Trust and Reputation Algorithms for Hierarchically Structured Peer-to-Peer Systems

by

Kalonji Kalala

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.Sc. degree in
Master of Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

This research focuses on the redesign of trust and reputation algorithms in the context of hierarchically structured Peer-to-Peer (P2P) networks with Chord, a scalable P2P lookup service for Internet applications. Chord, which is an open source project, is an overlay network based on a distributed hash table(DHT), and all peers in Chord are arranged around a circle.

In this work, we propose four adapted trust and reputation algorithms for hierarchically structured P2P networks: EigenTrust, PowerTrust, Absolute Trust and NodeRanking. EigenTrust is one of the most well-known trust and reputation algorithms, as well as the most simple. To calculate the reputation, EigenTrust needs to normalize trust and rely on pre-trusted peers. Like EigenTrust, PowerTrust relies on feedback and the use of a distributed ranking mechanism. It chooses a limited number of power nodes with a high reputation. By combining a random walk strategy and the power nodes, it improves accuracy of global reputation. AbsoluteTrust does not require normalization of trust, pre-trusted peers or any centralized authority. Weighted average combined with feedback from peers is employed to determine trust. NodeRanking relies on both individual reputation and social relationship to compute the trust value. NodeRanking evaluates reputation using local information. A node's reputation value can be readily determined by the number of references from other nodes in the network. These adapted algorithms are capable of handling a huge number of nodes disseminated in different rings, which improves complexity and reduces the number of malicious nodes in a hierarchical context.Furthermore, we describe the components of the hierarchical model architecture and present and discuss the results from the experiments. These experiments are employed to verify and compare reduction of downloads from malicious peers, load distribution and residual curl in flat structured networks and in hierarchically structured networks.

Acknowledgements

First, I would like to thank my supervisor Dr. Iluju Kiringa for agreeing to take me on as a student and for waking the researcher inside of me. It has been a great pleasure and honor working with this eminent researcher in his domain. He not only helped me understand research methodologies, but also provided me with the useful sense of direction required for my work. This research has granted me a deep understanding of trust and reputation in hierarchically structured peer-to-peer networks. I am also grateful for the support that I have received and the facilities that I have had access to at the School of Electrical Engineering and Computer Science.

I would like to acknowledge Tao Feng for her tremendous contribution in coding all algorithms designed in this research.

I would also like to extend my thanks to Professors Dr. Mbuyi Mukendi, Dr. Mubenga Kampotu, Dr. Manya Ndjadi and Dr. Kasengedia Motumbe for their support and help in all possible forms over the past ten years.

Finally, I would like to give my special thanks to my parents, brothers, sisters, and friends for their endless love, support, encouragement and blessings throughout all my years of study.

Dedication

I would like to dedicate this thesis to my lovely wife, Lusamba Kalonji Falone, and my children, Kalonji Kalala Ashtar, Kalonji Musenga Zozer and Kalonji Kabengele Uriel. Also, to my father, Kalala Bisonga, my mother, Kapinga Santu, my aunt Ngalula Muyaya, and my uncle Kabengele Tshimpanga.

Nomenclature

BATON	BAlanced Tree structure Overlay on a peer-to-peer Networks
CAN	Content Addressable Network
CPU	Central Processing Unit
CSP	Cloud Service Provider
DHT	Distributed Hash Table
F-P2P	Flat P2P
H-Chord	Hierarchical Chord
H-DHT	Hierarchical-Distributed Hash Table
HR	Home Ring
LPH	Locality Preserving Hashing
MR	Main Ring
P2P	Peer-to-Peer
REGRET	REputation model for GREgarious SocieTiEs
RRM	Remote Resource Management
SR	Super Ring
SR	Super Ring
TTL	Time To Live

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis studies trust and reputation systems in hierarchically structured P2P networks. A hierarchical Chord architecture is suggested. We propose four trust and reputation algorithms redesigned and analyzed for hierarchically structured P2P networks. The designated algorithms are built on top of Chord, a distributed lookup protocol.

1.1 Motivation

The development of the Internet has brought along with it the rapid progression of web technologies, such as e-commerce, data sharing systems, online social media, and social networks. These emerging technologies constitute part of the foundation upon which data sharing and online transactions are based. Centralized systems have presented a number of challenges in efficiently handling these new technologies, which introduced the need of decentralization systems. Nowadays, many companies and organizations use P2P systems. A P2P system is defined as a group of organized autonomous peers in which peers share distributed resources (files, computing and services) without any centralized entity[28]; a peer can be both a client and a server at the same time.

With the growing interest in P2P networks combined with the emergence of new concepts such as big data and Internet of things(IoT), the hierarchy design is presented to enhance and resolve disadvantages associated with pure P2P networks. For example, this includes the challenge of managing a network when the number of nodes increases exponentially and deteriorates the performance of the entire network, or handling the exponential increase in the number of things in a cloud environment. A hierarchical design is a solution to manage the complexity of P2P networks for many applications in IoT. The IoT may employ millions of nodes or equipments that require the exchange of information through P2P networks. The hierarchical model is easier to manage and more efficient than the pure P2P model. It also provides better fault isolation and is much more adaptable to the underlying physical network. The hierarchical model splits the entire P2P system into different layers that are mostly focused on solving local tasks and communicating with other layers in specific cases. This can include searching for an item that does not exist on the local layer, and allowing each layer to implement its own lookup system according to its requirements. The organization of the P2P model in hierarchy reduces the network traffic, while at the same time decreasing the workload. The hierarchical design decreases the lookup path length because the number of hops is significantly reduced in addition to the lookup latency, and each layer or group can implement its own lookup system according to it requirements. A DHT hierarchy is preferred in this model due to its flexibility.

Trust and reputation systems have emerged in pure P2P systems to solve the problem of malicious behaviors and fraudulent activities. These systems are based on all previous interactions, actions and feedbacks from all transactions performed among all peers connected to a network. Trust and reputation systems are the corner stone of commerce architecture, data exchange and online transactions. The main goal of trust and reputation systems is to evaluate the trustworthiness of peers, provide value to any transaction made among peers, and distinguish good peers from bad peers. These systems help in isolating malicious peers among a set of peers in the network. So far, there are many trust and reputation systems implemented for pure P2P networks, but there is yet to be a real investigation or study undertaken to implement trust and reputation in hierarchically structured P2P networks. This study offers a way to understand and master the complexity of P2P networks for applications in the IoT environment. As an application, we need to tackle the problem of trust and reputation in hierarchically structured P2P networks based on Chord. Chord is a DHT lookup protocol based on a ring topology[90].

1.2 Problems

There are many existing flat P2P networks with trust and reputation systems designed to run on top of them. These P2P networks run a number of different routing systems such as distributed hash table (Chord, CAN, Pastry and Tapestry), Gnutella and Freenet. There is currently a lack of research in implementing trust and reputation algorithms for hierarchically structured P2P networks. This thesis focuses on the implementation of trust and reputation algorithms for hierarchically structured P2P networks after implementing algorithms, analyzing their new complexity and discussing their performance.

Four algorithms using different characteristics such as trust model, reputation collection, reputation aggregation, reputation computation and trust management will be designed and implemented in hierarchically structured P2P network based on the Chord protocol. The main components of the three-tier hierarchy we build for the P2P network will be analyzed, defined and adapted to fit with our algorithms.

This thesis relies on the Chord protocol as the lookup system to build the hierarchy of nodes around rings. Chord is a DHT based system and one of the most popular structured P2P lookup protocols. A hierarchically structured P2P system is presented and adapted to handle and run these algorithms in local and top-level layers, allowing a group of nodes organized around superpeers in a group to join or leave the network without disrupting the entire network.

We extend four different trust and reputation algorithms for hierarchically structured P2P networks based on Chord, namely, EigenTrust, PowerTrust, Absolute Trust and NodeRanking. NodeRanking computes trust by calculating the number of relationship a node has with other nodes; PowerTrust use Bayesian method to generate local trust scores; EigenTrust determines trust by means of individual reputation of node in preceding transactions (using the sum of positive and negative ratings) and Absolute Trust determines trust by using number of satisfactory, neutral and unsatisfactory transactions. The choice of these algorithms is motivated by the desire to redesign algorithms using different types of reputation aggregations and computation systems. EigenTrust, Absolute Trust and PowerTrust are based on individual reputation, while NodeRanking is based on both individual reputation and social relationship. EigenTrust, PowerTrust are based on pre-trusted peers while Absolute Trust and NodeRanking are not based on pre-trusted peers and do not use a normalization of trust. NodeRanking is based on a centralized entity to retrieve results, while EigenTrust, Absolute Trust and PowerTrust do not rely on a centralized entity. We go on to analyze all redesigned algorithms and study their complexity in the hierarchical Chord

1.3 Methodology

In this thesis, we based our research on a the methodology using trust and reputation algorithms implemented for a flat P2P network. We redesigned algorithms to adapt them for a hierarchically structured P2P network running the Chord lookup protocol. Each algorithm is resigned according to its properties and characteristics. We then analyze the complexity of each algorithm in the new environment. Lastly, we conduct some experiments to generate results to apprehend the performance of each algorithm in a hierarchically structured P2P network running the Chord lookup system.

1.4 Contribution

This thesis provides the following contributions:

1. We redesigned four trust and reputation algorithms found in the literature by extending them to hierarchically structured P2P networks. We also studied their computation complexity. The redesigned algorithms are: EigenTrust, PowerTrust, Absolute Trust and NodeRanking.

2. We simulated the redesigned algorithms by using our own extension of an existing simulator for non-hierarchical networks. We focused on fraction of download, residual curl (to determine the convergence speed) and the malicious collective to assess the performance. The results of the simulation showed that Absolute Trust outperforms all the other algorithms on hierarchical Chord.

3. We detail the architectural properties of the hierarchically structured Chord model that handles all trust algorithms.

4. We specify all characteristics and properties of the architecture model and all entities including superpeers, normal peers, home and main ring.

1.5 Thesis Organization

This thesis has been organized as follows. Chapter 2 details the necessary background of P2P systems, trust and reputation systems. We introduce the paradigm of Peer-to-Peer systems, their various architectures, as well as their taxonomy. We present the routing in unstructured, structured and hierarchically structured P2P networks and provide further details about the flat Chord protocol and hierarchical Chord. We then go on to expound on trust and reputation systems, providing different definitions and concepts. Finally this chapter presents some trust and reputation systems, their taxonomy based on their functional mode, and a classification of trust systems based on reputation . Chapter 3, presents the architecture of the hierarchically structured P2P network based on Chord. In this chapter, we discuss the properties of nodes that we use for the hierarchical Chord. Two categories of rings are presented: the home and the main ring. Further to this, we redesign and discuss trust and reputation algorithms for a hierarchical Chord in great depth. we also resign, EigenTrust[44], PowerTrust[96], AbsoluteTrust[12] and NodeRanking[68] algorithms. For each algorithm, we present the adapted algorithm for a hierarchical Chord and a complete study of its complexity. In chapter 4, we present all experiment results using news algorithms and provide a summary of results. Chapter 5 includes the conclusion and future work.

Chapter 2

Background and Related Work

In this chapter we present details on the P2P computing paradigm, the taxonomy and its primary architecture. We also discuss the routing in P2P networks. Finally, we describe the Hierarchical-Chord structured network and trust systems.

2.1 Peer-to-Peer Computing Paradigm

2.1.1 The Paradigm

A P2P is a distributed system where peers share resources (files, computing and services)[85][73]. In P2P systems, all peers have the same responsibilities and capabilities. In a client-server system, the constraint is to have a server and many clients defined, while in the P2P system, this constraint is overtaken[19]. Thus, each peer has a symmetric role, meaning that a peer can be a client in one instance and a server in another instance[77, 72]. Peers exchange messages in a way that they do not require a central server and are linked to other peers. P2P systems are scalable and can be heterogeneous; peers can join or leave the network at will[62], and when one peer fails other peers continue to operate.

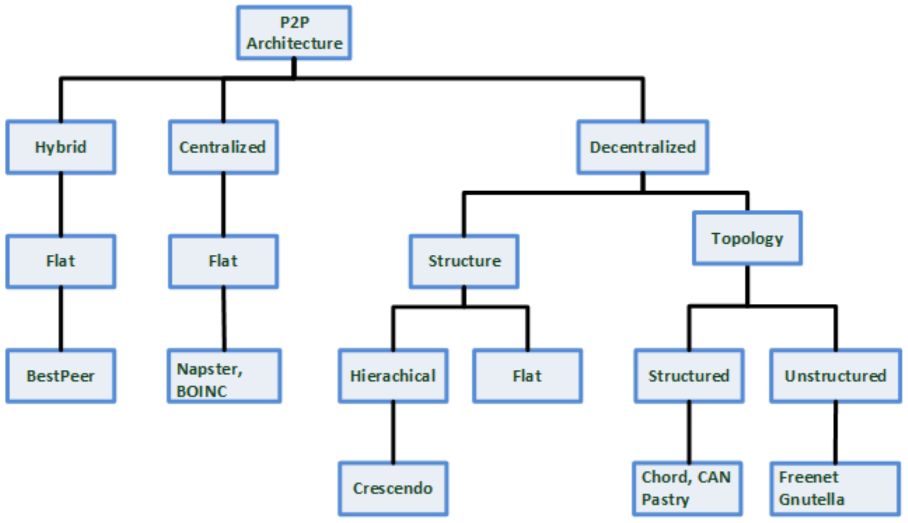


Figure 2.1: P2P System topologies (Adapted from Fig.2.1, chapter 2 of[85])

2.1.2 Benefits and Applications

The low cost of setting up and maintaining a P2P system makes it preferable to other existing systems. Each node or peer manages and applies full security control on its data. It can decide what data to make available to other nodes. Applications can share resources such as storage space, CPU cycles or bandwidth capacities and also share data by providing access privileges to retrieve data[46].

2.2 Architecture and Taxonomy of Systems

Peer-to-Peer systems are classified in 3 main categories: Centralized P2P, Decentralized P2P and Hybrid P2P[24]. Figure 2.1 presents the taxonomy of P2P systems.

2.2.1 The Centralized Systems

In the centralized systems[82], all peers are connected to a central server or a group of central servers, like in the "client-server" architecture. Napster[1] and Boinc[10] are examples of this category. Each peer must query a central server to find the position of files throughout the network[76]. A central server helps in searching files located in peers in the network. Figure 2.2 illustrates a centralized P2P .

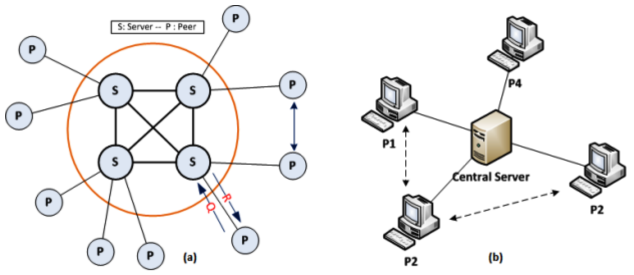


Figure 2.2: Centralized P2P systems with (a) a group of servers and (b) a single server

The disadvantages of this system are that[85] the central server is a single point of failure, exposed to malicious attacks, it can behave like a bottleneck (in case the of many simultaneous queries) and affect the performance of the network. Moreover, this system is not scalable and robust.

2.2.2 The Decentralized Systems

There is no need for dedicated central servers[61], all peers can behave as clients or servers, all nodes have the same responsibilities. There is not a single point of failure, high degree of performance, huge scalability an robustness. Furthermore, it is categorized into two main designs : "*the structure and the overlay topology* ". As represented in Figure 2.1, the structure is further divided into two groups : "*the flat (single tier) and the hierarchical (multiple tier)*". The overlay topology also is divided into groups : "*unstructured and structured*".

2.2.3 Hybrid Systems

Hybrid systems combine the features and advantages of centralized and decentralized architectures[54]. Peers called superpeers[53][79] or super nodes are provided with powerful characteristics and used to set up an upper level in the system and supply other peers called normal peers[92, 79, 52]. BestPeer++[22] and FastTrack[41] are P2P suitable for this category.

2.3 Routing in Decentralized Peer-to-Peer Systems

2.3.1 Routing in Unstructured Peer-to-Peer Network

In unstructured P2P systems the topology is not determined. These systems implement search systems that are not corresponding to the overlay network characteristics on which they are built[7]. They use different types of search systems such as flooding searches and random walks. In the random walk strategy, a query is redirected to an arbitrarily selected neighbour[55][61]. The process is iterated until the result of the query is found. In the flooding strategy, a node broadcasts a previously received query message to all of its neighbours. A neighbour forwards the query message to all of its neighbors until the result of the query is found[57]. The main drawback of this type of search systems is that it requires a larger bandwidth to function since it generates extensive data exchange among nodes. Gnutella[66][2] and Freenet[25] are examples of unstructured P2P systems.

2.3.2 Routing in flat Structured Peer-to-Peer Network

The Distributed Hash Table strategy is used as a lookup strategy in structured P2P systems to specify the location of objects in peers. The DHT provides all needed services to lookup objects in a decentralized P2P system[88]. The DHT retains mapped information about nodes and peers in the form of key/value so that data can be easily located in the overlay network. In DHT, each peer maintains a storage space to keep a hash table. Many structured P2P systems like Chord[83], CAN[69], Pastry[74] and Tapestry[94] are based on DHT. The indexing of data facilitates its discovery and search by any peer. Skip list is another strategy used in structured P2P systems for lookup information. Skip list structure works with elements arranged in an ordered manner[3] so that the search operation can be performed faster. Sorted linear linked lists are used to maintain elements[64].

2.3.2.1 Chord

Chord is distributed lookup protocol that helps find to data items located in different nodes in the network[83][40]. Chord is an overlay network[38]. It determines the node that carries the item that is looked up. Chord[78] is one of the well known distributed protocol based on a DHT[90]. It organizes all peers in a ring that maintains all keys in the range from $2^m - 1$. The value of m should be large enough to prevent collision by assigning the same ID to other nodes. Chord remains effective even when there are nodes leaving or joining networks, and it continues to answer to queries efficiently. Chord performs only one operation: it maps keys with corresponding nodes. SHA-19[67] (a consistent hash function) is the base hash function used by Chord to allocate each node and key an identifier(with m bits). Both the node's IP address and the key are hashed to determine the node's identifier and the key identifier. In chord[88, 81, 21] structure each node possesses a routing table named finger table that maintains the successors, predecessors and fingers of the node. Fingers are other peers that are tracked by each node. Every finger table contains m entries. Each node only possesses knowledge of its successors on the identifier circle in order to execute look up operations and knows little about distant nodes. Thus, each node can only maintain information for a small number of nodes, i.e. a total of *O(log N)* fingers[56], with N the number of nodes in the network. In the finger table of a node n, the identifier of the first node s (at i^{th} entry) that comes after n is determined by $s = successor(n + 2^{i-1})$.

The lookup operation is simple. Algorithm 1 represents the lookup operation. When a query is sent to a node, the node first needs to inspect its own local storage to ensure that it carries the desired data item. If it holds the desired data item, it simply sends the result to the requester. Otherwise, it redirects the query to its nearest successor node according to its finger table, as explained in[60, 59]. Subsequently, the nearest successor also redirects the query to its nearest successor, and so on, until the query reaches the node that carries the result. The result follows the reverse path to reach the requester node(the node that sent the query). With a system of N nodes, the complexity of the search algorithm is *O(logN)*. Figure 2.3 from [85] represents a node (N_{30}) that needs to lookup the key (K_{56}) stored at the node (K_{70}). (K_{30}) sends the query to node (K_{42}) which does not carry the

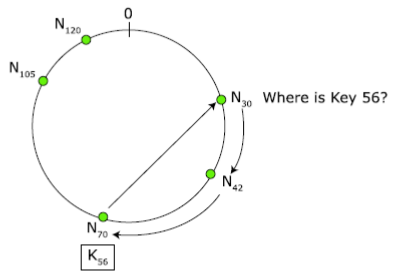


Figure 2.3: The query path from node 30 for key 56(Copied from Fig.3.6 in Chapter 3 in[85])

key and finally (K_{42}) forwards the query to node (K_{70}) that which does carry the key.

Algorithm 1: Chord lookup algorithm

```
1: // ask node n to find the successor of id
2: n.find_successor(id)
3: n'= find_predecessor(id)
4: // ask node n to find the predecessor of id  n = n'
5: while id ∉(n', n'.successor) do
6:     n' = nclosest_preceding_finger(id); return n';
7: end while
8: // return closest finger preceding id
9: n.closest_preceding_finger(id)
10: for i = m downto 1  do
11:     if finger[i].node ∉ (n,id) then
12:         return finger[i].node;
13:         return n;
14:     end if
15: end for
```

2.3.2.2 CAN

The scalable content addressable Network(CAN), a scalable structured P2P lookup service, is among the well-known P2P systems[39]. It performs a DHT to execute some basic query operations such as: deletion (key, value) pairs, insertion, and lookup [8]. Keys and values are mapped by the DHT[83]. The main idea in CAN is that its design is based on employing a virtual logical d-dimensional Cartesian coordinate to create a DHT[69], and the (key, value) pairs are kept in that virtual coordinate. Nodes organize themselves in a way that they create an overlay network representing the virtual coordinate space. For every node, the routing table contains the IP address and the virtual coordinate zone of close neighbors. Each node is responsible for limit virtual coordinates[85]. CAN space is organized in a way that it is split among all nodes that constitute the network at a specific point in the system.

2.3.2.3 PRR Trees, Pastry and Tapestry

The routing table of Tapestry[74] and Pastry[74] are similar because both of them are set up with PRR Trees. They considers the network distances when they build the routing overlay and the network topology to decrease the routing latency[83]. Pastry allows a lookup service with *O(log N)* steps. By hashing an IP address, each node is given a 128-bit node id identifier(nodeId). Each node in the Pastry network keeps a routing table, a neighborhood set of nodes and a leaf set. To route a given message, a node checks its routing table and determines the nodeId with the numerically closest key among all nodes, and sends the message to that node. Tapestry lookup service is performed in *O(log N)*. The address of a target peer is determined by the entry in the cell $(i, j)^{th}$ ID of the routing table. The identifier space of Tapestry is composed of 160-bit values.

2.3.3 Routing in Hierarchically Structured Peer-to-Peer Network

All DHT P2P systems we have described thus far are flat DHT's, meaning that they use a flat routing system; the key space is maintained in one dimension[37]. The hierarchical DHT (H-DHT) is built using a hierarchical routing system[31]. This P2P structure provides a number of advantages such as transparency, faster lookup time, less messages in the wide area, and the usage of heterogeneous peers. In flat systems, when the number of nodes increase drastically, the scalability of the entire system could be severely damaged and the communication among nodes could also become deficient[11]. The hierarchical system arranges nodes in a multi-layer architecture that tackles the weakness of the flat system.

Similar to flat structured P2P systems, in H-DHT, each peer is identified by an IP address. Also, in H-DHT, query messages are sent through the hierarchical structure and peers are assembled in groups. Each group is recognized by a unique identifier(id). This category also uses various tree structures to index data. Tree structures provide huge flexibility in carrying a large variety of queries. BATON[39] and P-Grid[6] are classified in this category; they use a balanced B+ tree. there is a novel distributed data structure called the Hierarchically Distributed Tree (HD Tree)[16][33] that can support the multi-dimensional range query from the P2P overlay.

2.3.3.1 Hierarchical Peer-to-Peer Systems

Garces & all[31] present the hierarchical DHT framework and focus on a hierarchical Chord instantiation.

In H-DHT, there are many group of nodes, each organized around its own overlay network and its own lookup service. Figure 2.4 represents the communication among groups in the overlay network. The lookup service used in each group is called a "*intra-group lookup service*". Each group contains a number of nodes called "superpeers". Superpeers are used as a gateway to connect all groups together[47]. Superpeers have ideal characteristics and properties in terms of storage and computing capacity, and are stable. Other peers in groups are called "normal peers", and based on their capacity, they can be elected to become superpeers when a superpeer leaves the network. A top-level overlay network is implemented to include all groups.

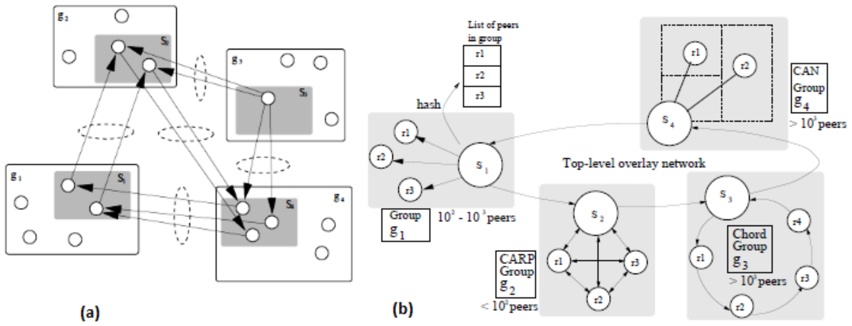


Figure 2.4: Hierarchical Model lookup process (Copied from[31])

Superpeers are integral to the lookup system because when a peer i in a group G_i needs to find a key k in the network, first the node i sends the query through superpeers in the group, then a superpeer in the group G_i redirects the query to superpeers in the group G_j

that is in charge of the key. Superpeers in the G_j will send the query to a specific peer $j \in G_j$ that is in control of the key k. The peer j will transmit the answer to the query by using the same path used to send the query message. To summarize, the query is first sent to superpeers that constitute a top-level overlay network then sent to a local overlay network to reach the local peer that is responsible for the key.

