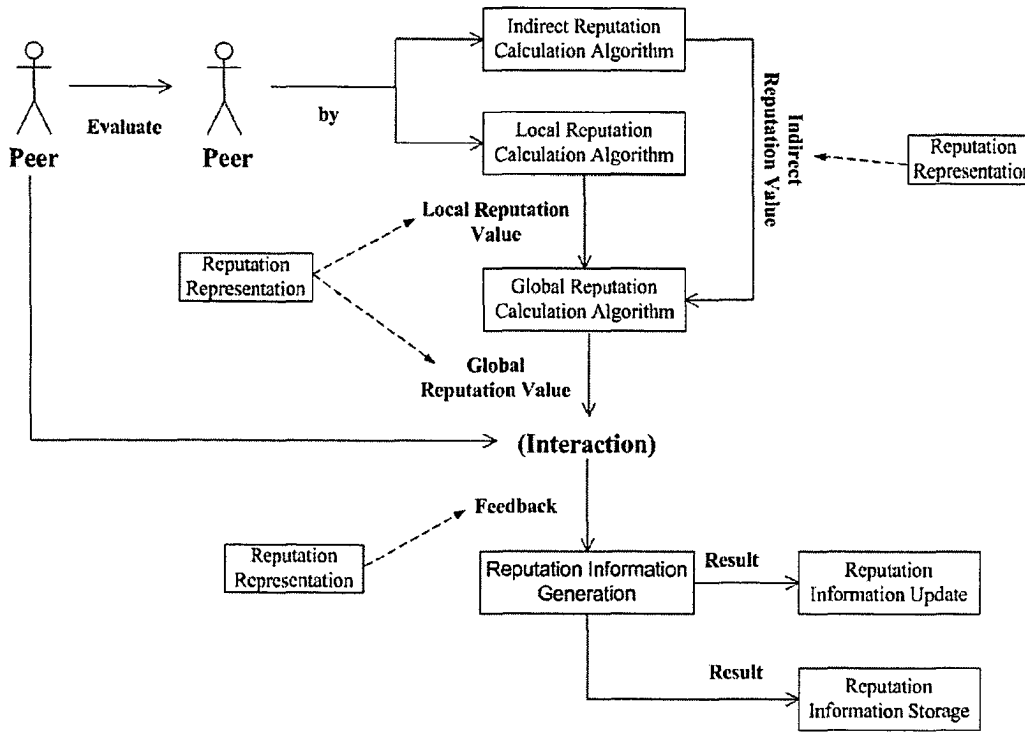

Figure 3-1. Reputation system overview

Before we go on to discuss those dimensions in further detail, we take a look at existing P2P reputation systems.

3.2 Overview of Existing P2P Reputation Systems

Rahman and Hailes [2, 3] proposed a method for reputation management that can be implemented in P2P networks. This model is based on Marsh`s model [34]. Actually it is an adaptation of Marsh's work to today's online environments. Some concepts were simplified and some were kept. But the main problem with their approach is that every agent must keep rather complex and very large data structures that represent a form of global knowledge about the whole network. In real world situations maintaining and updating these data structures can be a laborious and time-consuming task. Also it is not clear how the agents obtain the other agents' opinions about a target agent, and how well the model will scale when the number of agents grows.

Debit-Credit and Credit-Only Reputation Computation (DCRC and CORC) [5] facilitates local storage of reputations for fast retrieval. The difference between DCRC and CORC is that DCRC works by crediting peers' reputation scores for serving content and debits them for downloading. CORC credits peers' reputation scores for serving content, but does not debit them for downloading. Unlike most of existing reputation systems, both schemes utilize objective criteria for updating peer reputations by offering additional credits based on query processing and forwarding, and staying online. However, in order to secure the updates to reputations from peers which act in their self-interest, this partially distributed approach employs a central

reputation computation agent (RCA), which is contacted periodically by the peers to earn credit for their contribution to the network. The RCA does not affect the search and download function of the P2P networks.

In *EigenRep* [6, 7], two types of value: local and global, are being stored in the system. The local value is a peer's own view of the reputation of the peers it interacts with, and is stored by the peer itself. The global reputation of each peer is derived from multiple local values, and being handled by random peers in a distributed hash table (DHT) such as CAN or Chord. This method rules out the possibility of a malicious peer maligning the reputation of other peers through random selection of peers that calculate the global values and redundancy in global value[9]. A shortcoming is that the reputation evaluation process is based on a normalization assumption: The trust ratings held by a peer are normalized such that their sum equals 1. By this normalization, significant trust information is lost: e.g., if there are n identical trust ratings in the database, the normalized value of them will be 1/n whether the original values were the highest possible ratings or the lowest; or a single non-zero rating in the database will always be normalized to 1, regardless of its actual value.

Kinateder and Rothermel [8, 9] presented a reputation system starting from the participants' (e.g. requestor's and recommender's) requirements and ending with a proposed fine-grained reputation model. The reputation system consists of three

[9] The redundancy in global value means a peer's global reputation value is stored by multiple peers.

models: system model, trust model and knowledge model. In the trust model, the semantic closeness and other relationships between trust categories is modeled by a dependency graph, which has not been done before in a reputation system context.

Managing Trust [13] is based on the P2P system P-Grid. It integrates the trust management and data management schemes to build a fully-fledged P2P architecture for information systems. The reputations metric in this system are complaint-based only; the more complaints a peer receives, the less trustworthy it will be considered. An obvious shortcoming of such complaint-only reputation metric is that it only maintains negative feedbacks, providing no means for a trustworthy peer to build a reputation to be distinguished from a newcomer.

Mui and Mohtashemi [10, 11] surveyed the literature on trust and reputation models across diverse disciplines, and constructed a computational model of trust and reputation. The model defines reputation as a quantity relative to the particular embedded social network of the evaluating agent and encounters history, introduces trust as a dynamic, quantity between the trustor and the trustee which can be inferred from reputation data about the trustee, and proposes a probabilistic mechanism for inference among trust, reputation and level of reciprocity.

The *OpenPrivacy* [12] reputation system is based on a web-of-trust [52] style network of peer certifications. Integrity of the reputation information is preserved in

a certificate. Every certificate stores the value of the target's reputation and the confidence of the certificate creator, and is digitally signed by the private key of the certificate creator. A peer needs the public key of the certificate creator in order to verify the validity of the certificate and the information stored within.

P2PRep [14] is a reputation sharing protocol proposed for Gnutella. Every peer in the system stores its interaction experience with other peers (based on pseudonyms). Reputation communication is based on a distributed polling protocol. Service requestors can assess the reliability by polling peers. When no peer can have full knowledge of the network, each of them can record the trust it has in others on the basis of its own experience, and share this information with others. Before a peer consumes a service, it polls other peers about their knowledge of the service provider. At the end of the interaction, the service consumer updates its reputation of the provider and, at the same time, the credibility of the peers that provided an opinion on the provider.

PeerTrust [15] is a reputation-based trust management system for peers to quantify and compare the reputation of other peers in P2P networks. In this system, trust is also a non-decreasing scalar and is subjective, but differs from other reputation systems in that it is computed based on the following three factors: 1) the amount of satisfaction received by the other peers in the system, 2) the total number of interactions, and 3) a balancing factor to offset the impact of malicious peers that

misreport other peers` service. Each peer maintains a small database with a portion of the global trust data. Though this reputation storage scheme differs from that used in other reputation systems, it still requires cooperation from the peers for storing the reputations. Maliciousness is countered by having multiple peers responsible for storing the same database. Voting can be used if these databases differ. Trust is computed on the fly through querying (potentially) multiple databases over the network.

Pierre and Steen [16] presented a reputation-based trust model that allows users of a large-scale peer-to-peer system to build trust relationships with nodes that they did not necessarily know in the first place. In this model, each reputation value is based on the recommendations from other peers. Based on these recommendations, each peer infers a reputation value for other nodes in the system using Dempster-Shafer theory [35].

Poblano [17] is Sun's work on reputation in their JXTA peer-to-peer architecture. Poblano introduces a decentralized trust model with trust relationships not only between peers, but also between peers and content (what they refer to as "codat", code data). Reputation value in a peer is calculated based on the content this peer has stored in addition to its performance and reliability.

RCert [18] is another reputation-based approach that uses public key encryption to provide security properties such as authentication, integrity and non-repudiation. This protocol aims to provide efficient retrieval of reputation information by storing the content locally at the owner (server-side) in the form of certificates. The protocol also ensures that these certificates are secure from being modified by the owner. This approach provides an efficient mechanism for the retrieval of reputation content. Furthermore, all messages in RCert are unicast messages.

Selcuk and Uzun [19] propose a reputation system for P2P networks that helps identify malicious peers and prevent spreading of malicious content. This system provides a strong authentication mechanism for reputation-related message, and makes a clear distinction between the reliability of a peer as a provider of files and its reliability as a provider of trust information. This is a crucial distinction to preclude certain types of coordinated attacks. They also built a file download mechanism that enables early detection of disguised malicious files and terminates the connection before any significant amount of bandwidth is wasted.

Xrep [20] is an extension to P2PRep[14], in which two separate local repositories are maintained – one with reputation information about resources and another with reputation information about the servents. Each of these repositories stores binary reputation values. Peers update their local repositories for their resources and resource providers upon finishing transactions. The criteria for such updates are

subjective. To compute trust values for resources and peers, they enhance the two-phase Gnutella search and download protocol into four phases: resource query, voting, verification of the votes, resource download. The votes are evaluated without any consideration of the credibility or trustworthiness of the voters. No quantitative trust metric is specified to be used for choosing among alternative files for download.

Yu and Singh [21, 22] adapt the mathematical theory of evidence [51] to represent and propagate the reputation that agents give to each other. When evaluating the trustworthiness of any party, a peer combines its local evidence (obtained from direct transactions) with the recommendations received from others regarding the same peer. They also propose an algorithm to compute the trust index for an agent in the range [-1, 1] that was an outcome of local evidence and testimonies. This approach focuses primarily on reputations, but does not address the issues of integrity and authentication of messages.

3.3 Reputation Information Generation and Maintenance

Five design decisions should be taken to initialize and maintain reputation information: Reputation representation, reputation information generation, reputation information storage, and reputation information update.

3.3.1 Reputation Representation

Representing reputation information accurately is crucial for the successful interactions in peer-to-peer systems. Accurate reputation information can benefit participants by allowing us to predict the reliability of resources and resource providers. Reputation information is originated from users' feedback, and aggregated into reputation value through some algorithms (see section 3.4). Therefore, our discussion on reputation representation focuses on initial feedback and reputation value.

3.3.1.1 Initial Feedback

The initial feedback of users can take the form of a textual review, or a numerical rating. Numerical ratings and multiple-attribute ratings form the basis on which to compute reputation values. Multiple-attribute ratings are an alternative way of representing discrete numerical ratings. Attributes such as "Very good", "Good", "Fair", "Bad", "Very bad", can be easily translated to a numerical rating, with each attribute corresponding to a specific number. There exist several kinds of reputation

ratings, and each aims at a specific design goal. Figure 3-2 shows the initial feedback representation techniques used by the different systems analyzed.

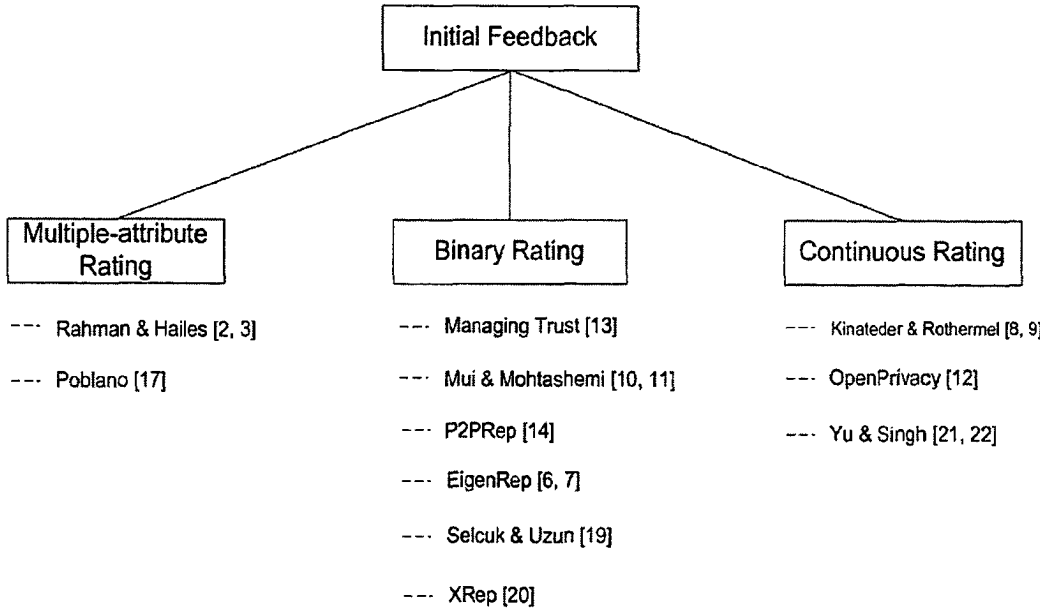

Figure 3-2. Initial feedback

1). *Binary Rating*

This representation only allows a distinction between complete trust in another agent or no trust at all. It is appropriate for calculating reputation value that expresses the expected outcome of a transaction with a specific peer. Binary rating is simple to

construct and allows unambiguous implementations. It is, nevertheless, rather restrictive because users are forced to choose between trusting another agent completely or not at all.

2). *Multiple-attribute rating*

In this scheme, each reputation value variable specifies a threshold which carries its own semantics. The semantic behind each reputation value is easy to use by people. For example, as we mentioned earlier, the semantics indicated by attributes "Very good", "Good", "Fair", "Bad", "Very bad" are easy to understand.

3). *Continuous Rating*

In this approach, reputation value is represented by real number, usually modeled as representing probability values. The reputation of an agent a_j in agent a_i's mind can be considered as the probability that in the next transaction, a_i will approve of a_j's rating. This scheme is useful for building a probability model; however, in comparison with the above two schemes, continuous rating values are hard to interpret with any meaningful accuracy. For example, how to interpret the meaning behind a rating value 0.34, good, very good, bad, very bad, or else? It is not easy to explain such rating accurately.

3.3.1.2 Reputation Value

Reputation value is the aggregation of initial reputation information or recommendations from recommenders. It identifies the reliability of users or resources, and is the basis for forming trust relationships. Different reputation values represent different levels of trust a peer may have in another peer. Figure 3-3 shows the reputation value representation techniques used by different systems analyzed.

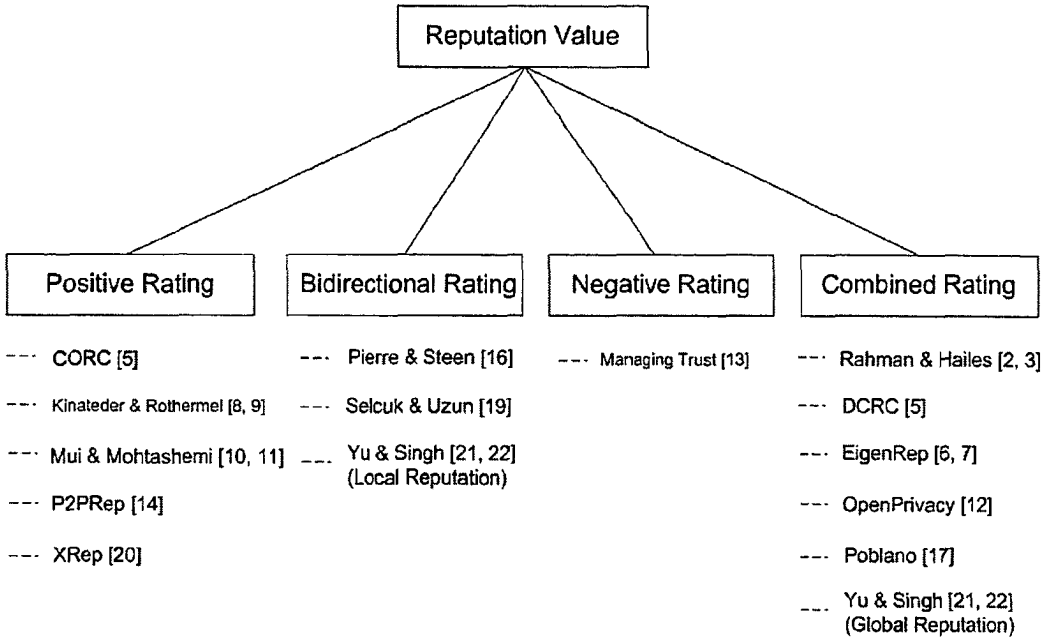

Figure 3-3. Reputation value

1). *Positive Rating*

Positive rating quantitatively describes the services a user has provided to the community. When there is no negative rating and users start at zero, there is no incentive for these users to change their names or identities. However, users may not be able to differ malicious users from new users since they may have the same reputation value, zero. Additionally, some systems assign zero value for negative rating; such kind of systems is also identified as positive rating systems. Reputation systems using positive rating offer an implicit disincentive for users to assume a new identity since reputations are built over time and having a long history of cooperation helps users to be chosen. Slander (i.e. malicious negative feedback) is not an issue in positive rating reputation systems, since no negative information is kept.

2). *Negative Rating*

In this scheme, reputation systems only evaluate complaints about a peer, that is, only negative feedback will be kept for a user. We can punish misbehaving users in such cases. It also provides a way of detecting and isolating malicious users. But there are several drawbacks as well. First, negative rating provides no means for a trustworthy peer to build a reputation to distinguish itself from a newcomer. Second, negative rating alone is not effective when malicious users can change their identities in order to refresh their reputation value whenever they have built up a bad reputation.

3). *Combined Rating*

In this scheme, rating can be positive or negative depending on the behaviors of users. A user who accidentally behaves inappropriately still has the chance to regain his reputation by behaving properly later. This approach prevents users from changing identities as frequently as in the negative rating scheme. The problem with this approach is that we may not be able to differ between the users who have more good (bad) behaviors than bad (good) ones from users who always behave properly (or improperly), since they may have the same ratings. Consequently, bad behaviors will be hidden beneath good behaviors, and go undetected by users.

4). *Bidirectional Rating*

This scheme provides both positive and negative rating simultaneously and, therefore, provides the most accurate picture of a user's reputation. The underlying belief is that, in real life trust relations, the existence of a single dishonest dealing in someone's history is usually considered a more significant indicator about that person's trustworthiness than a large number of honest transactions. To model this particular significance of cheating in a peer's history, we can use this scheme to explicitly differentiate between dishonest and honest transactions unlike in a combined rating. However, a problem is that the multiple nature of rating makes decision making more difficult. For example, if peer A has a higher positive rating and negative rating than that of peer B, it is hard to decide which one is more reliable, A or B? Since the one with more positive transaction records also has more negative ones, and vice versa.

3.3.2 Reputation information Generation

It is desirable to learn as much as possible from the user to lead to successful interactions between users. For this reason, the source of reputation information is an important aspect of a reputation system. There are two techniques for generating the initial reputation information, one based on explicit feedback, and another based on implicit extraction of user behaviors. Figure 3-4 shows the reputation information generation techniques used by the different systems analyzed.

