A Decentralized Key Database for Overlay Identification

Shimul Shil
A.K.M. Mahfuzur Rahman

School of Computing
Blekinge Institute of Technology
SE - 371 79 Karlskrona Sweden
This thesis is submitted to the School of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering. The thesis is equivalent to Fifteen weeks of full time studies.

Contact Information:
Author(s):
Shimul Shil
Email: shimul_ustc@yahoo.com
A.K.M. Mahfuzur Rahman
Email: itsmahfuz@yahoo.com

University Advisor:
Karel De Vogeleer
School of Computing
Blekinge Institute of Technology

School of Computing
Blekinge Institute of Technology
SE - 371 79 Karlskrona
Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57
ABSTRACT

Increasing popularity of extensively-organized and decentralized Peer-to-Peer (P2P) architecture emphasizes on the need to come across an overlay structure that can provide efficient content discovery mechanism, accommodate high churn rate and adapt to failures in the presence of heterogeneity among the peers. Since the 2-layer hierarchical models improves the P2P by offering some benefits to P2P paradigm and got popularity thereby, hierarchical models are brought into the account of current P2P researchers and still the process is under development. Many designs so far are introduced in hierarchical model like some hierarchical models has CAN or Kademlia algorithm inside them as DHT algorithm. A fundamental task of Peer-to-Peer applications is to find the peer efficiently that stores a desires data item. In our research work, we used 3-layers hierarchical model with distributed database architecture in different layer, each of which is connected through its root. The peers are divided into three categories according to their physical stability and strength. They are Ultra-superpeer, Superpeer and Ordinary Peer and we assign these peers to first, second and third level of hierarchy respectively. Peers in a group in lower layer have their own local database which hold as associated Superpeer in middle layer and access the database among the peers through user queries. In our 3-layer hierarchical model for DHT algorithms, we used an advanced Chord algorithm with optimized finger table which can remove the redundant entry in the finger table in upper layer that influences the system to reduce the lookup latency. Our research work finally resulted that our model really provides faster search since the network lookup latency is decreased by reducing the number of hops. The peers in such network then can contribute with improve functionality and can perform well in P2P networks.

Keywords: Chord, Decentralized, DHT, Hierarchical, Overlay, P2P.
ACKNOWLEDGEMENTS

At first, we are grateful to Almighty GOD, who gave us strength and courage to complete this work productively in time. A special thanks to Karel De Vogeleer, our supervisor, for his support and commitment throughout our research. We would like to thank him for his efforts in guiding us through our work and sharing our difficulties. We are also grateful to him for giving us his time whenever we needed it and for providing a great work environment during our work. It has been an honor working with him. We would also like to thank our Parents and family for their endless love, support, care and everything they have provided us with during our entire life. We admire their trust and patience that helped us on our long pursuit of study and solved many difficulties on the way. We would have not accomplished all this without them.

- Shimul Shil

- A.K.M. Mahfuzur Rahman
Table of Contents

Chapter 1: Introduction
1.1 Introduction: .................................................................................................................. 1
1.2 Problem Statement: ........................................................................................................ 2
1.3 Aims and Objective: ....................................................................................................... 3
1.4 Research Questions: ....................................................................................................... 3
1.5 Research Methodology: .................................................................................................. 3
1.6 Motivation and Contribution: ......................................................................................... 4
1.7 Related Work: .................................................................................................................. 5
1.8 Thesis Outline: ................................................................................................................. 6

Chapter 2: Theoretical Background
2.1 Overlay: ........................................................................................................................ 7
2.2 Peer-to-Peer Network: .................................................................................................... 8
   2.2.1 Classification Based on Appeared Time and Purpose: ........................................... 9
      2.2.1.1 First Generation P2P: ...................................................................................... 9
         2.2.1.1.1 Centralized System: .................................................................................... 9
         2.2.1.1.2 Decentralized System: ................................................................................ 9
      2.2.1.2 Second Generation P2P: .................................................................................... 9
      2.2.1.3 Third Generation P2P: ...................................................................................... 10
2.2.2 Classification Based on Architecture: ..................................................................... 11
   2.2.2.1 Centralized: ......................................................................................................... 11
   2.2.2.2 Decentralized: ..................................................................................................... 11
2.2.3 Classification Based on How the Peers are Connected in a Network: ....................... 12
   2.2.3.1 Structured: ........................................................................................................... 12
   2.2.3.2 Unstructured: ....................................................................................................... 12
   2.2.3.3 Ring: .................................................................................................................... 13
   2.2.3.4 Hybrid: ............................................................................................................... 13
   2.2.3.5 Hierarchical: ....................................................................................................... 14
2.3 Distributed Hash Table (DHT): ..................................................................................... 15
   2.3.1 Consistent Hashing: .................................................................................................. 16
2.3.2 DHT Algorithm: ........................................................................................................ 17
   2.3.2.1 Chord: .................................................................................................................. 17
   2.3.2.2 CAN: .................................................................................................................... 18
   2.3.2.3 Tapestry: .............................................................................................................. 18
2.4 Indexing: ....................................................................................................................... 19
   2.4.1 Local Indexes: .......................................................................................................... 19
   2.4.2 Centralized Indexes: ................................................................................................. 19
   2.4.3 Distributed Indexes: ................................................................................................ 19
2.5 Routing Procedure in P2P Routing: ............................................................................. 20
   2.5.1 Iterative Routing: .................................................................................................... 20
   2.5.2 Recursive Routing: ................................................................................................. 20

Chapter 3: Performance Analysis of P2P with Chord
3.1 System Framework: ....................................................................................................... 21
3.2 Basic Component of Chord Protocol: .......................................................................... 22
   3.2.1 Routing Table: .......................................................................................................... 22
   3.2.2 Stabilization Protocol: ............................................................................................. 24
3.3 Lookup Procedure: ......................................................................................................... 24
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Overlay Network</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>General P2P model</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>Centralized P2P network Architecture</td>
<td>11</td>
</tr>
<tr>
<td>2.4</td>
<td>Decentralized P2P network Architecture</td>
<td>12</td>
</tr>
<tr>
<td>2.5</td>
<td>Ring structure:</td>
<td>13</td>
</tr>
<tr>
<td>2.6</td>
<td>Hybrid P2P Structure</td>
<td>14</td>
</tr>
<tr>
<td>2.7</td>
<td>Hierarchical P2P Structure:</td>
<td>15</td>
</tr>
<tr>
<td>2.8</td>
<td>Distributed Hash Table</td>
<td>16</td>
</tr>
<tr>
<td>2.9</td>
<td>An Identifier circle with 3 lives peer</td>
<td>17</td>
</tr>
<tr>
<td>2.10</td>
<td>P2P Routing procedure</td>
<td>20</td>
</tr>
<tr>
<td>3.1</td>
<td>Chord routing model: with identifier circle consisting of eight peers &amp; finger Table</td>
<td>23</td>
</tr>
<tr>
<td>3.2</td>
<td>Query flow chart in Chord Identifier space</td>
<td>24</td>
</tr>
<tr>
<td>3.3</td>
<td>Chord routing model: path of query for key 29 starting at peer 5</td>
<td>25</td>
</tr>
<tr>
<td>3.4</td>
<td>Chord based distributed storage system:</td>
<td>26</td>
</tr>
<tr>
<td>3.5</td>
<td>Chord routing model: After peer 15 joining:</td>
<td>28</td>
</tr>
<tr>
<td>3.6</td>
<td>Chord: path of query for key 29 starting at peer 5 with update the finger table</td>
<td>30</td>
</tr>
<tr>
<td>3.7</td>
<td>Analytical result for lookup delay</td>
<td>31</td>
</tr>
<tr>
<td>4.1</td>
<td>3-layer Hierarchical P2P model</td>
<td>34</td>
</tr>
<tr>
<td>4.2(a)</td>
<td>Index of Superpeer</td>
<td>35</td>
</tr>
<tr>
<td>4.2(b)</td>
<td>Index of Ultra-superpeer</td>
<td>35</td>
</tr>
<tr>
<td>4.3</td>
<td>Lookup process comparison between traditional and hierarchical system</td>
<td>38</td>
</tr>
<tr>
<td>4.4</td>
<td>Query flow chart in 3-layer hierarchical P2P network.</td>
<td>39</td>
</tr>
<tr>
<td>4.5</td>
<td>Hierarchical model: query path for different keys from different peer</td>
<td>40</td>
</tr>
<tr>
<td>5.1</td>
<td>Average lookup hops considering the fixed participating peers and customized no of Ultra-superpeer</td>
<td>47</td>
</tr>
<tr>
<td>5.2</td>
<td>Average lookup hops considering the fixed Ultra-superpeer</td>
<td>48</td>
</tr>
<tr>
<td>5.3</td>
<td>Average lookup hops considering the all parameter are customized</td>
<td>48</td>
</tr>
<tr>
<td>5.4</td>
<td>Average lookup latency when the all parameter are customized</td>
<td>49</td>
</tr>
</tbody>
</table>
List of Tables