2.3.3.2 BATON

BATON is an acronym for : " BAlanced Tree structure Overlay on a peer-to-peer Network ". BATON uses a balanced tree as the topology of its overlay network. P-Grid[6] and P-Tree[26] are closed to BATON because they use trees as their overlay network. P-Grid lies on a structured binary tree topology and P-Tree lies on a structured combination of the B+ tree and a Chord ring topology[83]. BATON is the first P2P system to implement an overlay network on a binary balanced tree structure. The balanced tree structure is constructed from top to bottom. The particularity of BATON is that both leaf nodes and internal nodes are used to store data[85]. In BATON, each peer maintains many links; a link to its parent, its left child, its right child, its left adjacent peer, and its right adjacent peer. Figure[2.5] Moreover at the same level, a peer maintains two routing tables, one to control and keep peers on its left hand side and another for the right hand side[39].

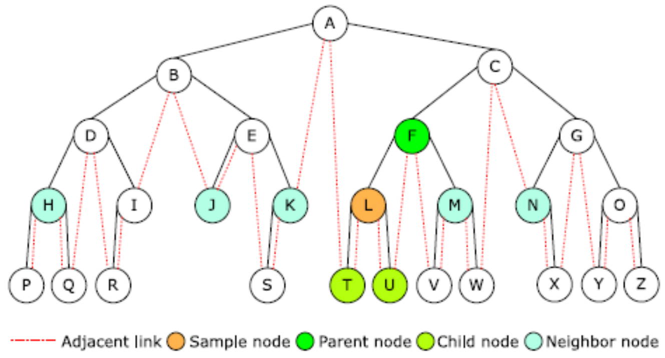


Figure 2.5: Baton structure(Copied from Fig.3.9 in[85])

2.3.3.3 Chordella

Chordella[36] is a example of a hierarchical P2P overlay application for heterogeneous mobile environments. Chordella is a two-tier P2P system. M. S. Artigas, P. G. Lpez, and A. F. Skarmeta[11] compare two hierarchal architectures limited to two layers.[35] Its upper level nodes are composed of superpeers (SP). The superpeers are organized around a DHT Chord ring. Each superpeer maintains a number of leafnodes (LN) with lower capacities. To join the network, a leafnode needs to contact a superpeer, a superpeer acts as a bootstrap for news leafnodes[97]. Figure 2.6 depicts the architecture of Chordella.

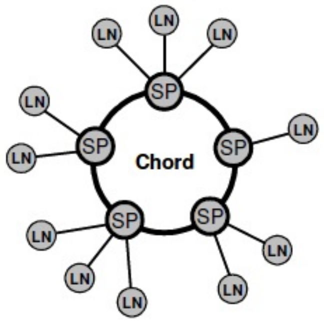


Figure 2.6: Hierarchical structure of the chordella system (Copied from Fig.5 in in[97])

2.3.3.4 Three Layer Hierarchical Model for Chord

Waqas & al.[89] present a three layer hierarchical model for Chord. This model has three types of nodes : *ordinary peers, superpeers and ultra superpeers.* All ordinary peers are connected to superpeers. Communication between two ordinary peers is made through a superpeer by means of a PING/PONG algorithm. Two ordinary peers cannot communicate directly. In this model, an ordinary peer can become a superpeer because a superpeer has a list of ordinary peers that have desirable characteristics. Thus, when a superpeer leaves the network, it must choose one of its ordinary peers to replace it. Like ordinary peers, all superpeers are connected by a single connection to ultra superpeers. Superpeers are located in the middle layer. Ultra superpeers constitute the upper level of the architecture. Ultra superpeers are organized around a DHT Chord protocol. Ultra superpeers have a higher availability and they stay in the network for a long period. When a ultra superpeers needs to leave the network it chooses a superpeer to replace it.

The lookup service is implemented in such a way that a query sent from an ordinary peer i is transmitted to the superpeer S_i that is responsible for peer i. If the superpeer S_i has the result to the query, it will simply send it to the requesting ordinary peer i. Otherwise, the superpeer S_i will redirect the query to its associated Ultra superpeer US_i. The Ultra superpeer US_i will examine its database to see whether it can find the result of the query. If not, the Ultra superpeer US_i will transfer the query to other Ultra superpeers according to its finger table.

2.3.3.5 Hierarchical Chord-Based Resource Discovery in Intercloud Environment

This model is based on the discovery of resources in the Intercloud environment[45]. In this model, data centres are organized around two chord ring structures. The first is called the "the home ring (HR)" and the second is called " the super sing (SR)". the HR connects co-located data-centres among among them. Each data-centre has a "*resource manager (RM)*" to handle internal resource maintenance. Each data-centre in a HR chooses a data-centre called "*remote resource manager(RRM)*" that participates in the SR structure so that all rings are connected. When a query is issued from a peer belonging to a data-centre in the HR, the data-center checks if it can find a result to the query. If so, the data-centre sends result to the peer, otherwise the data-centre redirects the query to the SR. Thus, a query is sent to the SR only when the data-centre from the HR cannot find the result to the query.

2.4 Hierarchical Chord

This section discusses the hierarchical P2P network based on Chord, as proposed by Garces & al.[31]. Similar to flat Chord, each peer and each key is identified by a group of m-bits.

All identifiers are organized around the ring of modulo 2^m. The hierarchical Chord inherits all the characteristics of flat Chord.

2.4.1 Hierarchical Lookup Service

The lookup service benefits from the hierarchical architecture. It is performed in two steps; first it locates the group that is in charge of the key, and inside the group it locates the peer that is in charge of the key. This is for a two-tier DHT, but for N-tiers DHT, the lookup service should go deeper in the hierarchy, by passing through groups until it reaches the group that is in charge of the key. For two-tier DHT, operations are executed in the following orders: A peer i from a group i addresses a query message to one of the superpeers S_i belonging to the group i One Superpeer of S_i at the top level lookup service transmits the query through (U, X) using superpeers, jumping from one group to another, until it reaches the group G_j that is in charge of the key k. All superpeers share their IP addresses so that they can collaborate to transfer queries from one group to another. A superpeer of group S_j of superpeers of group j receives the query and can then transfer it to the peer p_j belonging to group G_j that is in charge of the key k. When p_j responds back to the query, the response can be transmitted using the reverse the path used by the query message, or can be transmitted directly from peer j to the peer i.

2.4.2 Advantages

In hierarchical DHT, the average number of hops is considerably reduced in a lookup. The lookup latency is also reduced in a group when peers have close topology and when a cooperative cache is deployed among groups. Furthermore, each group can determine the type of lookup protocol it may perform such as Chord, CAN, Pastry and Tapestry. By using superpeers in two-level overlay networks, the H-DHT becomes much more stable than F-DHT. This increases the performance of lookup hops on average; for instance $\frac{1}{2}logN$ for Chord, where N is the number of peers. The H-DHT benefits from high *transparency* ; when a key k is searched, this search for the peer that is in charge of the key k is entirely transparent at the top-level algorithm. Furthermore, the change of the intra-group lookup algorithm by a group is entirely transparent to the top-level lookup algorithm. The failure of a peer j belonging to a G_j will be local to G_j. This will not have repercussions on routing tables in peers outside of G_j. Also, the number of nodes in groups would be smaller than the total number of nodes; thus, queries would be transmitted through a fewer number of hops. Finally, the fewer number of hops per lookup generates less exchanged messages, and the use of cache diminishes the number of messages that need to leave the group.

2.4.3 Characteristics

In hierarchical Chord, peers are aggregated in groups and each group can run a different lookup protocol. Each group has a unique designated group *id*. The directed graph *U,X*, with $X = \{g_1, ..., g_I\}$ = *Set of all groups* and U = *set of virtual edges among nodes (groups in this case)*. This overlay network describes directed edges linking groups but not particular nodes in a group. Each contains at least one superpeer.

2.4.4 Top-level Overlay Network

Each node in the top-level overlay network represents a group of nodes, and each group is composed of a set of superpeers and normal peers. Figure 2.7 depicts the proposed architecture . The lookup system at the top-level administers an overlay of groups. Also, at the top-level each node maintains both predecessor and successor vectors, in place of a pointer . Each finger, which is a vector as well, contains superpeers'IP addresses of other groups of nodes in other rings.

In the top-level overlay network, all peers are required to have stability and stay longer in the network. But when it happens that a superpeer from a group crashes or malfunctions, the new superpeer that is elected must update the predecessor and successor vectors of the groups. The idea behind this is that each group will coordinate the view with all its nearest groups.

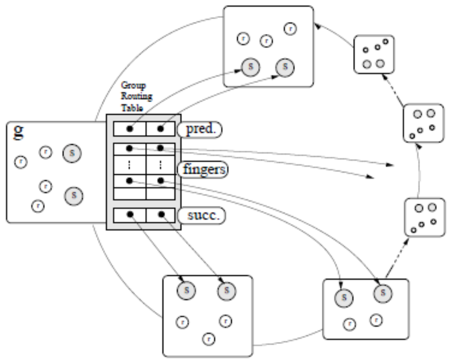


Figure 2.7: Hierarchical Model lookup process (Copied from[31])

2.4.5 Intra-Group Chord Ring

In the intra-group level depicted in Figure 2.4 *b*, different overlays can be implemented by groups. When the number of nodes in a group is great (thousands of nodes), DHT is preferable,and with N number of nodes the local lookup operation is executed in $O(logN)$ steps. When a new peer p joins a group, it is provided with the *id* of the group to be recognized, such as the name of the group. Then *p* contacts a node *p'* already participating in the group to request the IP address of the group's superpeer(s)for the group key g. If the group key sent to node p by superpeer(s) is g, then p can be connected to the group. Node p then informs to superpeer(s)regarding its CPU and bandwidth capacity. If the group id is not g, that means a new group should be constructed and p will be considered as the first peer or superpeer. A list of peers is maintained by superpeers to keep track of their duration and resources in the group;, this list is used to replace a superpeer that fails. The transfer of files is limited to the maximum available in the intra-group in order to reduce traffic among groups.

2.4.6 Content Caching

In a hierarchical Chord, we assume that all nodes are topologically close. The hierarchical structure allows the use of caches to keep query results of keys k in intra-group so that if a local node sends a query for key k that already exists in the cache, the response can be found locally. The idea is to limit traffic among groups.

2.5 Trust and Reputation Systems

Before presenting trust and reputation algorithms, it is pertinent to elucidate the various existing definitions on the concepts of trust and reputation.

2.5.1 Trust and Reputation Definitions

2.5.1.1 Trust Definitions

Definition 1: According to Castelfranchi & al.[20] : *"Trust is a mental state, a complex attitude of an agent x towards another agent y about the behaviour/action relevant for the result (goal) g "*.

Definition 2: Kumar & al.[49] adopt the definition - *"A peer′s belief in another peer′s capabilities, honesty and reliability based on its own direct experiences "*.

Definition 3: In[30], Barber & al. propose that - *"Trust is a subjective probability that relies on context and reputation, it describes how secure a situation is even though risk is associated with it"*. **Note** : In this work, definition 3 will be used to define trust.

2.5.1.2 Reputation Definitions

Definition 1: In Kumar & al.[49], reputation is defined as: *" peer′s belief in another peer′s capabilities, honesty and reliability based on recommendations received from other peers "*.

Definition 2: *"Reputation is generally said or believed about a persons or things character or standing"* Kwok & all[27].

Definition 3: Sabater & Sierra[75] suggest that: *"Reputation is the common or general estimate of a person with respect to character or other qualities"*.

Note : Definition 1 will be used as the working definition of reputation for this thesis.

2.5.1.3 Trust Characteristics

When computing trust, there are some characteristics that should be considered:
Environment incertitude :The value of trust is greatly beneficial when a large number of members need to collaborate to obtain results from an unpredictable environment, such as a P2P networks[18][87].
context-sensitive : Trust is limited in a certain context. A peer i can trust a peer j in a context x where peer j is efficient, but not trust peer j in a context y where j is ineffective[70].
Trust is subjective:Two peers can have completely opposite perceptions of third peer whose trustworthiness the are evaluating .
Trust is not transitive : Often, trust is not considered transitive; transitivity is seen as a transmission in one direction, but the move back is not always true[80].
Trust is unidirectional: There may not be reciprocity in the evaluation of trust between an agent and a subject.

2.5.1.4 Difference between Trust and Reputation

The main difference between the concepts of trust and reputation is that trust represents a personal and subjective belief, depending on an individual perception or conviction, whereas reputation is a public belief, a global perception or conviction[86].

2.5.1.5 Trust and Reputation Measurements

Feedback or ratings collected from peers (or members of a community) constitute the measurement in the deduction of reputation[86]. Trust and reputation systems help to determine good and bad nodes in P2P systems[43], by gathering , allocating, and totalling feedback regarding nodes'(participants) past behavior or past transactions[71]. Only good peers are selected as source peers[42]. We consider all interactions between two peers as transactions; for example : queries (send/receive), files or money exchange, CPU usage and data store, etc[58].

2.5.1.6 Trust Values

A trust system can use these four types of values to evaluate the trust of a peer : single, binary, multiple or continuous values. Single, binary and multiple values are discrete values. A simple value is either trust or non-trust, while the binary value has two states, one for trust, and the other for non-trust. The multiple values provide a flexible method to determine different measures of trust, such as very low trust, low trust, average trust, high trust, and very high trust. In this last case, a “very high trust ” can be seen as the trust state and “average trust” as an unknown. Continuous values provide a range of trust, for example a scale between 0 and 1.

2.5.1.7 Characteristics

According to[12], a good reputation system should have certain characteristics: It should have (a), indications of the exact past behaviors of peers, (b), low overhead, (c), ability to auto-adapt (c), no central authority, (d),ability to adapt to peer dynamics (peers joining and leaving), (e), robustness to malicious peers, (f),scalable and fast convergence (the convergence of the reputation aggregation should be sufficiently fast to show the true modification of peer behaviors because the reputation of a peer changes over time).

2.5.1.8 Trust Model Taxonomy

Trust models are classified into two main categories: credential and reputation models(as evident Figure 2.8)[85]. Some models such as PolicyMaker, Blaze et al.[15] and Trust-X[14] are trust systems built on the credential model. These credential models use only credentials to measure trust. A peer needs to look at another peer‘s credentials and determine if its credentials meet its policy, at which point it can trust the other peer. The reputation based system can be categorized into two groups: the trust System based on *individual reputation* and the trust systems based on *both individual reputation and social relationship.* The following section will present some relevant papers on trust systems based on individual reputation and social relationship.

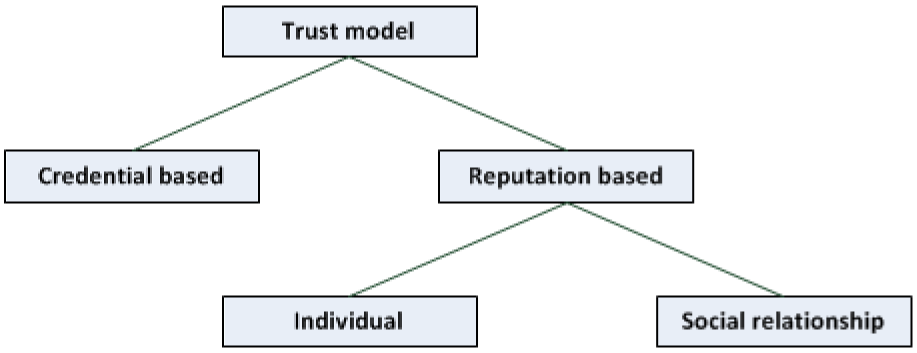


Figure 2.8: Trust model taxonomy

2.5.2 Trust Systems Based on Individual Reputation

2.5.2.1 PeerTrust

PeerTrust[85, 91] is a reputation based model for P2P online (ecommerce) communities. PeerTrust is a distributed trust reputation system and uses a transaction based feedback system to measure the level of trust and reputation. It is based on individual reputation as a trust model. Trust management is based on a distributed control. The drawback of this system is that, if a reputation of a peer is only based on the satisfaction (feedback) that it receives from other peers, it may not be sufficient to eradicate malicious peers. In this system, the computation of the reputation of peers is based on previous transactions. To establish the reputation of a peer, PeerTrust[85] provides five important features that should be exploited: 1) The feedback a peer obtains from other peers, 2) The feedback scope, such as the total number of transactions that a peer has with other peers, 3) The credibility factor for the feedback source, 4) The transaction context factor for discriminating mission-critical transactions from less or noncritical ones, and 5) The community context factor for addressing community-related characteristics and vulnerabilities. To determine the general trust metric, PeerTrust combines all parameters in a consistent manner. The metric is composed of two modules. The weighted average of the amount of satisfaction a peer receives for each transaction is represented in the first module of the metric. The weight considers the credibility of the feedback source to prevent the feedback from dishonest peers. The transaction context is useful in representing the transaction dependent characteristics. In the second module, the increasing or decreasing of the trust value derive from the community specific characteristics and situations contributes to the refinement of the first module of the metric. However, PeerTrust does present some drawbacks; namely, feedback messages among peers can generate high traffic volume in the network, which can create a bottleneck when the number of peers increases substantially.

2.5.2.2 The EigenTrust

EigenTrust[51][44][85][50] is a distributed trust reputation system based on individual reputation and uses distributed control. After each transaction, each peer keeps a list of reputations from other peers it interacted with. A peer conserves a record of all previous transactions in a local trust vector $\vec{c_i}$. Vector $\vec{c_i}$ consists of all local trust values c_{ij} that peer i has attributed to other peers j. It can be represented as $\vec{c_i} = (c_{i1}, c_{i2}, c_{i3}, ..., c_{in})$. All c_{ij} are positives because there are normalized as $c_{ij} = \dfrac{max(s_{ij}, 0)}{\sum(s_{ij}, 0)}$, and the sum of $(c_{i1} + c_{i2} + c_{i3} + ... + c_{in}) = 1$. All local trust values are represented in a matrix $[(c_{ij})]$ defined by C. A gossiping algorithm is used to assemble the global reputation t_i of the P2P network. The global trust value t_i determines the trust that the entire system places in the peer i . $\vec{t}$ determines the global trust vector of the entire system. EigenTrust evaluates the left principal eigenvector of a matrix of normalized local trust values, so that it can calculate the global trust value of a peer. It computes local reputation and global reputation and it uses transitivity to measure trust. This system needs a group of honest peers, pre-trusted peers$\vec{P}$ as start vectors to eliminate malicious peers.

Local trust values assigned to a peer i by other peers are used to determine the global reputation of peer i. Local trust values must be normalized to avoid malicious peers attributing randomly high local trust values to other malicious peers, and at the same time, they may attribute low trust values to good peers. Consequently, malicious peers may alter the system. Once computed, normalized local trust values are kept in a vector called: *the normalized local reputation vector* on the local peer. The normalized local trust values are aggregated[29]. To aggregate them, a peer i should ask its acquaintances their perspective regarding other peers as $t_{ik} = \sum c_{ij}c_{jk}$, where t_{ik} is the trust that the peer i

places in the peer k derived from querying its friends j[4]. Take it for granted that a peer i is informed about all c_{ij} of the entire network disposed in a matrix C, after that $\vec{t_i}$, the trust vector that describes the trust values for $\vec{t_{ik}}$ is defined by the formula : $\vec{t_i} = C^T \vec{c_i}$. To obtain a wide perspective, peer i should ask friends of its friends so that the formula can be written as :$\vec{t_i} = (C^{T^2} \vec{c_i})$. $\vec{t} = (C^{(T)^n} \vec{c_i})$, with n being large enough to allow $\vec{t_i}$ to converge the same vector for every peer i. $\vec{t_i}$ represents the global trust of the system. t_j, elements of the vector $\vec{t_i}$, represent the value of trust that the entire system places in j. By isolating malicious peers, EigenTrust allows users to download files from secured and reputable peers, also motivates users to share files by rewarding reputable peers.

Algorithm 2 represents the *"Basic Eigen Trust Algorithm"*. Pre-trusted peers $\vec{p}$ are incorporated and the formula $\vec{t}^{(k+1)} = (1-a)\vec{t}^{(k+1)} + a\vec{p}$ is used to address the issue of malicious collectives[63]. Malicious collectives are malicious peers that give to each other a high local trust value, and give very low trust value to other peers.

Algorithm 2: (Basic EigenTrust algorithm)

```
1: $\vec{t}^{(0)} = \vec{p}$;
2: **repeat**
3:     $\vec{t}^{(k+1)} = (C^T \vec{t}^{(k)})$;
4:     $\vec{t}^{(k+1)} = (1-a)\vec{t}^{(k+1)} + a\vec{p}$;
5:     $\delta = ||t^{(k+1)} - t^{(k)}||$;
6: **until** $\delta < \varepsilon$;
```

In the distributed EigenTrust algorithm 3, each peer i keeps its local trust vector $\vec{c_i}$, its own global trust value t_i and it computes its own t_i.

In secure EigenTrust algorithm 4, score managers are chosen based on the coordinates dictated by using hash functions for the peer's unique identifier. Each peer i can be a score manager for a peer j, so that it is given a group of daughters.

2.5.2.3 Absolute Trust

Absolute trust is a algorithm for aggregation trust in P2P networks for peers that only exchange files. The algorithm and the metric used determine the true past behavior of

Algorithm 3: (Distributed EigenTrust Algorithm)

```
1: $A_i$= Set of peers which downloaded files from peer $i$.
2: $B_i$= Set of peers from which peer $i$ has downloaded files
3: query all peers j $\in A_i$ for $t_j^0 = p_j$;
   Each peer $i$ do
4: **repeat**
5:     **Compute**
6:     $t_i^{k+1} = (1-a)(c_{1i}t_1^k) + (c_{2i}t_2^k) + ... + (c_{ni}t_n^k) + ap_j$;
7:     send $(c_{ij}t_i^{k+1})$ to all peers j $\in B_i$
8:     wait for all peers j $\in A_i$ to return $(c_{ji}t_j^{k+1})$;
       **until**
9: **until** $\delta < \varepsilon$;
```

Algorithm 4: (Secure EigenTrust Algorithm)

```
1: **for each peer** $i$ **do**
2:     Submit local trust values $\vec{c_i}$ to all score managers at position $h_m(pos_i)$, m = 1...M-1;
3:     Collect local trust values $\vec{c_d}$ and sets of acquaintances $B_d^i$ of daughter peers $d \in D_i$
4:     Submit daughter $d's$ local trust values $c_{dj}$ to score manager $h_m(pos_d)$, m = 1...M-1,$\forall j \in B_d^i$;
5:     Collect acquaintances $A_d^i$
       **Foreach** *daughter peer* $d \in D_i$ $i$ do
6:     Query all peers j $\in A_d^i$ for $c_{jd}p_j$;
7:     **repeat**
8:         **Compute**
9:         $t_d^{k+1} = (1-a)(c_{1d}t_1^k) + (c_{2d}t_2^k) + ... + (c_{nd}t_n^k) + ap_d$;
10:        send $(c_{dj}t_d^{k+1})$ to all peers j $\in B_d^i$
11:        wait for all peers j $\in A_i^d$ to return $(c_{jd}t_j^{k+1})$; **until**
12:    **until** $||t_d^{(k+1)} - t_d^{(k)}|| < \varepsilon$;
13: **end for**
```

peers[12, 13]. In contrast to previous algorithms such as EigenTrust[9] and PeerTrust[91], this aggregation algorithm doesn't need a normalization of trust, pre-trusted peers or any centralized authority . It is also completely decentralized. It uses the concept of weighted averaging and scaling of local trust to determine the reputation of peers. To determine the local trust value, the satisfaction of a peer is categorized by three levels : *satisfied, neutral and unsatisfied.* We have two peers, i and j, i downloads files from j.

Local Trust

A local trust metric that peer i assigns to j can be computed by :

$$T_{i,j} = \frac{n_g w_g + n_n w_n + n_b w_b}{n_t} \tag{2.1}$$

n_g = Number of satisfactory (good) files; n_n = Number of average or neutral files; n_b = Number of unsatisfactory (bad) files; n_t = Total number of downloaded files; w_g = Weight factor for satisfactory file; w_n = Weight factor for average or neutral file; w_b = Weight factors for unsatisfactory file. The variation of weight factors is supposed to vary in a linear manner from unsatisfactory file to satisfactory file, so that :

$$w_n = \frac{n_g + w_b}{2} \rightarrow T_{i,j} = \frac{1}{2}[(x - y + 1)w_g + (y - x + 1)w_b] \tag{2.2}$$

x : The fraction of satisfactory files, y : The fraction of unsatisfactory files.
Note : This metric guarantees that the local value will stay between w_g and w_b.