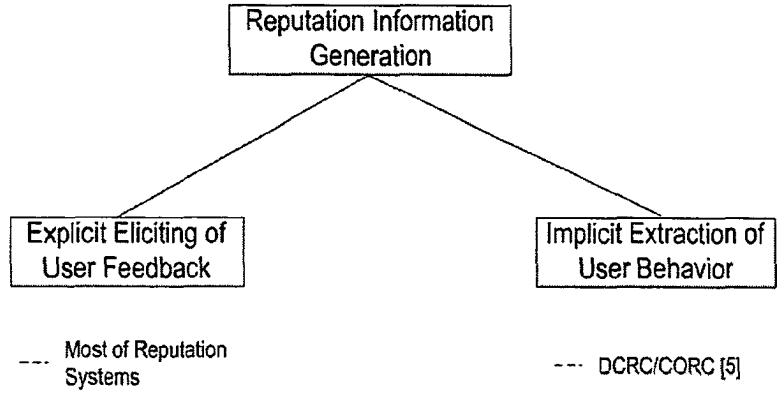

Figure 3-4. Reputation information generation

3.3.2.1 Explicit eliciting of user feedback

The technique relies on the resource consumer to provide feedback. A resource consumer rates a resource provider in term of the service[10] quality or capability. The rating is then collected by peers who are responsible for updating the corresponding

[10] Offered by resource provider.

reputation information. This approach is similar to "word of mouth" in real life. The problem is that in most online systems (such as eBay) feedback submission is voluntary. Thus, in the absence of concrete incentives, users may not provide feedback, or they may intentionally or unintentionally give untruthful feedback. A number of researchers are working towards developing mechanisms that provide strict incentives to users to both participate (i.e. provide feedback) as well as truthfully report their observations.

3.3.2.2 Implicit extraction of user behaviors

In this scheme, the system agents keep track of users' behaviors, so that the relevant reputation information can be automatically extracted by agents. An impressive amount of information about someone's past behaviors and interaction habits can be inferred in this way, such as the quantity of content and bandwidth. Such implicit reputation mechanisms are an intriguing complement to mechanisms that rely on explicit feedback. They can identify the contribution of users to the systems. A drawback of this scheme is that it can not provide any information for the quality of resources (or services). As a result, malicious peers may disseminate a lot of bogus files across the network to obtain a high rating.

3.3.3 Reputation Information Storage

In decentralized reputation systems, reputation data have to been stored in a decentralized manner. The storage of reputation information should support their later effective retrieval. There are three approaches to managing reputation information in a decentralized P2P fashion. Figure 3-5 shows the reputation information storage techniques used by the different systems we have analyzed.

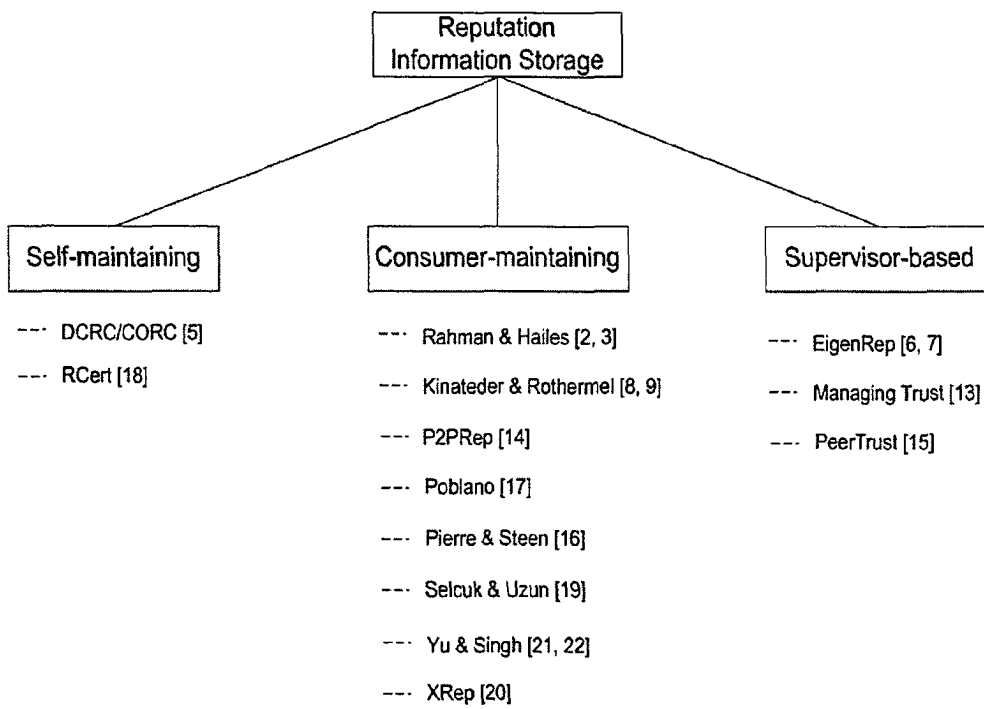

Figure 3-5. Reputation information storage

3.3.3.1 Self-maintaining

In this approach, reputation information is maintained by the owner. This greatly simplifies the problem of storing reputation information. In addition, the retrieval of reputation information can be done efficiently without any additional communication cost. However, by having the owner to store the reputation information, there is risk of information integrity if used without a monitoring mechanism. Malicious peers can easily report false reputation values.

3.3.3.2 Consumer-maintaining

In this approach, after each transaction, consumers maintain the reputation information of service providers. This method can prevent users from modifying their reputation information; however, since the reputation information is obtained on-demand, it incurs a high communication cost whenever users query for reputation information. Another drawback is that the information is distributed across the whole network, so information retrieval may be not as complete as with self-maintaining storage.

3.3.3.3 Supervisor-based

In this approach, each peer will be monitored by other special peers, called its supervisors. These supervisors hold the reputation value of the supervised peer. Any message related to reputation query, storing and updating will be directed to the

peers' supervisors. Similar to the second scheme, this approach could also protect reputation information against modification from its owners; moreover, it can effectively prevent collusion[11], which may happen in the other two schemes. However, building such supervisor-based mechanism is much more complex, and currently it is mostly applied in decentralized structured P2P networks[12].

3.3.4 Reputation Information Update

Since reputation is dynamic and non-monotonic, reputation value may rise or fall depending on various factors, including time and experience. So there is a need for updating reputation information in the system. Generally, updating is based on the experiences of peers with each other. Figure 3-6 shows the reputation information update techniques used by the different systems analyzed.

[11] Collusion means that a group of malicious peers collude to rate each other high and the rest of the peers low.
[12] There are two design rationales for such a supervising network. First, a peer can not choose where to locate itself, which prevents a peer from computing its own reputation value. Second, the choice of supervisors should look random to the supervised peer so that malicious users can not form small groups that control their own ratings. Currently, only structured P2P networks satisfy the above conditions through their network topologies, such as Chord (rings) and P-Grid (binary trees) [36].

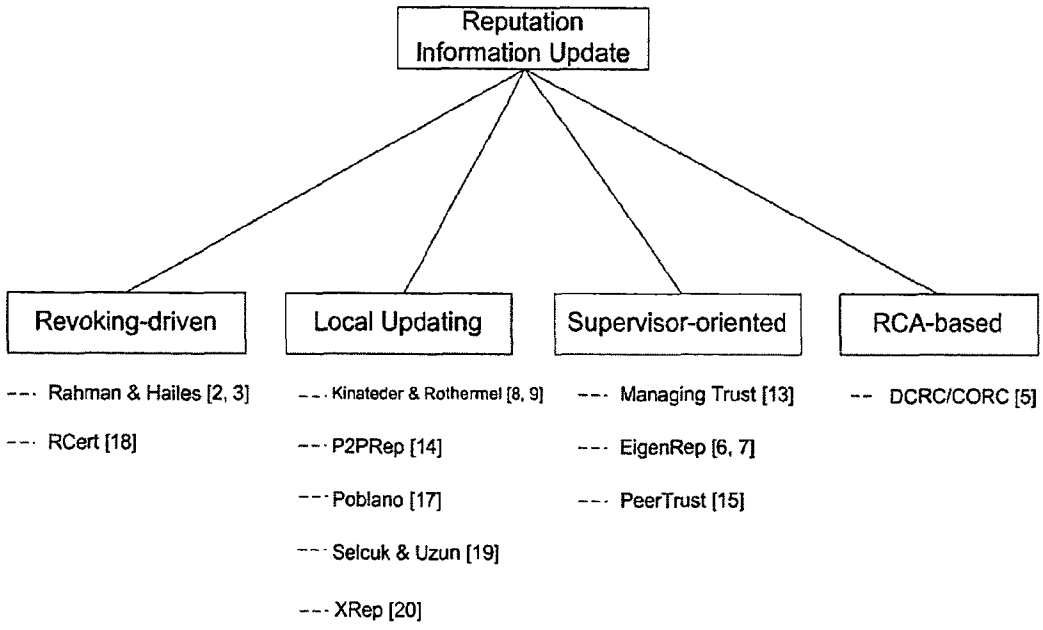

Figure 3-6. Reputation information update

3.3.4.1 Revoking-driven

After each transaction, a peer's reputation information will be adjusted based on his behavior in the transaction, and all the previous raters or requestors have to update their rating towards the peer. This method provides dynamic updating of reputation information; however, it is costly since it always (that is, unsolicitedly) propagates the updated information across the network.

3.3.4.2 Local updating

A peer will update its local trust repositories after a transaction. If the chosen service turns out to be malicious, all peers who offered that service can be assumed to be malicious, and their ratings must be downgraded along with the peer by whom the service was provided. If the reputation score calculated for the service was partly or entirely based on other peer's feedback, then the credibility rating of each peer who contributed to the selection of that service in the reputation query is updated by the requestor according to the outcome.

The updated reputation information is not distributed until explicitly requested from the network. As a result, if a peer lies about its resource provider, it is not detected, since this information is never broadcasted or provided (back) to the resource provider.

3.3.4.3 Supervisor-oriented

If the storage scheme is supervisor-based, reputation updating will be reported to and executed by the supervisors. The update algorithms will vary according to the underlying infrastructure.

3.3.4.4 RCA –based

This method uses a reputation computation agent (RCA) to periodically update peers' reputations in a secure, light-weight, and partially distributed manner. In this

mechanism, the updated rating of a peer will first be forwarded to the RCA. Each peer locally stores its reputation information for later fast retrieval, and periodically obtains the updated information from RCA. The RCA can monitor a peer's behavior to prevent malicious peers from modifying their local reputation data, and does not affect the search and download function of the P2P network. A problem with such mechanism is that it will be vulnerable to a central authority attack[13].

[13] That is, if the RCA is attacked and corrupted, the reputation system will not function properly.

3.4 Reputation Metrics

3.4.1 Local Reputation Calculation Algorithm

The local reputation value is a peer's local belief about another peer. It is derived from direct interactions the original peer[14] has with the correspondent[15], and can be propagated to others upon request. It can simply be adding individual ratings, or based on a more complex mathematical function. Figure 3-7 shows the local reputation value calculation techniques used by the different systems analyzed.

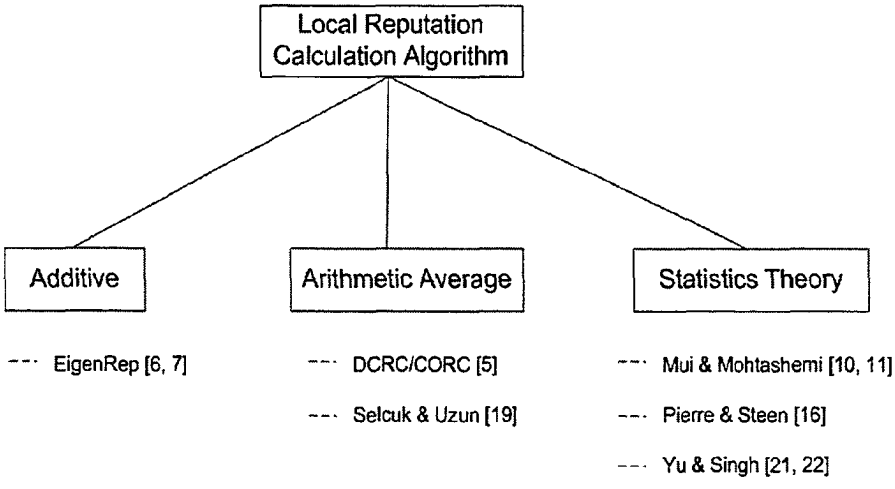

Figure 3-7. Local reputation calculation algorithm

[14] The peer who initializes a request.
[15] The peer who responds the queries issued by the original peer.

3.4.1.1 Additive

In this method, the local reputation value of a peer is the sum of the ratings assigned by the owner over time. This method is easy to implement, but not as accurate as the following two methods.

3.4.1.2 Arithmetic Average

In this method, the reputation score assigned to a target peer is determined as the average or a weighted average of its ratings assigned by the owners. This method provides more accurate evaluation about reputation value than the additive one; however, it is not as simple as the additive method in terms of implementation.

3.4.1.3 Statistics Theory

This approach uses a statistics (scalar) metric, such as *Bayesian Probability Theory* [53] or *Dempster-Shafer theory* (also known as the theory of belief functions) [54], to model the subjective probability that an agent will perform a particular action as expected. This approach is not as intuitive as the previous two approaches; however, it takes advantage of some probability theory models which have been widely applied in the prediction of future events.

3.4.2 Indirect Reputation Calculation Algorithm

In a large decentralized network, it is difficult to obtain first hand knowledge and experience about every entity. Thus, the only option open to network agents for coping with uncertainty is via others' opinions (or recommendations). Recommendations from others can be aggregated to generate indirect reputation. Indirect reputation differs from local reputation value in that the latter is based on direct interactions between reputation requestor and target peer. In general, if systems adopt self-maintaining or supervisor-based storage, then peers or the peers' supervisors can automatically receive feedback from consumers after each transaction; otherwise, reputation requestors have to actively query others for their opinions about target peers.

A peer may receive multiple recommendations (feedback) for a single target peer. Those recommendations must be combined to yield a single value. Figure 3-8 shows the Indirect Reputation Calculation mechanisms used by the different systems analyzed.

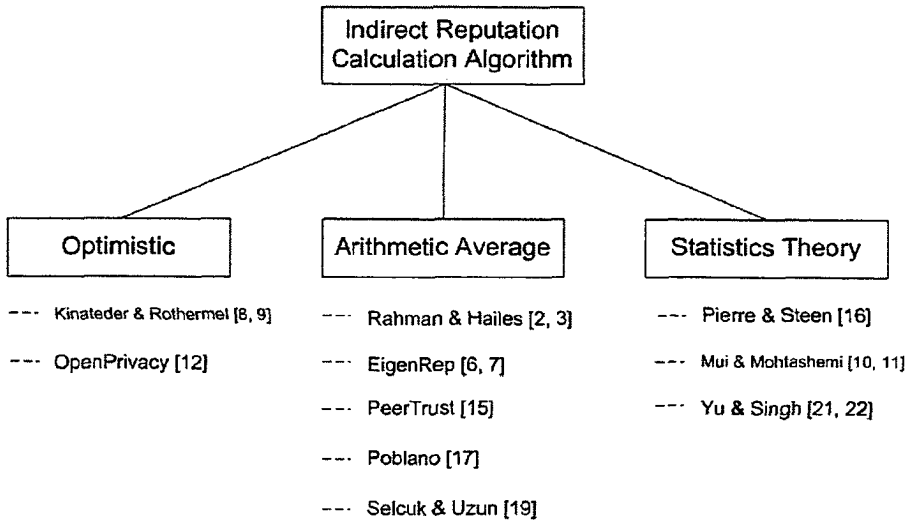

Figure 3-8. Indirect reputation calculation algorithm

3.4.2.1 Optimistic

This approach singles out the strongest recommendation. The strongest recommendation could be the one with highest reputation rating, or the one with highest positive reputation rating and lowest negative reputation rating. Comparing to other approaches, this one is easy to implement; however it can only provide a rough evaluation for a peer's reputation.

3.4.2.2 Arithmetic Average

In this method, to combine these recommendation trust paths into one, the arithmetic mean of the trust values of each path is taken to be the new single value. In other words, a reputation score is computed as the weighted sum of the reputation ratings

from other peers. Suppose A's weight for feedback of peer i is c_i and peer i's trust rating for target peer B is t_i, then A's reputation evaluation on B is: $\frac{\sum_{i=1}^{k} c_i t_i}{k}$. This method allows us to tolerate wrong recommendations provided by malicious peers. However, it is more complex than the previous one in terms of implementation.

3.4.2.3 Statistics Theory

This approach uses a statistics (scalar) metric, such as Dempster-Shafer theory; to model the subjective probability an agent will perform a particular action. It is done the same as for local reputation. It provides a sound statistics basis for combining feedback and for expressing reputation ratings, but need more implementation job.

3.4.3 Global Reputation Calculation Algorithm

Global reputation is the total belief a peer in another peer (or target peer). It combines the local belief (if any) with the opinions received from other peers. This reputation can be used for the final evaluation on the trustworthiness of the target peer. Figure 3-9 shows the global reputation algorithm used by the different systems analyzed.

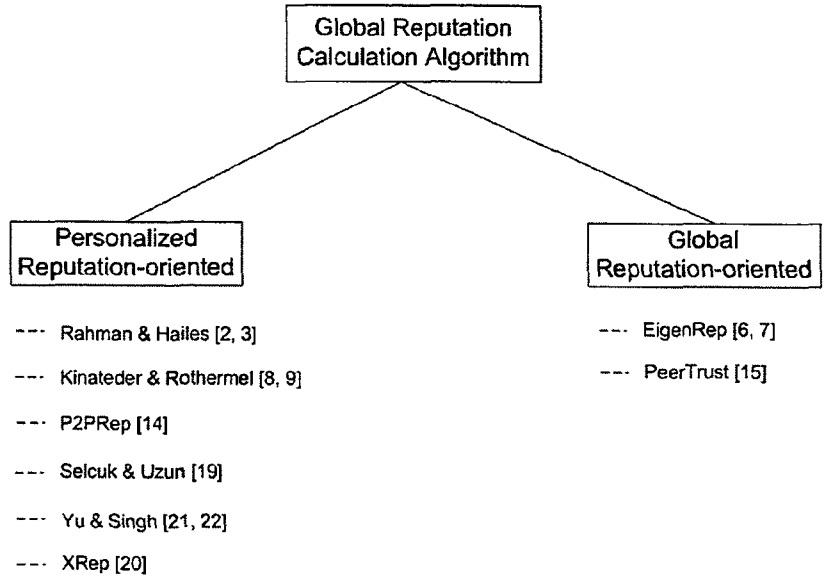

Figure 3-9. Global reputation calculation algorithm

3.4.3.1 Personalized Reputation-oriented

Personalized reputation means an agent is likely to have different reputations in the eyes of others. In this mechanism, each peer deduces target peer's global reputation value from the local reputation value and some of other peers' recommendations for the target peer. The point is that recommendations come from a subset of the whole networks.

This mechanism is similar to the generation of reputation in real life. It just contacts some peers in the system; therefore, this metric can only provide limited reputation information.

3.4.3.2 Global Reputation-oriented (EigenTrust Algorithm)

This mechanism differs from the first one in that it considers not some of peers' opinion but everyone's. That is, a peer queries not only his friends, but his friends' friends. He will continue in this manner until he has a complete view of the network, fortunately, if the network is large, such query process will stop at some points [6, 7]. When he stops the query, all global reputation value of the peers in the network will be available, that is, a peer's reputation is globally visible to all peers in a P2P network. This mechanism has a better performance, but requires much more communication overhead since it retrieves the reputation data of all peers in the network even when a peer is only interested in evaluating the trustworthiness of a particular peer or a small set of peers.