Table 2.1: Comparison with three Generation P2P characteristic................................................................. 10
Table 3.1: The expressions of other forms in Finger table.................................................................................... 22
Chapter 1

Introduction

1.1 Introduction

Over the last decade, Peer-to-Peer (P2P) computing model has been recognized as a more natural and flexible approach for sharing resources, compared to the traditional Client-Server model. Internet-scale decentralized architecture created an environment for millions of users to simultaneously connect and share content, thus creating cooperative groups to communicate with their associates [9]. P2P networks are logical overlay networks which are built on top of one or more existing networks. P2P overlay networks have raised deal of attention, not only for researchers/academicians but also in practical usage of the technology, for example in file sharing [4]. First implementations targeted of P2P are file-sharing services and distributed computing. One of the most important techniques in P2P is, it has good computational ability through distributed storage rather than a centralized system, denial of service attack vulnerable system. The distributed system is less vulnerable to attacks, robust and highly available. The P2P network will be all-key-based heterogeneous network that allows users to access the system anywhere, anytime. In many internet applications, networks may have very large centralized databases. Due to an unexpected user density in the P2P networks, the updating and querying loads on the location databases are increased exponentially [1].

Now a day, many systems face the consistency problem because of one central database system. P2P algorithm applies distributed architecture. P2P algorithms are a class of decentralized distributed systems collectively, called as Distributed Hash Tables (DHTs) that grant a lookup service same as hash table: \((key, value)\) pairs are stored in the DHT. The hash table links data (sometimes called values) with keys and supports efficient data insertion, lookup and participating peer can efficiently retrieve the associated value with a given key. The hash value is the index of the data item and key space are formed by the hash value. On a network, DHTs allocate a value to a peer which is known as peer ID and permit packets to be routed “by value” to whichever peer is currently responsible for that value [2].

Lists of features are provided by the P2P systems which include performance of lookups, efficient data location, redundant storage and distributed content placement. This feature discovers via DHT and its algorithms [8] [10]. Most of the DHTs require \(O(\log N)\) hops per lookup request with \(O(\log N)\) neighbors per peer, where \(N\) is the network size. CHORD, KADEMLIA, CAN and TAPESTRY are the most popular DHT algorithms and are used to provide content discovery at distributed and decentralized manner. For simplicity characteristic, we have chosen Chord which is approved across the research community. Chord is a well-organized distributed lookup service based on consistent hashing and provides only one operation: \textit{given a key and it efficiently maps the key onto a peer} [7]. DTH Chord can provide the necessary location management by creating a key-value pair for each participating peer, where key represents the identifier and value represents the location. On the other hand, list of services are present in hierarchical DHT designs. The acceptable properties of DHT such as scalability, better fault tolerance, more effective bandwidth consumption, self organization, robustness, and better adaptation to the underlying physical network are the key reasons for choosing hierarchical systems. We demonstrate that hierarchical systems propose to reduce the lookup path length than the traditional lookup service [5].

Overlay network is a virtual or logical network. Mobility management is significant to cooperation of the peers in overlay network based on the peers ID corresponds to data ID. Two important operations are usually used to carry out the mobility management in P2P overlay network: location update and handoff.
management. Location update is a process for a peer user to inform the network where it lies, while handoff is another process that transfers the metadata (data information) to the proper peer. During the churn (join and leave) time both operations are occurred. Traditional mobility management approaches are based on client or server paradigms, and suffer from their well-known shortcomings like single point of failure, congestion, bottlenecks [47]. In P2P file sharing, mobility management schemes improve the scalability, availability, robustness and performance.

A centralized database can raise many problems, like single point of failure that can slow down the entire system, dependence on additional network components and lack of reliability. Traditional Peer-to-Peer system is a client/server communication which uses hierarchical structure for better performance. It is an important mechanism to relieve scalability problem. Naturally hierarchical DHT design has a better fault isolation, effective bandwidth utilization, a superior adaptation to the underlying physical network and a reduction of the lookup path length as additional advantages [47]. It is more efficient and easier to manage than broadcasting or pure P2P structure. By dividing the whole system into several different layers and tries to solve local tasks inside their own layers. As a result, system workloads on the upper layer are greatly reduced [48].

We propose three layers DHT based overlay with decentralized key database architecture for overlay identification that relies on P2P algorithm: Chord. A distributed lookup service that is both scalable and decentralized and can be used as the basis for general purpose P2P systems is Chord. Simplicity, correctness, and performance are distinguished Chord from other lookup service. It is simple to route a key through a sequence of $O(\log N)$ other peers toward the destination [7]. This necessitates to explore the design and performance of high-throughput database technologies used in P2P networks to meet future demands to fetch efficiently the anticipated loads. This architecture effectively reduces the access loads on a centralized database by distributing query load into the decentralized databases as well as the signaling traffics [1]. This projected three layer P2P architecture with decentralized database consists of a number of database subsystems, each of which is connected to others through its root only.

Based on the Chord lookup service, overlay peers form a ring shaped DHT. Peers in Chord are placed on a ring. Each ring has an $m$-bit identifier space. To meet the challenges of present internet applications, we have developed a three layer hierarchy in a P2P overlay network by using Chord. The lower layer peers are designed to be deployed on resource-constraints. In P2P overlay network lower layer peers do not need to deliver high data rates and high computational power. The second layer is the middle one that will act as a medium between lower and upper layer. Middle layer is designed by the Superpeer. This Superpeer performs as a central server for lower layer peers. The upper layer is the core or backbone layer in this architecture. We design this layer with ring based Ultra-superpeer communication. Ultra-superpeer in upper layer also acts as a central server for associated middle layer Superpeers. Finally, each layer Superpeers perform as a central server for immediate lower layer peers. Each Ultra-superpeer maintains the pointer index and a finger table that contains the successor/predecessor list in Chord ring and index contains the all previous level peer ID and data list but does not store the data or document. Moreover, Superpeer in the middle layer only maintains the pointer index. Each of the Ordinary peers is attached to a Superpeer with point-to-point or mesh topology connection and this Superpeer belongs to both the Ordinary peers and the Superpeers. As a result local peers do not have to share the burden of possibly high maintenance traffic and the overlay network does not have to deal with their performance bottlenecks and low reliability [4].

1.2 Problem Statement

In P2P overlay network, a fundamental problem is to find efficiently a peer that shares a requested object through the ability of fault tolerance and reduce the query traffic load. Structured network does not support the complex queries and more expensive to maintain the network than unstructured. But
structured overlays provide scalable, efficient and accurate service to expend lots of labors to maintain the regular topology. On the other hand, finding rare data items are quite unpredictable for unstructured network. The major problem of centralized architecture is single point of failure, poorly scalable and database also overloaded because of one central server. Decentralized architecture uses flooding based query, its sends a query to the whole network for unpopular files, as a result query traffic load are extensively high.