Algorithm for Aggregation

There are three methods taken into account when evaluating global trust : One-to-many, many-to-one, and one-to-one. *One-to-many*: One peer is assessing many other peers; *many-to-one*: many peers are assessing one peer; *one-to-one*: One peer is assessing another peer. One-to-many generates a uniform evaluation because just one peer assesses other peers, while in many-to-one functions, there are potential contradictions, thus, the more proficient peer will receive more weight than others. In one-to-one, the evaluator and evaluatee are different; a direct comparison between them does not exist. The evaluation of an evaluator's competence is more precise, so that evaluation can be biased by a weight factor that corresponds to the competence of the evaluator. The bias can be expressed by :

$$Eval\_uniform\_out = [(Eval\_value\_in)^p.(w_e)^q]^{\frac{1}{p+q}} \tag{2.3}$$

Eval_value_in = evaluation done by an individual evaluator and w_e = weight factor assigned to this evaluator; $Eval\_uniform\_out$ = output uniform evaluation; *(p, q)*:suitably chosen constants. From equation 2.2 a peer i can determine the trust of all peers j from which it downloaded files because it uses a *one-to-many evaluation*. The evaluation of *global trust* t_i of a peer i can be shown by the equation :

$$t_i = \frac{\sum_{j\in S_i} T_{ji}t_j}{\sum_{j\in S_i} t_j} \rightarrow t_i = \frac{\sum_{j\in S_i} T_{ji}t_j}{e_i.C.t} \tag{2.4}$$

$S_i =$ A set of peers that download files from peer i; $T_{ji} =$ The local trust of peer i determined by peer j; $t_j =$ The global trust of peer j. From equation 2.4, the value of t_i stays between the minimum and maximum value of trust provided by peers belonging to S. Knowing that each peer i is evaluated by a group of peers S_i, a set can be viewed as a single peer, and thus be considered as a *one-to-one evaluation*. A set S is composed of m peers with global values $t_1, t_2, ..., t_m$. In a set, the opinions of more trustworthy peers hold greater weight so their opinions their have a greater influence on determining the global trust of the set. The global trust of the set is presented as :

$$t_s = \frac{\sum_{j\in S} t_j^2}{\sum_{j\in S} t_j} \tag{2.5}$$

The global trust ensures that the global trust will be influenced by peers with a high global trust value. The global trust is between the lowest and the greatest value of the global trust of peers in the set S. Including the bias in equation 2.3, global trust t_i of a peer i is represented by equation 2.6 :

$$t_i = \left[\left(\frac{\sum_{j\in S_i} T_{ji}t_j}{\sum_{j\in S_i} t_j}\right)^p . \left(\frac{\sum_{j\in S_i} t_j^2}{\sum_{j\in S_i} t_j}\right)^q\right]^{\frac{1}{p+q}} \tag{2.6}$$

The equation 2.6 expresses clearly the past behavior of any peer i. Algorithm 5 determines the source file to be selected by a peer i. All chosen peers are named "*source peers*". A reference value of global trust called " *global_ref* " is specified by each peer to determine which peer can or cannot be a source peer . To be chosen as a source peer, any peer should have a " *global_ref* " greater than that of the requesting peer. Requesting peers determine a TTL value for each query they send while searching resources. The value of the TTL decreases until it reaches the value 0, hence the query is not sent again. If no response is

Algorithm 5: For selection of source peer

```
Procedure
$Global\_ref \leftarrow \frac{(w_g + w_b)}{2}$
$TTL \leftarrow Const.$
top:
$Set\, Time\_Counter \geq 2 * TTL$
$i \leftarrow 0.$
Send the query for required file in Network;
while $i \leq Time\_Counter$ do
    Wait for response from the network;
    $i \leftarrow i + 1$;
end while
if $(Number\_of\_responding\_peers == 0)$ then
    if $TTL \geq (TTL)_{upper}$ then
        Terminate the query process;
    else
        $Increase\, TTL$;
        GOTO top
    end if
else
    Get the $Global\_Trust$ of all the responding peers from their trust holder peer;
    Select the peer with maximum $Global\_Trust$;
    if $Global\_Trust \geq Global\_ref$ then
        Download the required file;
        Download the required file;
        Send the feedback to trust holder peer of source peer;
        Stop;
    else
        if $TTL \geq (TTL)_{upper}$ then
            Terminate the query process;
        else
            increase TTL
            GOTO top
        end if
    end if
end if
end Procedure
```

received, the requesting peer resends the query with a larger TTL. In case of many peers responding to the query, the requesting peer will choose the more reputable among all source peers to download the needed file. After downloading the necessary file from the selected peer, the requesting peer transmits feedback of its transactions(with the source peer) to peers that carry the trust values of the source peer. In Absolute Trust, a peer i does not hold its global trust value; each peer may have more than one "*holder peer*" that is in charge of carrying its global trust value . The number of holder peers increases the security . Algorithm 6 represents how a global trust value of a peer is updated by a holder peer.

Algorithm 6: For Updating the Global Trust of peers

```
1: Input :  Local Trust of peer
2: Output :  Global trust on trust holder peers
3: Procedure
4: for each peer i do
5:     for all peer $j$, who is selected as source peer do
6:         Evaluate the received file;
7:         Assign the Local Trust value between $w_b$ to $w_g$;
8:         Send the local Trust to trust holder peer of peer $j$;
9:     end for
10:    if Peer $i$ is trust holder peer of peer $k$ then
11:        for all peer $j$, who is selected $k$ as source peer do
12:            Receive the Local Trust values $T_{kj}$
13:            Locate their trust holder peer;
14:        end for
15:        Initialisation;
16:        $Setp, q, previous\_t_k$, threshold;
17:        while $error \geq threshold$ do
18:            Receive the Global Trust $t_j$ from their trust holder peer;
19:            compute
               $t_i = \left[\left(\frac{\sum_{j \in S_i} T_{ji} t_j}{\sum_{j \in S_i} t_j}\right)^p \cdot \left(\frac{\sum_{j \in S_i} t_j^2}{\sum_{j \in S_i} t_j}\right)^q\right]^{\frac{1}{p+q}}$
               $error \leftarrow |t_k - previous\_t_k|$
               $previous\_t_k \leftarrow t_k$
20:        end while
21:    end if
22: end for
23: end Procedure
```

2.5.2.4 PowerTrust

PowerTrust is a reputation system that carefully and dynamically chooses a small number of power peers that have a high reputation value by deploying a distributed ranking mechanism; it uses the power law findings in peer feedbacks[96][23] . Furthermore, it determines a good reputation for power peers by examining the history of the network. By combining a look-ahead random walk strategy and the advantages of power peers, PowerTrust ameliorates global reputation accuracy as well as the rate of aggregation speed. PowerTrust uses trust overlay network (TON) to build trust relationships among peers. In PowerTrust, reputable peers can be dynamically replaced if they are less active or display inappropriate behaviors. Their presence is crucial to determining the local trust and global scoring processes. PowerTrust applies a Bayesian method to produce local trust scores (feedback scores)[95].

The architecture of PowerTrust is depicted in Figure 2.9. The TON is constructed on top of a P2P network, the *look-ahead random walk (LRW)* is in charge of updating the reputation score regularly, and *the regular random walk* supports the initial reputation aggregation. *The distributed ranking* collaborates with the *LWR* to recognize power peers in the P2P network. Power peers help the system to update the global reputation scores. The PowerTrust method is similar to that of EigenTrust[44]. The difference is that, PowerTrust deploys a Bayesian method to produce local trust values[93].

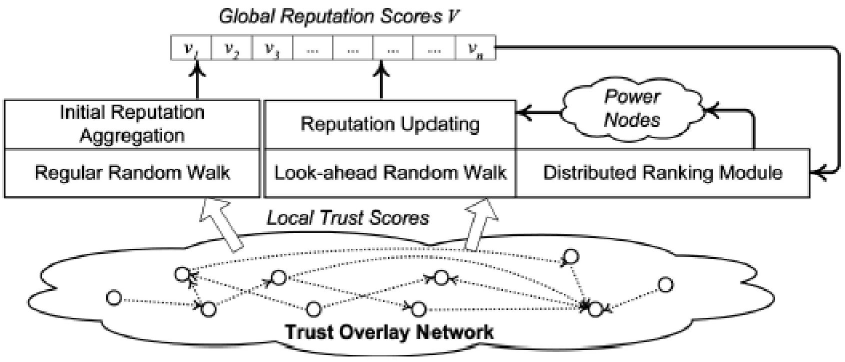

Global Reputation Scores V
v_1 v_2 v_3 ... ... v_n
Initial Reputation Aggregation
Reputation Updating
Power Nodes
Regular Random Walk
Look-ahead Random Walk
Distributed Ranking Module
Local Trust Scores
Trust Overlay Network

Figure 2.9: Functional modules in the PowerTrust system and the control flow pattern in local trust score collection and global reputation aggregation. (Copied from Fig.1 in[96])

TON is a directed graph in which links (directed edges) are marked with the feedbacks scores from two peers engaged in interaction. The source of the link provides the feedback score after having received a service from the destination of the link. In TON, a peer is depicted by a node and each edge is marked with the peer feedback score for the service allocated. All trust values are summed from other peers (represented by the incoming edges of a peer i) and the sum is used to compute the global reputation of node i. The *outdegree of a peer i* is the number of users to whom peer i sends feedback scores. The *indegree of peer i* is the number of users from whom peer i receives feedback scores.PowerTrust relies on Bayesian learning or on an average rating based on peer satisfaction to produce feedback scores. Each peer is recommended to normalize all generated feedback scores from it interaction with other peers.

Let the *trust matrix* $R = (r_{ij})$. This matrix is employed of the n-nodes TON. r_{ij} represents the *normalized local trust score.* r_{ij} is computed from $r_{ij} = \frac{S_{ij}}{\sum_j S_{ij}}$. S_{ij} and represents the most recent feedback score from peer i's evaluation of peer j. $0 \leq r_{ij} \leq 1$ for all $1 \leq i,j \leq n$, also $\forall i \sum_{j=1}^{n} r_{ij} = 1$. The matrix R fulfills all conditions of a stochastic matrix by having all entries as fractions and the sum of each of its rows sum as 1. $V = (v_i)$:

A normalized reputation column vector, $\sum_i v_i = 1$. V contains all global reputation scores v_i for n nodes. $V = (v_1, v_2, v_3, ..., v_n)$ constitutes the output of the PowerTrust system. The global reputation of each nodes is computed by (1). $\forall t = 1, 2, 3, 4, ..., k$, $\mid V_i - V_{t-1} \mid > \varepsilon$ reputation vector is recursively computed by :

$V_{(t+1)} = R^T \times V_{(t)}$ (1) Where, t : Number of iterations and R = The trust matrix.

The reputation vector V starts with an initial reputation vector V_0 and a small error threshold ε. In a network with n nodes $v_i = \frac{1}{n}$ as an initial value. The Markov random walk used in ranking web pages is used to stimulate the recursive process. This process is identical to that of a random knowledge surfer visiting nodes after nodes looking for a reputable node. The surfer chooses a neighbour according to the current distribution of local trusts. The converged global reputation vector is defined by the stationary distribution of the Markov chain.

PowerTrust employs three algorithms to display the initial construction, the distributed ranking and the updating process. Algorithm 7 describes the complete procedure to discover m peers that are more reputable. PowerTrust relies on DHT to perform *the distributed ranking mechanism*[34]. The distributed ranking mechanism uses the *locality preserving hashing (LPH)* to organize all nodes based on their global scores. LPH relies on two properties :
First property :" $H_{(v_i)} < H_{(v_j)}$, iff $v_i < v_j$, v_i and v_j,the global reputation of node i and j ". *Second property* :" if an interval $[v_i, v_j]$ is divided into $[v_i, v_k]$ and $[v_k, v_j]$, the corresponding interval $[H_{(v_i)} < H_{(v_j)}]$ ought to be divided into $H_{(v_i)} < H_{(v_k)}$ and $H_{(v_k)} < H_{(v_j)}$ ".

A score manager is attributed to each node to gradually collect its global reputation. For instance, when a new node i connects to the network, it is given a node j to be its score manager if node j is considered as a successor node of k_i. k_i is determined as the unique identifier of node i. To know the global reputation of node i, all other nodes should send a query with the key corresponding to k_i. For improving security against malicious peers, different hash functions can be implemented so that a malicious score manager will communicate an identical global reputation score.

Algorithm 7: Selection of top-m peers (power nodes)

```
Input :  Global reputations stored among score managers
Output :  m most reputable nodes
Procedure :
for each score manager $j$, suppose it is the score manager of node $i$ do
    hash reputation value $v_i$ to a hash value $H_{(v_i)}$ using a LPH function;
    insert the triplet $(v_i, i, j)$ to the successor node of $H_{(v_i)}$
end for
Initialize :  node $x$ = successor node of the maximum hash value
Set $p$ = number of triplets with highest reputation values stored in node $x$
loop:
if $p > m$ then
    return;
else
    node $x$ sends a message to its predecessor
    node $y$ finds the next $m - p$ highest reputation triplets
    node $x$ = node $y$
    $m = m - p$
    $p$ = number of triplets stored in node $y$
    goto loop
end if
end Procedure
```

Algorithm 8 illustrates the initial stage of global reputation aggregation, where each node i submits all local scores to the score managers of its out-degree neighbours.

Algorithm 8: Initial Global Reputation Aggregation

```
Input :  Local trust scores stored among nodes
Output :  Global reputation for every node
Procedure :
for each node $i$ do
    for all each node $j$, which is an out-degree neighbor of node $i$ do
        Send the score message $(r_{ij}, i)$ to the score manager of node $j$
    end for
    if node $i$ is the score manager of node k then
        for all node $j$, which is an in-degree neighbor of node $k$ do
            Receive the score message $(r_{jk}, j)$ from node $j$
            Locate the score manager of node $j$
        end for
        Set a temporary variable $pre = 0$; initialize the error threshold $\epsilon$
        and *global reputation* $v_k$ *of node* k
        repeat
            Set $pre = v_k$; $v_k = 0$
            for all received score pair $(r_{jk}, j)$, where $j$ is an in-degree neighbor of node $k$ do
                Receive the global reputation $v_j$ from the score manager of node $j$
                $v_k = v_k + v_j r_{jk}$
            end for
            Compute $\delta = \mid v_k - pre \mid$
        until $\delta < \epsilon$
    end if
end for
end Procedure
```

The algorithm 9 describes the entire reputation updating process. Based on the Markov chain, PowerTrust works like a random walk. The random surfer begins at any peer using the same probability. The probability of the surfer jumping directly to the power node is established as the *greedy factor* α. When the value of α increases, the surfer is eager to

Algorithm 9: Global Reputation Updating Procedure

```
1: Input : Local trust scores stored among nodes
2: Output : Global reputation scores for all nodes for use by score managers collaboratively to find the $m$ most
   reputable nodes using Algorithm 1
3: Procedure :
4: for each node $i$ do
5:     for all each node $j$, which is an out-degree neighbor of node $i$ do
6:         aggregate local trust scores from node $j$
7:         Send the score message $(r_{ij}, i)$ to the score manager of node $j$
8:     end for
9:     if node $i$ is the score manager of node k then
10:        for all node $j$, which is an in-degree neighbor of node $k$ do
11:            Receive the score message $(r_{jk}, j)$ from node $j$
12:            Locate the score manager of node $j$
13:        end for
14:        Set a temporary variable $pre = 0$; initialize the error threshold $\epsilon$
15:        and global reputation $v_k$ of node k
16:        repeat
17:            Set $pre = v_k$; $v_k = 0$
18:            for all received score pair $(r_{jk}, j)$, where $j$ is an in-degree neighbor of node $k$ do
19:                Receive the global reputation $v_j$ from the score manager of node $j$
20:            end for
21:            if node $k$ being a power node then
22:                $v_k = (1-\alpha)\sum(v_j \times r_{jk}) + \frac{\alpha}{m}$
23:            else
24:                $v_k = (1-\alpha)\sum(v_j \times r_{jk})$
25:            end if
26:            Compute $\delta =\mid v_k - pre \mid$
27:        until $\delta < \epsilon$
28:    end if
29: end for
30: end Procedure
```

attach to a power node. When the surfer reaches any node, it carefully chooses a neighbour based on the local trust distribution with a probability $1 - \alpha$. To be attached a power node, the surfer uses a probability α. After each pass of the aggregation, the power nodes are chosen again according to a new global reputation score. The adjustment of the *greedy factor* α allows to manage the gap separating the first and second largest eigenvalues of a transition matrix T. The largest eigenvalues $\lambda_1 = 1$, the second largest eigenvalue should be $\lambda_2 = 1 - \alpha$

2.5.3 Trust Systems Based on Both Individual Reputation and Social Relationship

2.5.3.1 REGRET

REGRET (A reputation model for gregarious societies)[65] is a trust system based on reputation. It measures the value of reputation and implements a distributed control to manage reputation. REGRET employs a reputation computation engine based on fuzzy model[17]. REGRET considers three dimensions of reputation: individual, social and ontological[84]. It is a unique distributed TRS system that employs various aspects of a community to derive a peer trust value. When a peer joins a network, the peer can rate the seller (another peer) and the community which the seller belongs to. In comparison to PeerTrust[91], REGRET can analyze a community as a whole, while PeerTrust only scrutinizes at analyzing the seller. An impression is an object that enables a peer to rate another peer on the scale of [-1, 1] based on a specific property relative to the seller. An impression is a subjective evaluation produced by a peer on a particular aspect of an outcome. Reputation in REGRET is seen as an ensemble of different pieces of information. If a peer evaluates reputations only with information from its direct cooperation with other peers, it is called individual dimension. When a peer uses information about another peer derived from members of the society and social relations, this is referred to as social dimension. The social dimension is based on the interaction between the members of a group[85]. A reputation of a peer in a group is influenced by the reputation of other peers in the group. Some properties may play a part in the evaluation of the reputation of a peer; for instance, when looking for a good university to study at, tuition fees, language of instruction and security may be considered as elements of its reputation. These elements upon which reputation is based referred to as the ontological dimension. The primary issue with the system is that the cost of computing the reputation of societies is costly in terms of preservation and assembly of reputation values. Moreover, making a decision regarding which node should be selected for association with a society is costly.

Algorithms Based on Trust Management Using a Distributed Control

2.5.3.2 NodeRanking

Like REGRET, NodeRanking[85] is another trust system based on both individual reputation and social relationship as a trust model; it executes a distributed control to manage trust. It does not rely on the feedback technique[68], as is the case with REGRET. The social network is built with information from various sources such as link in personal web pages, email traffic, commercial transactions, documentary information and knowledge exchanges, responses to answers, advice, or document authorship[68]. Data mining techniques can be used to obtain all this information. NodeRanking can estimate a reputation exclusively from local information using topological information. The reputation value of a node i can be evaluated by the number of references provided to it by other nodes in the network. The *degree of authority* is used as a measure[5]. The sum of all nodes that point to a node i should be a positive value. A random walker strategy is used to explore the graph. It begins by selecting a random node and pursues outgoing references to other nodes, according to[85]. Also, there is no need to be aware of all graphs for NodeRanking to operate. A gossiping algorithm manages sharing of knowledge among all peers that are involved in the P2P network. Furthermore, it needs a central entity in order to acquire all the results of the ranking. Algorithm 10 represents the random walker strategy to travel through the graph network . In order to construct the social network, knowledge from personal pages was extracted, with each personal page representing a node (a member of a community). The directed edges of the graph are computed :

$$w(a \to b) = w_{email}(a \to b) + w_{link}(a \to b) + w_{name}(a \to b) \tag{2.7}$$

The sum of the three factors determines the weight of a relationship: if $w_{email}(a \to b) = 1$, there exists an email address of member b in personal pages of member a; if not the value is equal to 0.

$$w_{link}(a \to b) = \sum_{\forall \alpha \in R(a,b)} \frac{2}{depth(\alpha, a) + depth(\alpha, b)} \tag{2.8}$$

R(a,b) represents a set of a resources (accessible through the web) of node b that are located in the personal page of a node a. $depth(\alpha, a)$ represents the depth of a resource α

in the personal page of node a. To extract the number of occurrences of b in node a :

$$w_{name}(a \to b) = \frac{1}{2} \sum_{i=0}^{i \leq \sharp name(a \to b)} \frac{1}{i^2} \tag{2.9}$$

Algorithm 10: NodeRanking algorithm

```
1: while (!converge()) do
2:     n = getNode(graph g);
3:     while (nnew!=null) do
4:         passAuthority(n);
5:         nnew = getNextNode(n,g);
6:         n = nnew
7:     end while
8: end while
9: end Procedure
```

getNode() = chooses a random normal node in a ring. **getNextNode(n,g)**= Based on information about a set of out-edges for each node, it gives back one of the out-neighbours nodes of node n to be explored. If the path is broken, it gives back a null node. The path breaks when the algorithm reaches a node already explored previously, or when a specific value of jumping probability is attained. When the random walker arrives at a node, it estimated the jumping probability using this equation :

$$Pr_{jump}(a) = \frac{1}{\sharp outEdges(a) + l} \tag{2.10}$$

The equation 2.10 determines the probability of a node to break the path. Nodes with a greater number of out-edges reduce the probability of breaking the path while a node with fewer out-edges increases the probability to break the path. Node b to be explored is chosen with density probability function determined by :

$$Pr_{choose}(n \to s) = \frac{w_{n \to s}}{\sum_{\forall l \in outNodes(n)} w_{n \to l}} \tag{2.11}$$

Equation 2.11 expresses the weight of the edge between a and b. The numerator determines the weight of the link connecting n and s. **passAuthority(node x)**: All nodes that x points to are delegated or transferred as part of the authority of x.

$$auth(y) = auth(y) + \left[\frac{Pr_{choose}(x \to y)auth(x)}{F_y}\right] \tag{2.12}$$

The value of the authority is kept by F_y in a restricted range of values, because the authority of a node tends to continuously increase as the random walker progresses. The initial value of F is the sum of the authority of all nodes that constitute the graph, and factor $F \geq 1$. The transition probability matrix of NodeRanking is :

$$M = J\frac{1}{N}\vec{1}\vec{1}^T + (\vec{1}\vec{1}^T - J)P \tag{2.13}$$

Where 1 represents a vector of 1s, N represents the number of nodes, and P represents the adjacent matrix normalized by rows. J represents the jumping probability matrix defined as 0. The diagonals of the matrix keeps jumping the probability Pr_{jump} of a node.
convergence() helps to determine the stationary state of all nodes based on a defined threshold φ. With the knowledge that each node is aware of its last increment authority, if the increment of a node is below the threshold φ, a node is seen to be in stationary state. The algorithm stops running when all nodes in the network are considered stationary. In this algorithm, nodes inform their state of becoming stationary.

2.6 Conclusion

In this chapter, we discussed the paradigms, benefits, applications, architecture and taxonomy of P2P systems. We presented the centralized, decentralized and hybrid P2P systems. Then, we looked at routing in decentralized P2P systems, by presenting routing in unstructured, flat structured, and hierarchically structured P2P system. In routing flat structured P2P systems, we addressed Chord, CAN, Pastry and Tapestry, while in hierarchical model we presented Baton and Chordella. Moreover, we discussed hierarchical Chord in particular, by showing its characteristics, advantages, and hierarchical lookup service. We also discussed top-level overlay network made up superpeers and the intra-group Chord ring that is implemented by groups. Finally, we discussed trust and reputation systems. We defined trust and reputation concepts, and spoke of their characteristics and differences. We presented the classification of trust and reputation algorithms based on individual reputation, such as PeerTrust, EigenTrust, AbsoluteTrust and PowerTrust, and those which are based on both individual and social relationship such as REGRET and NodeRanking. In the next chapter, we will redesign EigenTrust,AbsoluteTrust, PowerTrust and NodeRanking Trust and reputation algorithms, to adapt them for the hierarchical Chord P2P network, analyze their complexity, and present the system model architecture.

Chapter 3

Trust and Reputation Algorithms For Hierarchical Chord

We have examined hierarchically P2P systems in the previous chapter and also some architectures developed so far such as in[45][36][31]. In this chapter we present four trust and reputation algorithms for hierarchically Chord structured P2P networks. We begin by introducing the architecture of hierarchical Chord, then, we redesign the trust and reputation algorithms for our hierarchical Chord model. Based on our research we have chosen these four algorithms that we improve and adapt to the hierarchical Chord architecture, namely EigenTrust[44], PowerTrust[96], Absolute Trust[12] and NodeRanking.

3.1 Overview

EigenTrust, PowerTrust, Absolute trust and NodeRanking are trust systems based on reputation systems; Trust is based on feedback from other peers. EigenTrust collects reputation by polling, also, the EigenTrust algorithm relies on the random surfer model, flow model. This algorithm requires the presence of pre-trusted peers and necessitates normalizing of local trust values in order to aggregate them. PowerTrust relies on the assumption of the power law network on user's feedback. PowerTrust is similar to EigenTrust when it aggregates local trust, but it is different from EigenTrust on same ways; in PowerTrust pre-trusted peers can be replaced by more reputable peers in the network. The choice of pre-trusted peers is selected dynamically by electing the most reputable peers in the network. Absolute Trust does not require any kind of pre-trusted peer, power peers or a centralized entity, and it does not necessitate a normalization of trust. It uses the concept of weighted averaging and scaling of local trust. The computation of global trust is performed recursively; it then converges at a certain unique value. Finally, NodeRanking uses the degree of authority of a node to measure its reputation. It deploys a random walker strategy.

3.2 System Model Architecture

The model of our hierarchical Chord is similar to that developed in[45] and[31]. In our thesis, we determine more practical and advantageous ways to use the model that take current concerns into account in terms of resource distribution and the discovery mechanism of resources. This model is designed to be used in a two-tier hierarchy, at least, and should be extended to n-tier hierarchy. For the purpose of our research, we assume a three-tier hierarchy, where a ring of nodes can join the network through another ring of nodes already in the network. We also consider that a main ring is composed of superpeers that connect other nodes to form a ring, and that an m-bit *identifier* is attributed to each peer and each *key*. The model we use is based on a cloud service provider (CSP) that is made up of many data centre disseminated around the globe, for example, in each continent. Thus, each continent can be connected to the super ring by using a Superpeer called "***remote resource manager (RRM)***", detailed in the next section. Each continent can be composed of many home rings, as depicted in figure 3.2.

3.2.1 Superpeer

A superpeer S_i is a peer or a node that has ideal characteristics in terms of storage capacity, stability and computation. A superpeer is available for a long period in the network and in many cases it is used as a server by other computers. Generally, a superpeer does not quit the network. A group G_i may contain more than one superpeers. We assume that a superpeer that links two group of nodes is called "***(RRM)***". A RRM operates as a proxy, which is a liaison between two rings. When a group needs to join the network, it should select a superpeer to be RRM, or a RRM should be selected before a ring joins the network. In the cloud environment, we assume that each ring is composed of many superpeers that host different kinds of servers or data centre. Data centre are selected based on their characteristics such as availability,security, performance, capacity and data integrity. When a node i in a group G_i wants to communicate with a node j in a group G_j, the node i sends the query to its superpeers S_i set and one of its superpeers redirects the query to the node j in the group G_j. In our model depicted in the Figure 3.2, we consider that two groups are linked by a single superpeer. Thus, a superpeer, the RRM S_i connects a group G_i and a group G_j. A RRM working in a Chord lookup system is assumed to have two finger tables, one for each of the two connected rings, and maintains successors and predecessors from each ring.