Currently, this mechanism can only be applied in decentralized structured P2P systems. The decentralized unstructured P2P systems can not do reputation queries globally since they forward queries only to a limited subset of the peers (bounded by the time-to-live) [50].

3.5 Cross-Dimensional Analysis

Trust is inherent in all social interactions and reputation systems provide a good foundation to build trust in virtual communities. An effective reputation system must base their performance on their functionality. In this chapter, we have surveyed 15 systems following a functional approach that allows us to draw up a general taxonomy comprising seven dimensions and based on two main groups: the generation and maintenance of reputation information, and the reputation metrics.

We then carry out a cross-dimensional analysis using the results of the infrastructure-oriented approach that is a cross-dimensional analysis among all the reputation systems based on the same infrastructure. From such an analysis, we have detected that different underlying infrastructures may lead to different patterns in some particular dimensions, such as *reputation information storage*, *reputation information update*, and so on. We will try to determine the common patterns in each infrastructure based on those infrastructure-sensitive dimensions, and those unambiguous classified systems.

Table 3-3 illustrates a cross-dimensional analysis among decentralized unstructured P2P networks. Here, a "N/A" entry means that this system can not be clearly categorized in this dimension. Some common features can be derived from such an analysis. In this table, most systems adopt a *self-maintaining* or *consumer-maintaining* storage strategy. Most systems also only support a *personalized*

reputation-oriented strategy to global reputation calculation. In addition, those systems can apply any *reputation information update* methods except the *supervisor-oriented* one[16].

Table 3-3 Cross-dimensional analysis among decentralized unstructured networks

Name	Reputation Information Storage	Reputation Information Update	Global Reputation Algorithm
DCRC& CORC	Self-maintaining	RCA-based	N/A
P2PRep	Consumer-maintaining	Local Updating	Personalized Reputation-oriented
RCert	Self-maintaining	Revoking-driven	N/A
Selcuk &Uzun	Consumer-maintaining	N/A	Personalized Reputation-oriented
XRep	Consumer-maintaining	Local Updating	Personalized Reputation-oriented

Table 3-4 illustrates a cross-dimensional analysis among decentralized structured P2P networks; the notation "N/A" has the same meaning as Table 3-3. Some common patterns emerge from this table. In this table, most of the systems adopt a *supervisor-based* storage strategy, and *supervisor-oriented* update strategy. All of

[16] The supervisor-oriented updating strategy is bounded with the supervisor-based storage strategy (see section 3.3.4), and the latter can only be applied to decentralized structured P2P networks (see section 3.3.3).

them are available for the *global reputation-oriented* strategy when computing the global reputation value.

Table 3-4 Cross-dimensional analysis among decentralized structured networks

Name	Reputation Information Storage	Reputation Information Update	Global Reputation Algorithm
EigenRep (Secure Version of EigenTrust)	Supervisor-based	Supervisor-oriented	Global Reputation-oriented
Managing Trust	Supervisor-based	Supervisor-oriented	N/A
PeerTrust	Supervisor-based	Supervisor-oriented	Global Reputation-oriented

Some reputation management systems, such as OpenPrivacy, leave the final trust value calculation to an external 'plug-in' module that supports third-party implementations. As such it does not define how or whether trust should be calculated based on experience. Flexibility with regard to the choice of reputation metric may become a major trend for reputation systems.

We did not detect any other patterns. We believe that the lack of common features lies in the fact that most systems have been developed following ad hoc approaches to satisfy specific application requirements. In this sense, we think that the taxonomy provided in this paper can provide a useful guideline for researchers contributing to the future development of new reputation systems.

3.6 Summary

The unceasing growth of the Internet and its environment has brought the need for new technology to help users find what they are looking for. This chapter tried to collect the state-of-the-art elements of reputation systems for decentralized peer-to-peer networks. To achieve this goal, we have selected 15 peer-to-peer reputation systems which give detailed description regarding system design and cover almost all the existing design possibilities in this research area. Furthermore, we believe that some dimensions in this taxonomy are also applicable to other kind of networks.

According to our knowledge, there is currently no paper providing classification of reputation systems. However, Mui, et al. (2002) has proposed a taxonomy of different notions of reputation, and some of these have been adopted in our taxonomy, for example, the notions of direct and indirect reputation. However, Mui's taxonomy did not mention the functional aspects in a reputation system, e.g., reputation representation, reputation information storage and update, and reputation calculation algorithm, etc. Furthermore, Mui's notions of reputation have been studied by various researchers across a number of disciplines such as distributed artificial intelligence, economics, and evolutionary biology, among others. Therefore, the scope of these notions of reputation is much wider than the ones applied in current reputation systems. For example, at the topmost level, Mui, et al has described reputation as individual and group reputation. However, the existing reputation systems focus on the study of individual reputation.

In this chapter, we have analyzed 15 systems, using a functional approach, and have divided our taxonomy into two main groups: reputation information generation and maintenance, and reputation computation. From this, we obtained a basic list of seven dimensions and we have explained, within each of these dimensions, all the techniques used in the systems we analyzed. We followed that with a cross-dimensional analysis which we hope gives designer a set of guidelines that will help them in their work developing new P2P reputation systems.

To summarize, our taxonomy provides a comprehensive explanation of current decentralized P2P reputation systems which we hope will contribute towards progress in this area of research.

Chapter 4 Reptella Reputation System: A Case Study

Our case study is based on a P2P application for file sharing, which facilitates internet users to share files. A famous P2P file-sharing system is the Gnutella file-sharing system which takes the pervasive Gnutella network as the underlying infrastructure; therefore, our case study will be based on a Gnutella file-sharing system.

In a reputation-based P2P file-sharing system, users predict the level of risk in a file using the perceived reputation of its provider. Users may choose to trust only resources offered by providers with a good reputation, and they will avoid resources offered by providers that have a bad reputation. Consequently, a user may decide if interact with a provider or not based on the relevant reputation information. In this paper, we will alternately use resource and file to refer to the same object. Let us denote the user who requests file(s) as Resource Requestor and the user who responds such requests as Resource Provider. Figure 4-1 summarizes these concepts and situates them in a general model of reputation.

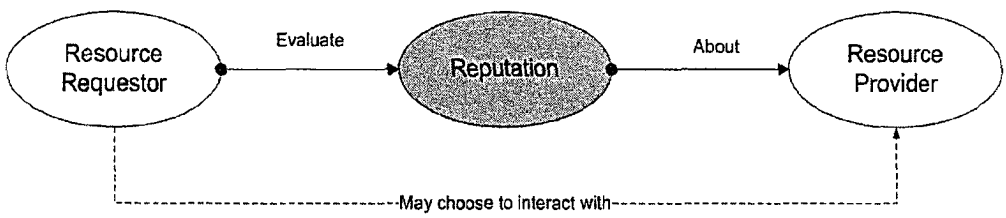

Figure 4-1. Model of reputation-based file-sharing system

Two important issues are involved in a P2P reputation system: (1) what reputation metrics are effective for computing peers' reputation value? (2) how to distribute, store, access, and update the reputation value effectively?

Those issues render the design of P2P reputation systems more complex. In this context, the classification of reputation systems with respect to a set of general dimensions is especially helpful, since it allows for a straightforward matching of system requirements and available techniques.

In this chapter, we describe in detail the design and implementation of a Gnutella-based[17] reputation management system: Reptella. Reptella is a reputation-based P2P file-sharing system. In this system, each participant can search, download, and contribute files. A reputation mechanism is proposed to evaluate the trustworthiness of resource providers with respect to providing high quality files or services.

Through this case study, we give a detailed example of how to apply our taxonomy during the design of a reputation system; furthermore, we could clarify the basic components for reputation management; finally, we discuss how to incorporate reputation system into an existing P2P network. Our purpose for this case study is to provide a guideline for those first-time reputation system developers; however, the techniques in our taxonomy can also help improve existing reputation systems.

[17] Based on Gnutella protocol.

4.1 Modeling a Gnutella-based P2P Reputation System

In this section, we will examine the specifics of our P2P reputation system, so that we are able to identify the system's demands and restrictions with respect to reputation management. We will see that these restrictions considerably confine the set of applicable schemes. On the other hand, other than those general properties addressed in the previous background chapter, there are particular properties being considered in the design of the Reptella system. Furthermore, the design of Reptella will concentrate on the generation, distribution and computation of reputation information, as we have discussed in our taxonomy; therefore, we will not consider some properties not pertinent to highlighting the desirable aspects of reputation management, e.g. context-dependency.

4.1.1 Design Issues

4.1.1.1 Term and definition

For the purpose of this discussion, we define the following terms.

Entity:	Any object in a network.
Recommendation:	Reputation information that is being communicated between two peers about another peer.
Recommender:	An peer providing recommendations.
Experience:	An interaction (or observation) of peer A with (about some behavior of) peer B.
Client:	When the computer is run as a client, i.e. a consumer of resource.
Server:	When the computer is run as a server i.e. a provider of resource.

4.1.1.2 Reputation Properties

We now describe and formalize some important properties of reputation, other than those mentioned in chapter 2. Our system design will be compatible with those properties.

- Reputation is multi-dimensional. Reputation can be represented in multiple dimensions. For example, in a file-sharing system, we can evaluate a resource provider's reputation along different aspects (download speed, quality of files, etc.).
- Reputation can be propagated between peers. If peer x knows peer y and peer y knows peer z, but x does not know z. x could predict z's reputation based on y's feedback about z. However, x may adjust its evaluation after it interacts with z.

- Reputation consists of two sets of ratings. One is to keep information about the trustworthiness of peers as resource providers, and the other indicates the credibility of peers as reputation recommenders. The reputation of a peer as resource provider may be different from that as reputation recommender. For example, it is possible that a peer may maintain a good reputation by performing high quality services but send malicious feedback to its competitors.

4.1.1.3 System Assumptions

As the discussed earlier, this case study focuses on applying our taxonomy as well as identifying the basic elements (or components) to build up a P2P reputation system. To highlight those aspects, we make the following two assumptions.

First, we assume that there is a single reputation value for each entity with respect to one aspect (or dimension). Although the reputation value for each entity is multi-dimensional, there is only one reputation value for each dimension. In other words, we won't take into account the context-dependent aspect in our reputation model. To our knowledge, this aspect is not fulfilled in the related work to the extent presented here.

Second, although it is preferable to enable the integrity aspect in our reputation system, we would like to defer the implementation to include security mechanisms into a future version of Reptella. In this case study, we assume that the information

will not be tampered with during transmission. We can incorporate one of existing security mechanisms into the Reptella system later to enhance the reliability of our reputation system.

4.1.1.4 Design Considerations

With the discussion of design considerations, we can identify what system aspects will be the focus of this case study. Furthermore, we will specify the system requirements related to each aspect. Later, we will give solutions for the fulfillment of those system requirements.

1.The reputation information should indicate the offer's quality.

The reputation information may disclose a peer's contribution, or capability to support high quality service, etc. Typically, in file-sharing systems, the resource selection is based on the offer's quality (e.g. the predicted download speed, the quality of file). So it is desirable to provide some solutions for this need.

2.The system should support meaningful feedback.

A practical issue of utmost importance for the correct functioning of our reputation-based system, is the correct phrasing of the question asked to the user for his evaluation of an interaction. We suggest those questions be easy to understand and unambiguous.

3.The reputation information should be stored securely.

Two concerns affect the reputation storage scheme: information integrity and fast retrieval of reputation data; however, these conflict with each other to a certain degree. In Reptella, we are more concerned about integrity.

4. The system should incur minimal overhead in terms of updating reputation information.

Since reputation is dynamic, it must be frequently updated. We require that reputation information is updated after each transaction, and at low computational cost. Additionally, the system should be robust to central point attacks, which aim at a central reputation authority.

5. The system should not give peers any incentive to change their identity.

Due to the anonymous nature of a P2P system, it is extremely easy for a peer to change its identity. This makes it easy for malicious peers to leave behind their bad reputation. It would impair the system in terms of detecting malicious behaviors.

6. The system should be fault-tolerant.

The system should be robust to malicious reputation information, e.g. the wrong recommendations given by malicious peers. The bad effect incurred by malicious information should be handled to a certain degree.

4.1.1.5 Our Solution

In this section, we will take advantage of the taxonomy in chapter 3, and decide about the most appropriate approach to satisfy the abovementioned system considerations. This process is illustrated in figure 4-2, which is followed by a detailed description.

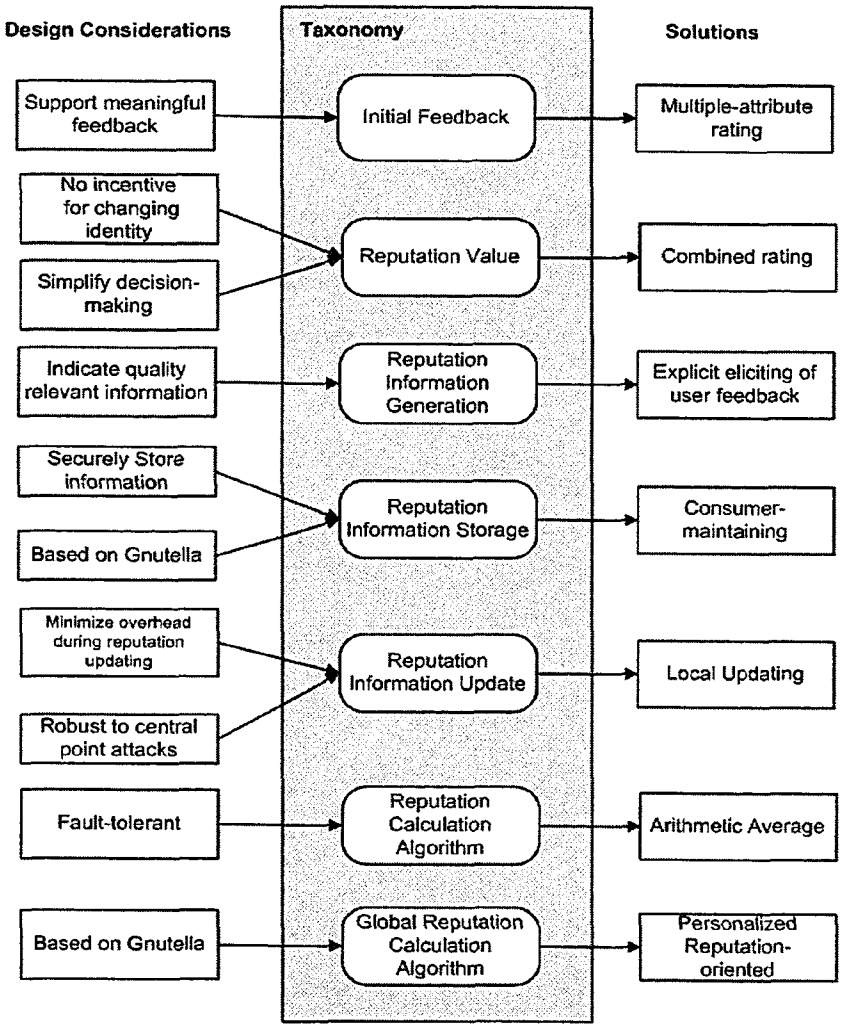

Figure 4-2. Apply taxonomy for design decision

1) This system extracts reputation information from explicit user feedback.

   The reason is that explicit feedback could provide information about the offer's quality, and that information will help user select resources.

2) This system will adopt multiple-attribute ratings for files and services.

   In general, the rank for the quality of files or download services is multi-level. Binary ratings can not provide such desirable levels of precision. On the other hand, the qualitative nature of reputation in this context makes it difficult to represent reputation with continuous values to any meaningful accuracy. Thus, continuous rating is not appropriate.

3) We adopt a consumer-maintaining strategy in the storage of reputation information to satisfy storage demand.

   We first exclude the supervisor-based strategy. Since this system is based on Gnutella network which is a decentralized unstructured P2P network. It is difficult to apply supervisor-based storage in this network. A self-maintaining scheme will induce peers to modify their own reputations, and since we do not implement a security mechanism here, we will not choose this strategy. As a result, a consumer-maintaining strategy, which distributes reputation information to the resource consumer and in a decentralized way, is the preferred choice.

4) To update information at low cost and avoid central point failure, the recommended scheme will be Local Updating.

   Since we do not support supervisor-based storage scheme, we can not apply a supervisor-oriented strategy. To avoid central point failure, we discard a RCA-based strategy. The revoking-driven scheme would result in too much communication overhead, which makes it inappropriate for this system.

5) To prevent malicious users from frequently changing their identity and at the same time, detecting and discouraging malicious users, the system should not assign any profit to newcomers, that is, the newcomer's reputation value should be zero or lower. To achieve this goal, we could apply a combined rating or bidirectional rating method. Here, we choose the former to simplify decision-making.

6) To make better predictions about users' behavior, this reputation system should be robust to any malicious recommendations.

   We will select the arithmetic average scheme to compute indirect reputation. Since the average of the recommendations can adjust the weights for different recommenders and the recommendation from lying recommenders will have less effect (or lower weight) on the process of recommendation aggregation. Statistical methods, such as the beta probability function, or Bayesian probability theory, are less intuitive. For our case study, we choose the easier-to-understand

average scheme. Additional, we also apply arithmetic average method in local reputation calculation, since this method can provide more accurate reputation evaluation than the additive one, and easier to understand when comparing to statistical methods.

Additionally, to avoid laborious and time-consuming computation of the global reputation and take into account the decentralized unstructured nature of the Gnutella network, we apply the personalized reputation-oriented technique for the global reputation calculation.

4.1.2 System Overview

In this section, we present a use case model of our system, describing the system actors and general system scenarios. System actors correspond to some of entities in our P2P reputation system. System scenarios describe the activities that system actors participate in. Figure 4-3 shows the use case diagram for our P2P reputation system.

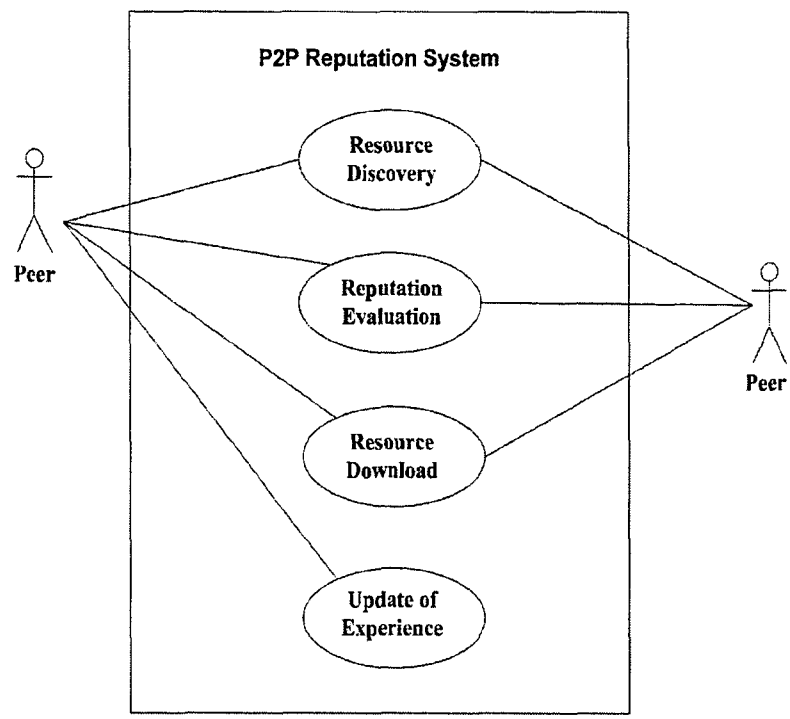

Figure 4-3. P2P reputation system use case diagram

4.1.2.1 Actors

The system has only one kind of actor: Peer.