1.3 Aims and Objectives

The aim of this thesis is to propose a decentralized P2P overlay architecture that does not involve any evolutionary technological changes and easy to implement without any difficulty. The decentralized overlay framework includes database, Superpeer and Ultra-superpeer based on DHT Chord to provide the overlay identification. The process will include:

- Analysis of the structured and unstructured P2P overlay network.
- Study the centralized and decentralized database architecture.
- Fundamental requirements and operations of DHT Chord for overlay identification.
- Analysis the key factor of system performance such as routing procedure, query traffic etc.
- Analysis the hierarchical architecture for P2P overlay identification.

1.4 Research Questions

In this thesis our main task is to answer these research questions:

- How to reduce the access loads on a centralized database system by distributing query load into the decentralized databases?
- How to increase the efficiency of the query process?
- How to reduce the lookup latency of the query process?
- How to make the system fault-tolerant compared to the centralized one?

1.5 Research Methodology

In our thesis, the goals of our study are achieved by adapting analytical method. In our 3-layer hierarchical model, we conduct our study for both structured and unstructured overlay network through applying the DHT Chord as well as the Superpeer based query process. Firstly, we analyze the structured and unstructured P2P overlay network and take the exploit the useful features found from those networks. After that, we analyze the DHT chord based lookup process and try to improve the lookup process by improving the finger table (routing table). We also analyze the Superpeer based query process. After analyzing the above mention terms, we intend to design a 3-layer hierarchical P2P model to achieve the goal. Consequently, through the mathematical expressions, we try to prove that our proposed model works better than that of with the conventional Chord lookup process. We put the different values for different parameters and analyze the results of different lookup processes. Thus, we compare and evaluate
the performance of the traditional Chord and the 3-layer hierarchical model in terms of the quantitative metrics considered in our study, i.e., average lookup hops and the average lookup latency.

1.6 Motivation and Contribution

P2P satisfied distribution system that has been organized for file, music and data sharing applications in excess of the internet. Perhaps the most commonly used forms of such applications are file sharing and content distribution applications. In P2P overlay, millions of users or systems are connected over the internet that can dynamically determine data stored at any peer in the network. P2P routing algorithms have been classified as: structured and unstructured overlay network.

First generation P2P systems mainly consist of unstructured overlays that organized peers into an arbitrary network topology and use flooding or random walks to lookup data items. But generally unstructured overlays are quite unpredictable for finding rare items because this requires visiting a large fraction of overlay peers [18].

To overcome the problematic bounds on the data discovery structured overlays are to be designed. Structured overlay constraints of peer graph and on data placement to enable the efficient discovery of data. Structured overlays provide scalable, efficient and accurate service but expend lots of labor to maintain the regular topology. Structure P2P overlay system is a class of distributed data structure that supports deterministic respectability for data lookup. It does not support the complex queries. Structured P2P overlays impose restrictions on peer placement in the overlay and hence, improve the efficiency of data lookup [45] [18]. One of the major problems is that structured graphs are more expensive to maintain than unstructured graphs and that the constraints imposed by the structure make it harder to exploit heterogeneity to improve scalability [45].

On the other hand, there are three categories of database architecture in P2P overlay network: centralized, decentralized and hybrid architecture. In centralized architecture, there is one central server that can allow accessing all the users from any place. So, organizing and maintaining a central server is the main task in this system. Additionally, the system is highly adaptive to peer joining and leaving. The key drawback of a centralized system is it is poorly scalable when the population of the system increases. In this system, server needs more computational capability and bandwidth to maintain a huge number of peers. Moreover, the server faces a single point of failure when the server is down and the whole system is broken [17] [24].

In pure decentralized system, there is no central server. These systems support the flooding based query, which broadcasts in the overlays, as a result the number of query messages exponentially increases with the number of overlay hops. In this system requesting peer sends the query to its neighbor peers. The query is replicated and forwarded by each peer to all its neighbors. Query message has to propagate most of the hops for finding an unpopular file. As a result query traffic load can be extensively high. To control the query traffic load in network, a Time-to-Live (TTL) field is included with query message. The TTL value will be decreased by one when the query message propagated to its neighbors. The query message is dropped if TTL value goes to zero. Moreover, it is not simple to fix the suitable TTL value. The network is unreasonably loaded for high TTL value and for low TTL value, peer could not find the file although file exists anywhere in the network [9] [18] [24].

Distributed system is self organized and has the capability of fault tolerance, scalability and redundancy. Moreover, distributed system also removes powerful central servers that may create single point of failures and performance bottlenecks. For this reason, distributed system is prettier than the centralized system [14].
Structured and unstructured P2P network cannot provide efficiency, flexibility, scalability, robustness and fault tolerance service individually [18]. The goal of this thesis is to design a three layer hierarchical architecture for P2P system with decentralized data sharing to take the advantages and reduce their disadvantages of both types of P2P networks. The proposed architecture has a few numbers of distributed servers that are responsible for user request and resources that are also connected in a Chord network is considered in this work.

This architecture has two parts. One part is the unstructured P2P network which provides freedom to peers join and leave with low cost and also no need to update the whole network during peers join and leave. Other parts are structured ring based core network that forms the backbone of the system. The ring based core network has more than one Ultra-superpeer. Unstructured Peer-to-Peer networks are attached to the Superpeers in the immediate upper layer network. Similarly Superpeer in the middle layer also connected to upper layer core network. In this architecture, Ultra-superpeer in upper layer contains the all necessary information that is to be shared in the network are enlisted in pointer index and also has a finger table that enroll the successor/ predecessor list in the ring based Chord DHT. Superpeer in middle layer only contains the pointer index of the associated Ordinary peer. In this architecture each peer communicates with other through the root. We have also focused on sufficient load balancing and fault-tolerant techniques to improve availability, reliability and performance of the system.

1.7 Related Work

Since the time, the concept of file sharing invented in late 90s, P2P networks had started to walk on its way and emerged a noticeable intension for researchers to till now and has been targeted of an intense study. Napster, what can be said as one of the starting P2P file sharing application which made P2P concept popular. It is the starting and then comes a number of structures, topologies, algorithms and concept to give the P2P network a significant look and still it is an ongoing process how to make the P2P network more efficient and acceptable to every manner for the users. The main task in P2P paradigm is to locate the resource or data efficiently in a network and for that reason several algorithms has been developed and they are mainly of two kinds, structured and unstructured. Chord [7], CAN [51], Pastry [40], Tapestry [40] and P-Grid[40] are some algorithms developed as highly structured systems and used hashing technique to store the metadata for peers and their resources. On the other hand, unstructured system like Gnutella [40], Kazaa [40] are some systems which used random data placement and became popular. There raised an idea also to combine the structured model and unstructured model together and named as hybrid. YAPPERS [18] is a such system which resulted with a scalable lookup service. Development of several algorithms is also going on as the algorithms mentioned above can result high latency and low efficiency. Chen D. and Yang S. tried to develop the Chord algorithm (TaChord) by introducing topology aware routing approach and also introduced the concept of Superpeer [39].

The systems and algorithm development process really helped to find data easily and quickly but there were still some questions arrived regarding scalability, stability, network latency etc. and to solve those issues several thoughts and ideas presented by the researchers day by day. As the systems which were flat and structured, the concept of hierarchy gave the system an advanced look. In the realm of P2P network, efficient routing is a matter to consider. In HIERAS [48], the authors tend to get better routing efficiency by allocating together the topologically adjacent peers matched someway by their identifiers and thus routing overhead is reduced. The authors [HIERAS] also pointed that link latency is smaller between two peers in lower level than the link latency between two peers in the higher level. So, lower level rings also created to support the latency questions and thus HIERAS provide better manageability for Pastry [40] and gave more routing efficiency for Chord like algorithms. CASTRO [40] is topologically aware version of PASTRY. As the systems were not considering the topology of the peers, Ratnasamy and Shenker [48] tend to group the topologically aware peers together but later probabilistic approach for location and routing [48] gave a way to locate a data faster in a way that a replica of a popular data can be created in
the closer peers. In some way some peer can become popular for having some popular resource and can become hotspot for a network. To reduce the load on this hotspot [48] the authors specifies some way to distribute the load on the hotspot. Introducing hierarchical approach made some system more successful like Gnutella [46], Kazaa [46]. The reason behind introducing hierarchy was to reduce search query traffic and search recall can be reduced if the networks can be divided into smaller groups and it was needed to introduce two or more level hierarchy with placing the more capable peers into the higher level whereas the less capable peer at lower level of hierarchy [48], CAP [40] is a two tier system which was proved as scalable and stable system.