3.2.2 Normal or Regular Peers

Normal peers are peers that participate in the overlay network in a group ring. Normal peers may have less characteristics than superpeers. But a normal peer can be elected to be a superpeer in case of the failure or departure of a superpeer. The system maintains a list of all normal peers with their characteristics and resources, and based on that, a normal peer can be elected to become a Superpeer. When a normal peer is the first to join a new ring, it is considered as a superpeer until there are other nodes, and an election can be organized to choose the nodes that have the features and resources to become superpeer. Each normal peer knows the name and IP address of its superpeers in the same group or ring. A normal peer may or may not stay longer in the network.

3.2.3 Node Group

In this system, nodes are organized in group G_i of nodes. A group is a flat system. Every group is organized around its own overlay network and lookup service. In our architecture we assume that all groups are organized around a DHT Chord lookup service. We assume that a node group is composed of normal and superpeers nodes, and all nodes are located in the same region. Each group is independent from other groups. We consider two types of groups: "*home group or home ring(HR)*" and "*super group or Super Ring (SR)*". We assume that there are many Home Rings and one Super Ring.

3.2.3.1 Home Group (Home Ring)

A home group or a home ring is a group of nodes around a ring organized around a lookup service called " *intra-group lookup service*". In this work, we use home ring (HR) to designate home group. Each HR is designated by an identifier that is unique. A group i is appointed by G_i and contains a number of nodes, normal peers as well as superpeers. Home rings build the hierarchy of the network and they are located in different places. Figures 3.2 and 3.3 depicts the architecture of the hierarchical model. In this model, we assume that there are many HRs connected among them using RRM peers. Figure 3.1 depicts the organisation of a HR that we assume in our model. A Home Ring is considered to be autonomous so that files exchange will be confined as much as possible among peers in a HR. It functions like a F-P2P but it communicates with other HRs in hierarchical network through RRM's. A HR may have a great number of nodes; it can connect many other rings and extends the hierarchy of the entire tree, its length.

3.2.3.2 Super Ring (SR)

SR is the main ring that connects all other HRs located in geographically distant locations. As depicted in Figure 3.2 and Figure 3.3, SR is considered as the first group, the first ring or the main ring. It is useful in propagating queries among HR that are connected to it. A SR is organized around a Chord lookup system. A SR is designed to connect many

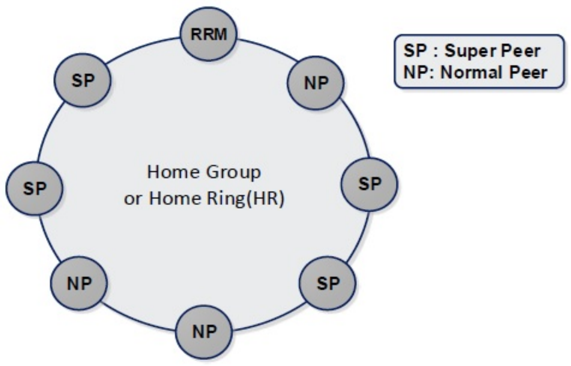


Figure 3.1: Home Group (Home ring)

other rings that are distant. It helps in propagating queries and data flow among different other groups of nodes (Home rings). We assume that all nodes participating in the SR are superpeers, and also, there is only one SR where many HRs are connected. In a tree, the SR is considered as the root of the tree.

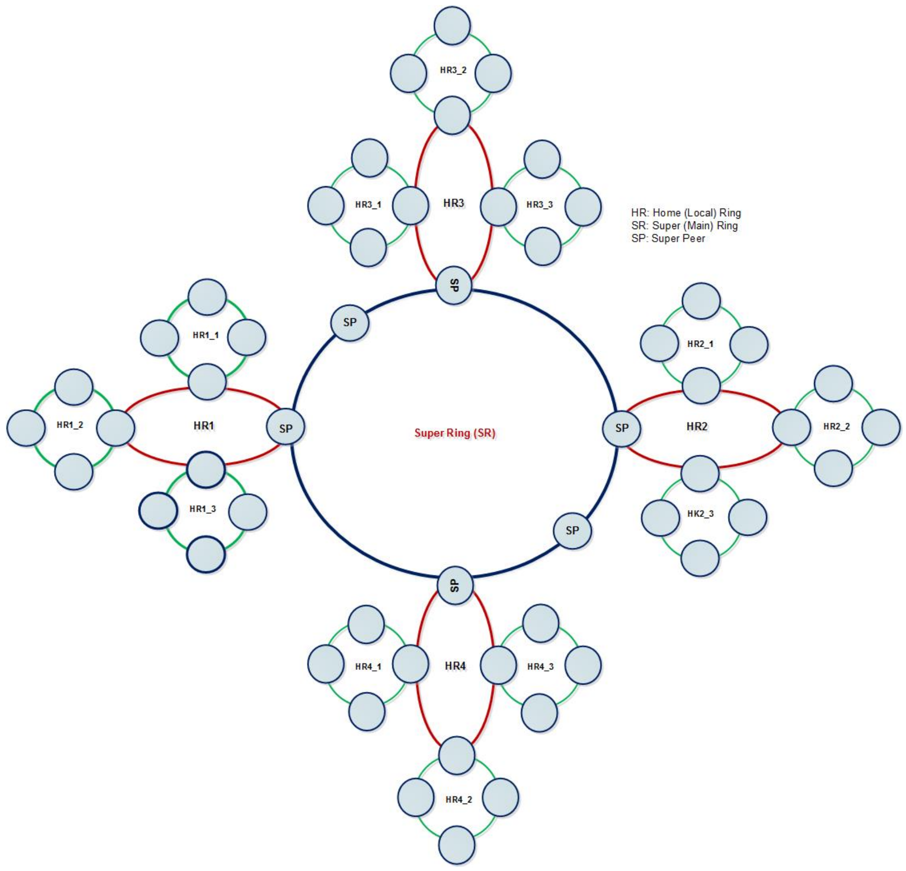


Figure 3.2: Hierarchical Chord model

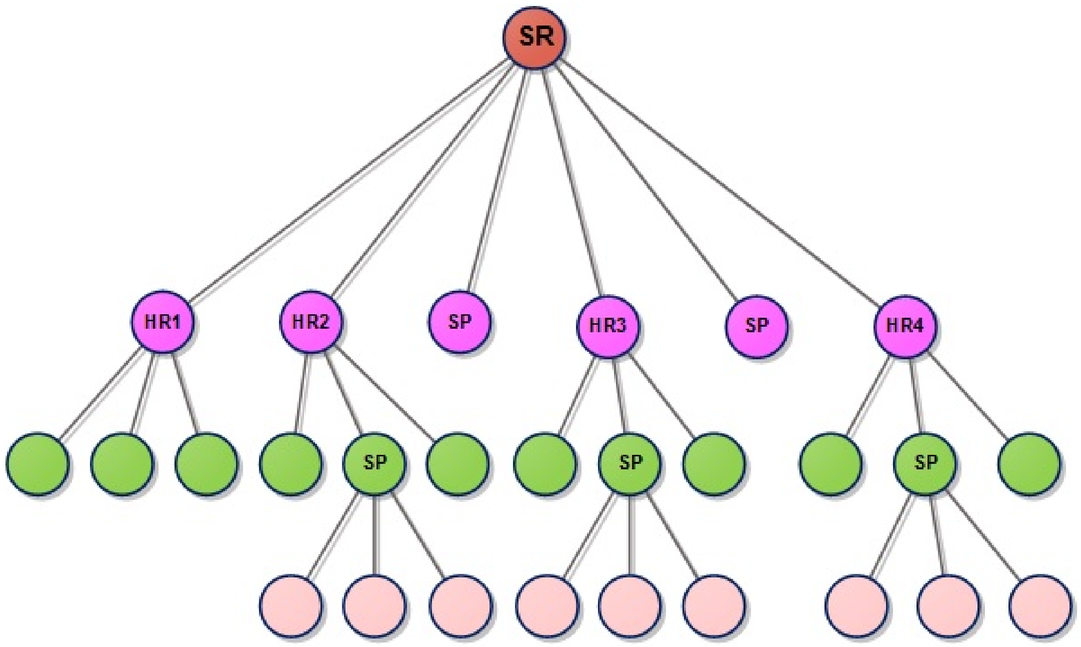


Figure 3.3: Hierarchical Chord model

3.2.4 Nodes in the Hierarchy

In this project, we use a three-layer structure. In the layered structure, the lower the layer of the ring, the smaller the average latency between nodes in the ring. In the overall P2P network, the H-Chord contains many P2P overlay networks in different layer (namely, P2P ring). Nodes are distributed in different layers. For example, if the user set the size of the ring to be four, the three-layer H-Chord structure will be shown as in Figure 3.2. There are 52 nodes in the system totally, the 4 nodes (superpeers) belong to the main ring (level-1), the 12 nodes are added to level-2, and the other 36 nodes are level-3 peers.

3.2.5 Lookup Service

We consider that all groups apply a DHT chord lookup service. For the entire network, the lookup service uses the hierarchical architecture like in[31]. As in a flat DHT Chord, every peer manages a group of successors, predecessors and finger vectors in the routing table. Every normal peer keeps the IP address of the superpeers (in the HR) that are connected to other groups. We also assume that a superpeer in a group (at the top level) keeps a vector of successors and predecessors groups. The lookup service, first determines a group that is responsible for the key then finds the peer that is in charge of the key. The hierarchy is organized around N-tiers levels, thus the lookup needs to determines all levels that must be browsed to reach the group that is responsible for the key, and at the same time, the peer that is in charge of the key. The query hops from one group to another through superpeers (RRM). Figure 3.4 depicts the hierarchical model lookup process. Figure illustrates how a query browses different groups of nodes until it finds the answer to the query. A query is transmitted to other peers and groups of nodes only if the answer cannot be found locally. A query passes by the main ring to reach other groups of nodes. The lookup service should be fast because the system must determine the group and hops to the group that has the key. To maintain the high performance of the entire hierarchy, a number of hops from one group to another should be determined during the lookup process. For example if the number of hops is fixed to four, that means a query from a local HR should browse through four other rings to find the result of the key. If a query browses all four rings and does not find the result, it is assumed that there is not an answer to the query. To overcome this limitation, the system is assumed to have query cache in HR to keep responses of previous queries.

3.2.6 Assumptions

Based on our architecture depicted in Figure 3.2, for our algorithms we have made a number of assumptions and considerations. First, we consider that there are a number of superpeers in each HR, at least one. We also suppose that all superpeers in the network are pre-trusted peers and all “RRM peers”are pre-trusted peers as well. Each HR has a “RRM ”, superpeers and normal peers. We assume that SR is composed of the RRM peers (that connect HRs) and other Superpeers. Also, we consider that selected superpeers contain two finger tables from both rings it interconnects with, and their finger tables are vectors containing the ID of superpeers that are successors and predecessors. A RRM can only connect two different rings at most. Finally, we assume that some superpeers and RRM in

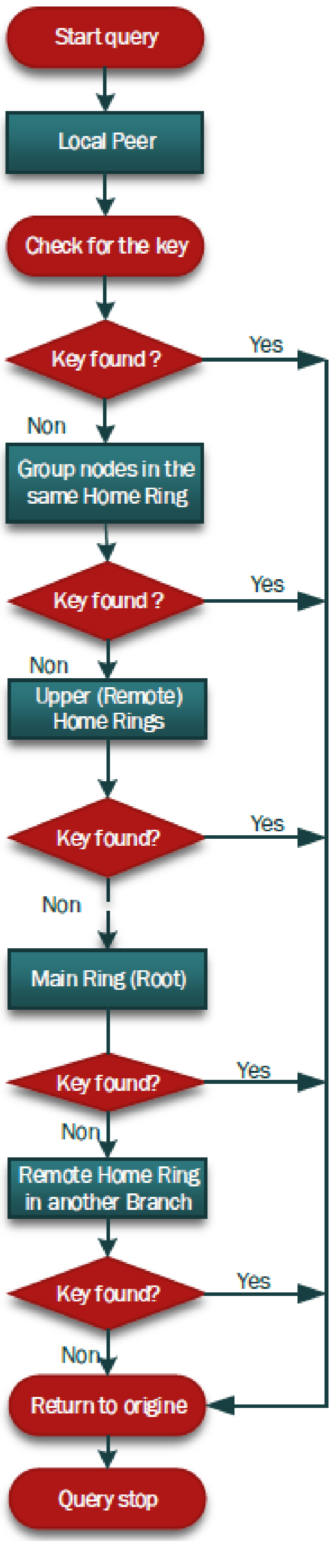


Figure 3.4: Hierarchical Model lookup process

every ring stay longer in the network. They are selected to be set up in the network, and considered to be stable.

3.3 EigenTrust Algorithm for H-Chord

EigenTrust algorithm is one of the more well-known algorithms designed for P2P flat systems. Flat P2P networks are systems that use a key space carries on one dimension. In this section,we redesign an EigenTrust algorithm for Hierarchical Chord P2P networks. EigenTrust is an algorithm suitable for reducing the number of inauthentic files in a P2P system based on file sharing. In our model based on CSP, many data-centre share data and the problem of inauthentic files can become poignant in trust management. This EigenTrust algorithm is applied to the hierarchical architecture model and to any other Chord-based hierarchical P2P system . To apply this algorithm in this architecture, we assume that HR′s should communicate to exchange information about the score of nodes when for example a node i from a group G_i downloads files from a node j in group G_j.

3.3.1 EigenTrust Algorithm

The presented Algorithm 11 takes into account the hierarchical aspects of the network. A HR can join the network at any time,but for a node to provide feedback about other nodes, it should have interacted with other peers for a certain number of transactions. The number of transaction should be determined. The distributed algorithm considers a RRM with two finger tables from both rings it links, that which eases communication and the lookup service.

In the algorithm, some important information about group G_j, RRM_j and pre-trusted peers (superpeers) P_j is added to improve the lookup process in the hierarchical structure. A group is identified by G_j. Thus, at each level, pre-trusted peers are assumed to be involved in the computation of trust and reputation. The algorithm computes the local trust of peers by using score managers of each peer to keep the score, and then aggregating all trust scores of a peer to determine its global reputation in the network. Because this

Algorithm 11: (EigenTrust for Hierarchical Chord)

```
1 $A_s^i$= Set of peers (from local and remote HR) which have downloaded files from peer $i$ .
2 $B_s^i$= Set of peers (from local and remote HR) from which peer $i$ has download files
3 $G_j$ = Group $id$
4 $P_j$ = : Pre-trusted Superpeers
5 $RRM_j$ : Pre-trusted Superpeer (gateway).
6 $hops_j$ : Number of hops from ring to ring
7 . $Rpeer_j$ : Number of remote peers to send scores
1: for each ($G_j$) do
2:     $P_j \leftarrow const$
3:     $hops_j \leftarrow const$
4:     $Rpeer_j \leftarrow const$
5: end for
6: for each peer i in a ($G_j$) do
7:     do
8:     Submit Local trust values $\vec{c_i}$ to all score managers at positions $h_m(pos_i), m = 1, 2, ...M - 1$
9:     if Local trust value of peers $\in$ other HR's then
10:        Transfer their scores to the local RRM
11:        determine the $id$ of the remote HR
12:        Local RRM sends value to remote RRM of the HR, to the score managers
13:    end if
14:    Collect all Local trust values $\vec{c_d}$ and set of $B_d^i$
15:    Submit daughter d's Local trust values $c_{dj}$ to score managers $h_m(pos_i), m = 1, 2, ...M - 1$ with $j \in B_d^i$
16:    Collect acquaintances $A_d^i$ of daughter peers
17:    Communicate with other HR to collect acquaintances $A_d^i$ of daughter peers
18:    for each daughter peer d $\in D_i$ do
19:        query all peers $j \in A_d^i$ for $c_{jd}P_j$
20:        if peers $j \in$ to another HR then
21:            send queries to RRM to transfer them to indicated HR.
22:        end if
23:        repeat
24:            Compute
                   $t_d^{k+1} = (1 - a)(c_{1s}t_1^k) + (c_{2s}t_2^k) + ... + (c_{ns}t_n^k) + aP_j;$
               send $(c_{dj}t_s^{k+1})$ to all peers j $\in B_d^i$
25:            sent to RRM all $(c_{dj}t_s^{k+1})$ for other HR's.
26:            wait for all peers $j \in A_i^d$ to return $(c_{jd}t_s^{k+1})$;
27:            wait for all peers $j \in A_i^d$ from other HR's to return $(c_{jd}t_s^{k+1})$;
28:        until $|t_s^{k+1} - t_s^k| < \varepsilon$
29:    end for
30: end for
```

algorithm works in a hierarchical environment, transactions made outside HR's, are used to bring the scores of all peers score from other HRs to local ring in order to compute the global reputation of a local peer.

In a hierarchically structured P2P, the number of local trust values reported by a peer i is limited because a network may have a huge number of peers. The algorithm adds a variable $Rpeer_j$ to limit the number of remote peers that can send scores of a peer i located in a local ring. But practically, a peer i in a network has a limited number of interactions with other peers. This algorithm allows a peer to have its score managers only located in the same HR to optimize the algorithm and to reduce traffic in the network. Peers outside of a local HR can report scores of peers in another ring to their RRM or superpeers, which will then transfer scores to score managers of peers into their corresponding HR′s. Any time a peer has to report the score of another peer that is not in the same HR, it transmits it to the RRM, since the RRM works in two HR's.

We also assume that a group can only interact with a limited number of other groups. The number of hops from one ring to another rings for lookup purpose is determined and limited by the variable $hops_j$, since in many cases, a query is sent outside the group only when the group does not have an answer to the query. Not all nodes will interact in the H-Chord network, and the number of nodes in every HR is reduced comparatively to the number of nodes in one flat Chord. Small rings (home rings) will execute the algorithm faster than one large flat ring; this allows the algorithm to converge faster.

3.3.2 Complexity of the Algorithm

Proposition 1. *The algorithm is executed in $O(n^2)$.*

Proof. To compute the complexity of the algorithm, we need to take into account the local and remote computation of node scores. Essentially, local computation involves many peers relative to remote computation due to restriction of the number of hops and remote peers that can send feedback for a peer in a local HR. Also, the system allows the use of cache to keep results of queries. The algorithm is run by each HR and RRMs are involved in the exchange of scores. When the algorithm is run for the first time, it determines pre-trusted peers, RRM and the number of hops it is permitted to use while sending a query out-site of a local ring. The first operations of all nodes in the H-CHord concern the computation of trust and the reputation of a chosen node. The idea in a hierarchical structure is to keep queries in the local ring. A query is forwarded out of the HR only if there is not any node (in the home ring) that can provide a response to a query, meaning that, the interaction among rings will be reduced and used only for specific purposes, like exchanging information about scores of nodes or data. Finally, we assume that in the computation of the reputation of a node i, the number of HR's from which peers will send feedback after having performed transactions with node i must be limited to increase the performance of the algorithm, and the number of remote peers from other HR's to a local ring is limited to improve the performance of the algorithm. Also, the number of nodes in a local ring is reduced so that the performance of the algorithm is improved. □

3.4 PowerTrust Algorithm for H-Chord

In the flat PowerTrust, when power nodes become less active, fail or demonstrate abnormal behavior, they are replaced. The system organizes an election for new power nodes. In hierarchical structure, we assume that each HR has a number of known predefined power nodes and normal nodes can be elected to become power nodes based on their higher reputation and their capacities in the HR. RRM peers are also replaced only in the event that they fail. Thus, PowerTrust helps in selecting more reputable peers among normal peers. Algorithm 12 describes the process of selecting reputable peers among normal peers. Algorithm 13 initializes the global reputation aggregation, and Algorithm 14 initiates the global Reputation updating procedure.

3.4.1 PowerTrust Algorithm of Selection Power Nodes

Algorithm 12 is executed in each HR. We assume that power nodes and RRM of each group G_i are predetermined. Also, because of their autonomy, each HR keeps all score manager nodes locally and all processes are executed without communicating with other HR's. After the execution of the algorithm, every group adds a number of normal peers to the group of power nodes. In the algorithm, we assume that there are power nodes and normal nodes in a HR. Also a HR may have more power nodes (super peers) than normal peers. If the number of power nodes is greater or equal to that of normal peers, we assume that there are enough reputable nodes that can participate in the computation of the trust and reputation of other nodes in the HR. Thus, this algorithm of selecting reputable nodes cannot be executed in a HR when the number of power nodes is greater or equal to the number of normal peers. Moreover, for the algorithm to be run, nodes should have executed a certain average number of transactions in a ring previously.

Algorithm 12: Selection of m most reputable normal nodes

```
Input :  Global reputations stored among score managers
Input :  predetermined Power nodes
Input :  predetermined RRM
Input :  Each group $G_i$
Output :  m most reputable normal nodes
Procedure :
if (number of power nodes $\geq$ number of normal nodes) then
    stop the execution;
else
    if (node == predetermined Power node) then
        return;
    else
        for each score manager $j$, suppose it is the score manager of node $i$ do
            hash reputation value $v_i$ to a hash value $H_{(v_i)}$ using a LPH function;
            insert the triplet $(v_i, i, j)$ to the successor node of $H_{(v_i)}$
        end for
        Initialize :  node $x$ = successor node of the maximum hash value
        Set $p$ = number of triplets with highest reputation values stored in node $x$
    end if
end if
loop:
if $p > m$ then
    return ;
else
    node $x$ sends a message to its predecessor
    node $y$ finds the next $m - p$ highest reputation triplets
    node $x$ = node $y$
    $m = m - p$
    $p$ = number of triplets stored in node $y$
    goto loop
end if
end Procedure
```

3.4.1.1 Complexity of the Algorithm

Proposition 2. *The algorithm is executed in $O(n)$ times.*

Proof. Each HR executes the algorithm in $O(n)$ (n the number of nodes) and does not need to communicate with other HRs to determine the most reputable nodes. Each score manager uses a LPH function to hash reputation value v_i to a hash value H(v_i) and another loop is executed to find the highest reputation triplets among normal peers. The algorithm will run faster because every HR has super nodes and a reduced number of nodes in comparison to a flat structure and there are already some reputable nodes (pre-defined superpeers and RRM). If the first condition of the algorithm is fulfilled, the algorithm stops because the number of power nodes is greater than the number of normal nodes. Another advantage is that, every HR is managing a reduced number of nodes relative to a flat P2P network. This will allow the algorithm to run fast. □

Algorithm 13: Initial Global Reputation Aggregation

```
1: Input :  Predetermined Power nodes
2: Input :  Predetermined RRM
3: Input :  Group $G_i$
4: Input :  Local trust scores stored among nodes
5: Output :  Global reputation for every node
6: Procedure :
7: (the procedure is applied to all HR from which peers participate in resource exchange)
8: for each node $i$ do
9:     for all each node $j$, which is an out-degree neighbor of node $i$ do
10:        Send the score message $(r_{ij}, i)$ to the score manager of node $j$
11:        if node $j$ is in another HR then
12:            Transfer the score message to the $RRM_i$ in the $HR_i$ of node $i$
13:            $RRM_i$ transmits the score message to the corresponding score manager of $j$ in its $HR_j$
14:        end if
15:    end for
16:    if node $i$ is the score manager of node k then
17:        for all node $j$, which is an in-degree neighbor of node $k$ do
18:            Receive the score message $(r_{jk}, j)$ from node $j$
19:            Locate the score manager of node $j$
20:            if node $j$ is in another HR then
21:                node $j$ transmit the the score message from one to another RRM until to the HR in the $RRM_i$
                   of node $i$
22:            end if
23:        end for
24:        Set a temporary variable $pre = 0$; initialize the error threshold $\epsilon$
25:        and global reputation $v_k$ of node k
26:        repeat
27:            Set $pre = v_k$; $v_k = 0$
28:            for all received score pair $(r_{jk}, j)$, where $j$ is an in-degree neighbor of node $k$ do
29:                Receive the global reputation $v_j$ from the score manager of node $j$
30:                $v_k = v_k + v_j r_{jk}$
31:            end for
32:            Compute $\delta = \mid v_k - pre \mid$
33:        until $\delta < \epsilon$
34:    end if
35: end for
36: end Procedure
```

3.4.2 Initial Global Reputation Aggregation

Algorithm 13 initializes the global reputation aggregation for all nodes in the network. We assume that we need to limit the amount of feedback from nodes in other HR's, and the number of HR's from which nodes send feedbacks. Also, RRMs are solicited to transfer scores of a peer i from one HR to another until it reaches the score manager of peer i in its HR, a RRM facilitates the collection of scores from different HR's.

3.4.2.1 Complexity of the Algorithm

Proposition 3. *Algorithm 13 is performed in $O(n^2)$.*

Proof. For n HR, the algorithm is executed for n nodes. The number of nodes in a HR is reduced in comparison to one large flat ring so that the algorithm can run fast. Queries are sent out of a ring only if there are not responses from the local ring. Thus, the flow of communication among rings is reduced and does not generate a huge among of data. The presence of caches will somehow help to solve the problem. The communication among nodes is optimized in the ring. The algorithm gathers scores of all out-degree and in-degree neighbours of nodes in the network. RRMs in HR's are used to transfer the score from one ring to another. To improve the complexity, the number of hops of scores need to be limited, in particular, hops from one node to another node and from one HR to another. The number of transactions after which the algorithm can run to aggregate reputation should be defined. □

3.4.3 Global Reputation Updating Procedure

Algorithm 14 works similar to Algorithm 13 but it is only used to maximize the advantages of power nodes. The run time is the same as that of Algorithm 13.

Algorithm 14 solves the problem of joining and leaving nodes in HR's. When nodes join and leave a HR, the transition matrix should be computed again to update the global reputation of nodes. In a hierarchical environment, the HR should communicate and exchange messages about the scores of nodes and update their global reputation.