A peer may act in different roles for different system activities as follows.

Resource Provider and Resource Consumer (or Resource Requestor)

Those roles are played by a peer that provides files (Resource Provider) and a peer that downloads files (Resource Consumer or Resource Requestor). In a P2P system, a peer can act as a resource provider as well as resource consumer, since there is no true distinction between server and client.

Rater, Ratee, and Recommender

There are three different roles a peer plays in reputation-related activities. A peer may rate a file provider whom it just downloads a file from, in this context, the former acts as a Rater, and the latter that provides files will be termed Ratee. A peer may query other peers for their opinions about a target peer's trustworthiness. The peer that responds with opinions is referred to as Recommender.

4.1.2.2 System Scenarios

In this section, we will describe activities of the P2P reputation system. They are related to resource discovery, reputation evaluation, resource download and update of experience (see figure 4-4).

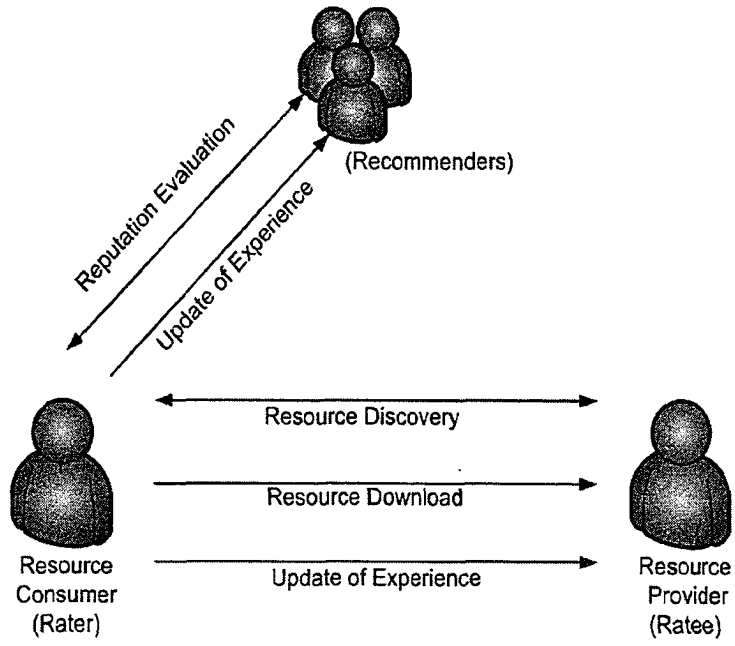

Figure 4-4. P2P reputation system activities

4.1.2.2.1 Resource Discovery

Resource discovery is a standard P2P system activity: a query message is broadcast by a peer to all its neighbors in order to search for a specific resource. When a peer receives a query message which he satisfies; he will respond to this message with the necessary information about the resource and its location (e.g. IP, port number). In our Gnutella-based system, query and response messages follow the *Query* and *QueryHit* messages in the Gnutella protocol (see section 2.1.2 for a review of the Gnutella protocol).

4.1.2.2.2 Reputation Evaluation

Upon receiving the responses to a resource query, a requestor will rank each recommended resource according to the provider's reputation. If the requestor has had a sufficient number of previous interactions with a given resource provider, the reputation computation for that provider can rely completely on the requestor's local reputation repository. Otherwise, a reputation query for that resource provider is issued to the peer's friends. A peer's friends consist of all reputable peers[18] during the peer's past transaction history. The reputation query will also be propagated through friends of friends links, until it meets a dead end (or its TTL has been exhausted). Each peer capable of fulfilling the reputation request will respond to the requestor. The details on collecting and aggregating the received reputation information are going to be discussed in Section 4.1.3.

4.1.2.2.3 Resource Download

After evaluating each resource provider's reputation, the rank of a resource can be obtained via the resource provider's reputation. The rank of a resource indicates some properties related to the resource, such as quality, and estimated time to download this resource. Those properties are valuable and helpful for users to make a resource selection later. When all the resources are ranked, those with positive ranks

[18] Reputable peers are those peers whose reputation values are above a threshold.

will be recommended to the user. The user is able to select an appropriate resource and contact the resource provider directly for a transaction.

4.1.2.2.4 Update of Experience

After the transaction is complete, the resource consumer will be asked to give feedback on this transaction. If the feedback is positive, the reputation for the resource provider is upgraded. Otherwise, if the resource turns out to be malicious, the reputation of resource provider will be downgraded.

The credibility rating, which is an indication of a recommender's reliability, is also updated in this process. If the reputation score calculated for the resource provider was partly or entirely based on the queried reputation, the credibility rating of every peer who contributed to the selection of that resource in the reputation query is updated by the requestor according to the outcome.

4.1.3 Reputation Management

In this section, we describe in detail the maintenance and evaluation of system reputation information, including reputation records, evaluating direct trust, disseminating the reputation query, processing the received recommendations, computing the global reputation value, organizing the results, collecting user feedback and updating the experience. Those functions construct the core process for reputation management. Figure 4-5 illustrates this core process.

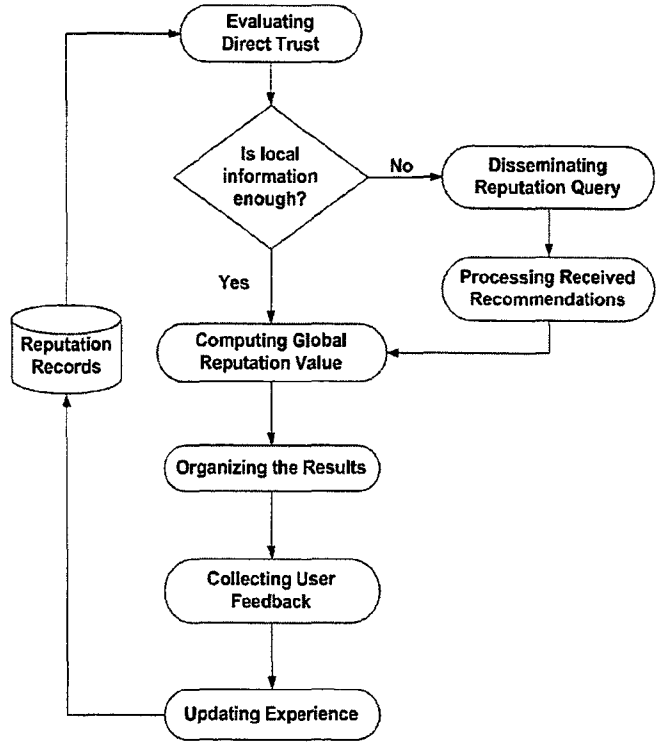

Figure 4-5. Reputation management

4.1.3.1 Reputation Records

In our system, the outcomes of past transactions are stored in reputation vectors, maintained by the peers that make the download. Two types of reputation values are used, and they relate to the following types of trust relationships:

1. Direct reputation value (or trust rating): This value identifies to what degree a peer will trust another peer as a resource provider.
2. Recommender reputation value (or credibility rating): This value is to measure the credibility of a peer as a reputation recommender.

The direct reputation value is stored in the trust vector. This vector is variable-length. For each peer, the trust vector keeps its up-to-date reputation value and the number of past transactions. Recommended reputation values are stored in the credibility vector. Similar to direct reputation, for each peer the credibility vector keeps its up-to-date credibility rating and the number of past recommendations received from that peer.

4.1.3.2 Evaluating Direct Trust

When the responses to a resource query arrive, a reputation score for each file is calculated. The rank of a file is determined by the reputation value of the peer who recommended the file. It has two dimensions: rank by download speed and rank by the quality of the file. We use the notation θ_T to denote the number of transactions to be considered for a peer's reputation calculation. We use the notation T_n for denoting

the number of transactions with a peer. The reputation score for a peer is calculated locally if there is sufficient local reputation information; that is, if $T_n \geq$

θ_T . If there is not sufficient information, then a reputation query about this peer is issued.

4.1.3.3 Disseminating a Reputation Query

As mentioned above, a reputation query is issued when there is not enough local information about the peers who offer files. Reputation queries can be made more efficient by combining all IDs to be queried into a single query message, reducing the number of query and response messages to be handled. The reputation request is sent out through the peer's neighborhood or friends which consist of the reputable peers interacted with before. The set of reputable peers contains those peers whose credibility is (recommendation reputation value) above a threshold and those peers whose direct reputation values are above the threshold. The request message contains the reputation query set and the trust chain other than other necessary elements.

A trust chain is used to compute the trust of the requestor in the recommender and is built during the request dissemination. It contains at intermediate identity I_i a list of intermediate trust value T_j ($1 <= j < i$) which is the credibility of I_{j-1} in I_j with I_0 being the requestor.

Each identity I_j's agent capable of fulfilling the recommendation request (therefore with one or more fitting recommendations in store) is creating a recommendation message that contains the responder's identity, host address, a request identifier, the stored recommendation, the trust chain whose last link is the trust of I_{j-1} in I_j, and the current chain which records the content of trust chain when the reputation query first reaches the current identity.

If the trust chain is still strong enough[19], the request is formed anew and sent on to other potential recommenders again including the own trust in the next recipient (hereby forming the aforementioned trust chain). At some point this dissemination of request stops, either due to the trust chain being too weak, too many hops, too much time passed or no fitting further recipients being available (see figure 4-6, 4-7).

4.1.3.4 Processing Received Recommendations

Upon the arrival of the responses to this query, a recommendation (indirect reputation value) is calculated for each peer. One simple option [10] to determine the resulting indirect reputation value R from a single trust chain (with T_1 denoting the credibility rating of the requestor in identity I_1, T_i is the credibility rating of I_{i-1} in I_i and T_n denoting the trust rating of the recommender on the target peer) is taking the product of all reputation values[20].

---

[19] Reputation systems may set a threshold for the length of trust chain. The trust chain with length below the threshold is considered strong one.

[20] Two possible methods are plausible: additive and multiplicative. The problem with additive weight is that if the chain is "broken" by a highly unreliable link, the effect of that unreliability is local to the immediate agents

$$T = \prod_{i=1}^{n} T_i$$

The same recommendation, which is issued by the same peer on the same target, can be received via completely different paths through the neighborhoods of the intermediary, which leads to different trust chains (see figure 4-6). We will evaluate those trust chains and produce a single reputation value for each received recommendation. Here, we apply the following optimization: we take the best rating in the set of trust chains for the same recommendation as the resulting trust rating with regard to that recommendation.

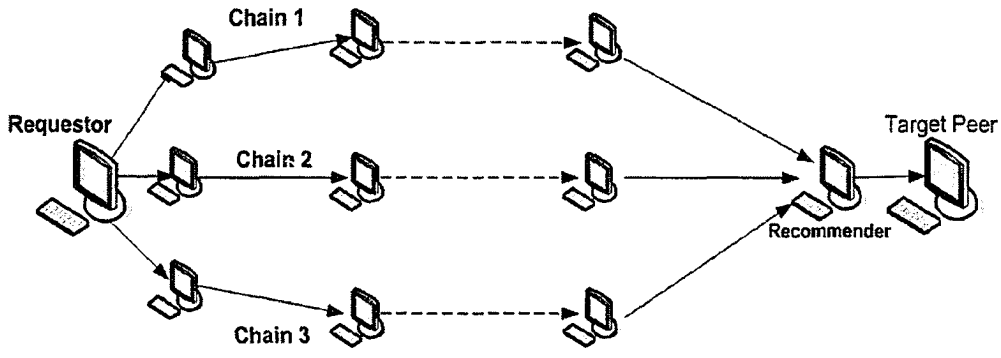

Figure 4-6. Illustration of trust chain for the same recommendation

around it. In a long social chain however, an unreliability chain is certain to cast serious doubt on the reliability of any estimate taken from the chain as a whole. On the other hand, a multiplicative weighting has "long-distance" effect in that an unreliable link affects any estimate based on a path crossing that link [10].

A requestor may have multiple recommendations for a single target (see figure 4-7) and thus the recommendations must be combined to yield a single value. For this moment, we have adopted the average approach, so that we can even out the impact of any single recommendation. The final single trust rating for each target peer is the average of all the distinct recommendations for that peer:

$$\frac{\sum_{i=1}^{n} C_i R_i}{n}.$$

C_i is requestor's credibility rating for recommender i, and R_i is the direct reputation value of the recommender on the target peer.

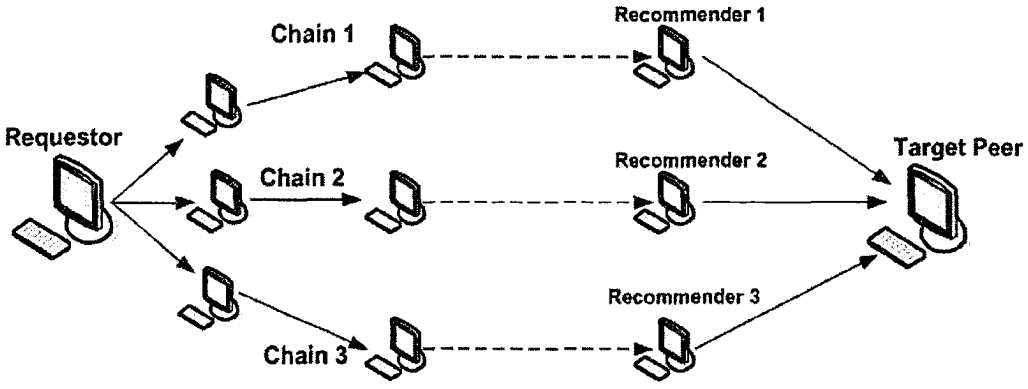

Figure 4-7. Illustration of trust chains for distinct recommendations

4.1.3.5 Computing a Peer's Global Reputation Value

If there are enough local transactions to evaluate a peer's trustworthiness, the peer's global reputation value equals to his local reputation value. Similarly, if we have no previous experience with the peer, we will take the peer's indirect reputation value as

its global reputation value. Otherwise, the global reputation value about a peer is the average of the peer's direct and indirect reputation value. Those files whose recommender's reputation value can not be obtained through the above evaluation process (both direct and indirect one) are ranked *zero*.

4.1.3.6 Organizing the Results

At the end of the evaluation, each file is ranked according to its provider's reputation score. Each file will be assigned a quality rank and a speed rank based on the reputation value of file provider. We filter out those files with either negative quality rank or negative speed rank, or both. Finally, we display the collected files with the quality and download speed rank to users. And it is upon the user to select the peer to download the file from.

4.1.3.7 Collecting User Feedback

After each transaction, a resource consumer is asked to rate this experience. In this system, a rating is made in term of two dimensions: download speed, and quality of file. The download speed discloses the capability of a provider in supporting a high speed download service. The quality of the file indicates the likelihood that a peer provides a file highly related to its category. Those reputation values and their description are given below.

Table 4-1 Reputation value semantics with respect to download speed

Value	Meaning	Description
-1	Unsuccessful	File transferring fails due to interruption or other malicious behavior.
0.2	Low	File download speed is below the normal speed.
0.5	Medium	File is downloaded at a normal speed.
0.75	High	File download speed is faster than the normal one.
1	Very High	File downloading is very fast.

Table 4-2 Reputation value semantics with respect to quality of file

Value	Meaning	Description
-1	Malicious	File is corrupted, empty (0 byte), or other malicious ones.
0.2	Low	File is a little related to its category.
0.5	Medium	File is related to its category in a medium level.
0.75	High	File is highly related to its category.
1	Very High	File is related to its category in very high level.

4.1.3.8 Updating Experience

After collecting feedback, the reputation value for file provider is updated according to the outcome: $T_{new}= (T_{old}* N_{tran} +feedback)/(N_{tran}+1)$

T_{new} is the updated trust rating, T_{old} is the old trust rating, N_{tran} is the total number of transactions. The feedback is the user's evaluation for this transaction. The credibility ratings of related recommenders are updated in a similar way, but it is more complex to evaluate the outcome which in this context refers to the evaluation for this recommendation. The updated reputation or credibility vectors will result in the change in the peer's neighborhood (or friends) as well since the peer's neighborhood consists of those peers whose reputation and credibility above some thresholds.

4.2 Implementation of Reptella

In this section, we will describe the details of the implementation of Reptella. This system implements the solutions that were discussed in section 4.1. This section describes the system architecture, system protocols (including the description of reputation message format), and system interactions (or flow of events).

4.2.1 System Architecture

We have implemented Reptella as an extension to an existing implementation of the Gnutella protocol: Comtella [55, 56]. Comtella system aims to motivate users to participate in P2P networks, therefore, it implements Gnutella protocol as well as some specific functions to achieve system goal. However, our Reptella system reuses and extends only those components related to Gnutella protocol. In this section, we give an overview of the Reptella system architecture with the detailed introduction for each component. We will describe the extensions we made to the original Comtella system in section 4.3.

Figure 4-8 shows the primary subsystems involved in the implementation of Reptella. In this context, we use the terms subsystem, component and manager interchangeably. Viewed from a higher level, the Reptella system contains three subsystems: backend subsystem, UI subsystem and repumod subsystem. The backend and UI subsystem implement Gnutella protocol, and are derived from the Comtella system. We introduce them based on an understanding of the source code,

since no design documents are available currently. The RepuMod subsystem is what we incorporate for reputation management. It implements the reputation mechanism in this Gnutella-based reputation system. Next, we describe each component in further detail.

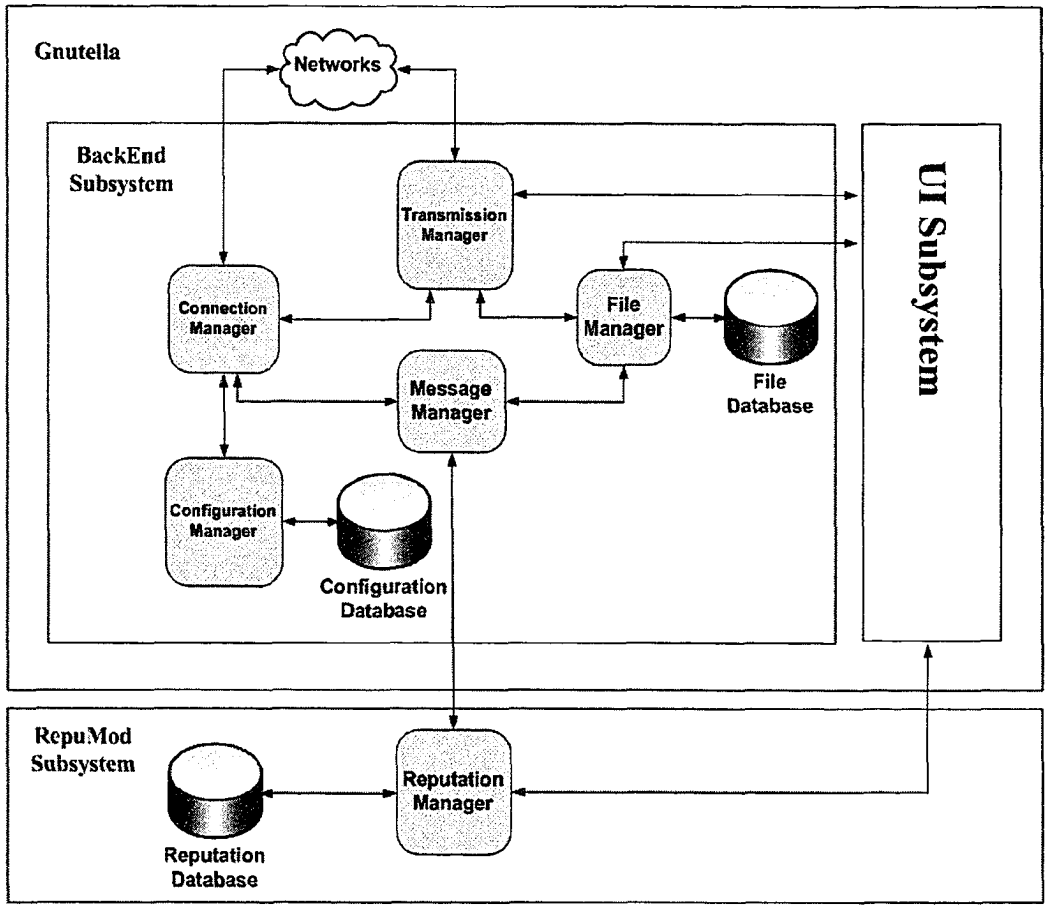

Figure 4-8. System architecture

4.2.1.1 Backend Subsystem

The backend subsystem is the core of Gnutella. It contains message/routing manager, connection manager, file manager, transmission manager and configuration manager.