To support the hierarchical model it was relevant to think about the placement of the peers regarding the capability, stability and performance of a peer. The considering terms, just mentioned, creates the distinction between peers and placing of the peers at their appropriate hierarchy level got an attention. The highest capable peers are then placed at top level hierarchy and the lowest one the bottom layer hierarchy. The peers placed at higher level hierarchy are called Superpeer. Superpeer based P2P networks also got attention in the P2P paradigm for its performance. Kazaa [46] is widely used Superpeer model. EduTella [46] is another Superpeer model which is based on schema. Hypercup [46] is another Superpeer model which can access digital resources. So the Superpeer leads in a cluster and if symmetric peers can be placed in a same cluster, the performance of the system can be increased. This concept offered at this point which gave better routing performance. Gnutella [46] uses this technique and for selecting Superpeer it uses dynamic mechanism. BROCADE [40] is also a Superpeer model which maintains a two level peer system. SBARC is also a Superpeer based system which puts loads on Superpeer and also caches routing information to achieve better routing. It also reduces routing latency and makes sure about efficient utilization of free storage space. Morpheus [46] is another application based on Superpeer model which has network compatibility features with other P2P file-sharing application like BitTorrent [46], NeoNet [46], G2 [46], Gnutella [46] etc. So, hybrid hierarchical model along with Superpeer based model really gave P2P application more feasibility, acceptability and still the research is going on for its improvement.

1.8 Thesis Outline

The thesis consists of 6 chapters each of them explains the following:

Chapter 2: Theoretical Background: This section provides the overview of the basic components of decentralized overlay framework. Description of the architecture and necessary features of DHT, P2P network and overlay network are given in this chapter.

Chapter 3: Performance Analysis of P2P Overlay with DHT Chord: In this section describes the basic features of Chord, a theoretical analysis. This chapter also provides a detailed explanation of various features and operation of Chord.

Chapter 4: Hierarchical P2P with Decentralized Database Framework: This section provides a detailed operation of hierarchical model as location discovery. Section 4.1 covers the system model, 4.3 covers the basic requirements like routing table, data structure, Superpeer selection, migration etc. Section 4.4 covers query procedure of the framework including peer placement, content discovery, and peer departure etc.

Chapter 5: Analytical Evaluation of Hierarchical P2P Model: In this section we provide the analytical result of hierarchical model with different parameters.

Chapter 6: Conclusion and Future Work: In this section we conclude the entire work with some suggestions regarding the improvements in the future work.
Chapter 2

Theoretical Background

This chapter introduces the overview of the theories behind the P2P overlay framework as well as hierarchical decentralized architecture.

2.1 Overlay

The invention of overlay networks was first made prior to the existence of the data network, when a computer system used to be found connecting over telephone line utilizing the modem. Overlay networks are built on top of one or more existing physical networks. P2P overlay networks are functioning on top of the IP network [43]. Overlay network is followed by a network semantic layer above basic transport protocol level and according to the content held by a peer, the network topology is arranged. For example, distributed systems such as client/server application, Peer-to-Peer networks because their peers run on the top of the internet [28] [43]. Finally, P2P network service is to be implemented on the overlays that are unavailable in the physical IP network.

![Overlay network diagram](image)

Figure 1: Overlay network.

The peers in the overlay network are connected with virtual links and many physical links came into play to make it possible for the peers to communicate. As the nature of the links between the peers is logical, the IP network topology on which the overlay is created also differs from the P2P overlay network topology. Also, the topology is determined by specific algorithm which helps to determine between which peers the virtual link should be created [38] [43].

Overlay networks share four qualities [28]:

1. **Routing**: The process of selecting the path for data to travel from the source to the destination.
2. **Discovery**: The process of finding the location of a peer in the overlay network.
3. **Scalability**: The ability of the network to handle an increase in the number of peers without significant performance degradation.
4. **Robustness**: The network's ability to maintain functionality in the presence of failures or changes in the network topology.

These qualities are essential for the efficient operation of overlay networks, especially in the context of P2P systems where peers are dynamically joined and left.
• Guaranteed data retrieval
• Provable lookup-time typically $O(\log N)$ where $N$ is the number of peers in the network.
• Automatic load balancing
• Self-organization

Overlay networks, along with the DHT (Distribute Hash Table) offers some other features, for example insertion of a key, removal of a key and querying a key. The key is generated here by applying some consistent hashing technique based on the resource held by a peer. Secure Hash Algorithm (SHA-1) is the algorithm to use in consistent hashing for DHT [28]. Thus overlay network does not support keyword search and that’s make the difference between overlay network and P2P overlay network in a way that in overlay network, identifier or key generated from the content is the basic for lookup process [38] [43].

Peers in an overlay network look for resources on the basis of identifiers that are derived from the content which does not support keyword based queries directly [28]. Resources can be of any type, for example, it can be a multimedia file. Whether it can be support large scale networks When any match is found, a session between the requester and the holder of the resource is created usually and gets the resource thereby [38] [43].

2.2 Peer-to-Peer Network

What is a Peer-to-Peer network? It is a network in which peers have equal capability and responsibility. P2P systems and applications are distributed systems with no hierarchical organization and centralized control [7] and all peers in the network act as both server and client according to specific rules. However, in P2P systems, classified as a server or a client is difficult because they might have server and client relationship with other peers at the same time. In a file sharing application like, bit torrent, a peer will perform as a client by getting some files from other peers, on the other hand it will also act as a server at the same time by sharing part of a file with the neighboring peers in the networks [8]. Therefore, in a pure P2P system, the main uniqueness of these systems is the similarity of all participating peers. Basically P2P technologies are primarily used for file sharing with the help of suitable distribution method.

![General P2P model](image-url)

Figure 2: General P2P model.

The present P2P network is a resultant state over many development phases, research works by meeting the purposes, needs of users and organizations. So, at present there are so many P2P networks and each P2P network holds so many different properties which makes it harder to classify P2P networks in a unique criteria. But to do so, we can classify the current P2P networks based on some criterion, i.e. Appeared time and purpose, Architecture and Topology.
2.2.1 Classification Based on Appeared Time and Purpose

Since the time the concept of sharing resources invented to today, P2P appeared in many ways to meet user needs and purposes. We can classify the current P2P network based on its appeared time and purpose into three generations [50].

1) First Generation P2P
2) Second Generation P2P
3) Third Generation P2P

2.2.1.1 First Generation P2P

The first generation P2P system was introduced as a system which combines servers with P2P routing and had centralized network architecture. In such network, the querying is done in client-server fashion. Based on central server coordination, we can divide P2P systems by centralized and decentralized system [24] [50].

2.2.1.1.1 Centralized System

In centralized systems, all the data address information is stored in an index server. The first task for a peer is to search at the server for the data the peer is looking for by making a request and desired address of the data is returned. In the next step it can access the peer directly which stores the data. Napster is one typical example of such system. Such systems are easy to manage and simple but not suitable for large scale systems as they have some obvious problems.