3.4.3.1 Complexity of the Algorithm

Proposition 4. *Like Algorithm 13,Algorithm 14 runs in $O(n^2)$.*

Proof. A number of peers will stay longer in the network, such as superpeers and RRM peers, and they are more reputable. This will reduce the number of peers that join or leave the network, and the transition matrix will be computed with minimal changes. □

Algorithm 14: Global Reputation Updating Procedure

```
Input :  Predetermined Power nodes
Input :  Predetermined RRM
Input :  Group $G_i$
Input :  Local trust scores stored among nodes
Output :  Global reputation scores for all nodes for use by score managers collaboratively to find the $m$ most normal
reputable nodes using Algorithm 1
Procedure :
for each node $i$ do
    for all each node $j$, which is an out-degree neighbor of node $i$ do
        aggregate local trust scores from node $j$
        Send the score message $(r_{ij}, i)$ to the score manager of node $j$
        if node $j$ is in another HR then
            send the score message to the RRM in the HR of node $j$
        end if
    end for
    if node $i$ is the score manager of node k then
        for all node $j$, which is an in-degree neighbor of node $k$ do
            Receive the score message $(r_{jk}, j)$ from node $j$
            Locate the score manager of node $j$
            if node $j$ is in another HR then
                send the score message to the RRM in the HR of node $i$
            end if
        end for
        Set a temporary variable $pre = 0$; initialize the error threshold $\epsilon$
        and *global reputation* $v_k$ *of node* k
        repeat
            Set $pre = v_k$; $v_k = 0$
            for all received score pair $(r_{jk}, j)$, where $j$ is an in-degree neighbor of node $k$ do
                Receive the global reputation $v_j$ from the score manager of node $j$
            end for
            if node $k$ being a power node then
                $v_k = (1 - \alpha) \sum(v_j \times r_{jk}) + \frac{\alpha}{m}$
            else
                $v_k = (1 - \alpha) \sum(v_j \times r_{jk})$
            end if
            Compute $\delta =\mid v_k - pre \mid$
        until $\delta < \epsilon$
    end if
end for
end Procedure
```

3.5 Absolute Trust Algorithm for Reputation Management in Hierarchical Chord

In hierarchical structure, Absolute Trust is assumed to run in many groups of nodes organized in rings. It uses two algorithms to compute the reputation of peers. Algorithm 15 is run for selecting source peers in local and remote rings and Algorithm 16 is run to update the global trust of peers for their previous transactions in local and remote rings. For each algorithm, the same assumptions will be made to adapt them for hierarchical Chord. In the flat Absolute Trust algorithm, there is no need for pre-trusted peers or power peers. However, in hierarchal Absolute Trust, the presence of pre-trusted peers, superpeers or power peers makes it not logical, but important. This is, because in Hierarchical environment, there are many flat networks interconnected which create a hierarchy. All flat networks are connected to each other through superpeers and RRMs. Thus, Absolute Trust will be run in every ring.

3.5.1 Algorithm for Selection of Source Peer

Algorithm 15 represents the selection of source peers in H-Chord. Superpeers, Group Gi and RRM RRM_i are considered as input. To find source peers, the algorithm will select peers in other HR's, thus, another variable needs to be added to determine the number of hops (h_i) from one ring to another to find source peers, in other words, to determine the number of rings that need to be visited to discover source peers. Also, the search of source peers focuses on the HR where a query is generated to see if a result could be found. If no result is found, then it will be extended to other rings. This will help to prioritize the local ring where the query is from to encourage local exchange and avoid huge message traffic that can impact the performance of the algorithm. Furthermore, the number of source peers to find in the network can be limited to improve the performance of the algorithm. If the number of source peers does not reach the limited number, then the search can be extended to other rings. If there are two source peers with the same global trust score, the source peer that is in the same ring where a query is issued and should be selected. Using hierarchical Chord, we assume that each peer i has score managers in charge of keeping its trust score. Score managers are allocated to peers by means of the DHT because Chord is used. Score managers are involved in updating the global trust of a peer. The location of a score manager of a peer is determined by using a hash function on a peer ID (IP address and a TCP port). After having evaluated files from a peer, a feedback about the transaction is sent to it score manager. If the peer for which the feedback should be sent is in another ring, its score should be sent to its score managers in the same ring through RRMs. Superpeers can be employed as score managers for normal peers because they are trustworthy.

Algorithm 15: For selection of source peer

```
Input :  Superpeers
Input :  Group $Gi$
Input :  RRM $RRM_i$
Input :  number of hops (number of rings to visit) $h_i$
Input :  limit of source peers $L_i$
Output :  Source peers
Procedure
$Global\_ref \leftarrow \frac{(w_g+w_b)}{2}$
$TTL \leftarrow Const.$
$h_i \leftarrow Const$
$L_i \leftarrow Const$
for each Group $Gi$ do
    top:
    $Set\,Time\_Counter \geq 2 * TTL$
    $i \leftarrow 0.$
    Send the query for required file in Network;
    while $i \leq Time\_Counter$ do
        Wait for response from the network;
        $i \leftarrow i + 1;$
    end while
    if $(Number\_of\_responding\_peers_{\in}\_Home\_Ring == 0)$ then
        Send the query for required file to RRM to extend search in remote Rings ;
        if $TTL \geq (TTL)_{upper}$ then
            Terminate the query process;
        else
            $Increase\,TTL;$
            GOTO top
        end if
    else
        limit the Number_of_responding_peers to $L_i$
        Get the Global_Trust of all the responding peers from their score managers;
        Select the peer with maximum Global_Trust;
        Give a privilege to Global_Trust of local peers;
        if $Global\_Trust \geq Global\_ref$ then
            Download the required file;
        else
            if the required file is in a peer $p_k$ in another $HR_k$ then
                the required file is transferred to its $RRM$
            end if
            Send the feedback to the score manager of source peer;
            Stop;
        else
            if $TTL \geq (TTL)_{upper}$ then
                Terminate the query process;
            else
                increase TTL
                GOTO top
            end if
        end if
    end if
end for
end Procedure
```

3.5.1.1 Complexity of the Algorithm for Selection of Source Peer

Proposition 5. *Algorithm 15 is performed in $O(n)$.*

Proof. As input, the algorithm considers ring groups, RRM, the number of hops, limit of source peers and the $global_{ref}$. Superpeers and RRMs are considered to be reputable peers, and in many cases, they will have many resources because the majority of of them are servers. Thus, as servers, they will be among source peers and the search of data or items can be fulfilled without transferring a query to a remote ring. This will increase the possibility of selecting source peers in local rings and also function to speed up the execution of the algorithm. □

3.5.2 Algorithm for updating the Global Trust of Peers

To improve Algorithm 16 in hierarchical Chord, the number of downloads after which the algorithm has to run is restricted, and we assume that the number of *Local Trust scores* that a peer should report must be limited to improve the performance of the algorithm. We emphasize on the fact that the interaction of a node with other nodes is delimited. A node commonly case communicates with a certain number of other nodes based on affinities. In a HR, all nodes have certain affinities and try to communicate among them before trying to interact with nodes in other HR's.

3.5.2.1 Complexity of the Algorithm

Proposition 6. *Algorithm 16 runs in $O(n^2)$.*

Proof. Each peer performs a certain number of operations, such as evaluating received files. Having assumed that the exchange among local rings will be improved, this algorithm will run fast. Furthermore, a number of peers (superpeers) function as servers with an extensive amount of huge amount of resources to be shared. □

3.6 NodeRanking Algorithm for Reputation Management in Hierarchical Chord

NodeRanking is based on both individual reputation and social relationship. In our architecture we assume that all normal peers point to the superpeers and RRM, because every

Algorithm 16: For updating the global trust of peers

```
1: Input :   Local Trust of peer
2: Input :   Superpeers
3: Input :   Group $Gi$
4: Input :   RRM $RRM_i$
5: Output :   Global trust score,manager peers
6: Procedure
7: Number of download to run the algorithm $(NDN_i) \leftarrow Const.$
8: Number of Local Trust to report per peer $\leftarrow Const.$
9: for each peer i do
10:     for all peer $j$, who is selected as source peer do
11:         Evaluate the received file;
12:         Assign the Local Trust value between $w_b$ to $w_g$;
13:         Send the local Trust to the score manager of peer $j$;
14:         if $Score\ manager\ in\ another\ Ring$ then
15:             send the local trust to $RRM_i$
16:         end if
17:     end for
18:     if Peer $i$ is score manager of peer $k$ then
19:         for all peer $j$, who is selected $k$ as source peer do
20:             Communicate with $RRM_i$;
21:             Receive the Local Trust values $T_{kj}$
22:             Locate their score manager;
23:         end for
24:         Initialisation;
25:         Set $p, q, previous\_t_k$, threshold ;
26:         while $error \geq threshold$ do
27:             Receive the Global Trust $t_j$ from their trust holder peer;
28:             compute
                $t_i = \left[\left(\frac{\sum_{j\in S_i} T_{ji}t_j}{\sum_{j\in S_i} t_j}\right)^p \cdot \left(\frac{\sum_{j\in S_i} t_j^2}{\sum_{j\in S_i} t_j}\right)^q\right]^{\frac{1}{p+q}}$
                $error \leftarrow |t_k - previous\_t_k|$
                $previous\_t_k \leftarrow t_k$
29:         end while
30:     end if
31: end for
32: end Procedure
```

normal node maintains links to its superpeers and RRM. Thus, In hierarchical Chord superpeers and RRM are pointed by many nodes. Their authority is set up to be higher, since superpeers and RRM are trustworthy nodes. The authority of a node is based on the authority of its in-nodes. In the first instance, all normal peers are initially granted the same authority. Algorithm 17 depicts NodeRanking in hierarchical Chord. The social network among nodes can be built by sharing news group, forums, personal web pages, images, emails, files and videos. As in[68], we assume that nodes share emails, links and names of members to construct a social network. The ranking will be based on information from these data sources. In H-Chord, the algorithm begins by selecting a random normal node in a HR. Based on its references, it continues to other nodes in the same HR. NodeRanking uses the Random walker strategy to browse the entire network. Each selected node performs a number of operations, such as allocation of a part of its authority to all nodes

Algorithm 17: NodeRanking algorithm for H-Chord

```
Input :  Superpeers $S_i$
Output :  Node reputation
Input :  Predetermined RRM
Input :  Group $G_i$
Input :  Group $G_i$
Input :  $Rpeer_j$ : Number of remote peers to send scores
Procedure
for each group  do
    $hops_j \leftarrow const$
    $Rpeer_j \leftarrow const$
    for each selected node $n$  do
        while (!converge()) do
            n = getNode(graph g);
            while (nnew!=null) do
                passAuthority(n);
                nnew = getNextNode(n,g);
                if (getNextNode(n,g))== null then
                    Jumping probability is reached
                else
                    the path is broken
                end if
                n = nnew
                if n belongs to a local HR then
                    go to n
                else
                    send a request to the RRM
                end if
            end while
            submit ranking result to a designated Superpeers $S_i$
        end while
    end for
end for
end Procedure
```

it points to.

NodeRanking needs to use a central entity to build the network graph; thus, in this architecture, superpeers in every HR are used for this purpose. They can be chosen or elected for centralizing node scores. They are employed to recuperate the results of the ranking process for all nodes in a local ring. Every node has its reputation carried by its respective superpeers or RRM. RRM are involved in propagating the trust of nodes from one HR to another. Since this architecture is Chord based, we assume that the random walker will follow the path based on information in finger tables of every node in a HR. It will travel from one node to another around the ring. Every ring runs the algorithm and transfer ranking results to its superpeers. The reputation of a node is based on the number of references from others nodes.

getNode() chooses a random normal node in a ring, and based on information on its finger table, it travels around the ring. **getNextNode(n,g)**determines the next node to be visited after node n using information from its out-edges set and the finger table.

We assume that a node may have out-neighbours from it own HR or from other HRs. If a node has local out-neighbours and distant neighbours, it must give higher priority to local out-neighbors (nodes in the same HR). Thus, its next out-neighbor should be a local node. Otherwise, a request should be sent to the RRM to keep the score ranking of the out-neighbours and to refer to the next ring. Since a RRM superpeer behaves like a member of another ring, it links two groups of nodes . The probability of a node being chosen (Pr_{choose}) is the same as in the NodeRanking flat algorithm . To determine the *jumping probability* we should take in account that a node may have *local out-edges* in the same HR and *remote out-edges* in other HRs. *passAuthority(n)* also deals with out-edges from different HRs. In H-Chord, the factor F is initialized for every node as the sum of the authority of all nodes in all the rings. We may assume that the factor F only includes all nodes in HRs that are connected directly to its HR, particularly in its direct neighborhood. To determine F, we may also limit the number of hops among HRs. In H-Chord, the algorithm is run in every node and RRM to transfer ranking results from one ring to another. In the same way, the **convergent()** function can be performed faster, because nodes are organized in small rings, and the number of hops are limited. Each HR determines the convergence of its nodes by means of superpeers and RRM superpeers. Every node notifies it convergence to its superpeers or RRM superpeer for exchange with other rings.

3.6.0.1 Complexity of NodeRanking Algorithm

Proposition 7. *This algorithm runs in $O(n^2)$.*

Proof. This algorithm browses all nodes and includes the jumping probability that depends on local information. A transition probability matrix is used to find the stationary state of the Markov chain. In hierarchical Structure, communication and exchange among peers is optimized to be executed locally. Also, this algorithm is based on social interaction in a community, so a ring is seen as a community and every ring has its own central server to gather all ranking process results. The number of remote peers to send scores for a local peer is determined and the number of nodes in a local ring is reduced relative to the number of nodes in the entire hierarchy □

3.7 Conclusion

In this chapter, we began by presenting the system model architecture environment of hierarchically Structured P2P Chord. We then discuss normal peers, superpeers and remote resource manager(RRM), and presented their characteristics and functions according to our system model architecture. Subsequently, we discussed the organization of nodes in groups by introducing the local group or home group, organized around an intra-group lookup service, which is composed of normal nodes, super nodes and one RRM node to connect with other local rings or to a super ring. We assumed that the super ring, also called the main ring, is only made up of with super nodes. We then explored about the lookup service and illustrated the hierarchical model lookup process and some assumptions we made for this architecture. After, we redesigned EigenTrust, PowerTrust, Absolute trust and NodeRanking to work in a hierarchical environment based on Chord. For each algorithm,according to its characteristics, we made appropriate assumptions and considerations to improve the algorithm performance in a hierarchical environment and presented their complexity. For EigenTrust we assumed that pre-trusted nodes will be maintained at each level of the hierarchy and the score is transferred from one ring to another by using RRM. For PowerTrust, we used three algorithms: one for selecting power nodes, another for initializing global reputation and the last for updating global reputation. The algorithm for selecting power nodes cannot be executed when the number of power nodes are greater or equal to the number of normal peers. For Absolute Trust, we employed two algorithms one for selecting source nodes and the second for updating the global trust. Its selecting source nodes added a variable to limit the number of hops between nodes in rings. Finally, we assume that in NodeRanking algorithms, all nodes are pointed to superpeers or RRM. We also suppose that each ring has a central entity to collect and maintain node scores. The next chapter will discuss the experiments and results to determine the performance of our trust and reputation algorithms.

In the following chapter, we will describe the experiments we performed and the results we obtained to illustrate the performance of our trust and reputation algorithms in a hierarchically structured P2P network. Moreover, we will compare the performance of each algorithm in a flat structured ring relative to its performance. in two many hierarchically structured rings.

Chapter 4

Experimental Results

4.1 Introduction

In these experiments, we will evaluate the performance of trust and reputation algorithms in an hierarchical structured P2P networks using Chord lookup service. We will use existing experiments for each trust and reputation algorithm applied on flat Peer-to-Peer systems and apply them in our scheme; we will compare the results.

4.2 Experimental Setup

To run our experiments, we used an open-source simulator called PeerfactSim [32][48] developed by Technischen Universitt Darmstadt. Written in Java, PeerfactSim.KOM is a simulator designed for large scale distributed P2P systems. Its goal is to evaluate interdependencies in multi-layered P2P systems. PeerfactSim is composed of a single-threaded P2P simulator and designed in a modular and scalable way. An XML configuration file is used to group all modules that constitute a simulation. Moreover, modules are reusable to produce new simulations.

PeerfactSim does not currently support hierarchical Chord. One of the tasks was to modify the source code with an implementation of the flat Chord P2P system structure and extend it to hierarchical Chord. This is so that the simulator can automatically generate a hierarchical P2P network with the possibility of allowing a user to specify the length of the tree and determine the number of nodes. At the end, some trust and reputation algorithms can be implemented for analysis.

All experiments were performed on a standalone computer with Intel(R) Core(TM) i5-3320M CPU @2.60GHz, 64 bits, using 8.00 GB. For coding we use Eclipse IDE for Java Developers, Version: Mars.1 Release(4.5.1) and jdk1.8.0_91. The Windows 7 Professional operating system was used.

For the experiments, we want to compare simulation results from a flat chord P2P system(with only one ring) to that of a hierarchical Chord P2P system. For this purpose, we consider that the hierarchy is limited to a three-tier DHT, so that we will have a three-level lookup service.

We will compare and analyze experimental results between flat algorithms and hierarchical algorithms. Moreover, we will compare the performance of all algorithms in the hierarchical network.To evaluate the performance of each algorithm, we use the fraction of download, residual curl and the malicious collective. Each experiment is executed 10 times. We assume that when a flat network has 100 nodes, the entire hierarchical network will have 100 nodes (all rings included). Then, we will increase the number of nodes to 500 and 1000 respectively in both networks, and compare results. For the sake of clarity and fair comparison, we assume that both flat and hierarchical networks have the same number of nodes and that all rings in the hierarchical network have the same number of nodes.

Table 4.1 represents the distribution of nodes in the hierarchical network.

Nodes repartition	
Nodes per hierarchy	Nodes per ring
100	8
500	16
1000	16

Table 4.1: Number of nodes per ring

4.3 Experimental Results and Analysis

In this, section, we perform experiments on the algorithms, obtain and interpret the results. To gain results, we run flat and hierarchical algorithms. In the experiments, we have placed our focus on fraction of download, residual curl and the malicious collective. Our experiments compare the fraction of download, the residual curl and malicious collective for each trust and reputation algorithm.

In the simulation test, 100, 500 and 1000 nodes are added to the flat peer-to-peer network and three-layer H-Chord. In order to make a fair comparison, we suppose that both the flat and H-Chord networks have the same number of peers. Because peers are joining and leaving the ring, the finger entries of the peers will not all be accurate.

4.3.0.1 Load Distribution

To capture the heterogeneity of the peers, we suppose that there are two categories of peers:

- Stable peers for node 1 to node 10;
- Unstable peers for the other nodes.

For the hierarchical organization, we select super-peers from a set of stable peers and we suppose that there is at least one stable peer in each group. For the flat organization, we choose nodes from the two categories randomly.

4.3.0.2 Convergence Speed

Convergence speed determines the number of iterations a trust and reputation algorithm runs to compute trust of nodes and then to provide their reputation. Convergence is another aspect of concern since nonlinearity may result in a large number of iterations and render the system inefficient. Considering the procedure used for the P2P network, the following factors are found to affect the convergence speed:

- Convergence criteria and tolerance: from each trust and reputation algorithm.
- Type of divisions in substructures: hierarchical and flat networks.

4.3.0.3 Malicious Collective

For all our simulations, we choose to have the same fraction of malicious peers in both hierarchical and flat Chord, in order to readily compare the fraction of download from peers.

4.4 Results for 100 Nodes

In this section, we present the results from our experiment using 100 nodes.

4.4.1 Load Distribution

4.4.1.1 EigenTrust

Figure 4.1 represents the results of the simulation. We can see that load distribution for the H-Chord network is concentrated on nodes with higher stability in each level of the hierarchy. Conversely, in the flat EigenTrust network, load distribution is selecting data sources with more scattered patterns.

4.4.1.2 PowerTrust

Figure 4.2 shows results for PowerTrust using 100 nodes in the network. The results demonstrate that for both algorithms, load distribution is mostly concentrated on a certain number of nodes (since the PowerTrust algorithm allows selection of a limited number of

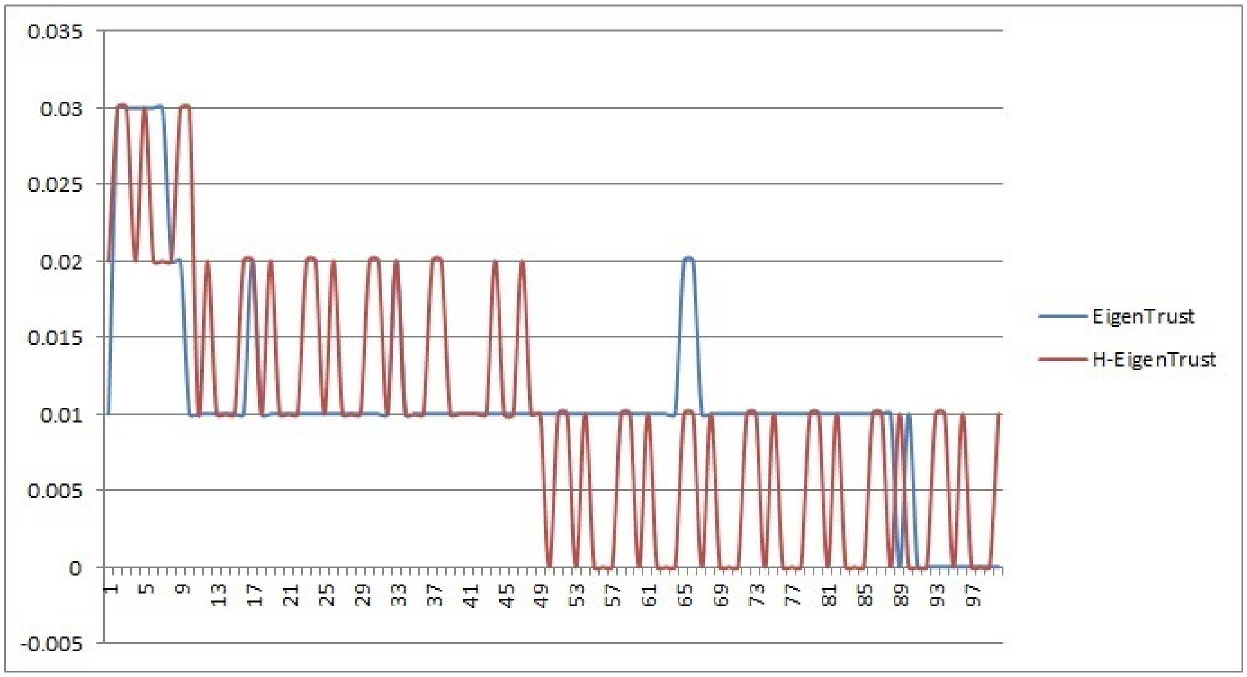


Figure 4.1: Results of Fraction of downloads in Flat and Hierarchical EigenTrust for 100 nodes

power nodes). Then, the load distribution stays constant for all other nodes. The flat PowerTrust load distribution curve is higher on the first nodes, then stays constant with a peak at a certain point. In H-PowerTrust, power nodes are selected in the first two levels of the hierarchy.

4.4.1.3 Absolute Trust

Figure 4.3 illustrates the fraction of downloads in flat and hierarchical Absolute Trust for 100 nodes. In H-absolute the load distribution is highly concentrated in a limited number of nodes (mostly superpeers). While in flat, we can see some scattered peaks on certain nodes.