Message/routing manager

This manager administrates all received and sent Gnutella messages. All messages are stored to be able to route back response messages of other hosts. When a Gnutella message is received from the network, this component parses the message and contacts the file manager if the message is a Query or QueryHit message. Further, it contacts the reputation subsystem to see if a reputation message was received. It is also involved in preparing Gnutella request/response messages based on inputs it receives from the GUI component, file manager and connection manager. Additionally, it builds routing tables for all kinds of messages, e.g. Query message, QueryHit message, reputation-related messages (see 4.2.2.1), etc. The QueryHit message processing involves an entry to reputation subsystem.

Connection Manager

The connection manager handles all connection requests and starts the corresponding workers (Acceptor or ConnectionMaintainer).Successfully established connections are then processed by the corresponding workers (message/routing manager, or transmission manager). Connections are administrated so that at any moment the

system maintains a desirable number of network connections for the peer. It is also responsible for issuing handshake requests and generating ping/pong messages.

File Manager

This component manages all local shared files and is responsible for all incoming Gnutella resource requests. It receives triggered events from the GUI whenever the user shares, unshares or changes a file's property. It is also responsible for indexing all the shared files. When a resource request is received from the network, the message/routing manager contacts the file manager to search for files matching the query content, and if a match is found, the resource manager will inform the message/routing manager to issue a QueryHit message.

Transmission Manager

The transmission manager component monitors all downloads/uploads to/from the local host. It communicates with the connection manager and the file manager to initiate, transfer and terminate downloads/uploads of resources. After the resource searching is done, if user selects a resource to download, a direct connection will be established between the resource requestor and the resource provider and the data transfer is performed over HTTP. Information about transfers, for example, the average transmission speed or status of the transferring, will be displayed with each transmission. Additionally, the upload bandwidth limit is implemented based on the reputation information.

Configuration Manager

This manager manages network-configuration parameters. Those parameters include variables such as standard listening port, connection address, browser, pong-caching version, message extensions version, and so on. Those parameters will be stored locally to be persistent after initialization.

4.2.1.2 UI Subsystem

This is the GUI component which provides data structures for the user interface used in the implementation of Reptella system. This component also triggers events based on inputs provided and executions issued by the user and invokes appropriate actions on the GUI, backend subsystem or repumod subsystem.

4.2.1.3 RepuMod Subsystem

The RepuMod subsystem is the core of Reptella. It includes reputation manager and reputation database.

Reputation Manager

This component implements the reputation-related algorithms and manages reputation messages received from the backend subsystem. It parses reputation

messages and takes appropriate actions. This component is also responsible for forming reputation request/response messages and dispatching them to the network. It communicates with the reputation database to load the reputation information into the run-time environment and to save run-time configuration changes.

Reputation Database

This database stores reputation and credibility value for each peer interacted with before. It also stores the history of the transactions for the owner of the database. It also stores information about the neighbors of a peer. This database is accessed by the reputation subsystem when required.

4.2.2 Protocol Description

This section details the format for system reputation messages, illustrates the corresponding implementation, and finally gives a description about the Reptella protocol based on the message flow.

4.2.2.1 Message Formats

In the Gnutella-based Reptella system, resource request messages are sent in the Gnutella Query message and response messages are sent in the Gnutella QueryHit message. Both messages have been introduced in the background chapter, and their formats are documented in the Gnutella Protocol Specification v0.4 [26]. Here, we only describe the messages particular to system reputation management: RepuQuery Message and Recommendation Message.

Message Description

Message Header

Both messages are preceded by a message header which is the same as a general Gnutella message header. To give a complete understanding about the reputation message formats, we will first introduce the Gnutella message header (see table 4-3).

	Message ID		Payload Descriptor	TTL	Hops	Payload Length	
Byte offset:	0	15	16	17	18	19	22

Table 4-3. Gnutella message header

Message ID: A 16-byte string uniquely identifying the message on the network.

Payload 0x00=Ping

Descriptor: 0x01=Pong

0x40=Push

0x80=Query

0x81=QueryHit

The above are the default payload descriptors in Gnutella Protocol. To identify the reputation messages in our reputation system, we extend the Gnutella payload descriptors with the following reputation-related descriptors:

0x20=RepuQuery

0x21=Recommendation

TTL: Time to Live. The number of the message will be forwarded by Gnutella servents before it is removed from the network. Each servent will decrement the TTL before passing it on to another servent. When the TTL reaches 0, the message will no longer be forwarded.

Hops: The number of times the message has been forwarded. As a message is passed from servent to servent, the TTL and Hops field of the header must satisfy the following condition:

$$TTL(0) = TTL(i) + Hops(i)$$

Where TTL(i) and Hops(i) are the value of the TTL and Hops fields of the header at the message's i-th hop, for $i \geq 0$.

Payload It identifies the length of the message immediately following this header.

Length: Servents should rigorously validate the field for each message received.

Immediately following the message header, is a payload consisting of one of the abovementioned payload descriptors.

RepuQuery (0x20)

	Number of Query	Length of Trust Chain	Query Criteria	Trust Chain	
Byte offset	0	1	2	3	...

Table 4-4. RepuQuery message

Number of Query: The number of queried targets. The target is the peer about whom the requestor asks for an opinion (or recommendation).

Length of Trust Chain: This length indicates how many intermediaries are in the dissemination path.

Query Criteria: A set of search strings. This set contains Number-of-Query elements. This field identifies all the target peers whose reputation information is requested.

***Trust Chain*[21]:** A list of intermediary reputation values. The reputation value R*i* indicates to what degree peer *i-1* trusts peer *i* as a reputation recommender.

Recommendation (0x21)

	Number of Hits	Length of Trust Chain	Length of Current Chain	Port	IP Address
Byte offset	0	1	2	3 4	5 8

	Recommendations	Trust Chain	Current Chain	Client GUID
Byte offset	9 ...	... ...	... ...	n n+16

Table 4-5. Recommendation message

Number of Hits: The number of query hits in the set of recommendations.

Length of Trust Chain: This length indicates how many intermediaries help direct the path to the recommender.

Length of Current Chain: This length indicates, at a specific point during the formation of this recommendation, how many intermediaries have been in the partly built trust chain.

[21] See section 4.1.3.3 and 4.1.3.4.

Port:	The port number on which the responding host can accept incoming connections.
IP Address:	The IP address of the responding host.
Recommendations:	The set of responses to the corresponding reputation query. This set contains Number-of-Hits elements.
Trust Chain:	It is the list of all the trust values which leads to the recommendation. The trust chain is the evaluation for the reliability of the recommendation.
Current Chain:	It is partly formed trust chain for this recommendation, and indicates how many intermediaries have participated in this recommendation at this point.
Client GUID:	A 16-byte string uniquely identifying the responding servent on the network. This is typically some function of the servent's network address [26].

Message Implementation

RepuQuery Message

In RepuQueryMessage Class (figure 4-9), there are three variables: minimumSpeed, query, and trustChain, which correspond to the three RepuQuery message fields: Minimum Speed, Query Criteria, and Trust Chain, respectively.

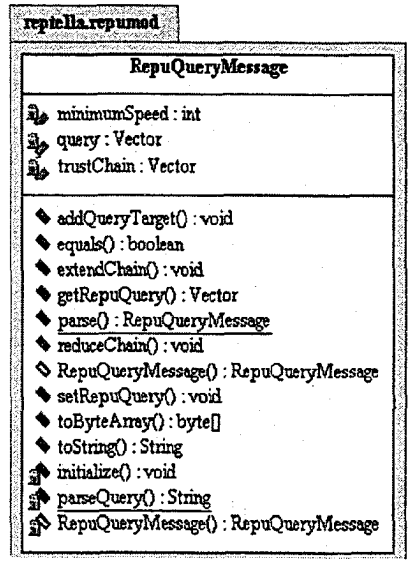

Figure 4-9. RepuQueryMessage class

We do not explicitly identify *Number of Query* and *Length of Trust Chain* as variables in this class; instead, we generate them dynamically whenever we are going to send out a RepuQuery message (figure 4-10). The reason is that the information only functions at the time message receiver parses *RepuQuery* message, and is useless after that (figure 4-11).

```
public byte[] toByteArray(){

                        ... ...

    int i=0;              // Number of Query
    for(int j = 0; j < query.size(); j++)
   {
       byte abyte1[] = ((String)query.get(j)).getBytes();
      if(abyte1 != null)
      {
            bytearrayoutputstream.write(abyte1);
            bytearrayoutputstream.write(new byte[] {
              0
            });
            i++;
      }
   }

                        ... ...

    int k=0;          //Length of Trust Chain
   for(int j = 0; j < trustChain.size(); j++)
  {

    byte abyte2[] = ((String)trustChain.get(j)).getBytes();
    if(abyte2 != null)
    {
            bytearrayoutputstream.write(abyte2);
            bytearrayoutputstream.write(new byte[] {
            0
    });
            k++;
    }
}

            ... ...

    bytearrayoutputstream1.write(new byte[] {
            (byte)i
    });
    bytearrayoutputstream1.write(new byte[] {
             (byte)k
    });

            ... ...
}
```

Figure 4-10. Send out RepuQuery message

```
public static RepuQueryMessage parse(ByteBuffer bytebuffer)
{
                              ... ...

          int j = abyte0[i] & 0xff;       //Number of Query
          int t=abyte0[i+1] & 0xff;       //Length of Trust Chain

                              ... ...

          int k = 0;
          do
          {
           if(k >= j)
             break;
           String q = parseQuery(bytebuffer);
           if(q== null)
             break;
           repuQuerymessage.addQueryTarget(q);
           k++;
          } while(true);


         int y= 0;
         do
         {
         if(y >= t)
           break;
         String re= parseQuery(bytebuffer);
         if(re== null)
           break;
         repuQuerymessage.extendChain(re);
         y++;
         } while(true);

         ... ...
  }
```

Figure 4-11. Parse RepuQuery message

Recommendation Message

In RecommendationMessage Class (figure 4-12), the five variables: clientGUID, currentChain, host, recommendations and trustChain correspond to the five fields: Client GUID, CurrentChain, Port and IP address, Recommendations, and Trust Chain in Recommendation message, respectively.

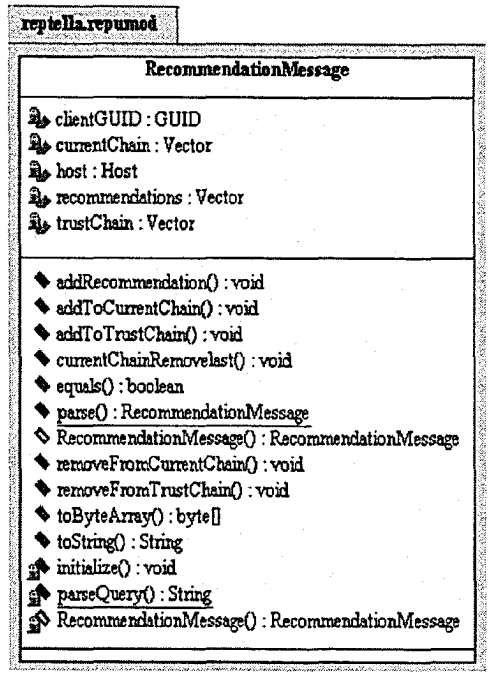

Figure 4-12. RecommendationMessage class

Similar as for the RepuQueryMessage class, we do not explicitly identify *Number of Hits, Length of Trust Chain* and *Length of Current Chain* as variables in this class; instead, we generate them dynamically whenever we are going to send out a *Recommendation* message (figure 4-13). Those data are used to help

Recommendation message receivers parse that message, and will be discarded after parsing the message (figure 4-14).

```
public byte[] toByteArray()
{
                    ... ...

    int i = 0;    //Number of Hits

    for(int j = 0; j < recommendations.size(); j++)
    {
     Tuple tup=(Tuple)recommendations.get(j);
                    ... ...

     {
       bytearrayoutputstream.write(abyte01);
       bytearrayoutputstream.write(new byte[] {
         0
       });
       i++;
     }
    }

    int k=0;    //Length of Trust Chain
    for(int j = 0; j < trustChain.size(); j++)
    {
                    ... ...
                    k++;
     }

    int cur=0;     //Length of Current Chain
    for(int j = 0; j < currentChain.size(); j++)
    {
                    ... ...
                    cur++;
     }
                    ... ..
     bytearrayoutputstream1.write(new byte[] {
       (byte)i
     });
     bytearrayoutputstream1.write(new byte[] {
       (byte)k
     });
     bytearrayoutputstream1.write(new byte[] {
       (byte)cur
     });
                    ... ...
  }
```

Figure 4-13. Send out recommendation message

```
public static RecommendationMessage parse(ByteBuffer bytebuffer)
{
                    ... ...
      int j = abyte0[i] & 0xff; // Number of Hits
      int t=abyte0[i+1] & 0xff;  //Length of Trust Chain
      int cur=abyte0[i+2] & 0xff;  // Length of Current Chain
                    ... ...
      int k = 0;
      do
      {
        if(k >= j)
          break;
       String q = parseQuery(bytebuffer);
                    ... ...
        k++;
      } while(true);

      int y= 0;
      do
      {
        if(y >= t)
          break;
       String re= parseQuery(bytebuffer);
                    ... ...
        y++;
      } while(true);

      int z= 0;
      do
      {
       if(z >= cur)
        break;
       String current= parseQuery(bytebuffer);
                    ... ...
        z++;
      } while(true);
                    ... ...
}
```

Figure 4-14. Parse recommendation message

4.2.2.2 Protocol Description

Each transaction of the Reptella protocol involves three types of entities in the network – a requestor, a set of providers, and a set of recommenders. The exchange of messages between these entities has been described below. For the sake of simplicity of this discussion, figure 4-15 shows only one provider (P) and n recommenders.

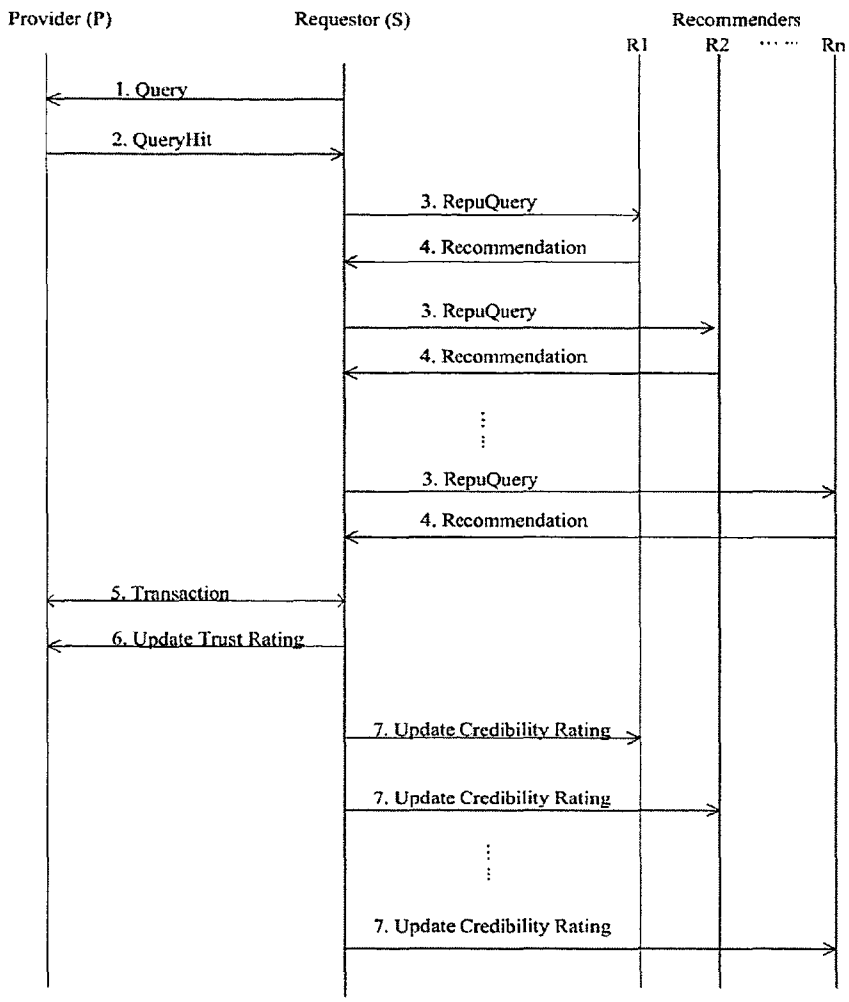

Figure 4-15. Reptella protocol message flow

The requestor (S) who wishes to search for a resource broadcasts a Query message into the network (1). Any providers (P) who are currently sharing the resource respond to this request message by sending the QueryHit message (2). The above steps are the same as the initial steps in the Gnutella protocol. However, in addition, the requestor sends their Reputation Query (3), in the trailer of the QueryHit message. The recommenders who have previous experience with the provider will respond with recommendation (4) about the provider.

The requestor evaluates direct reputation, processes collected recommendations, and computes the global rating for each provider. Accordingly, the providers who have a rating higher than a predefined threshold value are recommended to the requestor. The requestor then selects the appropriate provider and begins the transaction, using the HTTP protocol, just as in Gnutella (5).

At the end of the transaction, the requestor is suggested to rate the service that was provided by the provider. The peers may consider multiple factors such as quality of file, bandwidth provided, etc, while rating the transaction. Based on the rating, the requestor will update the trust rating (6) and credibility rating (7) for the provider and recommenders, respectively.

Figure 4-15 focuses on the illustration of a general message flow in a reputation-based peer-to-peer system, and it does not indicate any trust chain which may be

dynamically constructed during a reputation query. In figure 4-16, we will illustrate in detail one of such trust chains corresponding to one of recommendations in figure 4-15, e.g. the recommendation from R1.