1) If the central server fails for some reason, the system will not work anymore.
2) The server may face huge traffic and storage on the server may become massive

2.2.1.2 Decentralized System

Decentralized systems from those days are not based on central servers, but most of them do not have deterministic location of data. They used broadcasting mechanism for searching the desired data. When a peer receives a request for a key which represents the data item, it tries to retrieve the item locally if possible. If not, then the request is forwarded to another peer. Depending on the result to come if it is failed or successful, the report of failure or desired data is sent back to the requested peer with the same path of incoming request. The routing performance by caching or forwarding decision may depend on former routing information in enhanced by some systems enhance. Gnutella, Freenet etc. are typical example of such system. The disadvantages of these systems are [24]:

1) Content location information is not so guaranteed to be reliable.
2) To make the system with better scalability and to get better routing performance, it requires extensive cost

2.2.1.2 Second Generation P2P

To grasp the better scalability and query efficiency, second generation P2P networks usually utilize the DHT technique. Deterministic search guarantees and load-balancing is provided by them but they are not so fault-tolerant and resilient.
In second generation P2P system architecture has no central indexing server, but considerable holds some structures. The term “Structure” in P2P means that here the network topology is tightly controlled. For example Mesh, Ring, d-dimension Torus, K-ary tree, SkipList. The resource placement of such network is not random. They are located at specific location so that they will help successive queries more efficient. Thus second generation P2P system often supports DHT like interface which are quite common in the research literature for P2P network. To make the search procedure efficient such network uses precise algorithm. Exact placement algorithms and specific routing makes the systems to efficiently handle transient number of peers. A partial match query is not supported in these systems. They support precise-match queries [50]. Based on the inspiration technique for file distribution, second generation P2P overlay system, execute higher levels of robustness and resource utilization [30].

2.2.1.3 Third Generation P2P

The first generation P2P is scalable and easy to be organized. Implementing the DHT technique, the second generations P2P are providing some sort of benefits. For example, their search mechanism is deterministic as well the query path is short in length. They are also resistant to random peer failure but the adversarial attack is one which they could not conquer. So, system with greater fault-tolerant capability and also replication of objects along with multipath backup, which can make the system more resilient is needed to be designed and third generation P2P networks are currently offering such system [50].

In third generation, the proposed P2P network aims to provide systems with high resilient capability [50]. The first generation P2P networks had many problems and the decentralized systems in the second generation P2P were able to overcome those problems but always felt the lack of a central server which introduces new issues and overhead came along with it. The second generation P2P networks thus facing the problem of massive network traffic generated in the network as the lookup process is done here in broadcast manner. Hybrid network architecture in the third generation P2P network has arrived to solve this issue in an excellent manner by using supper peers that have more features than the Ordinary peers and act as servers in central P2P network. This Superpeer contributes as a gateway and recourse lookup in the P2P networks [43].

The table below tends to compare among P2P networks introduced in different generations [43].

Table1: Comparison with three Generation P2P characteristic.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>1st generation</th>
<th>2nd generation</th>
<th>3rd generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability</td>
<td>Medium</td>
<td>Low</td>
<td>Very high</td>
</tr>
<tr>
<td>Resiliency</td>
<td>Low</td>
<td>Very high</td>
<td>Medium</td>
</tr>
<tr>
<td>Query efficiency</td>
<td>Bad</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Operation control</td>
<td>Very high</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Search coverage</td>
<td>Very high</td>
<td>Medium</td>
<td>High</td>
</tr>
</tbody>
</table>
2.2.2 Classification Based on Architecture

2.2.2.1 Centralized

The early P2P networks in first-generation followed centralized network architecture. The query procedure for such architecture was implemented in client-server approach. The figure below demonstrates the lookup process in centralized architecture.

![Centralized P2P network architecture](image)

Figure 3: Centralized P2P network architecture.

When a peer needs a resource, it first sends a query to the server where all the information about the connected peers and their resources are reserved. The server then searches the requested resource and replies with the address of the peer which holds the requested resource. Then a session is created between the sender and requester of the resource and direct data transfer can take place between those peers.

2.2.2.2 Decentralized

The major problem in first-generation networks is to maintain the central servers. In the central server architecture, if server is failed, the whole network is down. To overcome the drawback, second generation P2P networks introduce the decentralized server i.e. distributed servers where every peer has equal functionality and responsibility for routing the messages. This kind of network architecture is called decentralized [43]. In decentralized networks, there is an absolute regularity in peer without any central management. Each peer is performing as both client or server, i.e., each peer may issue requests and serve/forward requests of other peers [9].

Content search in decentralized P2P networks support flooding based algorithm where the query messages have a Time-To-Live (TTL) field. The TTL value denotes how many hops the query message
has travelled to find the accurate data. An example of the lookup process in a decentralized P2P network is Gnutella, which is highly robust, but the query routing overhead comes in large scale network [43].

Figure 2 depicts the Peer X wants to lookup a resource and forward the query request as a flooding based in the network with the TTL field value of 2. The requested resource is found in peer Z. If the TTL value is 1, peer X cannot retrieve the resource. So that resource query depends on the TTL value. In large scale network, lots of traffic are generated in this process, as a result it is not scaled well and the lookup process is too slow than the first generation centralized P2P networks [43].

2.2.3 Classification Based on How the Peers are Connected in a Network

2.2.3.1 Structured

Structured overlay is the first one to describe in the context of overlay architecture. In overlay communication routing of information is one of the major procedure in the network and the process in structured overlay, is maintained here mutually with every peer in the network which helps to reach all the peers and their respective resource in the overlay. Peer in the structure overlay has a local routing table which is used by the forwarding algorithm. This routing table is initialized by the bootstrap procedure when a peer newly joins in the network. A key is generated by hashing the data item that is identifying the data. Peers in the network are arranged by structured graph that maps each key to a responsible peer. The peer who is responsible for a specific data, hold the pointer for the data. Example of structure overlay is DHT. In DHT, key-based routing (KBR) is used for puts and gets index operations. Peers are individually responsible for maintaining a section of the DHT index. [9] Example of structured overlay includes Chord, CAN, PASTRY, Kademlia, ROMEO etc [35] [43].

2.2.3.2 Unstructured

Unstructured network, in contrast to structured P2P system do not follow any structured network. In such network, content of the resources and address are stored in an unrelated way. So the links are formed arbitrarily. Every peer in the network can join any time and contact any peer for incorporation. They can have a replica of other peer’s existing information at times as well. [9][43]
Floodling and random walking mechanism is used in an unstructured network for broadcasting query message to overall network. So, messages can reach several peers and specific data information can be held by more than one peer. Each peer has the freedom choose any other peer to be its neighbor and it can store the data it owns. But unstructured overlays cannot find rare data items efficiently because this requires visiting a large fraction of overlay peers. In such system, replica of some popular can be found at different locations which makes it easy to find any data item in the network. Example for unstructured overlay includes Gnutella, Freenet etc [35] [43]. Unstructured overlays are widely used in popular applications because they can perform complex queries more efficiently than structured overlays.

2.2.3.3 Ring

The ring based topology led the way in the context of overlays in the Chord network. Chord generates an m-bit identifier by using SHA-1 based consistent hashing algorithm for every peer. This is mapped onto a circular identifier space (key-space). The identifiers for the peers generated by Chord are distinct and it is not concerned which algorithm is used to generate the identifier. Identifier for the resource also is to be generated and those identifiers to the resources are also unique. So, there are many keys in the key-space and an individual key maps one particular peer with least identifier in the key which is greater or equal to the key. This way a peer becomes a successor for a key and peers are set for specific resource or resources in the network. All the keys in the key-space, in this mechanism, are linked with the peers in the network and peers become interconnected and communicate with each other [9] [22]. Finally, the ring structure is formed circular. Each peer has information of successor (next live peer in the identifiers circle in clockwise) and predecessor (previous peer) peer. This architecture is generally used in P2P network without central coordination.

2.2.3.4 Hybrid

The problems with centralized and decentralized systems gave rise to some steps how to step away from those problems. To do so, hybrid system along with hierarchical design has appeared recently and been used by many popular P2P application to have a better scalable performance [9]. The first generation P2P networks had many problems and the decentralized systems in the second generation P2P were able to overcome those problems, but always felt the lack of a central server which introduces new issues and overhead came along with it. The second generation P2P networks thus facing the problem of massive network traffic generated in the network as the lookup process is done here in broadcast manner. Hybrid network architecture in the third generation P2P network has arrived to solve this issue and is trying to solve the issue in an excellent manner [50].
In hybrid architecture, some peers called Superpeers have more capabilities and functionalities than that of an Ordinary peer in that network. The Superpeers have the same capability, functionality and responsibility as a central server in a centralized network has and the Superpeers are only responsible for peer and resource lookup in the whole network. And, each of the Ordinary peers, called only a peer, uses one of the Superpeers in the network as its gateway to the network. The figure below is an illustration of lookup process in a hybrid network [43]. The advantage of hybrid decentralized systems is that they are simple to implement, and they locate files quickly and efficiently.