4.4.1.4 NodeRanking

Figure 4.4 represents the fraction of downloads in flat and hierarchical NodeRanking for 100 nodes. Load distribution is centralized around nodes with higher stability in H-

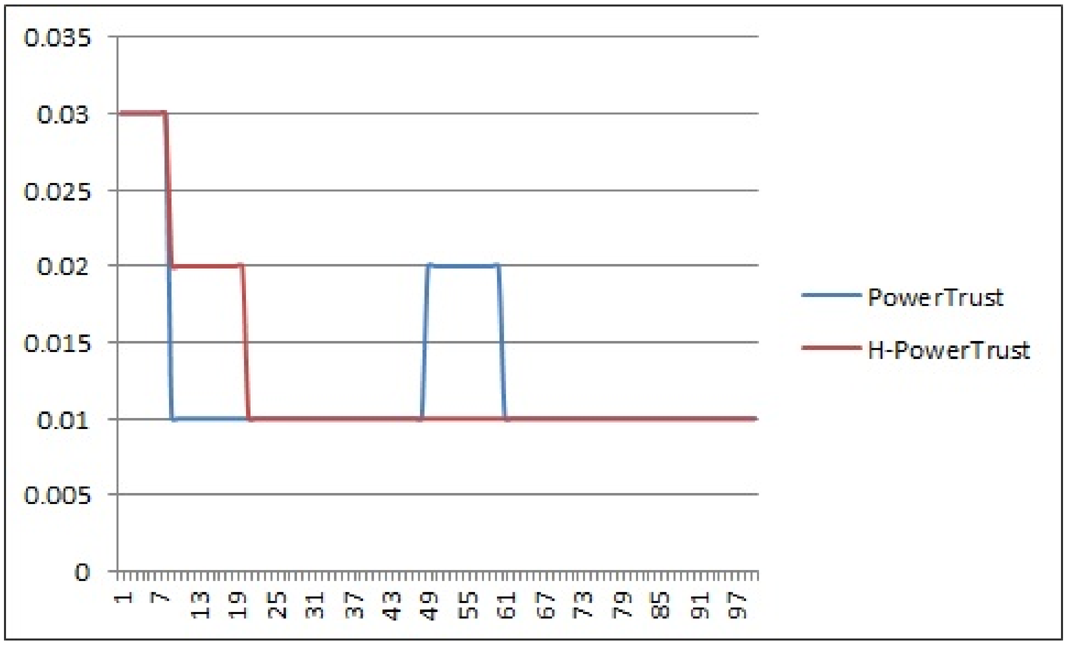


Figure 4.2: Results of Fraction of downloads in Flat and Hierarchical PowerTrust for 100 nodes

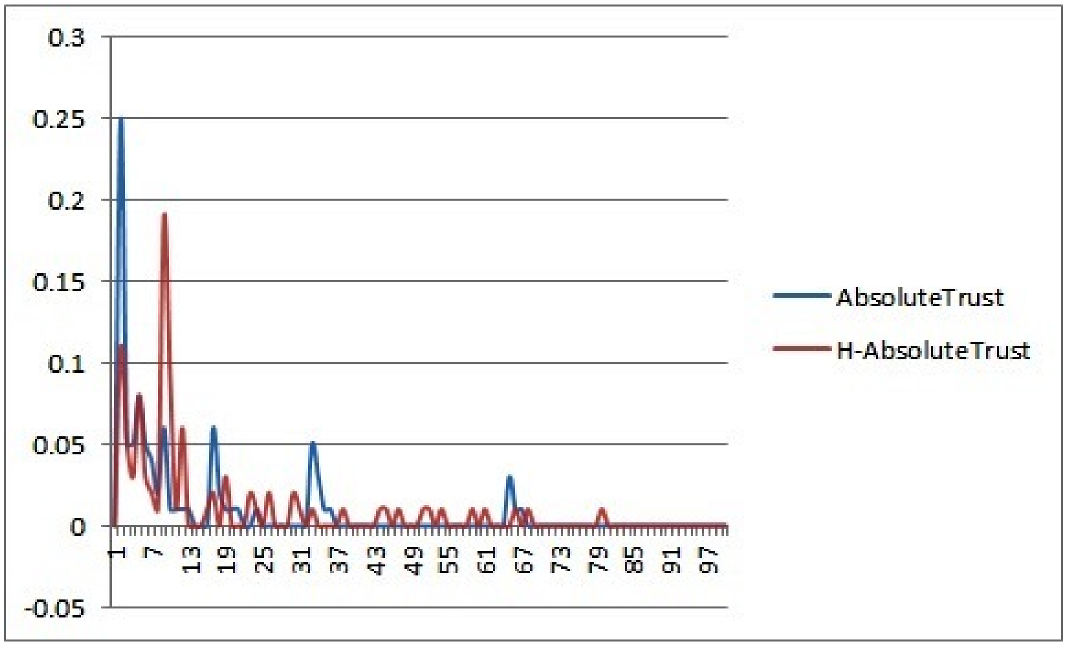


Figure 4.3: Results of fraction of downloads in flat and hierarchical Absolute Trust for 100 nodes

NodeRanking. In H-NodeRanking, the load distribution is limited to certain nodes in the different levels of the hierarchy, based on the importance of nodes in the rings, and while other nodes are ignored.

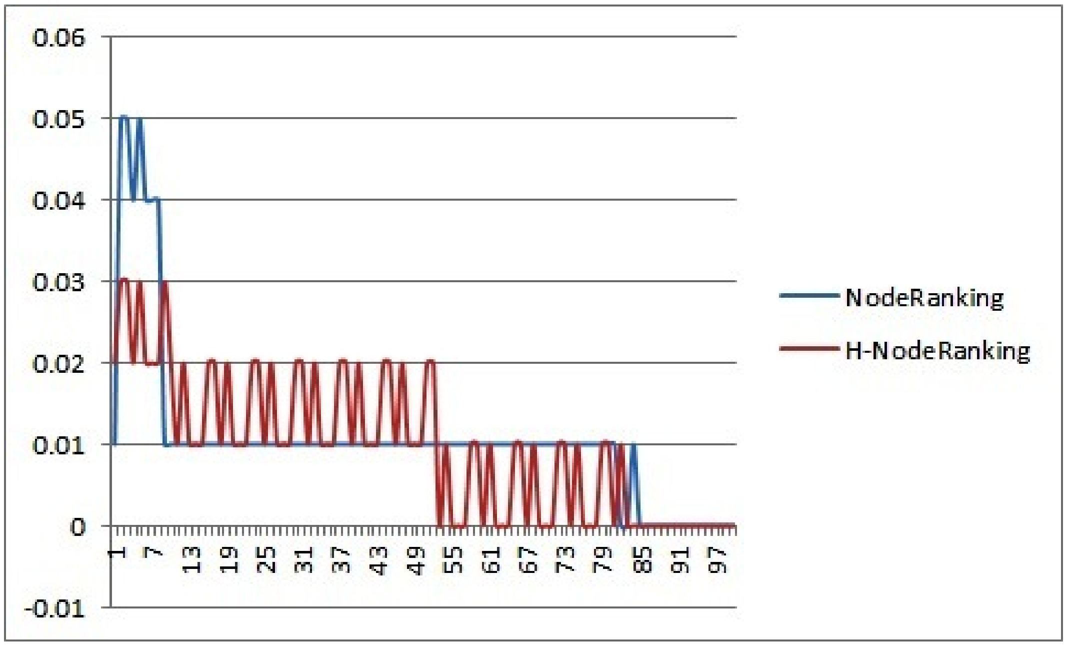


Figure 4.4: Results of fraction of downloads in flat and fierarchical NodeRanking for 100 nodes

4.4.2 Load Distribution Comparison of all Hierarchical Algorithms for 100 nodes

Figure 4.5 represents a comparison of all hierarchical algorithms in a network composed of 100 nodes. In this data, we can observe that for a network 100 nodes, NodeRanking presents the best performance of fraction of downloads, while certain algorithms show a slight upward trend at the beginning, followed by a gradual drop. Although the curve is not regular, we can observe peaks on more stable nodes. H-Absolute Trust comes in the second position followed by PowerTrust in the third position. At this stage of experiments, H-EigenTrust demonstrates the worst performance.

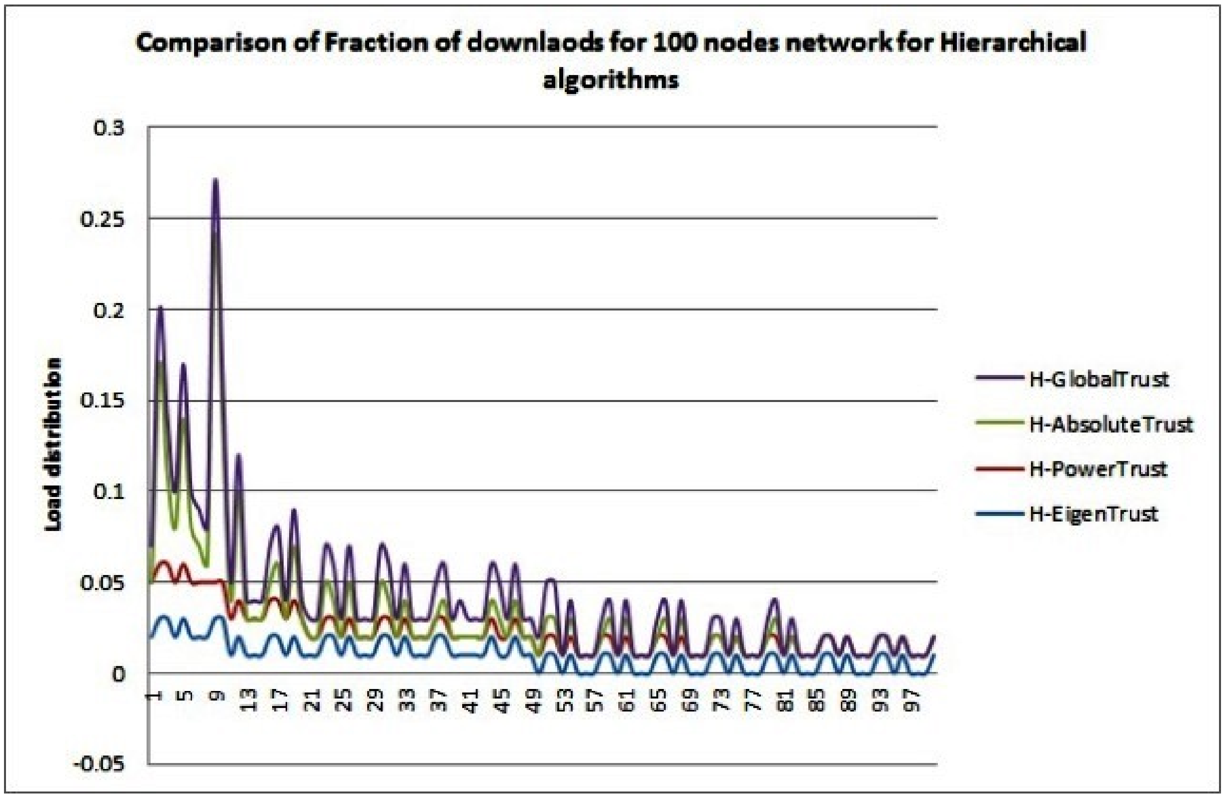


Figure 4.5: Comparison of fraction of download of all hierarchical algorithms in a 100 nodes P2P network

4.4.3 Malicious Collective

In this section, we compare all results from all four trust and reputation algorithms for 100 nodes, for both flat and hierarchical networks. Figure 4.6 depicts the fraction of inauthentic downloads when the fraction of malicious peers is about 43.56 %. We can see the difference between a flat and hierarchical algorithm and compare the percentage of inauthentic downloads. The hierarchical EigenTrust, Absolute Trust and NodeRanking are improved significantly in the hierarchical network. Hierarchical PowerTrust has a slight improvement compared to the flat PowerTrust.

In Figure 4.7, we only compare the results from trust and reputation algorithms in a hierarchically structured P2P networks. We can see that among all trust systems, Absolute trust presents the greatest diminution of inauthentic downloads, followed by NodeRanking, EigenTrust, with PowerTrust. PowerTrust has the worst performance.

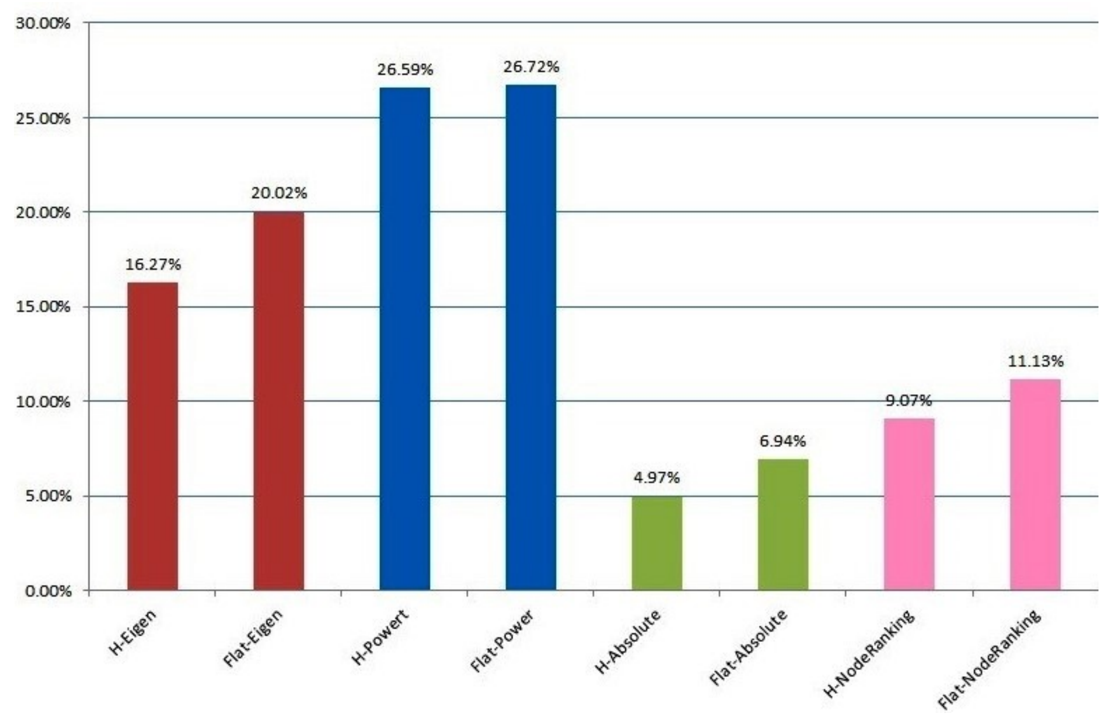


Figure 4.6: Trust-based reduction of inauthentic downloads in a network with 100 nodes where a fraction of peers form a malicious collective

4.4.4 Convergence Speed

Figure 4.8 depicted the convergence speed of each algorithm in both a hierarchical and flat 100 nodes network. Hierarchical algorithms present a fast convergence relative to flat algorithms, except for PowerTrust. Figure 4.8a demonstrates that H-EigenTrust converges faster than Flat-EigenTrust, just after about 3 iterations. In Figure 4.8b, H-PowerTrust converges after 3 iterations while Flat-PowerTrust takes more iterations. The same can be said for Figure 4.8c when we compare H-AbsoluteTrust and Flat-AbsoluteTrust. Figure 4.9 shows the convergence speed of all hierarchical algorithms. In a network with 100 nodes, H-EigenTrust, H-PowerTrust and H-Absolute have almost the same convergence speed while H-NodeRanking presents the slowest speed. They converge after less than 3 iterations while H-NodeRanking converges after 3 iterations.

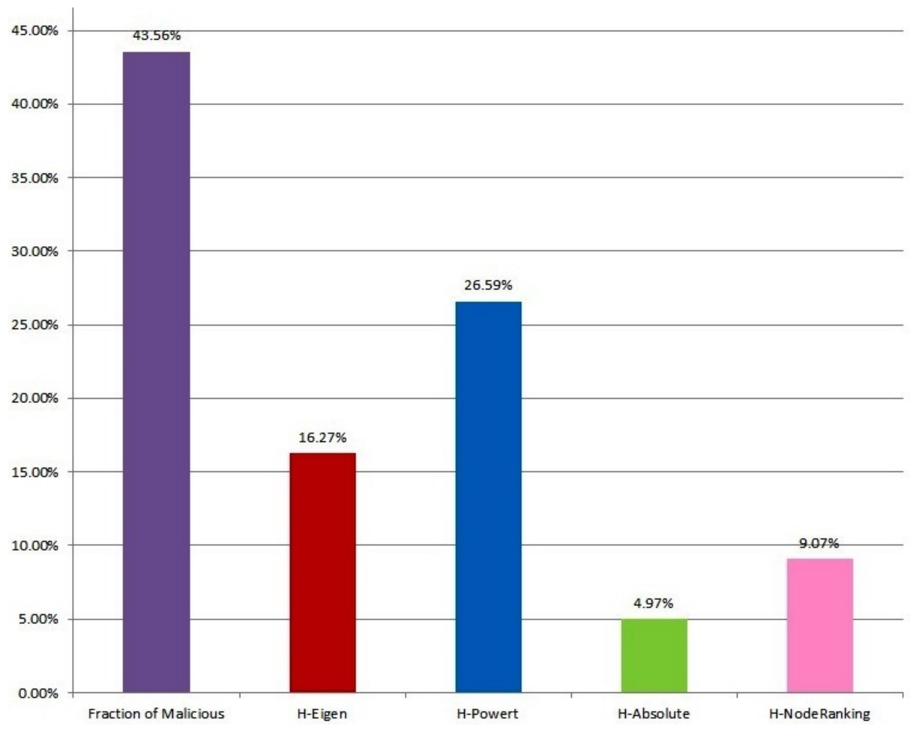


Figure 4.7: Trust-based reduction of inauthentic downloads in hierarchical network with 100 nodes where a fraction of peers forms a malicious collective

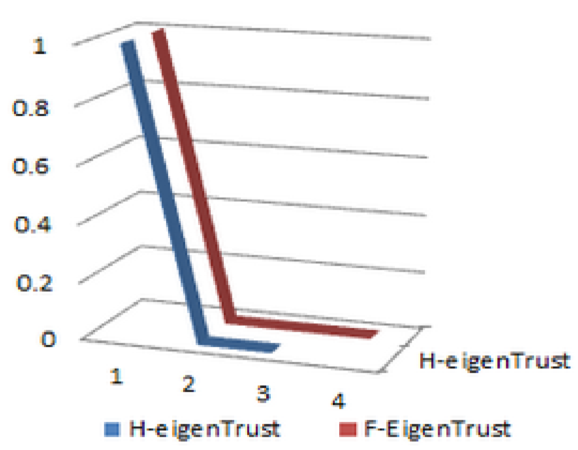
4.5 Results for 500 Nodes

In this section, we increase the number of nodes to 500. We will present the results from experiment using 500 nodes. We limit the number of nodes to 16 in each ring.

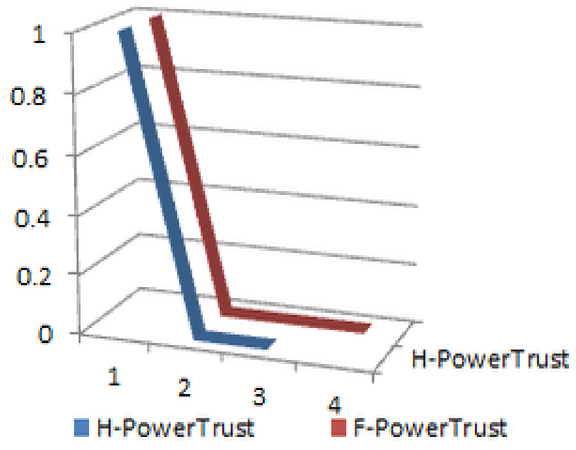
4.5.1 Load Distribution

4.5.1.1 EigenTrust

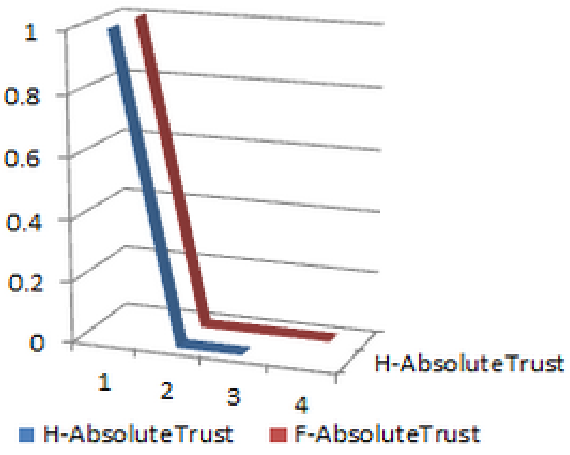
Figure 4.10 illustrates the outcome of fraction of downloads in flat and hierarchical EigenTrust for 500 nodes. When the number of nodes increases, EigenTrust changes and extends the load distribution to more stable nodes as depicted by Figure 4.10a, while in a flat system, depicted in Figure 4.10b, the distribution is scattered, with peaks in certain nodes. This implies that superpeers are mostly the first peers to be solicited for downloading because of their characteristics, importance and long presence in the network.



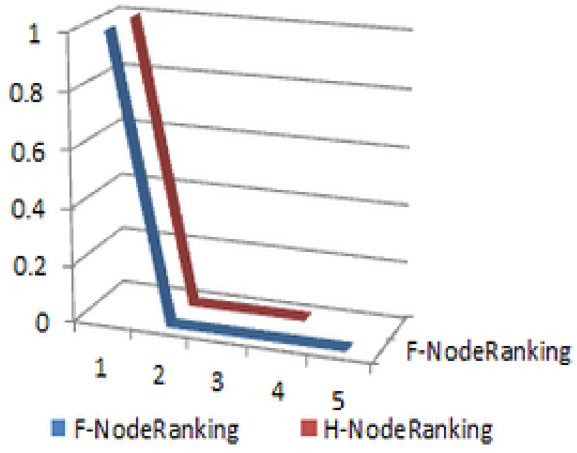
(a) EigenTrust Convergence



(b) Power Trust Convergence



(c) Absolute Trust Convergence



(d) NodeRanking Convergence

Figure 4.8: Convergence speed in 100 nodes network

4.5.1.2 PowerTrust

Figure 4.11 provides results of fraction of downloads in flat and hierarchical PowerTrust for 500 nodes Load distribution is using a slight number of stable nodes, then decreases quite sharply and stay constant, as depicted in Figure 4.11a and Figure 4.11b.

4.5.1.3 Absolute Trust

Figure 4.12 shows experiment results of Absolute Trust. Figure 4.12a and Figure 4.12b depict the results of experiments to represent the load distribution.

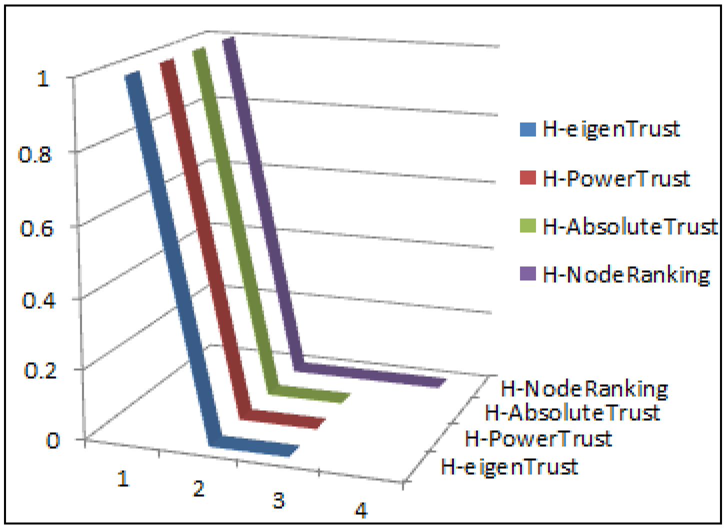
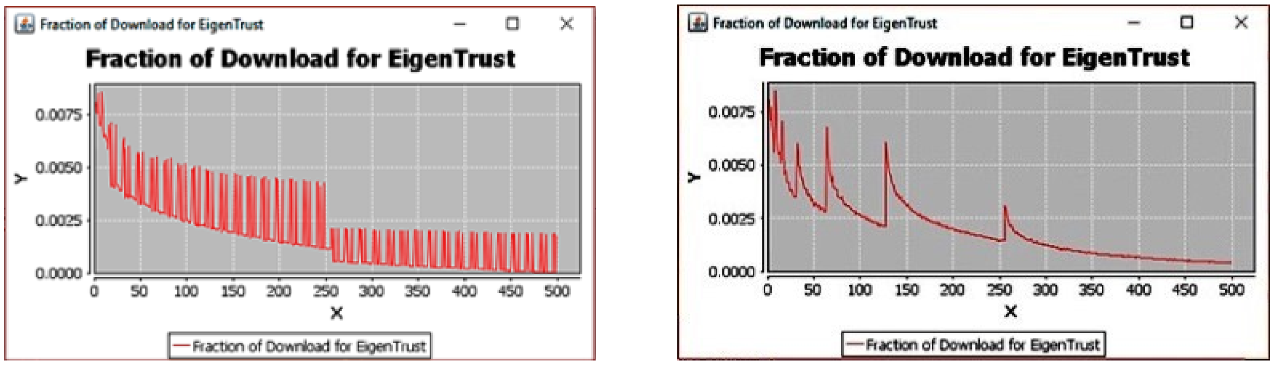


Figure 4.9: Convergence speed of algorithms in 100 nodes hierarchical network



(a) Hierarchical EigenTrust

(b) Flat EigenTrust

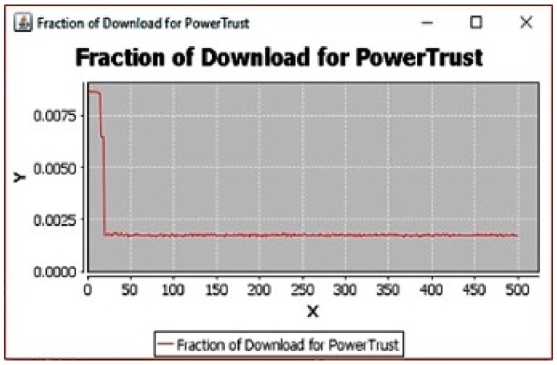
Figure 4.10: Results of fraction of downloads in flat and hierarchical EigenTrust for 500 nodes

4.5.1.4 NodeRanking

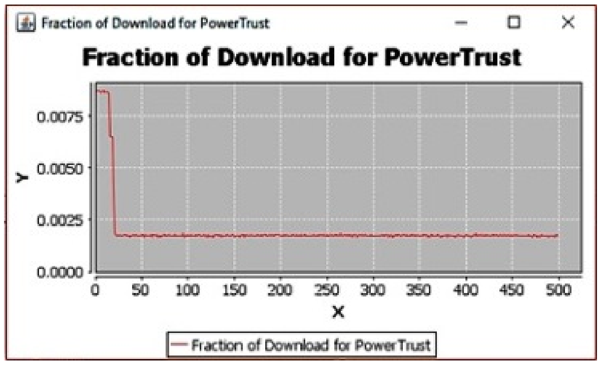
Figure 4.13a and Figure 4.13b depict the results of experiments to represent the load distribution.