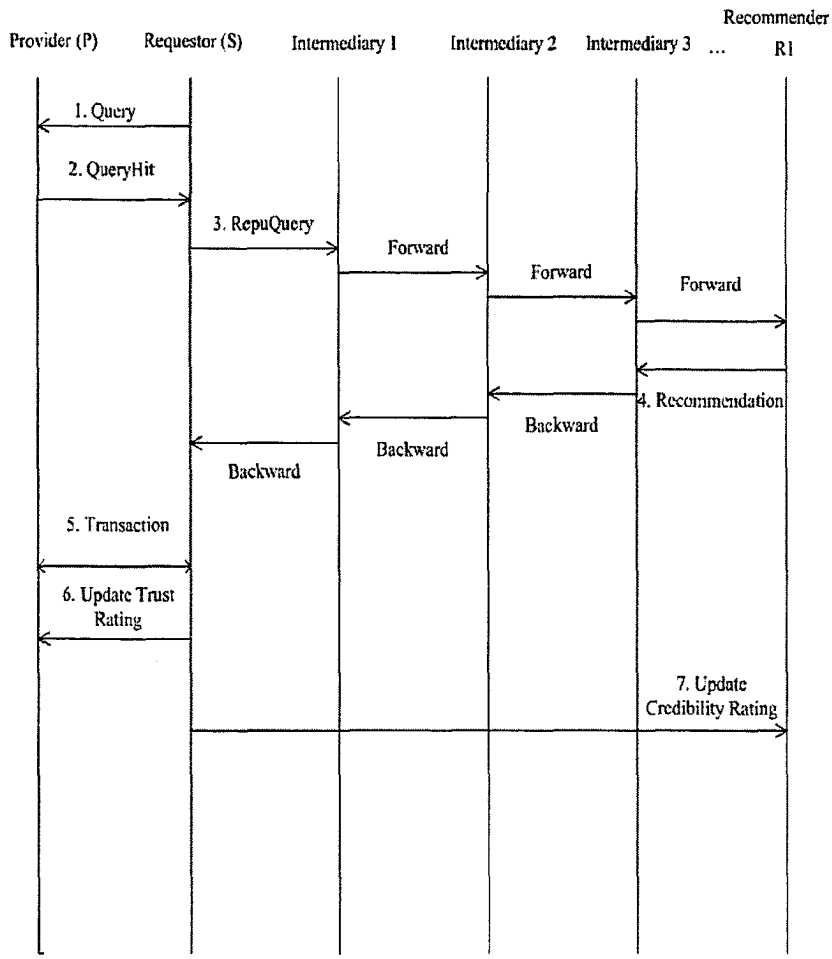

Figure 4-16. A trust chain in protocol message flow

In the above figure, the *Intermediary* denotes a peer who forwards a *RepuQuery* message to his neighbors and at the same time attaches his credibility ratings for

those neighbors to the *RepuQuery* message. The *Backward* indicates the *Recommendation* is sent back to the requestor along the same way the *RepuQuery* came from.

4.2.3 System Interaction

In a P2P file-sharing system, there are two basic system activities: resource discovery and resource download. After incorporating reputation mechanism, both processes are enriched. This section describes the interactions between the components of Reptella in detail. It depicts the invocations made on the Reptella Reputation Subsystem that were added as extension to the existing code base of Gnutella.

On starting up, system will execute a sequence of initialization operations: the connection manager contacts configuration manager for information to configure the network, and sends connection requests as part of the peer discovery process. Furthermore, the connection manager will issue Ping/Pong messages. Figure 4-17 shows the sequence diagram for initializing Reptella. This process follows the Gnutella protocol.

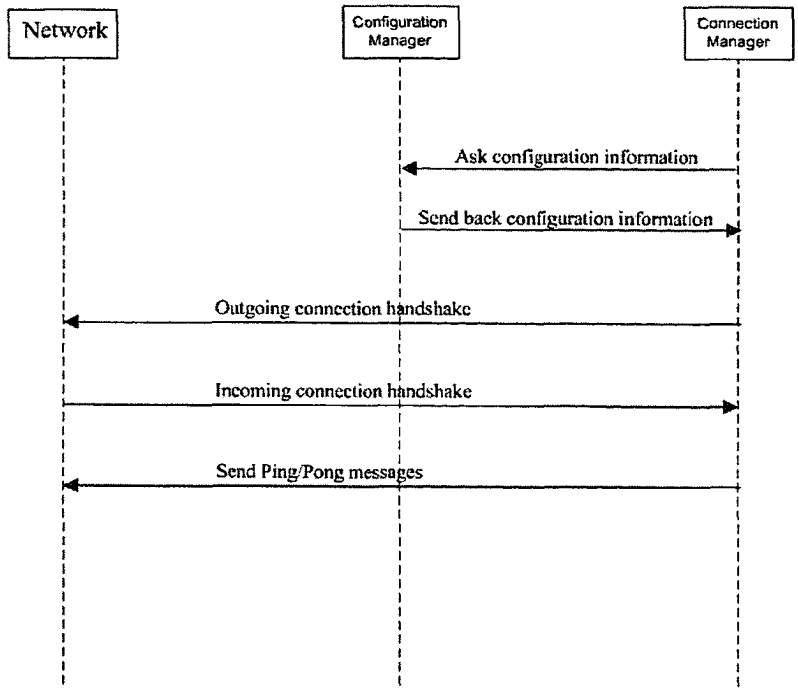

Figure 4-17. The sequence diagram for initializing Reptella

As soon as the initialization is complete, the system is ready for processing any incoming or outgoing requests.

Resource Discovery

Resource discovery in Reptella system involves four components: File Manager, Message/Routing Manager, UI Subsystem, and Reputation Subsystem (see figure 4-18).

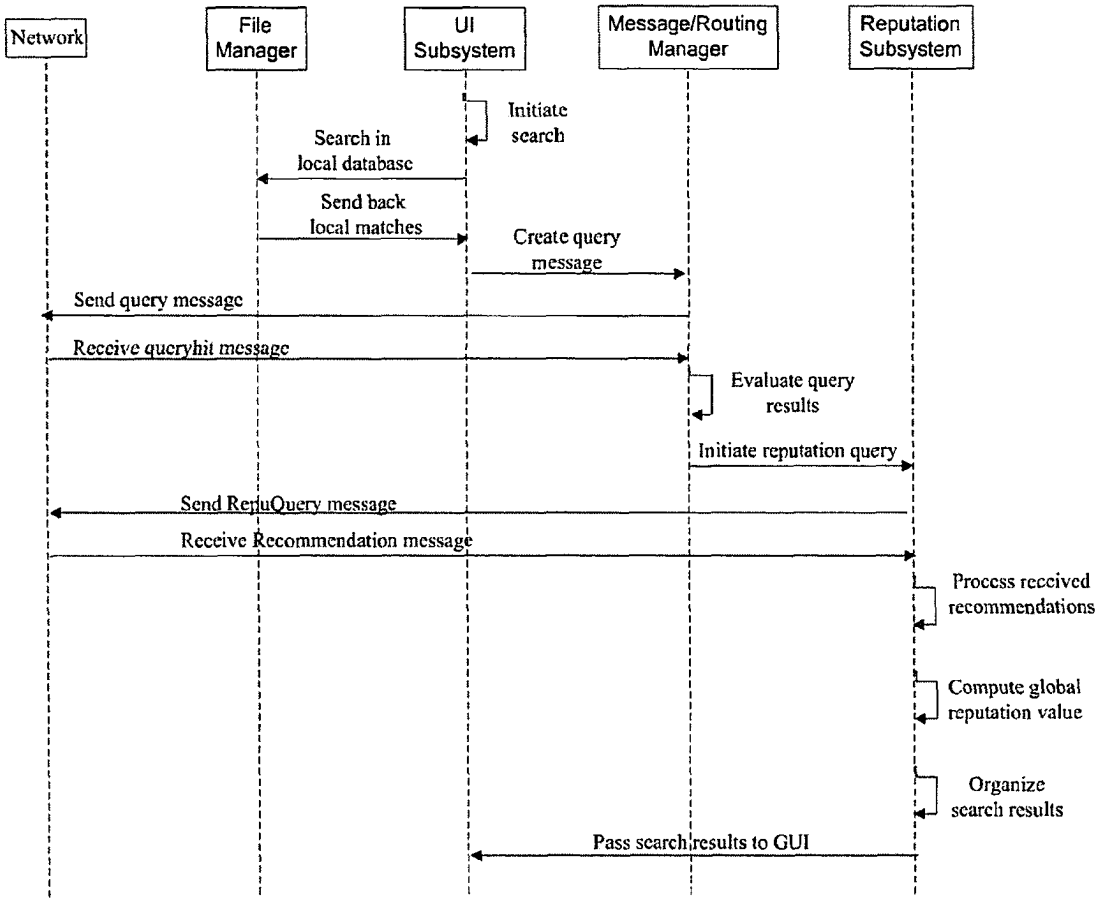

Figure 4-18. The sequence diagram for resource search

When the user wishes to send a search request, he/she selects a search category and clicks the "Search" button to start searching (see figure 4-19).The GUI contacts file manager and makes a call onto the message/routing subsystem simultaneously. The file manager checks if there are local files matching the search request. If there are,

the results will be sent back and displayed on the GUI. At the same time, the message/routing subsystem initiates a search request, which in turn sends a query message into the network. After receiving all the replies to the query, the servent will evaluate the query results, and group those providers with whom it has not had enough previous transactions. Based on the query results and the grouped providers, the servent generates and sends out a reputation query to network through his/her friends (see figure 4-20).

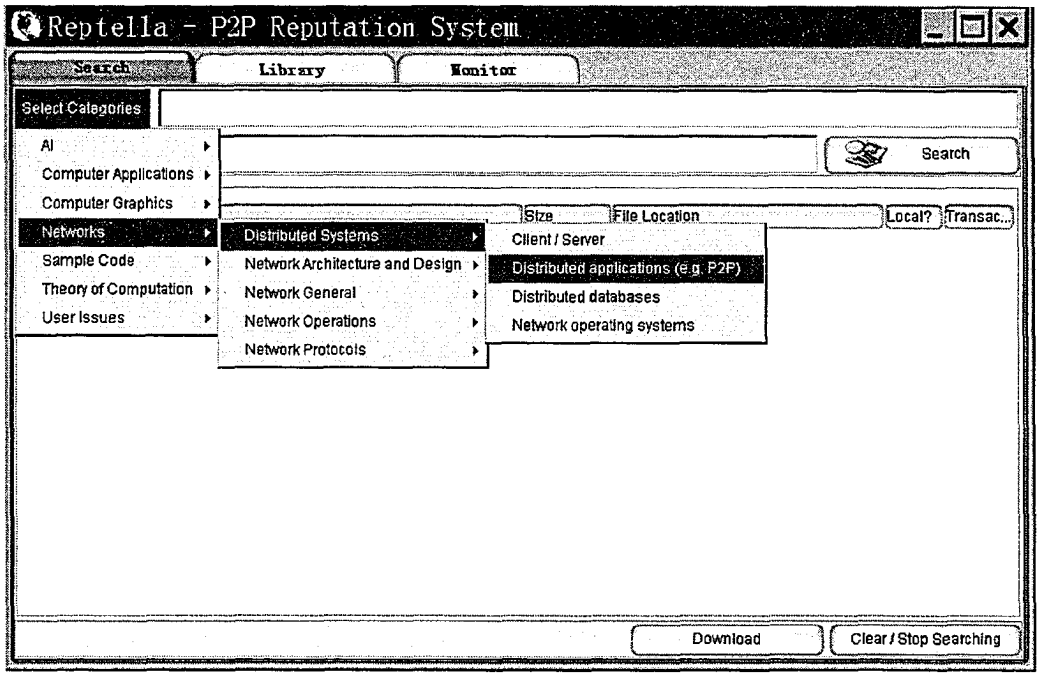

Figure 4-19. Initialize search

In figure 4-20, the requestor first checks the number of transactions that it has with each resource provider in the past. Those peers who have no enough transactions with the requestor will be grouped into a set. If the result set is empty or the requestor

has no neighbors (or friends), the evaluation for each resource provider will be based on local reputation information. Otherwise, a RepuQuery message will be created and sent out through the requestor's neighbors to obtain recommendations from other peers.

```
public synchronized static void evaluateQueryResult(){
                                        ... ...

    Enumeration en=cache.keys();
    for(;en.hasMoreElements();){
       servname=(String)en.nextElement();
        trans=RepuModeling.getInstance().getTransNumberFor(servname);

        if(trans<RepuMod.TRANSACTION_LOW_BOUND)

          repuQuery.add(servname);

    }
  Hashtable agent_values = RepuModeling.getInstance().getNeighbours();
              ... ...
  if((repuQuery.size()==0)||(agent_values.isEmpty())){ //local information is enough.

    // Rank received resource according to local reputation information,
    // and send the results to GUI for display.
    return;
  }

  Enumeration enumeration=agent_values.keys();

  RepuQueryMessage repuQueryMessage=
                    messageFactory.newRepuQueryMessage(repuQuery,trustChain);

  Connections connections=NetworkController.getConnections();
  do {
     if(!enumeration.hasMoreElements())
              break;
     String s5 = (String)enumeration.nextElement();

     if(s5 == null || s5.length() <= 0 || vector.contains(s5))
              continue;
     Enumeration enumeration2 = connections.getConnections();

     do{
                    ... ...
       connection2 = (Connection)enumeration2.nextElement();

                    ... ...

     } while(// connection2 is not in the set of friends);
  String value=(String)agent_values.get(s5);
  RepuQueryMessage rMsg=messageFactory.newRepuQueryMessage(repuQuery,(Vector)trustChain.clone());
  rMsg.extendChain(value);
                                  ... ...
     connection2.send(rMsg);

     } while(true);
}
```

Figure 4-20. Evaluate query results and issue reputation query

After received all the recommendation messages, the servent will first process those recommendations to obtain the indirect reputation value for each resource provider, described in figure 4-21.

```
public synchronized static Vector processRecommendations(){

  Vector result=new Vector();
  Enumeration en=tree.keys();
 ... ...

  double recq;
  double recs;
  for(; en.hasMoreElements();){
   String serv=(String)en.nextElement();
   Hashtable re=(Hashtable)tree.get(serv);
   Enumeration en1=re.keys();

   for(; en1.hasMoreElements();){
    String recommender=(String)en1.nextElement();
    Tuple tup=(Tuple)re.get(recommender);
   ... ...
   }

   RepuMod rm=RepuModeling.getInstance().getRepuMod();
   recq=rm.computeRecQualityRating(quality);
   recs=rm.computeRecSpeedRating(speed);

   if((recq>=0.0d)&&(recs>=0.0d)){
    ... ...
    }
   result.add(j,new Tuple(serv, new Tuple(new Double(recq),new Double(recs))));
     }
 }

  return result;

}
```

Figure 4-21. Process received recommendations

Next, the servent computes the global reputation value for each resource provider, described in Figure 4-22[22].

```
public synchronized static Vector computeGlobalTrust(){

   Vector global=new Vector();
   Vector recs=processRecommendations();

  for (int i=0;i<recs.size();i++){
    Tuple tup=(Tuple)recs.get(i);
    String provider=(String)tup.getFirst();
    Tuple value=(Tuple)tup.getSecond();
    double rq=((Double)value.getFirst()).doubleValue();
    double rs=((Double)value.getSecond()).doubleValue();
    double gq=RepuModeling.getInstance().getRepuMod().computeGlobalQualityRating(provider,rq);
    double gs=RepuModeling.getInstance().getRepuMod().computeGlobalSpeedRating(provider,rs);

    if((gq>=0.2d)&&(gs>=0.2d)){
     Tuple gRating=new Tuple(provider,new Tuple(String.valueOf(gq),String.valueOf(gs)));
     int j;
     for(j=0;j<global.size();j++){
      Tuple temp=(Tuple)global.get(j);
      Tuple value1=(Tuple)temp.getSecond();
      double q=((Double)value1.getFirst()).doubleValue();
      double s=((Double)value1.getSecond()).doubleValue();
      if(q<gq)
        break;
       else if((q==gq)&&(s<gs))
        break;
     }
     global.add(j,gRating);

    }

   }

  return global;

 }
```

Figure 4-22. Compute global reputation value

[22] In the Reptella system, we use multiple-attribute rating to represent user feedback. The lowest positive rating provided by user is 0.2. In our current system design, we discard those resources provided by peers whose reputation values are lower than the lowest positive rating. However, this value (or threshold) may vary in response to different design considerations.

Finally, the servent organizes the search results, ranks each file according to the provider's reputation, and passes them on to the user interface, which is responsible for displaying the results to the user, described in figure 4-23.

```
public synchronized static void organizeResults(){

  Vector recResult=computeGlobalTrust();
  QueryResultCache queryResultCache=StorageController.getQueryResultCache();

  for(int i=0;i<recResult.size();i++){

   Tuple tup=(Tuple)recResult.get(i);
   String serv=(String)tup.getFirst();

   Tuple value=(Tuple)tup.getSecond();
   String q=(String)value.getFirst();
   String s=(String)value.getSecond();

   int trans=RepuModeling.getInstance().getTransNumberFor(serv);

   if(queryResultCache.containsKey(serv)){

     Hashtable files=(Hashtable)queryResultCache.get(serv);
     Enumeration en=files.keys();
     for(; en.hasMoreElements();){
      String filename=(String)en.nextElement();
      Vector fileRecord=(Vector)files.get(filename);
      String uid=(String)fileRecord.remove(0);
      String sp=(String)fileRecord.remove(2);
      String index=(String)fileRecord.remove(2);

      ... ...
      fileRecord.add(0,q);
      fileRecord.add(String.valueOf(trans));
      BackEnd.frameAddNewQueryHit(fileRecord,uid,sp,index);
      queryResultCache.remove(serv);
     }
   }
  }

  if(!queryResultCache.isEmpty()){

        // Sent out those files without indirect reputation value to GUI.
  }
  BackEnd.frameQueryHitEntryFinish();
}
```

Figure 4-23. Organize the search results

The GUI displays the discovered files with corresponding ranks (see figure 4-24). The columns with titles "Quality" and "Speed" identify each file's quality rating and speed rating, respectively[23]. The last column titled with "Transactions" indicates the number of previous direct transactions the servent has with the file provider.