Although, hybrid design came to play a good role in P2P system, it has still single point of failure problem. Also, being vulnerable to restriction, legal action, malicious attack, surveillance, and technical failure are disadvantage of these systems. [38]

2.2.3.5 Hierarchical

Unstructured and structured P2P networks can be said as flat systems. So in these systems, there is not any hierarchy [53]. But in Client/Server computing system, hierarchical design has great significance and extensively implemented. In hierarchical model, a flat network is split into smaller regions, i.e., DNS (Domain Name System) is example of such design where hierarchical model is implemented and thus achieved scalability. To make any network a better scalable one, this mechanism can be applied and also proved as more efficient in some extent [48].

Recently, in the literature of many hierarchical P2P design, DHT is introduced to achieve advantages over multi-layer architecture when the most reliable peer is placed at the top. So, hierarchical design for DHT is classified into two categories, one is homogeneous design and another is Superpeer design. In homogeneous design all peers play equal roles for Superpeers who are literary small portion of the participating peers in the network with greater capability and functionality. CORAL, Canon and Cyclone are examples for homogeneous design, on the other hand HONET, HIERAS can be said as the example for super-peer design.
In hierarchical architecture, there are two levels of hierarchy though systems can be designed with multiple levels of hierarchy. Introducing multilevel in the system in a nested fashion and a tree is the way to make the intermediate connection among the different levels of the nested overlays and thus makes the system to have overall better performance. In system with two level of hierarchy, the top level of an overlay consists of super-peers and the lower level consists of the normal peers in the network [47]. The emergence of introducing such design came as the P2P network is becoming day by day and need to be more efficient and scalable [9]. Different routing algorithms are used in different overlay levels in a hierarchical structure.

2.3 Distributed Hash Table (DHT)

In the process of P2P development, it was needed to make the system independent from central services as much as possible and DHT came into play to make the P2P system as distributed. DHT is a class of distributed system which partitions the key space among participating peers in that system [36]. DHT determines the peer in the system which is responsible for storing the data and retrieve the data that are stored as (key, value) pairs among a number of peers [43]. Fault-tolerance Scalability, self organization and robustness are the pleasing attribute of DHT [12]. DHT make it easy to find an object from large scale network on foundation of the object’s key [43]

In DHT-based P2P systems, file association is done with the help of keys and the keys are produced by hashing the name of the file. Each peer in a system handles a portion of the hash space which stores a certain range of keys. To search a key in the system, the system will return an identity which is stored in the table and generated from hashing the IP address of that peer where the object is located. Thus, DHT allows the peers to put and get files based on their key [24].

Two basics Remote Procedure Calls (RPC) are normally used in DHT algorithm for communication and assistance among the DHT peers, which can be introduce as [12]:

1) Put (key, value): this method is used to store a key/value pair into the DHT;
2) Get (key): this method is used to retrieve the information stored in the DHT that is associated with the given key.

Earlier Key-value pairs in DHT are distributed among the peers in the system and mapping method is needed to decide on which peer specific pair should be placed. Consistent hashing is used in DHT for such mapping and it divides the key space into partition [38]. This method uses distance conceptually to map a key to specific peer and distance is a logical concept not related to network distances of peers. This way a peer which is physically in Sweden, could be nearer to a peer in Mexico than a peer in Sweden.

### 2.3.1 Consistent Hashing

To implement a DHT, consistent hashing has been constructive to distribute the key-value pairs over the peers of P2P overlay network. Consistent hash function allocates an $m$ –bit identifier for each peer and key by using the hash function such as SHA-1. Peer’s identifier and key identifier’s are obtained by hashing the peer’s IP address and key [7]. Using a mapping method, we can find out the specific pairs that should be stored on the peer. This mapping function is called the consistent hashing in support of DHT [8]. In DHT, consistent hashing partitions a key space among the distributed peers such that the peer responsible for any key can be determined efficiently. On average $K/n$ keys need to be remapped only by using consistent hashing, where $K$ is the number of keys, and $n$ is the number of slots or peers.

During insert a new key value pairs into the hash table and to find out the key from the overlay, consistent hash function is being used. The design of consistent hashing is to be tolerating the churn i.e. peers join and leave the overlay without negligible interruption. When a peer joins the network, certain keys allocate previously to the successor, which become allocated for joining peers. Similarly, all allocated keys are reallocated to its successor peer when peer leaves the overlay network [7] [43]
Figure 9: An Identifier circle with 3 lives peer.

Figure 9 shows an identifier circle with m=3. The circle has three peers: 0, 1 and 3. The successor of identifier 1 is peer 1, so key 1 would be located at peer 1. Similarly, key 2 would be located at peer 3 and key 6 at peer 0.

2.3.2 DHT Algorithm

To meet the purpose of P2P, DHT is an intangible idea. DHT fulfills two main purposes of P2P, to become fully independent of central lookup server and the other one is to build it up as more tolerant in the network. DHT algorithm is used to store and retrieve key value pairs to mapping between keys and values in a distributed approach are implemented by the peer [12]. To execute the DHT idea, a number of algorithms have been designed. Chord, CAN, Kademlia, Tapestry, Pastry are most popular implementation of structure P2P DHT [36]. Other algorithms which are less popular in addition to above mentioned algorithms such as Koorde, DKS, Viceroy, OpenHop, Accordion and Dipsea. Now we will take a short look on the design and techniques of these algorithms in this section [8].

2.3.2.1 Chord

In P2P, Chord is one of the most popular decentralized lookup services. Chord formed a ring base identifier where peer IDs and keys are placed on the circle. A Chord circle can have maximum $2^m$ peers and the range from 0 to $2^m-1$ where m denotes the number of bits in the identifier. The identifiers are generated by using consistent hash function and in the same identifier space they are distributed uniformly [7] [8]. Peer identifiers are generated from hashing the peer’s IP address and key identifiers from hashing the name of the file or keyword [29]. A key is assigned to the first peer whose identifier is equal or follows the identifier of the key [8].

In Chord DHT, SHA-1 hash function is used for hashing the peers and keys. Each Chord peer maintains the routing table called finger table contain the m number of entries. In the Chord ring, a successor of peer is the next peer clockwise and the predecessor is the next peer in anti-clockwise. If a ring is fully
occupied, for peer 2, its successor will be peer 3 and its predecessor will be peer 1. Chord maps keys onto a peer and both peers and keys are assigned \( m \)-bit identifier. If there is \( N \) number of peers and \( K \) number of identifiers in a Chord ring, each peer will be responsible for \( K/N \) keys [8] [30] [50].

Chord supports only one operation: given a key, it maps the key onto a peer and both peers and keys are assigned \( m \)-bit identifier. If there is \( N \) number of peers and \( K \) number of identifiers in a Chord ring, each peer will be responsible for \( K/N \) keys [8] [30] [50].

### 2.3.2.2 CAN

CAN (Content Addressable Network) is another popular DHT algorithm which was designed to be scalable, fault tolerant, and self-organizing and provide hash table functionality [30]. CAN is a system that using a \( d \)-dimensional Cartesian coordinate space to execute the distributed hash table concept which are hyper-rectangles is called zone [38]. In CAN, the peers in it are maintaining a routing table with \( O(d) \) entries. This routing table keeps the IP Address and virtual coordinate zone of the peer’s neighbors. A peer in CAN route a message towards a destination point in the coordination space [30].

CAN apply greedy forwarding method to route a message towards its destination by using neighbor coordinate [30]. The first task of a peer is to determine which neighboring zone is closest to destination point. Then the peer looks up the IP address of the peer in that zone by using the routing table [52].

When a peer wants to join, it will try to find a peer already in that overlay network and select a point of \( d \)-dimensional space which is completely logical [30]. Then it tries to find which zone to split and also tends to update the routing table. To find the peer it will use bootstrapping peer which tells the joining peer about the IP addresses of the existing peers and task of selecting the zone came easy for the peer when it knows the IP address of other participating peers. And if any peer departs from the overlay network, the CAN network should know that one peer is departing and the departing zone of the peer have been merged or taken-over by neighboring peer, along with with the routing table is updated across the network [52].