4.5.2 Malicious Collective

Figure 4.14 shows that, with more nodes, we can see that the percentage of inauthentic downloads increases very slightly compared to 100 nodes for EigenTrust, NodeRanking and Absolute trust, for both the flat and hierarchical P2P. The flat and hierarchical PowerTrust

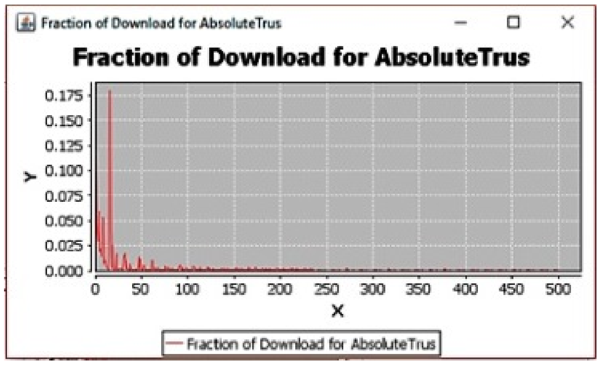


(a) Hierarchical PowerTrust

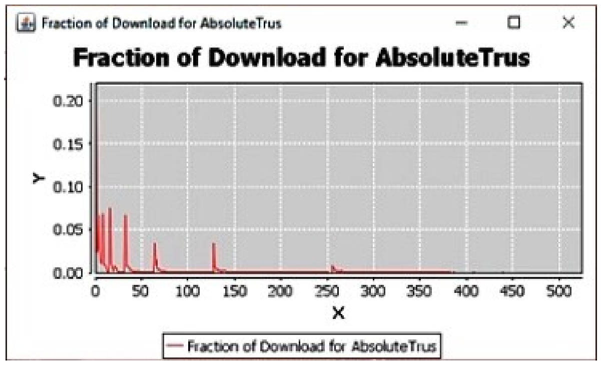


(b) Flat PowerTrust

Figure 4.11: Results of Fraction of downloads in Flat and Hierarchical PowerTrust for 500 nodes

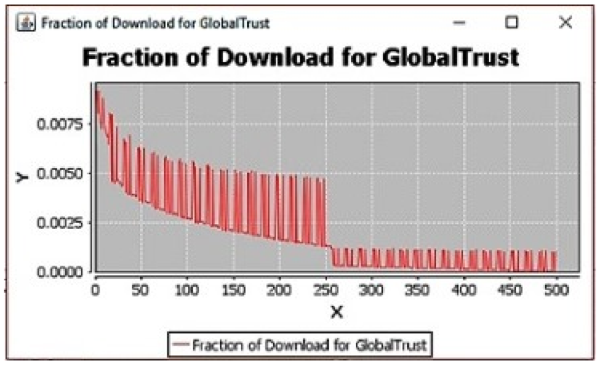


(a) Hierarchical Absolute Trust

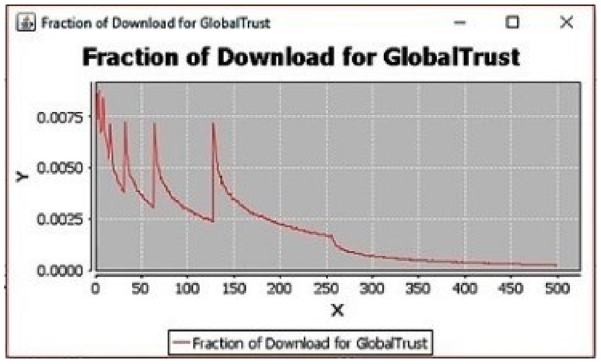


(b) Flat Absolute Trust

Figure 4.12: Results of Fraction of downloads in Flat and Hierarchical Absolute Trust for 500 nodes



(a) H-NodeRanking with 500 nodes



(b) Flat NodeRanking with 500 nodes

Figure 4.13: Results of Fraction of downloads in Flat and Hierarchical NodeRanking for 500 nodes

increases dramatically. Absolute Trust decreases the number of inauthentic downloads for both flat and hierarchical networks.

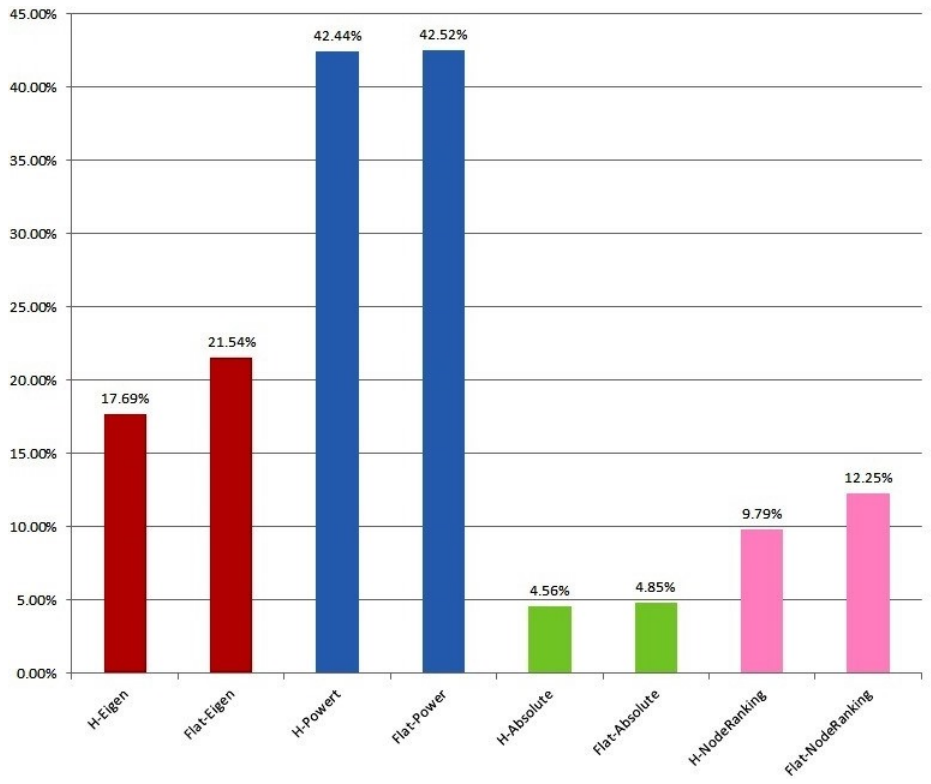


Figure 4.14: Trust-based reduction of inauthentic downloads in hierarchical network with 500 nodes where a fraction of peers forms a malicious collective

Figure 4.15 Illustrates the comparison of hierarchical trust and reputation algorithms. As we can see, the hierarchical Absolute Trust decreases the number of inauthentic downloads, while Hierarchical EigenTrust and NodeRanking increases it slightly.

4.5.3 Convergence Speed

Figure 4.16 represents the convergence speed of all hierarchical algorithms. With 500 nodes, H-NodeRanking and H-PowerTrust present better convergence speed than H-EigenTrust and H-AbsoluteTrust.

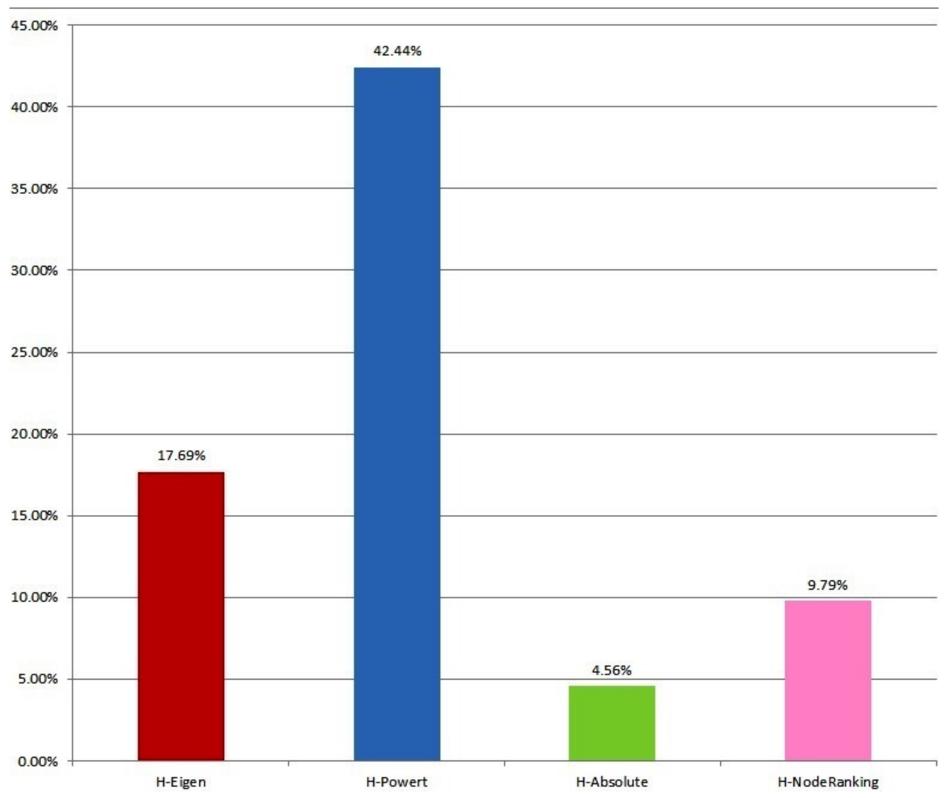


Figure 4.15: Trust-based reduction of inauthentic downloads in hierarchical network with 500 nodes where a fraction of peers form a malicious collective

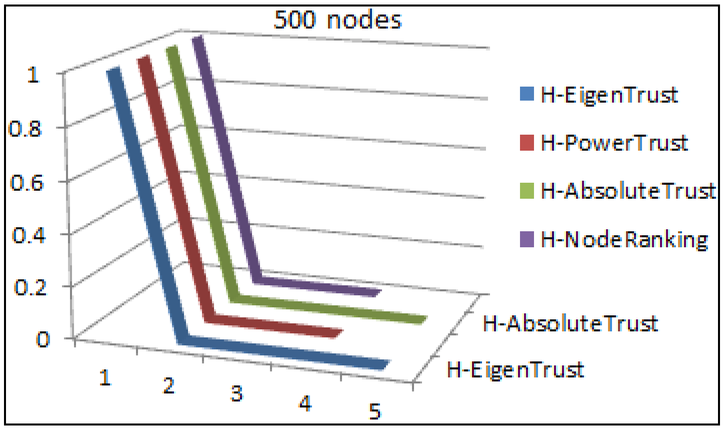


Figure 4.16: Convergence speed of algorithms in 500 nodes hierarchical network

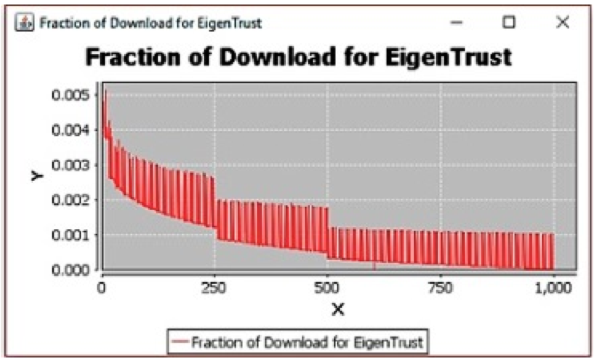
4.6 Results for 1000 Nodes

In this third experiment, we increase the number of nodes to 1000 nodes and apply our algorithms to obtain new results.

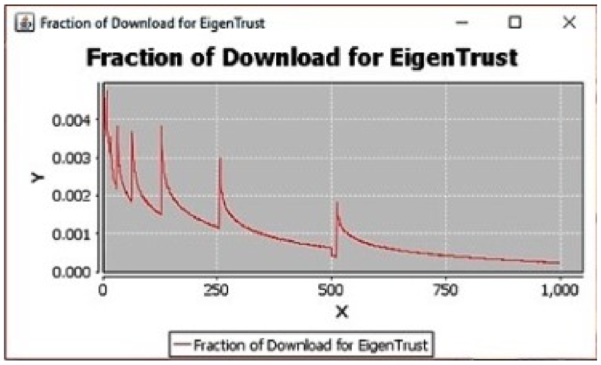
4.6.1 Load Distribution

4.6.1.1 EigenTrust

Figure 4.17a and Figure 4.17b depict the results of the experiment.



(a) H-EigenTrust with 1000

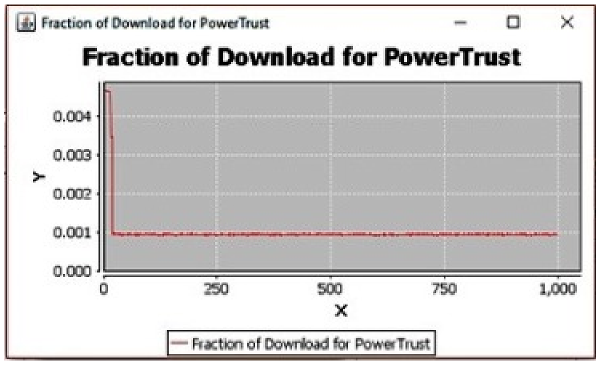


(b) Flat EigenTrust with 1000

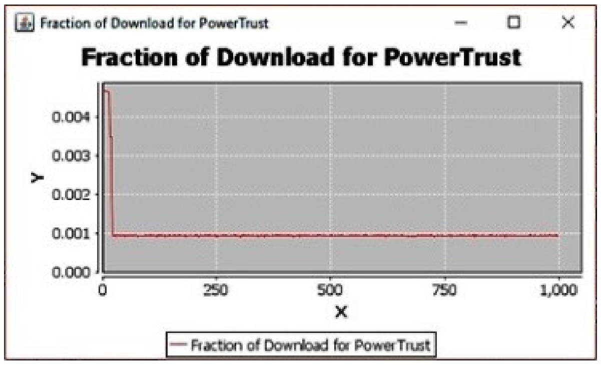
Figure 4.17: Results of fraction of downloads in flat and hierarchical EigenTrust for 1000 nodes

4.6.1.2 PowerTrust

Figure 4.18a and Figure 4.18b depict the results of experiments to represent the load distribution.



(a) H-PowerTrust with 1000 nodes

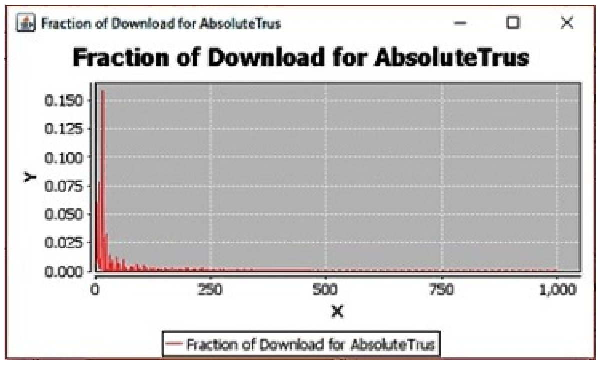


(b) Flat PowerTrust with 1000 nodes

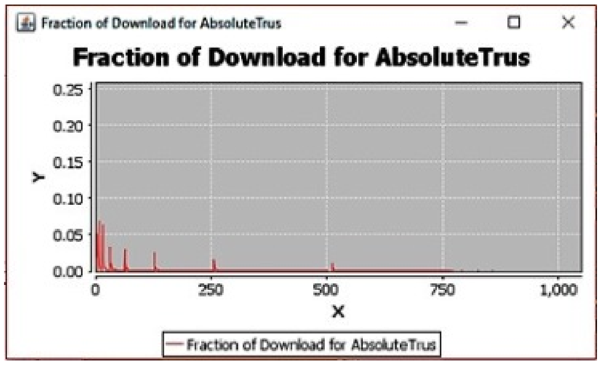
Figure 4.18: Results of Fraction of downloads in Flat and Hierarchical PowerTrust for 1000 nodes

4.6.1.3 Absolute Trust

Figure 4.19a and Figure 4.19b depict the results of experiments to represent the load distribution.



(a) H-Absolute Trust with 1000 nodes

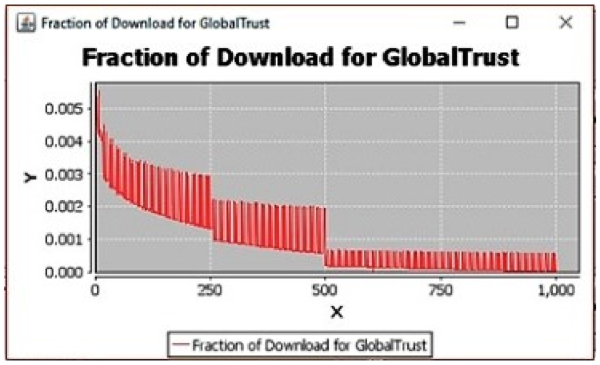


(b) Flat Absolute Trust with 1000 nodes

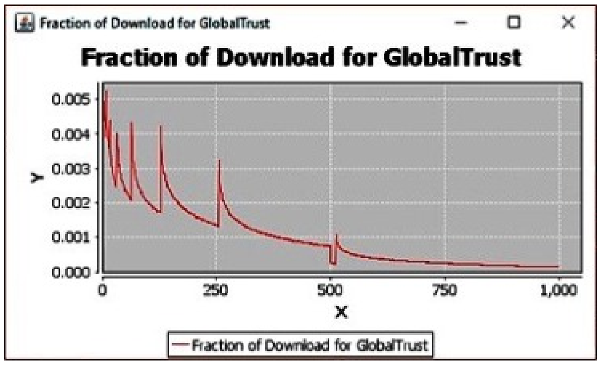
Figure 4.19: Results of Fraction of downloads in Flat and Hierarchical Absolute Trust for 1000 nodes

4.6.1.4 NodeRanking

Figure 4.20a and Figure 4.20b depict the results of experiments to represent the load distribution.



(a) H-NodeRanking with 1000 nodes



(b) Flat NodeRanking with 1000 nodes

Figure 4.20: results of fraction of downloads in flat and hierarchical NodeRanking for 1000 nodes

4.6.2 Malicious Collective

With 1000 nodes, we increase the fraction of the malicious collective to 50 %. Figure 4.21 shows the outcome of the experiments. We can observe that all trust and reputation hierarchical algorithms present better performance than flat algorithms. In Figure 4.22 4.22, we compare only the performance of hierarchical algorithms and can see that Absolute trust achieves the highest performance, followed by NodeRanking, EigenTrust and PowerTrust.

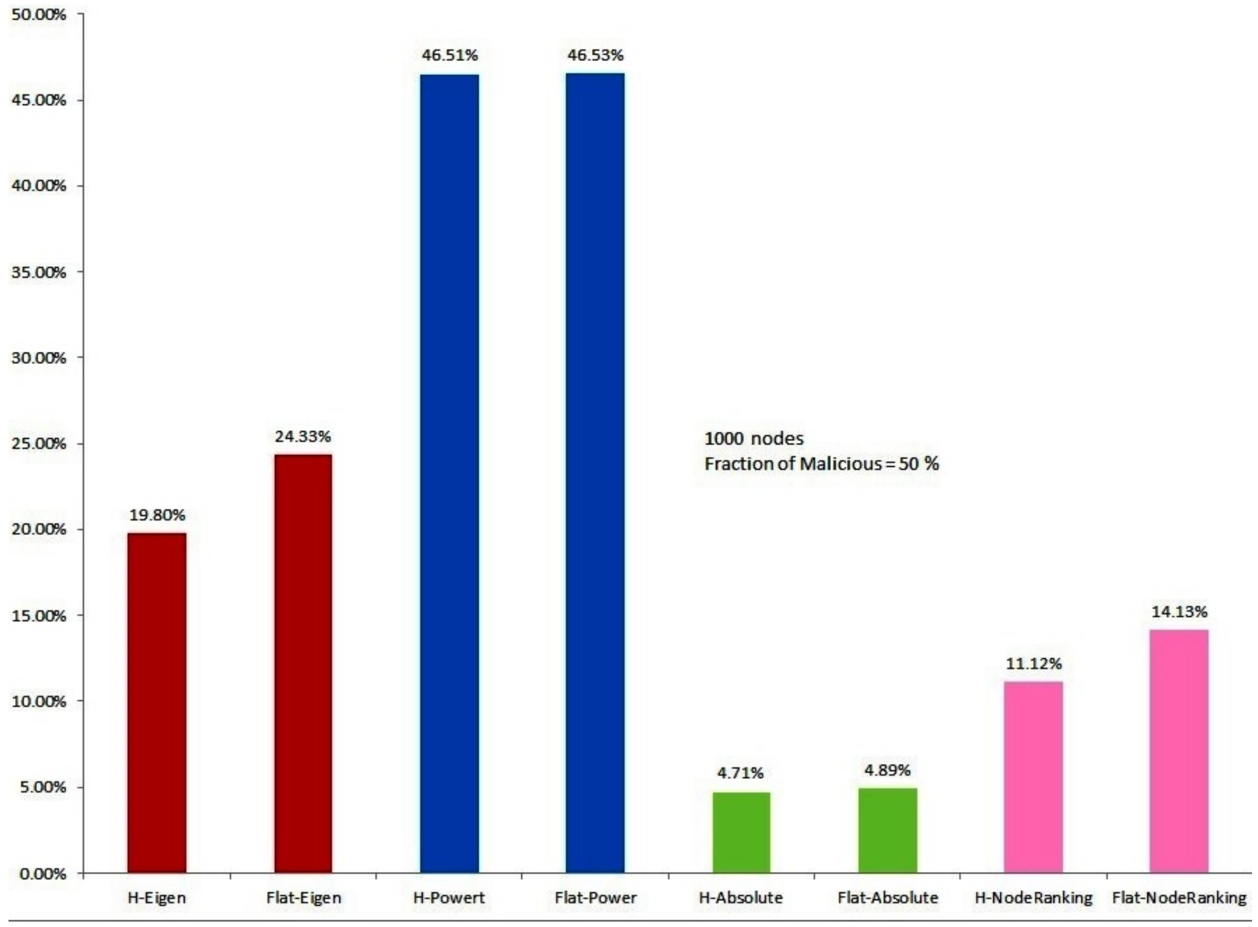


Figure 4.21: Trust-based reduction of inauthentic downloads in hierarchical and flat network with 1000 nodes where a fraction of peers form a malicious collective

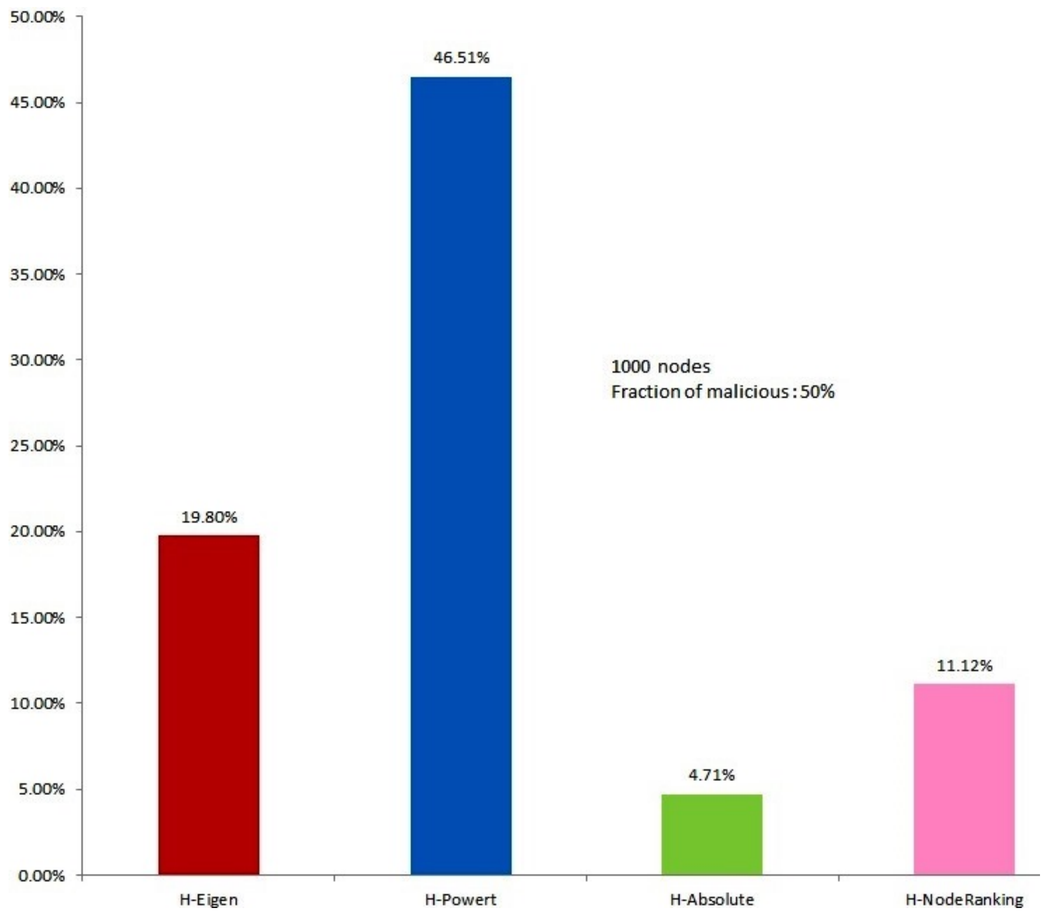


Figure 4.22: Trust-based reduction of inauthentic downloads in hierarchical network with 1000 nodes where a fraction of peers form a malicious collective

4.6.3 Convergence Speed

Figure 4.23 depicts the convergence speed of all hierarchical algorithms. H-EigenTrust and H-AbsoluteTrust have the most optimal convergence speed among all algorithms. They converge in less than 5 iterations, while H-PowerTrust and H-NodeRanking take more time to converge.