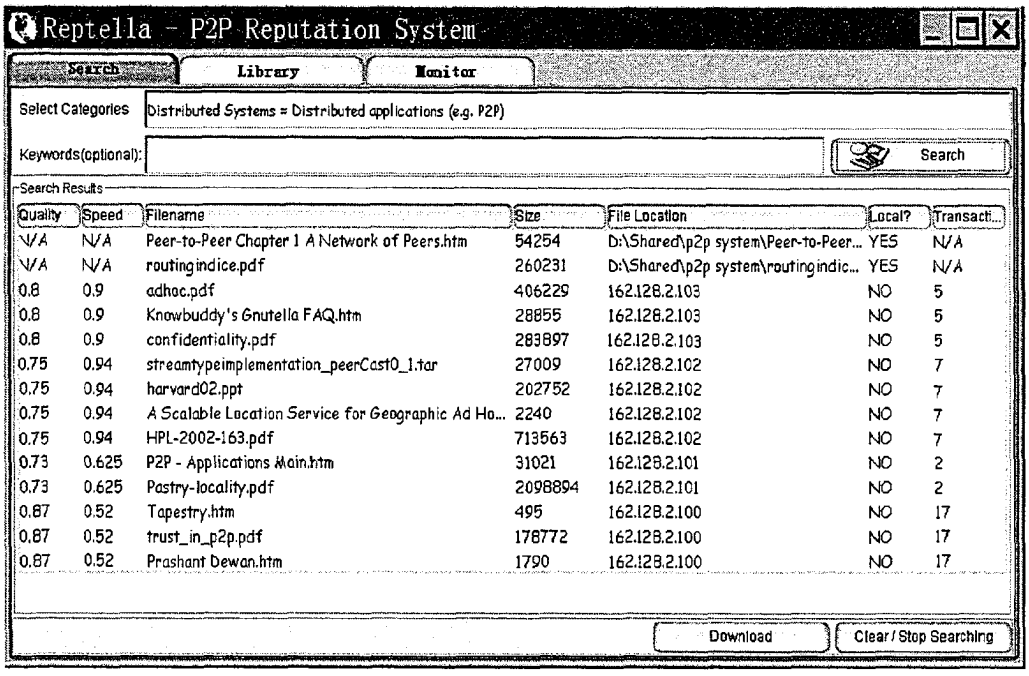

Figure 4-24. The interface showing search results

[23] The evaluation for each file is based on the file provider's reputation; therefore, the quality and speed rating of each file also represent the reputation of file provider in terms of the provider's capability in providing high quality files and high speed downloading service.

Resource Download

Figure 4-25 shows the flow from downloading file to updating corresponding reputation information according to the transaction outcome. After user selects a file and click "download" button, a download request is issued. The UI subsystem contacts the Transmission Manager to initiate the download. The Transmission subsystem establishes an HTTP session with the provider and begins transfer of the file. The file is saved locally. After completion of the download, UI subsystem initiates an interface for user to provide feedback (see figure 4-26). Currently, we make it mandatory for user to rate each transaction. If user does not provide rating, the rating UI can not be closed (see figure 4-27). After receiving user rating, reputation subsystem updates the corresponding trust rating (for provider) and credibility rating (for recommenders) (see figure 4-28). This change will be saved to local reputation database.

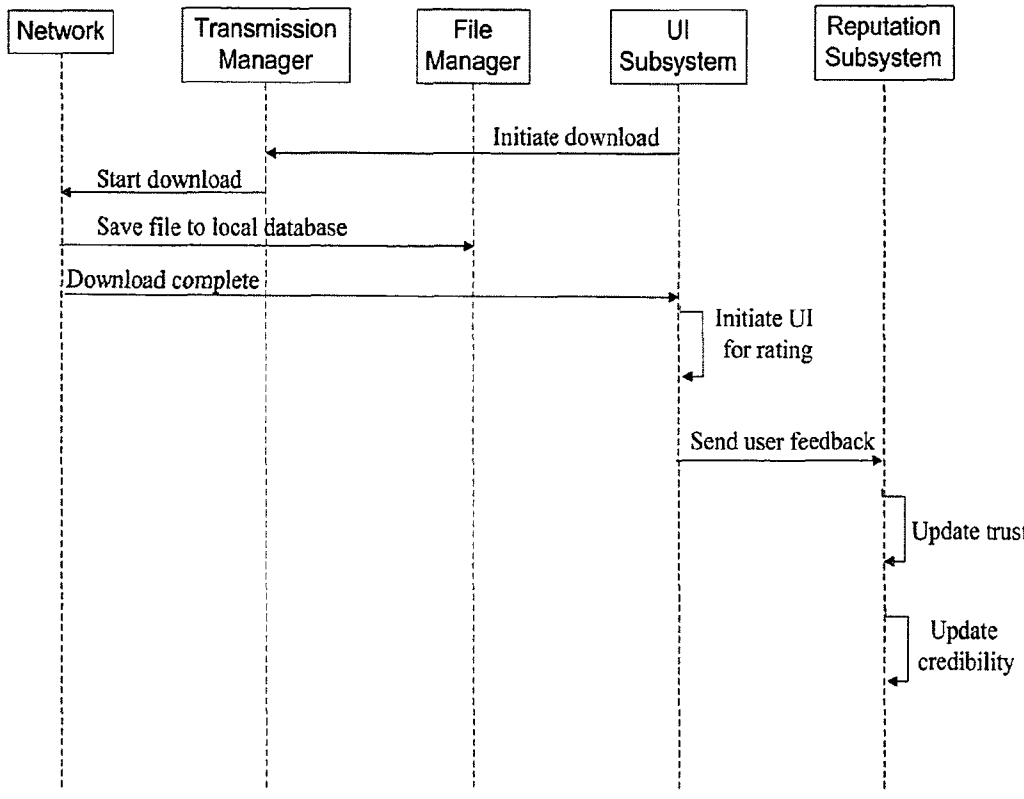

Figure 4-25. Resources download and update of experience

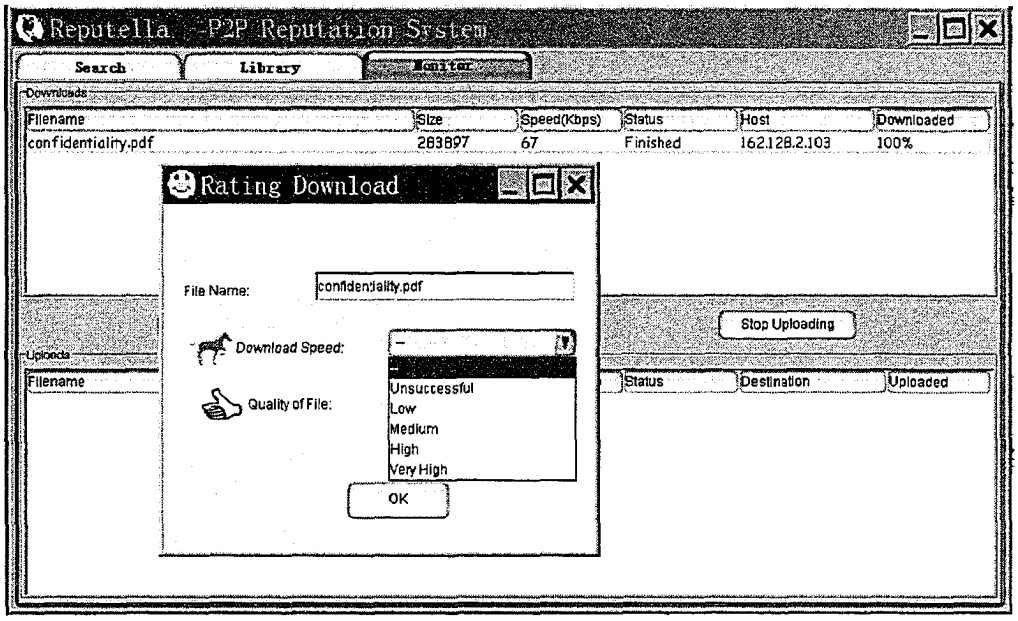

Figure 4-26. The rating UI

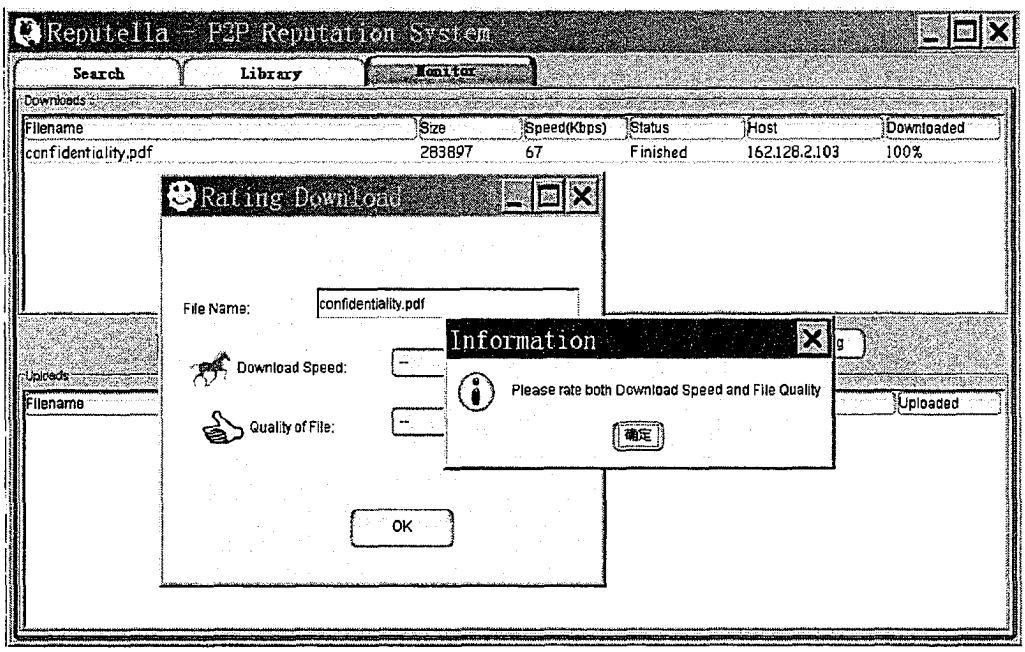

Figure 4-27. The error message for illegal rating

```
void jButton1_actionPerformed(ActionEvent e) {


   double speedRate;
   double qualityRate;
   String speed=(String)jComboBox1.getSelectedItem();
   String quality=(String)jComboBox2.getSelectedItem();
   if((speed.equals("--"))||((!speed.equals("Unsuccessful"))&&(quality.equals("--")))){
      JOptionPane.showMessageDialog(this, "Please rate both Download Speed and File Quality",
"Information", 1);
      return;
   }

   RepuModeling.getInstance().getRepuMod().updateReputation(servName,quality,speed);
   RecommendationCache cache=StorageController.getRecommendationCache();

  if((cache!=null)&&(!cache.isEmpty())){


     Hashtable recommenders=(Hashtable)cache.get(servName);

          ... ...

     if((recommenders!=null)&&(recommenders.isEmpty())){
       Enumeration en=recommenders.keys();
       for(;en.hasMoreElements();){
         String rec=(String)en.nextElement();
         Tuple tup=(Tuple)recommenders.get(rec);
         double q=Double.parseDouble((String)tup.getFirst());
         double s=Double.parseDouble((String)tup.getSecond());

                    ... ...

         RepuModeling.getInstance().getRepuMod().updateCredibility(rec,servName,q,s);

       }
     }
  }
     this.hide();

  }
```

Figure 4-28. Update experience

4.3 Incorporating Reputation System into Existing P2P Network

Based on the design and implementation of our Reptella system, we can describe more generally how to incorporate a reputation system into an existing P2P network. In this section, we describe how an existing P2P network can be extended with a reputation system during system design and system implementation. Our purpose is to relieve developers from the laborious job involved in developing a peer-to-peer reputation system from scratch.

4.3.1 Incorporate Reputation System during System Design

In an original P2P system, the system interaction involves two phases: 1) resource discovery and 2) transaction. A typical system transaction is that a peer queries resource(s) through network, and selects an appropriate resource from the returned resources to start the transaction. After integrating a reputation mechanism into this P2P networks, there will be two more phases: reputation request and update reputation. Correspondingly, a typical reputation-based transaction will be as follows: a peer will query for (a) particular resource(s) using regular P2P protocol queries. It then will receive a number of offers from various peers (providers) within the network, who are willing to provide that resource(s). *Then the requestor peer will query for the reputation values of the providers, and according to the query result,* select a peer to receive the service. After interacting with the chosen provider, *it will then rate the provider based on its performance*

Such incorporation won't change the original P2P messages, but could still invoke them in a new context. There is no interaction between the new messages—reputation messages, and the existing messages (or original messages). Therefore, such incorporation is safe with respect to side effects on the existing messages. Figure 4-29 illustrates this design level transformation.

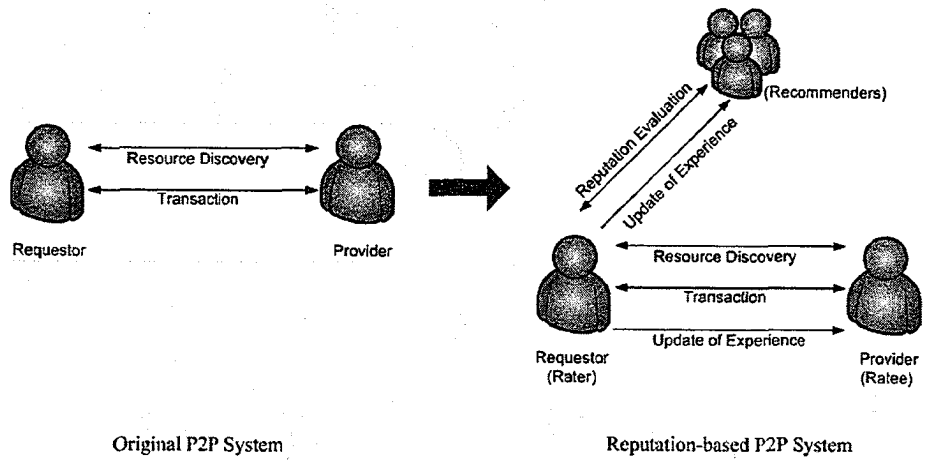

Figure 4-29. Incorporate reputation system during system design

4.3.2 Incorporate Reputation System during System Implementation

In this section, we describe how to implement a reputation mechanism for an existing P2P system. It involves extensions to existing P2P subsystems: backend and UI subsystem[24], as well as the building of a new subsystem: reputation subsystem. In our system, we have encapsulated most of those extensions into the reputation subsystem in order to highlight all the tasks needed for incorporating a reputation system into an existing P2P network. We will identify those encapsulated extensions in the following.

4.3.2.1 Extension to the Backend Subsystem

Most extensions related to the backend subsystem are done for the message/routing component. Those extensions can be classified as function-level or class-level. Function-level extensions mean adding more functions to an existing class, and class-level extensions mean generating new classes extending the current component. The former is trivial, and relies on the specifics of implementation. Thus, we only introduce class-level extensions, including

1. The extensions of messages and message handlers, which have been encapsulated into the reputation subsystem;
2. The generation of helper classes which facilitate the reputation-based resource search.

[24] Generally, we can group the subsystems or components of a P2P system into backend subsystem and UI subsystem. The backend subsystem supports the fundamental functions of a P2P system, e.g. sharing storage and communication among peers [57]. The UI subsystem provides a friendly graphic interface for user to communicate with the system. We can always recognize such components in P2P systems.

We will exemplify them through our Gnutella-based reputation system.

Extend Messages

The Gnutella protocol has five kinds of messages: *Ping, Pong, Query, QueryHit*, and *Push*. We have introduced them in background chapter. In the original implementation of Gnutella protocol[25], those messages have been implemented as the subclass of class *Message*, described in figure 4-30. After incorporating the reputation system, the set of messages will include two more members: RepuQueryMessage and RecommendationMessage. The reputation-enriched set of messages is shown in figure 4-31.

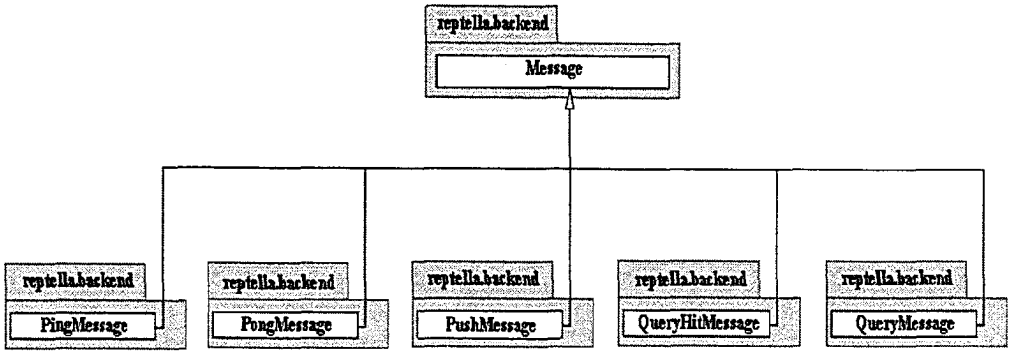

Figure 4-30. The set of messages in original Gnutella system

[25] The one achieved by Comtella, including backend subsystem and UI subsystem.

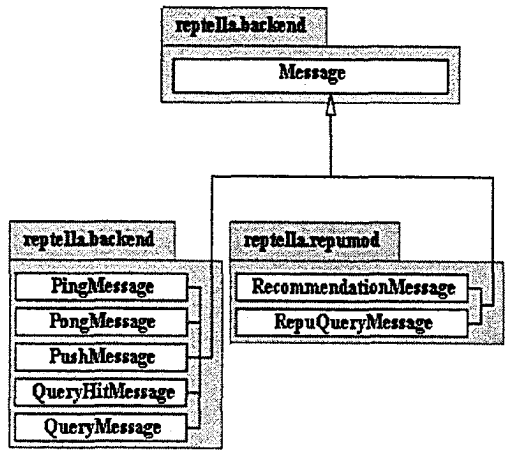

Figure 4-31. The set of messages in reputation enriched Gnutella system

Extend Message Handlers

Message handlers are responsible for dealing with incoming messages. There are five kinds of message handlers in the original implementation of Gnutella protocol[25]: PingHandler, PongHandler, PushHandler, QueryHandler, QueryHitHandler (see figure 4-32), corresponding to the five Gnutella messages respectively. Similar to the set of messages, after incorporating the reputation mechanism, the set of message handlers will also be extended as shown in figure 4-33.

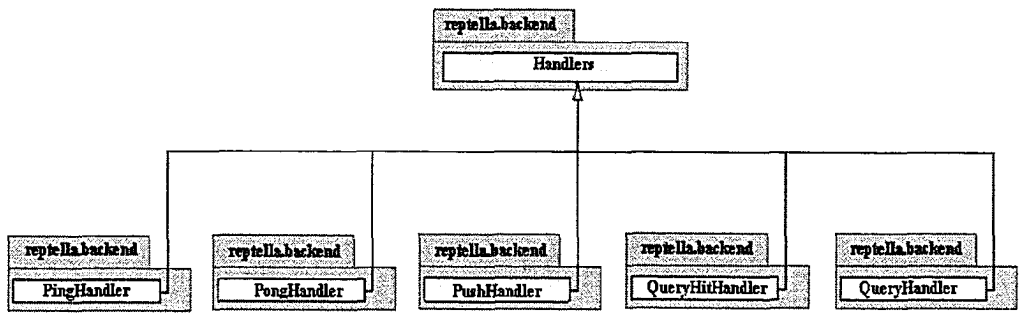

Figure 4-32. The set of message handlers for original Gnutella system

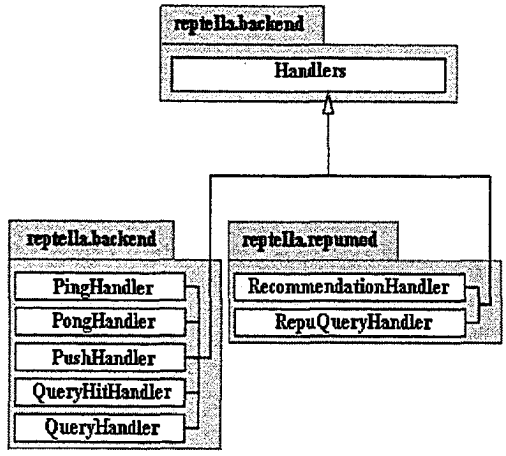

Figure 4-33. The set of message handlers for reputation enriched Gnutella system

Helper Classes

Two helper Classes have been added to the message/routing component, which direct the resource search into reputation subsystem at the end of QueryHit message processing: one is *QueryResultCache* (see figure 4-34), and the other *QueryTimerThread* (see figure 4-35). The *QueryResultCache* class caches the query result and passes the final set of query result to reputation subsystem for the initiation of reputation-based message communications. The *QueryTimerThread* controls the time for a resource query, and triggers opening the entry for the corresponding reputation query as soon as the resource query times out.