Each peer keeps its neighbor’s peer ID locally, and the routing task is performed by forwarding the request to nearest region to the key position. In this method, expected search length (or P2P hop) for CAN is \( O(d \sqrt{N}) \) and state information reserved locally is \( O(d) \) [30] [50].

### 2.3.2.3 Tapestry

Tapestry is one of the basic intends at offering high-performance, decentralized object location, scalable P2P lookup algorithm and location-independent routing of messages to nearby endpoints [30]. This algorithm based system update cached content at times and thus promises to get rid of failures as much as possible [50].

Tapestry peers are assigned random peer ID uniformly from a large identifier space and typically 160 bit values are used together with radix which results 40-digit hexadecimal identifiers. To generate peer ID, SHA-1 hashing algorithm is used. Each identifier is identified to a peer called root ‘G’. The peer is not the root peer, if peer ID is G. This peer will then use the peer IDs and IP addresses from the routing table to find its neighbor. Each neighbor map has multiple levels where each level contains links to peer matching up to a certain digit position in the ID [53].
Tapestry formed strings of number to mapping peer and key identifiers, and to correcting a single digit, lookups is forwarded to the accurate peer at a time. To keep away from the single point of failure at the query process, tapestry uses multiple roots for each data object [30].

The primary \( i^{th} \) entry in the \( j^{th} \) level is the ID and the closest peer location having prefix \((N,j-1)+i\), and IDs of base \( B \) (\( B \) is a Hexadecimal value). Such routing approximately take \( \log_{B}N \) (where \( B = 16 \) is the base of Log Function) logical hops in a network which has \( N \) sized namespace and IDs of Base \( B \) [53].

### 2.4 Indexing

Today, in P2P designs assume some structure, a more instructive taxonomy in lieu of networking taxonomy. P2P research is set to benefit from database research. Databases are designed to offer an organized mechanism for storing, managing and retrieving information. The invention of appropriate indexing mechanisms and query optimizations enable data independence that is to decouple data indexes from the applications that use the data. Database indexes have an analog in P2P’s distributed hash tables (DHTs). An index is a collection of terms with pointers which will provide us the information about the documents. There are three types of index and they are Local index, central index and distributed index [33] [43].

#### 2.4.1 Local Indexes

Local index based P2P system is becoming rare at present and in these system, local index only keeps their own content [33]. The self data references are received only by the peers in local indexing system. Peers do not keep references for data that the other peers are storing. Decentralized P2P network architecture only uses local indexes. The basic query process for local indexing system is flooding based. As the rich queries are also enabled, the search system is not limited to a simple key lookup. Local indexes are good when there are many replicas of the data items exist in the network. Local indexes are very inefficient when finding a data item which exists only in a single peer in the network [43]. As of the system, a huge volume of query traffic is generated and there is also no guarantee to a match of the finding item [33].

#### 2.4.2 Centralized Indexes

The P2P networks which are using centralized indexes, has a server to which the peers can connect and all the references to the stored items are kept in this sever pool [43]. System with centralized data indexes have also been called \emph{hybrid} because the index is centralized but the data is distributed [33]. The centralized index system can be derived up to three levels of specification – unchained, chained and fully replicated. In unchained servers, the peers can contact or have index of a data which is stored in the server to which it is connected without providing the index information in other servers [33] [43]. In chained servers, query will be forwarded if the peer cannot find the data in the server it is connected with; it can connect or contact other server for its desired data item. In fully replicated system, replica of all the data are indexed in all the servers and peers can contact all the servers and get the data accordingly from any server [43].

#### 2.4.3 Distributed Indexes

Now a days, distributed indexing is becoming popular and most of the P2P system applies this kind of indexing. Not only that peer which hold a specific data but also some other peers stores the reference of that data in such indexing process. When a peer receives a query request, its first check its own table for the peer with keys closest to the target otherwise forward the query message to the closest peers. P2P architectures which use Chord and Kademlia, apply distributed indexing. Such system generates multiple
replicas and a specific data can be stored in a number of peers, which depends on how many replicas are allowed in the system [33] [43].

2.5 Routing Procedure in P2P Routing

Routing is the significant element in P2P overlay network. Without efficient routing procedure, finding a peer in the network is too much difficult. In DHT Chord, there are two type of routing procedures used to find a peer, which are: iterative and recursive routing.

2.5.1 Iterative Routing

An intermediate peer in iterative routing, replies with a response message after it gets a query message. Response message keeps the information of the next hope. If a Chord topology is applied, peers are positioned in ring shape (Figure 10 is an illustration of this). In fact, the task of communicating the next hop is up to initiating peer. Also, the initiating peer supervise the routing process and takes the decision about how much time it will wait to declare the next hop peer as absent. To reach the target peer, $2N-1$ messages is required for a lookup path of $N$ peers iterative routing. We summarized that, peers communicate with the originator only in iterative routing process [43] [55].

![Figure 10: P2P Routing procedure.](image)

2.5.2 Recursive Routing

Recursive routing is same as iterative routing but it works in recursive manner. In the process of recursive routing, the intermediate peer simply forward the query to next hop peer. It is not needed for intermediate peer to inform the initiating peer about this forwarding task. That implies that there is no control of initiating peer over routing table when it has already sent the query message in the network. Depending on the scenarios of the travelling of response message from destination peer to initiating peer, recursive routing is of different kinds. They are symmetric recursive, forward-only and direct response. Figure above is an illustration of symmetric recursive routing. In recursive routing procedure, to achieve the target peer, $N$ number of messages is required for a lookup path of $N$ peers [43] [55].
Chapter 3

Performance Analysis of P2P Overlay with Chord

Overview of Chord

Chord is an efficient distributed lookup service for structured P2P network, using a globally consistent hashing protocol to ensure that any peer can participate in and leave the network which provides support for only one operation: given a key, it efficiently maps the key onto a peer [7]. Chord uses one-dimensional circular identifier space with module $2^m$ for both peer identifiers and data keys which forms a ring of peers. The identifier ranges from 0 to $2^m - 1$, where $m$ is the number of bits in the identifier using SHA-1 as the base hash function ($2^m = N$ defines the maximum number of peers supported) [14]. Peer’s identifier (P_ID) can be obtained by hashing the peer’s IP address and the key identifier (D_ID) is created by hashing the data key, for example, hashing a file name [10]. In Chord protocol, same identifier is not possible for two peers or resources in the overlay system because the identifier length $m$ must be a large enough integer. The peer is called successor peer of key $k$, identify by successor($k$) if key $k$ is assigned the first peer and whose identifier is equal or follows key $k$ in the identifier space numerically(clockwise) [25]. This identifier circle is called the Chord ring. Simplicity, correctness, and performance distinguish Chord from other lookup services. It forward the query routing message toward the destination through a sequence of $O(\log N)$ other peers [7].

3.1 System Framework

In DHT Chord, peers are first organized into the Chord ring as virtual peers. In Chord ring, peers may appear from different places in the underlay networks and are logically interconnected in the overlay network. Peers takes the place in Chord ring overlay according their peer ID, which are generated from hashing the IP address of the peer. Resource is assigned to a peer on the Chord ring according to its own key value. The IP address provides the physical location of the file and is obtained when a user inserts the file to be shared [14]. In the Chord lookup services, each key for a data item is assigned to the live peer whose peer identifier is “closest” to the key [40]. Each live peer maintains a finger table that contains the successor list of this peer. In the finger table, there are $m$ number of entry, where $2^m$ is the number of peers in the ring. One important objective in Chord ring is, every peer does not know any information about the other half part of the Chord ring peers and its lookup bound is limited to its own half part of the ring [25].

Two basic commands used in DHT Chord are:

**Put (key, value):** This is used to store a key/value pair into Chord peer using DHT to find the successor peer of the hashed key on the identifier circle.