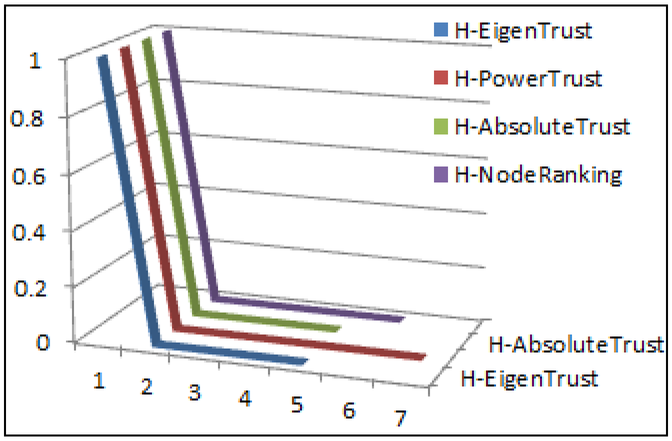


Figure 4.23: Convergence speed of algorithms in 1000 nodes hierarchical network

4.7 Conclusion

In this chapter, we presented the results of our experiment. We began by introducing our experimental setup and presenting the simulator, our computer features and the programming environment. We then described the distribution of nodes in the hierarchical network and explained our method of performing the experiments and analyzing and interpreting the results. We focused our experiments on the fraction of downloads,the residual and malicious collective. We simulated flat and hierarchical trust and reputation algorithms. We compared each flat algorithm with its hierarchical correspondent using 100, 500 and 1000 nodes, respectively, in a hierarchically structured P2P network, and presented the results. The comparison we studied was based on load distribution, malicious collective and residual curves to determine the convergence speed. Each simulation is illustrated by a chart. Evaluation of the results revealed that almost all hierarchical algorithms presented better performance than flat algorithms. The tests we performed revealed that Absolute Trust achieved the most optimal performance among all the tested algorithms. Absolute Trust was followed by EigenTrust, which also achieved a satisfactory performance.

Chapter 5

Conclusion and Future Work

5.1 Summary

In this thesis, we began by introducing the peer-to-peer computing paradigm, expounding on its architecture and taxonomy and the centralized, decentralized and hybrid systems within it. We also discussed routing in decentralized P2P systems; routing is performed in unstructured or structured P2P systems. Topology is not defined in an unstructured P2P, while in a structured P2P, topology is defined around a DHT. We introduced some overlay lookup systems based on DHT, such as CAN, Pastry, Tapestry and Chord, and discussed Chord at great length in particular. Chord arranges all peers around a ring and keeps all keys. It executes only one operation: it associates keys with corresponding nodes, and each node maintains a finger table that contains successor and predecessor nodes. The concept of routing in hierarchically structured P2P systems based on DHT and BATON was also explicated. We presented hierarchically structured characteristics and introduced existing hierarchical models based on Chord. We applied the hierarchical DHT with Chord, which inherited all Chord characteristics, and extended them to a hierarchical structure where peers are assembled in groups recognized by a unique identifier. The hierarchical organization solves the challenge faced in flat P2P systems where the number of nodes increases and affects the performance of the entire network. Moreover, it improves the query latency in group nodes in the hierarchy and generates less messages in the wide area. At different levels of the hierarchy, each group may run its own overlay and lookup service.

As for trust and reputation systems, we discussed their characteristics, measurements, and their classification in two categories: the credential model and the reputation model. We presented a number of definitions about trust and reputation and selected pertinent ones for our own purpose. Subsequently, we detailed some primary trust systems based on individual reputation as well as on both individual and social relationship reputation. In particular, we chose to use the EigenTrust, PowerTrust, Absolute trust and NodeRanking trust systems. Next, we introduced the hierarchical Chord architecture model we defined by presenting different component groups and node characteristics. We defined the types of nodes as normal and super nodes. For super nodes, we named nodes that connect different rings in the hierarchy as RRM nodes.

This research is comprised of two stages. First, we selected and redesigned EigenTrust, PowerTrust, Absolute trust and NodeRanking trust and reputation algorithms for a hierarchically structured P2P system based on Chord overlay, then analyzed and determined their complexity. For each algorithm we made assumptions to make it suitable for the hierarchical Chord. We simulated the algorithms and directed our experiments toward aspects of fraction of download, residual curl and the malicious collective. Second, we compared the results for our trust and reputation algorithms in flat and hierarchal P2P systems. We used the load distribution, residual curl, and malicious collective downloads to evaluate and test the performance of each algorithm. The simulation test included 100, 500 and 1000 nodes respectively. The results revealed that hierarchical trust and reputation algorithms achieve better performance than flat algorithms and converge faster than flat algorithms. We can confirm that the reduced number of nodes in hierarchical rings improve the performance of P2P systems. Finally, the experimental results demonstrated that among all four hierarchical trust algorithms, Absolute Trust had the best performance, followed by NodeRanking, EigenTrust and PowerTrust.

5.2 Future Work

Before coming to a close, we would like to offer some possible directions for future investigation in this area of research:

- *Extension to n+ 3 tiers architecture:*

This research implemented a hierarchically structured P2P network limited to a three-tier architecture to run trust and reputation algorithms. An extension to more tiers or levels could be an interesting direction for future research.

- *implement with BATON or HD Tree:*

We redesigned and analyzed trust and reputation algorithms for hierarchically structured peer-to-peer networks using Chord as a lookup service. Future work can be focused on the redesign of these algorithms using other types of lookup services based on hierarchies such as HD Tree or BATON which has been gaining popularity. Trust and reputation could be applied to HD Tree or BATON, and compared to the performance of the lookup services used in this paper.

- *Extend to a heterogeneous hierarchically structured P2P network:*

This research only focuses on a homogeneous hierarchically structured P2P network based on Chord as a lookup service. Future studies can also implement trust and reputation algorithms in a heterogeneous hierarchically structured P2P network using rings running different lookup services. While, for instance, the main ring is running Chord or another lookup service at the first level, each ring at level two can run a different lookup service such as HD Tree, CAN, BATON, Pastry or Tapestry.

- *apply to a IoT application:*

Applications of IoT are gaining traction by involving many devices or features such as sensors. A suitable area in which to apply this concept would be the fleet management problem in the IoT environment. Fleet management in IoT could be a specific application that can rely on trust and be applied on hierarchically structured P2P networks. Vehicles or tracks can be considered as the leaf nodes of the tree and be organized around a local ring with super-peers that can function as the points at which data can be merged on the local ring. Moreover, many rings can be connected together and constitute a huge hierarchically structured P2P network. Examining reputation and trust in this context could be an attractive research direction.

References

[1] Napster home page . http://www.napster.com/, 2016. Accessed: February 15, 2016.

[2] Gnutella project. http://www.gnu.org/philosophy/gnutella.en.html, 2016. Accessed: February 26, 2016.

[3] Skip list. https://en.wikipedia.org/wiki/Skip_list/, 2016. Accessed: March 01, 2016.

[4] EigenTrust. https://https://en.wikipedia.org/wiki/EigenTrust/, 2016. Accessed: March 07, 2016.

[5] Lecture 3: PageRank Algorithm -The Mathematics of Google Search. http://www.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture3/lecture3.html, 2016. Accessed: May 29, 2016.

[6] Karl Aberer, Philippe Cudré-Mauroux, Anwitaman Datta, Zoran Despotovic, Manfred Hauswirth, Magdalena Punceva, and Roman Schmidt. P-Grid: A Self-organizing Structured P2P System. *ACM SIGMOD Record*, 32(3):29, 2003.

[7] William Acosta and Surendar Chandra. On the need for query-centric unstructured peer-to-peer overlays. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8, April 2008.

[8] Derar K. Al-Omari, Vijay K. Gurbani, and Tricha Anjali. A novel architecture for a computer network defense (CND) system using Content Addressable Networks (CAN). *2012 IEEE Globecom Workshops, GC Wkshps 2012*, pages 758–762, 2012.

[9] Adrian Alexa and Anja Theobald. Reputation Management in P2P Networks : The EigenTrust Algorithm by. *WWW '03 Proceedings of the 12th international conference on World Wide Web*, pages 1–14, 2003.

[10] David P. Anderson. BOINC: A system for public-resource computing and storage. In *5th International Workshop on Grid Computing (GRID 2004), 8 November 2004, Pittsburgh, PA, USA, Proceedings*, pages 4–10, 2004.

[11] Marc Sánchez Artigas, Pedro García López, and Antonio Fernandez Gómez-Skarmeta. A comparative study of hierarchical DHT systems. In *32nd Annual IEEE Conference on Local Computer Networks (LCN 2007), 15-18 October 2007, Clontarf Castle, Dublin, Ireland, Proceedings*, pages 325–333, 2007.

[12] Sateesh Kumar Awasthi and Yatindra Nath Singh. Absolute trust: Algorithm for aggregation of trust in peer-to- peer networks. *Computing Research Repository*, abs/1601.01419, 2016.

[13] Sateesh Kumar Awasthi and Yatindra Nath Singh. Generalized analysis of convergence of absolute trust in peer-to-peer networks. *IEEE Communications Letters*, 20(7):1345–1348, 2016.

[14] Elisa Bertino, Elena Ferrari, and Anna Cinzia Squicciarini. Trust-x: A peer-to-peer framework for trust establishment. *IEEE Trans. on Knowl. and Data Eng.*, 16(7):827–842, July 2004.

[15] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *1996 IEEE Symposium on Security and Privacy, May 6-8, 1996, Oakland, CA, USA*, pages 164–173, 1996.

[16] Azzedine Boukerche and Yunfeng Gu. Hierarchically distributed tree. *Proceedings - IEEE Symposium on Computers and Communications*, pages 91–96, 2011.

[17] Colin Boyd, Audun Jøsang, and Roslan Ismail. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

[18] Amira Bradai. *Secured trust and reputation system : analysis of malicious behaviors and optimization. (Gestion de la confiance et de la réputation sécurisée : analyse des attaques possibles et optimisation).* PhD thesis, Telecom & Management SudParis, Évry, Essonne, France, 2014.

[19] John F. Buford and Heather Yu. Peer-to-peer networking and applications: Synopsis and research directions. In *Handbook of Peer-to-Peer Networking*, pages 3–45. Springer US, 2010.

[20] Cristiano Castelfranchi and Rino Falcone. Principles of Trust for MAS: Cognitive Anatomy, Social Improtance, and Quantification. *International Conference on Multi-Agent Systems*, pages 72–79, 1998.

[21] Miguel Castro, Peter Druschel, Ayalvadi J. Ganesh, Antony I. T. Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *5th Symposium on Operating System Design and Implementation (OSDI 2002), Boston, Massachusetts, USA, December 9-11, 2002*, 2002.

[22] Gang Chen, Tianlei Hu, Dawei Jiang, Peng Lu, Kian-Lee Tan, Hoang Tam Vo, and Sai Wu. BestPeer++: A Peer-to-Peer Based Large-Scale Data Processing Platform. *2012 IEEE 28th International Conference on Data Engineering*, pages 582–593, 2012.

[23] Kang Chen, Haiying Shen, Karan Sapra, and Guoxin Liu. A Social Network Based Reputation System for Cooperative P2P File Sharing. *IEEE Transactions on Parallel and Distributed Systems*, 26(8):2140–2153, 2015.

[24] Yi Hui Chen, Eric Jui Lin Lu, Yao Tsan Chang, and Shiuan Yin Huang. RDF-Chord: A hybrid PDMS for P2P systems. *Computer Standards and Interfaces*, 43:53–67, 2016.

[25] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*, pages 46–66, 2000.

[26] Adina Crainiceanu, Prakash Linga, Johannes Gehrke, and Jayavel Shanmugasundaram. Querying peer-to-peer networks using p-trees. In *Proceedings of the Seventh International Workshop on the Web and Databases, WebDB 2004, June 17-18, 2004, Maison de la Chimie, Paris, France, Colocated with ACM SIGMOD/PODS 2004*, pages 25–30, 2004.

[27] Claudia Eckert, Sokratis K. Katsikas, and Günther Pernul, editors. *Trust, Privacy, and Security in Digital Business*, volume 8647 of *Lecture Notes in Computer Science*. Springer International Publishing, Cham, springer edition, 2014.

[28] Eyal Even-Dar, Alex Kesselman, and Yishay Mansour. *Peer-to-Peer Systems and Applications*. Springer Berlin Heidelberg, 2003.

[29] Xinxin Fan, Ling Liu, Mingchu Li, and Zhiyuan Su. Eigentrustp^{++}: Attack resilient trust management. In *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2012, Pittsburgh, PA, USA, October 14-17, 2012*, pages 416–425, 2012.

[30] Diego Gambetta. *Can We Trust Trust?*, chapter Trust: Making and Breaking Cooperative Relations, pages 213–237. Department of Sociology, University of Oxford, 1980.

[31] Luis Garcés-Erice, Ernst W. Biersack, Pascal Felber, Keith W. Ross, and Guillaume Urvoy-Keller. Hierarchical peer-to-peer systems. In *Euro-Par 2003. Parallel Processing, 9th International Euro-Par Conference, Klagenfurt, Austria, August 26-29, 2003. Proceedings*, pages 1230–1239, 2003.

[32] Kalman Graffi. PeerfactSim.KOM: A P2P System Simulator - Experiences and Lessons Learned. In *IEEE P2P '11: Proceedings of the International Conference on Peer-to-Peer Computing*, 2011.

[33] Yunfeng Gu and Azzedine Boukerche. HD Tree: A novel data structure to support multi-dimensional range query for P2P networks. *Journal of Parallel and Distributed Computing*, 71(8):1111–1124, 2011.

[34] Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys*, 42(1):1–31, 2009.

[35] Quirin Hofstätter. Performance impacts of node failures on a chord-based hierarchical peer-to-peer network. In *Networked Services and Applications - Engineering, Control and Management, 16th EUNICE/IFIP WG 6.6 Workshop, EUNICE 2010, Trondheim, Norway, June 28-30, 2010. Proceedings*, pages 256–258, 2010.

[36] Quirin Hofstätter, Stefan Zöls, Maximilian Michel, Zoran Despotovic, and Wolfgang Kellerer. Chordella - A hierarchical peer-to-peer overlay implementation for heterogeneous, mobile environments. In *Proceedings P2P'08, Eighth International Conference on Peer-to-Peer Computing, 8-11 September 2008, Aachen, Germany*, pages 75–76, 2008.

[37] Feng Hong, Minglu Li, Xinda Lu, Jiadi Yu, Yi Wang, and Ying Li. Hp-chord: A peer-to-peer overlay to achieve better routing efficiency by exploiting heterogeneity and proximity. In *Grid and Cooperative Computing - GCC 2004: Third International Conference, Wuhan, China, October 21-24, 2004. Proceedings*, pages 626–633, 2004.

[38] Jinfeng Hu, Yinghui Wu, Ming Li, and Weimin Zheng. Improvement of Routing Structure in P2P Overlay Networks. *Grid and Cooperative Computing*, pages 292–299, 2004.

[39] Hosagrahar Visvesvaraya Jagadish, Beng Chin Ooi, and Quang Hieu Vu. BATON: a balanced tree structure for peer-to-peer networks. *Proceedings of the 31st international conference on Very large data bases*, pages 149–160, 2005.

[40] Saeed Javanmardi, Mohammad Shojafar, Shahdad Shariatmadari, and Sima S Ahrabi. FR TRUST: A Fuzzy Reputation Based Model for Trust Management in Semantic P2P Grids. *International Journal of Grid and Utility Computing*, pages 1–11, 2014.

[41] Audun Jøsang. Trust and reputation systems. In *Foundations of Security Analysis and Design IV, FOSAD 2006/2007 Tutorial Lectures*, pages 209–245, 2007.

[42] Audun Jøsang, Elizabeth Gray, and Michael Kinateder. Simplification and analysis of transitive trust networks. *Web Intelligence and Agent Systems*, 4(2):139–161, 2006.

[43] Kamesh and Sakthi Priya. Security enhancement of authenticated RFID generation. *International Journal of Applied Engineering Research*, 9(22):5968–5974, 2014.

[44] Sepandar Kamvar, Mario Schlosser, and Hector Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. *12th International Conference on World Wide Web (WWW)*, page 640, 2003.

[45] Lohit Kapoor, Seema Bawa, and Ankur Gupta. Hierarchical Chord-Based Resource Discovery in Intercloud Environment. *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 464–469, 2013.

[46] Anastasios Kementsietsidis. Data sharing and querying for peer-to-peer data management systems. In *Current Trends in Database Technology - EDBT 2004 Workshops, EDBT 2004 Workshops PhD, DataX, PIM, P2P&DB, and ClustWeb, Heraklion, Crete, Greece, March 14-18, 2004, Revised Selected Papers*, pages 177–186, 2004.

[47] Dmitry Korzun and Andrei Gurtov. Hierarchical architectures in structured peer-to-peer overlay networks. *Peer-to-Peer Networking and Applications*, 7(4):359–395, 2014.

[48] Aleksandra Kovacevic, Sebastian Kaune, Hans Heckel, Andre Mink, Kalman Graffi, Oliver Heckmann, and Ralf Steinmetz. PeerfactSim.KOM - A Simulator for Large-Scale Peer-to-Peer Networks. Technical Report Tr-2006-06, TU Darmstadt, 2006.

[49] Sandeep Kumar, Chander Diwaker, and Amit Chaudhary. Reputation system in Peer-to-Peer network: Design and classification. *Journal of Global Research in Computer Science*, 2(8), 2011.

[50] Heba Kurdi. HonestPeer: An enhanced EigenTrust algorithm for reputation management in P2P systems. *Journal of King Saud University - Computer and Information Sciences*, 27(3):315–322, 2015.

[51] Heba Kurdi, Sarah Alnasser, and Munira Alhelal. Authenticpeer: A reputation management system for peer-to-peer wireless sensor networks. *International Journal of Distributed Sensor Networks*, 11:637831:1–637831:11, 2015.

[52] Kwang Jo Lee, Jae Ho Choi, and Sung Bong Yang. Fuzzy inference-based super peer selection for a practical double-layered mobile peer-to-peer system. *Ad-Hoc and Sensor Wireless Networks*, 21(3-4):327–351, 2014.

[53] João Leitão and Luís Rodrigues. Overnesia: A resilient overlay network for virtual super-peers. *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, 2014-January:281–290, 2014.

[54] Xukang Lu, Qishi Wu, Runzhi Li, and Yunyue Lin. On tree construction of super peers for hybrid P2P live media streaming. *Proceedings - International Conference on Computer Communications and Networks, ICCCN*, (ii), 2010.

[55] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Author retrospective for search and replication in unstructured peer-to-peer networks. In *ACM International Conference on Supercomputing 25th Anniversary Volume*, pages 64–82, 2014.

[56] Jouni Mäenpää and Gonzalo Camarillo. Study on maintenance operations in a chord-based peer-to-peer session initiation protocol overlay network. In *23rd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2009, Rome, Italy, May 23-29, 2009*, pages 1–9, 2009.

[57] Spiridoula Margariti and Vassilios Dimakopoulos. A novel probabilistic flooding strategy for unstructured peer-to-peer networks. In *15th Panhellenic Conference on Informatics, PCI 2011, Kastoria, Greece, September 30 - October 2, 2011*, pages 149–153, 2011.

[58] Sergio Marti and Hector Garcia-Molina. Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks*, 50(4):472–484, 2006.

[59] Adán Medrano-Chávez, Elizabeth Pérez-Cortés, and Miguel Lopez-Guerrero. A performance comparison of Chord and Kademlia DHTs in high churn scenarios. *Peer-to-Peer Networking and Applications*, 8(5):807–821, 2015.

[60] Alberto Montresor, Mark Jelasity, and Ozalp Babaoglu. Chord on demand. *Proceedings - Fifth IEEE International Conference on Peer-to-Peer Computing, P2P 2005*, 2005:87–94, 2005.

[61] Keerthi Nelaturu. Content Management and Hashtag Recommendation in a P2P Social Networking Application. Master's thesis, University of Ottawa, 2015.

[62] Wee Siong Ng, Beng Chin Ooi, Kian-Lee Tan, and Aoying Zhou. Peerdb: A p2p-based system for distributed data sharing. In *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India*, pages 633–644, 2003.

[63] Miroslav Novotny and Filip Zavoral. Resistence against malicious collectives in bubbletrust. In *Proceedings of the 2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies*, PDCAT '11, pages 56–61, Washington, DC, USA, 2011. IEEE Computer Society.

[64] Thomas Papadakis, J. Ian Munro, and Patricio V. Poblete. Analysis of the expected search cost in skip lists. In *SWAT 90, 2nd Scandinavian Workshop on Algorithm Theory, Bergen, Norway, July 11-14, 1990, Proceedings*, pages 160–172, 1990.

[65] Isaac Pinyol and Jordi Sabater-Mir. Computational trust and reputation models for open multi-agent systems: A review. *Artificial Intelligence Review*, 40(1):1–25, 2013.

[66] Marius Portmann, Sookavatana, Ardon, and Seneviratne. The cost of peer discovery and searching in the Gnutella peer-to-peer file sharing protocol. *IEEE International Conference on Networks, ICON*, pages 263–268, 2001.

[67] Fips Publication. Archived Publication Secure Hash Standard. *Public Law*, 2:100–235, 1987.

[68] Josep Pujol, Ramon Sangüesa, and Jordi Delgado. Extracting reputation in multi agent systems by means of social network topology. *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 467–474, 2002.

[69] Syvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content Addressable Network. *Proc of ACM SIGCOMM*, TR-00-010:161–172, 2000.

[70] Indrakshi Ray, Indrajit Ray, and Sudip Chakraborty. An interoperable context sensitive model of trust. *Journal of Intelligent Information Systems.*, 32(1):75–104, 2009.

[71] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[72] Vladimir Rocha, Fabio Kon, Raphael Cobe, and Renata Wassermann. A hybrid cloud-P2P architecture for multimedia information retrieval on VoD services. *Computing*, 98(1-2):73–92, 2016.

[73] Rodrigo Rodrigues and Peter Druschel. Peer-to-Peer systems. *Communications of the ACM*, 53(10):72, 2010.

[74] Antony Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. *Middleware 2001*, 2218(November 2001):329–350, 2001.

[75] Jordi Sabater and Carles Sierra. Reputation and social network analysis in multi-agent systems. In *The First International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2002, July 15-19, 2002, Bologna, Italy, Proceedings*, pages 475–482, 2002.

[76] Stefan Saroiu, Krishna P. Gummadi, and Steven D. Gribble. Measuring and analyzing the characteristics of Napster and Gnutella hosts. *Multimedia Systems*, 9(2):170–184, 2003.

[77] Rüdiger Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. *Proceedings First International Conference on Peer-to-Peer Computing*, pages 101–102, 2001.

[78] Chithra Selvaraj and Sheila Anand. A survey on Security Issues of Reputation Management Systems for Peer-to-Peer Networks. *Computer Science Review*, 6(4):145–160, 2012.

[79] Rajesh Sharma and Anwitaman Datta. SuperNova: Super-peers based architecture for decentralized online social networks. *2012 4th International Conference on Communication Systems and Networks, COMSNETS 2012*, 2012.

[80] Wanita Sherchan, Surya Nepal, and Cecile Paris. A Survey of Trust in Social Networks. *ACM Computing Surveys*, 45(4):47–47:33, 2013.

[81] Rob Sherwood, Seungjoon Lee, and Bobby Bhattacharjee. Cooperative peer groups in NICE. *Computer Networks*, 50(4):523–544, 2006.

[82] Ralf Steinmetz and Klaus Wehrle. What is this *"Peer-to-Peer"* about? In *Peer-to-Peer Systems and Applications*, pages 9–16, 2005.

[83] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *Conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '01)*, 11(1):149–160, 2001.

[84] Girish Suryanarayana and Richard Taylor. A survey of trust management and resource discovery technologies in peer-to-peer applications. *ISR Technical Report # UCI-ISR-04-6*, page 61, 2004.

[85] Quang Hieu Vu, Mihai Lupu, and Beng Chin Ooi. *Peer-to-Peer Computing - Principles and Applications.* Springer, 2010.

[86] Omar Abdel Wahab, Jamal Bentahar, Hadi Otrok, and Azzam Mourad. A survey on trust and reputation models for Web services: Single, composite, and communities. *Decision Support Systems*, 74:121–134, 2015.

[87] Yao Wang and Julita Vassileva. Trust and reputation model in peer-to-peer networks. In *3rd International Conference on Peer-to-Peer Computing (P2P 2003), 1-3 September 2003, Linköping, Sweden*, pages 150–157, 2003.

[88] Zhe Wang and Naftaly H. Minsky. Towards secure distributed hash table. In *Collaborative Computing: Networking, Applications, and Worksharing - 11th International Conference, CollaborateCom 2015, Wuhan, China, November 10-11, 2015. Proceedings*, pages 257–266, 2015.

[89] Waqas imtiaz, Shimul Shil, Mahfuzur Rahman. Three Layer Hierarchical Model for Chord. *International Journal of Advanced Computer Science and Applications(IJACSA)*, 3(11):96–101, 2012.

[90] Isaac Woungang, Fan Hsun Tseng, Yi Hsuan Lin, Li Der Chou, Han Chieh Chao, and Mohammad S. Obaidat. MR-Chord: Improved Chord Lookup Performance in Structured Mobile P2P Networks. *IEEE Systems Journal*, 9(3):743–751, 2014.

[91] Li Xiong and Ling Liu. PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):843–857, 2004.

[92] Beverly Yang and Hector Garcia-Molina. Designing a super-peer network. In *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India*, pages 49–60, 2003.

[93] Yuan Yao, Sini Ruohomaa, and Feng Xu. Addressing common vulnerabilities of reputation systems for electronic commerce. *Journal of Theoretical and Applied Electronic Commerce Research*, 7(1):1–20, 2012.

[94] Ben Zhao, Ling Huang, Sean Rhea, Jeremy Stribling, Joseph Anthony, and John Kubiatowicz. Tapestry: A global-scale overlay for rapid service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004.

[95] Runfang Zhou. *Scalable Reputation Systems for Peer-to-peer Networks.* PhD thesis, Los Angeles, CA, USA, 2007. AAI3262732.

[96] Runfang Zhou and Kai Hwang. PowerTrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Transactions on Parallel and Distributed Systems*, 18(4):460–473, 2007.

[97] Stefan Zoels, Zoran Despotovic, and Wolfgang Kellerer. Load balancing in a hierarchical DHT-based P2P system. *Proceedings of the 3rd International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2007*, pages 353–361, 2007.