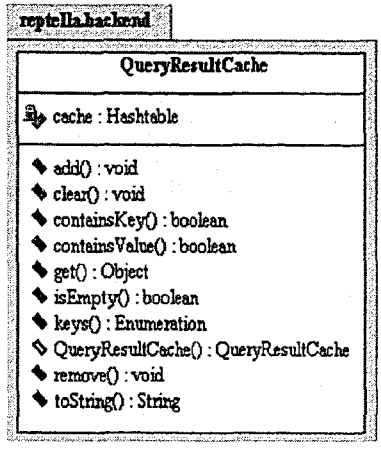

Figure 4-34. QueryResultCache class

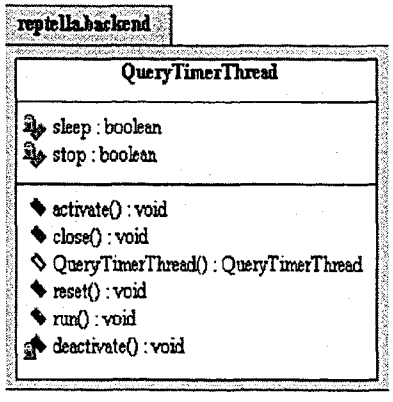

Figure 4-35. QueryTimerThread class

4.3.2.2 The Extension for UI Subsystem

The extensions for UI subsystem include:

1. Modify the user interface so that the reputation information related to each resource can be shown to the user,

2. Create a new user interface for the user to provide feedback after each transaction.

The first extension is the function level, we won't discuss it here. For the second extension, we suggest to implement an independent RatingUI class, so that we can easily redesign the feedback UI in the future without modifying the main user interface. Figure 4-36 is our RatingUI class. We encapsulate this class into reputation subsystem.

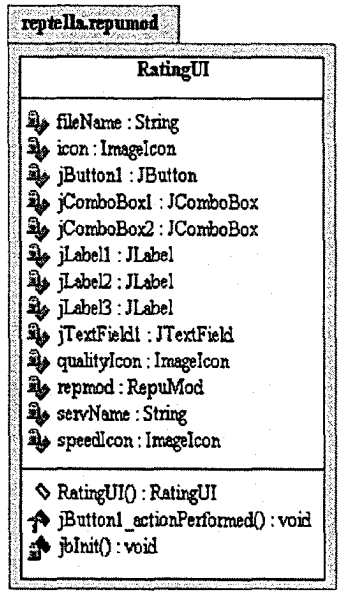

Figure 4-36. RatingUI class

4.3.2.3 Building the Reputation Subsystem

System reputation management is achieved through the reputation subsystem. It is suggested to encapsulate all reputation relevant functions, including:

- represent reputation value;
- compute direct, indirect and global reputation values;
- update reputation information, e.g. trust rating and credibility rating;
- set neighborhood for the dissemination of reputation query;
- maintain reputation database;
- reward peers based on reputation;

into one component, this may be one class or one core class supported by several helper classes. Obviously, such component is the core of reputation subsystem. An advantage is that the redesigning of reputation metrics will only involve one or several classes-- the core of reputation subsystem. The core component communicates with the backend subsystem and UI subsystem, as well as the reputation database. In our reputation subsystem (called RepuMod subsystem), we use a class named RepuMod to act as the system core. We describe the relationship between the RepuMod subsystem and backend subsystem as well as UI subsystem in figure 4-37. Due to space limitations, we only give the complete definition for the RepuMod class. The backend and UI components are represented as class-like entities in this model.

In figure 4-37, Class *Credibility* and *Reputation* represent the objects stored in reputation database. Class *RepuModeling* acts as a channel through which *RepuQueryHandler* or *RecommendationHandler* can callback the functions in RepuMod class. Class *RecListenerEntry* caches and processes the received

recommendation, computes the global reputation value with the aid of the RepuMod class, organizes the search results, and passes the ranked search results to the user interface (see figure 4-38).

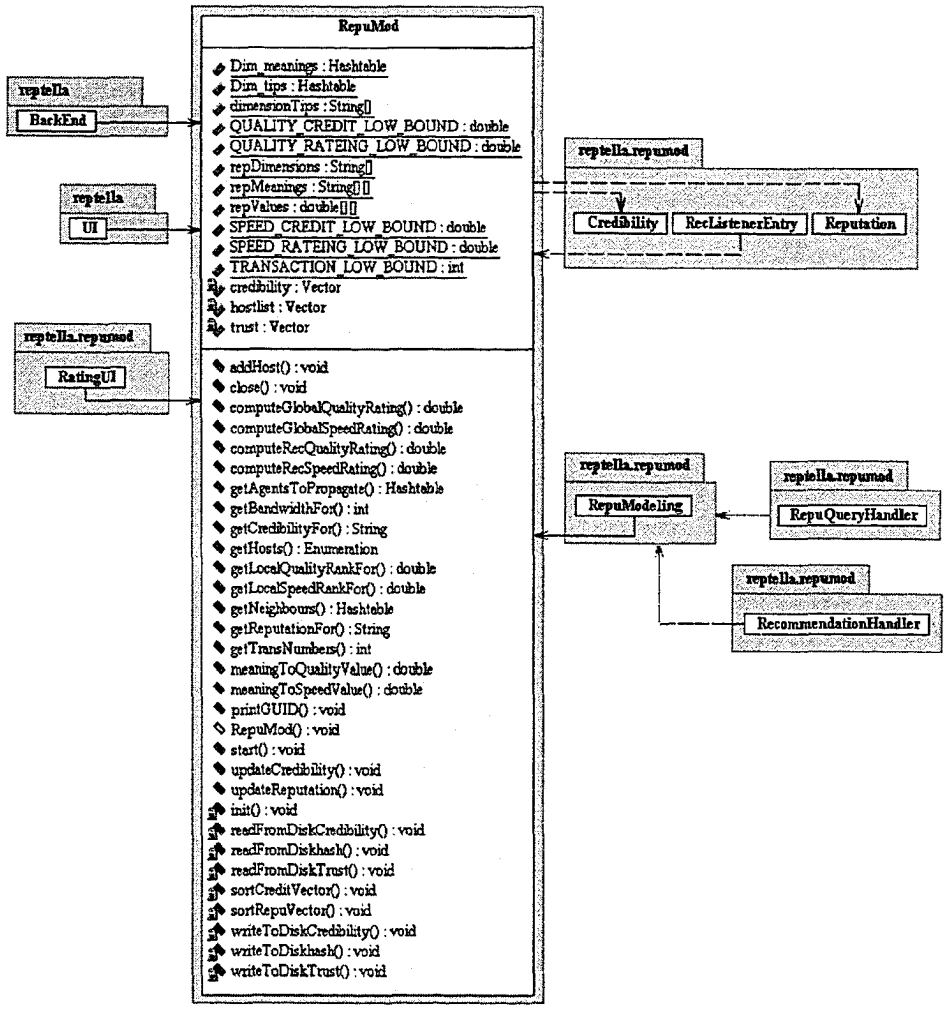

Figure 4-37. RepMod-centered class diagram

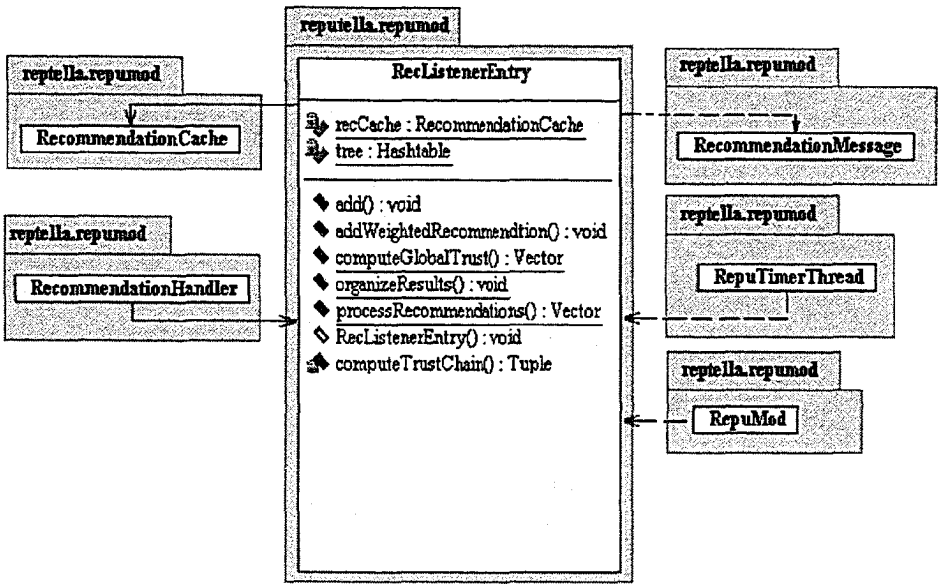

Figure 4-38. RecListenerEntry-centered class diagram

Chapter 5 Conclusions and Future Works

5.1 Conclusions

In this thesis, we investigate the core mechanism to build a decentralized P2P reputation system. We present a taxonomy of the decentralized peer-to-peer reputation systems, and develop a reputation system: Reptella which takes advantage of this taxonomy in our case study. This work identifies and analyzes different approaches to build reputation systems in peer-to-peer networks. Through the design and implementation of Reptella system, we disclose the general reputation management process and describe in detail some important issues we encountered in reputation management, e.g. trust chain.

In the Reptella system, we extend the Gnutella protocol with reputation-related messages to help peers obtain the information related to the rank of resources, and that information will benefit peers in choosing reliable resources in order to make effective transactions later. We describe the system components and highlight the data flow between these components using sequence diagrams. Finally, we present a detailed analysis on how to incorporate reputation system into an existing P2P system. To summarize, through our research, we provide an overview of decentralized P2P reputation systems, and guide future development in this research area.

5.2 Future Works

Our work can be extended to detect and prevent deception in a P2P reputation system. Presently, if the initiator provides a malicious rating at the end of the transaction (e.g. rates a good transaction with a bad rating), there is no mechanism to prevent or punish such misbehavior. Thus, further research could address this issue. Another direction of research is to attempt to extend the taxonomy to include other types of P2P networks, including centralized, and hybrid. Additionally, it is desired to incorporate more mechanisms, for example, security, identity, and incentive mechanisms, etc, in future to enhance the current reputation system.

Bibliography

[1] R.Axelrod. The Evolution of Cooperation. Basic Books, New York, 1984.

[2] Alfarez Abdul-Rahman and Stephen Hailes. A Distributed Trust Model.

In Proceedings of the 1997 New Security Paradigms Workshop, pages 48–60,

September 1997.

[3] Alfarez Abdul-Rahman and Stephen Hailes. Support Trust in Virtual Communities. *In Proceedings of the Hawai'I International Conference on System Sciences,* Maui, Hawaii, Jan 4-7 2000.

[4] A. Jøsang and R. Ismail. The Beta Reputation System. *In Proceedings of the 15th Bled Electronic Commerce Conference, Bled, Slovenia,* June 2002.

[5] Gupta, M., Judge, P., and Ammar, M. A Reputation System for Peer-to-Peer Networks. *In ACM 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video* (2003).

[6] Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. The Eigentrust Algorithm for Reputation Management in P2P Networks. *In Proceedings of the Twelfth International World Wide Web Conference* (2003).

[7] Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. EigenRep: Reputation Management in P2P Networks. In *Proceedings of the Twelfth International World Wide Web Conference* (2003).

[8] M. Kinateder and K. Rothermel. Architecture and Algorithms for a Distributed Reputation System. *In Proc. of the First International Conference on Trust*

Management, ser. LNCS, P. Nixon and S. Terzis, Eds., no. 2692. Crete, Greece: Springer-Verlag, May 2003, pp. 1–16.

[9] M. Kinateder and S. Pearson. A Privacy-Enhanced Peer-to-Peer Reputation System. *In Proc. of the 4th International Conference on Electronic Commerce and Web Technologies (EC-Web 2003), ser. LNCS, K. Bauknecht, A. M. Tjoa, and G. Quirchmayr, Eds., no. 2738. Prague, Czech Republic: Springer-Verlag,* Sept. 2003, pp. 206–215.

[10] L. Mui, M. Mohtashemi, C. Ang, P. Szolovits, A. Halberstadt (2001). Bayesian Ratings in Distributed Systems: Theories, Models, and Simulations. *MIT LCS Memorandum.*

[11] L. Mui, M. Mohtashemi, A. Halberstadt (2002). A Computational Model of Trust and Reputation. *Proc. 34th Hawaii International Conference on System Sciences.*

[12] K.A. Burton. Design of the OpenPrivacy Distributed Reputation System. http://www.peerfear.org/papers/openprivacy-reputation.pdf, May, 2002.

[13] K. Aberer and Z. Despotovic. Managing Trust in a Peer-2-Peer Information System. *In Proceedings of the 10th International Conference on Information and Knowledge Management (ACM CIKM),* New York, USA, 2001.

[14] Cornelli, F., Damiani, E., and Capitani, S. D. Choosing Reputable Servents in a P2P Network. *In Proc. of the Eleventh International World Wide Web Conference* (2002).

[15] L. Xiong and L. Liu. Building Trust in Decentralized Peer-to-Peer Communities. In *International Conference on Electronic Commerce Research (ICECR-5)*, Oct. 2002.

[16] Guillaume Pierre, Maarten van Steen. A Trust Model for Peer-to-Peer Content Distribution Networks. Draft Paper. Nov. 2001.

[17] R.Chen and W.Yeager. Poblano: A Distributed Trust Model for Peer-to-Peer Networks.*http://www.jxta.org/project/www/docs/trust.pdf*, Sun Microsystems, 2002.

[18] Ooi, B. C., Liau, C., and K.L.Tan. Managing Trust in Peer-to-Peer Systems Using Reputation-Based Techniques. In *International Conference on Web Age Information Management* (August 2003).

[19] Selcuk, A.A. Uzun, E. and Pariente, M.R. A. Reputation-based Trust Management System for P2P Networks. *Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium.* April, 2004.

[20] Damiani, E., Di Vimercati, D. C., Paraboschi, S., Samarati, P., and Violante, F. A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. *In Proceedings of the 9th ACMconference on Computer and communications security (2002), ACM Press, pp. 207–216.*

[21] B. Yu and M. P. Singh. A Social Mechanism of Reputation Management in Electronic Communities. *In Proceedings of Fourth International Workshop on Cooperative Information Agents, pages 154-165*, 2000.

[22] B. Yu and M. P. Singh. An Evidential Model of Distributed Reputation Management. *In Proceedings of First International Joint Conference on Autonomous Agents and Multiagent Systems, pages 294-301*, 2002.

[23] L. Mui, M. Mohtashemi, and A. Halberstadt. Notions of Reputation in Multi-Agents Systems: a Review. *In Proceedings of First International Joint Conference on Autonomous Agents and Multiagent Systems, pages 280-287,* 2002.

[24] Wang Y., Vassileva J. Trust and Reputation Model in Peer-to-Peer Networks. *In Proceedings of the Third IEEE International Conference on Peer-to-Peer Computing.* September 1-3, 2003, Sweden.

[25] *James Howison.* An Introduction to the Literature on Online Reputation Systems for the MMAPPS Project. http://www.mmapps.org/private/material/papers/RepLitIntro /sources. Last accessed August 18, 2003.

[26] Gnutella: The Gnutella Protocol Specification v0.4, Document Revision 1.2, Clip2, http://www.clip2.com, protocols@clip2.com.

[27] Napster: http://www.napster.com.

[28] Kazaa: http://www.kazaa.com/.

[29] The freenet project. http://freenet.sourceforge.net.

[30] Gnutella website. http://gnutella.wego.com.

[31] Morpheus website. http://www.morpheus.com.

[32] The Gnutella Developer home page, http://gnutelladev.wego.com/.

[33] G. Zacharia. Collaborative Reputation Mechanisms in Electronic Marketplaces. *In proceedings of the 32nd Hawaii International Conference on System Sciences*, 1999.

[34] Stephen Marsh. Formalising Trust as a Computational Concept. Ph.D Thesis, University of Stirling, 1994.

[35] Peter Lucas and Linda van der Gaag. Principles of Expert Systems. Addison-Wesley, 1991.

[36] D. Dutta, A. Goel, R. Govindan, and H. Zhang. The Design of a Distributed Rating Scheme for Peer-to-Peer Systems. *Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[37] Michael Gertz, Tamer Ozsu, Gunter Saake, Kai-Uwe Sattler. Data Quality on the Web. *In Dagstuhl Seminar report, Schloss Dagstuhl, 31.8.-5.9.2003,* 2003.

[38] Abdul-Rahman A. and Hailes S. A Distributed Trust Model. *In Proceedings of the 1997 New Security Paradigms Workshop, 48-60.* ACM, 1997.

[39] Azzedin F. and Maheswaran M. Evolving and Managing Trust in Grid Computing Systems. *IEEE Canadian Conference on Electrical & Computer Engineering (CCECE '02)*, May 2002.

[40] Cornelli F. and Damiani E. Implementing a Reputation-Aware Gnutella Servent. *In Proceedings of the International Workshop on Peer-to-Peer Computing*, Pisa, Italy, May 24, 2002.

[41] Webopedia. http://www.webopedia.com/TERM/p/peering.html.

[42] Ispa. http://www.ispa.org.za/faq_iep.htm.

[43] Website. http://simonwoodside.com/projects/ict/ixp.html.

[44] Joe's BGP Page. http://joe.lindsay.net/bgp.html.

[45] Grand Central. http://www.grandcentral.com/.

[46] Website. http://www.disappearing-inc.com/P/pseudonymity.html.

[47] V.Shmatikov and C. Talcott. Reputation-based Trust Management. *In Workshop on Issues in the Theory of Security (WITS)*,2003.

[48] OpenP2P. http://www.openp2p.com/pub/a/p2p/2001/12/14/topologies_one.html.

[49] Distributed hash table. http://www.infoanarchy.org/wiki/wiki.pl?Distributed_Hash_Table.

[50] http://lists.gnu.org/archive/html/gzz-commits/2003-02/msg00261.html.

[51] Glenn Shafer website. http://www.glennshafer.com/books/amte.html.

[52] Web of Trust. http://www.rubin.ch/pgp/weboftrust.en.html.

[53] Bayesian Probability Theory. http://www.cis.hut.fi/harri/thesis/valpola_thesis/node12.html.

[54] Dempster-Shafer theory. http://www.ife.ee.ethz.ch/~stiefmei/ ws1/ ws1_0405/Ex07/article48.pdf.

[55] Comtella project website. http://bistrica.usask.ca/madmuc/comtella.htm.

[56] Bretzke H., Vassileva J. (2003). Motivating Cooperation in Peer to Peer Networks, *Proceedings User Modeling UM03, Johnstown, PA, June 22-26, Springer Verlag LNCS 2702*, 2003, 218-227.

[57] L. Jean Camp. Peer to Peer Systems. *The Internet Encyclopedia* ed. Hossein Bidgoli, John Wiley & Sons (Hoboken, New Jersey) 2003.