**Get (value):** This can be used to retrieve the value from DHT, which is associated with the given key. Chord is used to trace the peer on the identifier circle with the requested key/value pair.
Chord protocol simplifies the design of P2P systems and applications by addressing the following problems [7].

**Load Balancing:** Load balancing is when several peers are connected together to share functions as a virtual peer. Chord has the capability to extend the keys evenly over the participating peers, which provides a degree of natural load balancing among the peers. This can reduce overload on hot peers.

**Decentralization:** Chord is a distributed system without any central coordination. In the overlay network, no peer is more important than any other participating peer. This improves the ability of Chord to make the system more robust and fault tolerant.

**Scalability:** It is the ability of a system application to continue the function well when it is changed in size or volume in order to meet a user need. In Chord ring overlay, lookup function works efficiently even with thousands or millions of peers join in the system. Even though the number of peers increases dramatically, there will be a minimal effect on performance and availability [38].

**Availability:** Chord automatically matches its finger tables whenever a peer joins or leaves the network as well as peer failure. This makes it able to deliver the key/value pair every time a query is made despite failures in the underlying network [7].

**Flexible naming:** In structure of the key lookup, there are no constraints i.e. the key space provided by Chord is flat. This gives applications a large amount of flexibility as they map their own names to Chord keys that can increase performance [7].

**Reliability:** Chord is a full decentralized system and there is no single point of failure. Additionally, these systems store a reference to a resource in the nearest peers instead only storing one reference in the next one. It provides reliability to adjacent peer failure by replicating keys at some constant (x) number of successive peers and storing log(N) successor addresses. Moreover, stabilization protocol runs on each peer that maintains correctness of references [24].

### 3.2 Basic Component of Chord Protocol

To explain the operation of Chord, it is necessary to explain the following parameters.

#### 3.2.1 Routing Table

In order to find a key efficiently, each peer maintains a routing table also called finger table with log₂*N records where N is the number of peers. It has three fields called *finger. Start, finger.int* and *finger.succ* entry [29]. Finger Table consists of Peer ID which is an IP address and port pair. Finger table is used during query process to find the responsible Peer ID or the closest Peer in the overlay network [14].

The finger table expressions are shown in Table 2.

**Table 2: The expressions of other forms in the Finger Table**

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finger.start</td>
<td>( (n + 2^{i-1}), 1 \leq i \leq m ) where ( 2^m ) is the number of peers in the overlay</td>
</tr>
<tr>
<td>Finger.interval</td>
<td>( (n + 2^{i-1}), (n + 2^i), 1 \leq i \leq m )</td>
</tr>
<tr>
<td>Finger[i]</td>
<td>( (n + 2^{i-1}) \mod 2^m, 1 \leq i \leq m )</td>
</tr>
<tr>
<td>Finger. successor</td>
<td>The next live peer on the identifier circle, namely finger[i]</td>
</tr>
<tr>
<td>Predecessor</td>
<td>The previous live peer on the identifier cycle</td>
</tr>
</tbody>
</table>
Since, the fingers increase according to the power of two to ensure that each query step reduce the distance to the goal by half, as a result, Chord lookup complexity of $O(\log N)$ [28].

![Chord routing model: with identifier circle consisting of eight peers & finger table.](image)

**Figure 1:** Chord routing model: with identifier circle consisting of eight peers & finger table.

Example Figure 1 shows that, the finger table of peer 5. The first finger of peer 5 points to peer 8, as peer 8 is the first peer that succeeds $(5 + 2^0) \mod 2^5 = 6$. Similarly, the last finger of peer 5 points to peer 25, as peer 25 is the first peer that succeeds $(5 + 2^4) \mod 2^5 = 21$.

Two important characteristics of this system are: each peer knows more about peers closely following it on the identifier circle. Next, usually finger table does not hold adequate information to determine the successor of an arbitrary key [7].
3.2.2 Stabilization Protocol

Every peer in the ring based Chord overlay uses a stabilization protocol that runs periodically in the background to update the successor pointers and the finger tables in the event of dynamic joins, leaves or failure [30] [43]. Actually, stabilization protocol is needed when a peer joins or leaves the system. The protocol that updates the finger pointers is called fix fingers. In order to give the guarantee for correctness of lookups in the set of participating peers changes, Chord must ensure that each peer’s successor pointer is up to date [7]. Lookup success rate and overall bandwidth consumption are generally included in these constraints.

3.3 Lookup Procedure

![Query flow chart in Chord Identifier space.](image)

Figure 2: Query flow chart in Chord Identifier space.
The object lookup protocol is responsible for locating a data object within a P2P overlay network. In Chord overlay network, when a peer query something, it basically hashes the data key (file name) to generate key identifier i.e. D_ID and searches for a peer with that key. Based on the successor pointer in finger table, queries requesting for a given identifier can be moved around the identifier circle until it gets the peer that contains the key identifier this is the peer that query maps to [7].

A stabilization protocol is used in Chord lookup service to maintain the successor pointer to ensure the correct lookups. For finding the correct location of an object, Chord requires $O(\log N)$ messages, where $N$ is the total number of peers in the overlay network.

If we consider the Chord ring in Figure 1, Here $m=5$ and for that reason at most 32 peers can take place simultaneously in this Chord identifier circle. In this figure only 1, 5, 9, 14, 19, 25, 28 and 30 are live.
peers, however others are logical peers. Key 10 is assigned to peer 14, peer 19 holds key 15 and key 17 while peer 30 is contained key 29 and so on.

Figure 3 depicts the lookup process for identifier 29 at peer N5 on the Chord identifier circle. According to the key peer mapping method, peer N5 gets a query request for key K29 from a peer X residing outside the network and checks the keys are assigned here or not. If it fails to find the key, it will use the mapping function to check which of its neighbors are closed to the key/value location [8]. Finger table is used to determine the interval in which key K29 is assigned. In this case, key 29 is located in the interval [21, 5), that is related to start logical identifier 21 and the successor is 25. After that, peer N5 forwards the query request to peer N25. Having received the query, it will check the key allocated here. If no, it will search its finger table for given key 29. Peer 25 will find that the closest point peer ID before identifier 29 is peer N28 in interval [27, 29) which is related to the successor peer N28. Similarly, Peer 28 finds that identifier 29 is greater than N28 but smaller than N30 which is the successor of N28. As a result, N28 consequently asks peer 30 where the key 29 along with an IP address of the requested peer X. Finally, peer N30 finds key 29 and instantly initiates the link with X [49] [25].

The query process concludes that, the requested data ID will be searched in its finger table. If the data ID is equal or small than the peer ID in the finger table, the peer ID will be selected as a responsible peer for data ID. Otherwise the query will be forwarded to the last entry in the finger table [14]. The example shows that, the searching for a key is therefore an \(O(\log N)\) procedure.

### 3.4 Replication

Data replication typically refers to the provision by storing the same information on different hosts [12]. The data replication system is a key component of any file sharing system. Peers may not always be alive in the system that is one of the significant attribute of P2P distribution systems. As a result, the stored data on the leaving peer will be no longer available [51]. To avoid this problem, data replication is too important. Data replication is also necessary for both load-balancing and fault-tolerance. In DHT Chord, the successor peer becomes responsible for ID and data of the leaving peer i.e. data is replicated to its successor list of peers [14]. Replication improves availability, reliability and performance of the system by removing single point of failure as well as reducing communication overhead and data access latency [9]. Finally, replication improves the system scalability with acceptable response times.

![Figure 4: Chord based distributed storage system.](image-url)
Figure 4 shows a distributed storage process in Chord protocol. It has three layers. The upper layer provides a file system metadata and mapping operations on the lower level block operations. The middle layer i.e. “block storage” layer is used to insert and retrieve data blocks identifier with unique key by implementing distributed hash table. It would take care of storage and replication of blocks. The block storage layer uses Chord to identify the peer that is responsible for storing a block. To read or write the block, the layer communicates with the block storage server on that peer [7].

3.5 Key-Value Insertion

Chord is a well-organized distributed lookup system based on consistent hashing that performs unique mapping between an identifier space and a set of peers [10] i.e. storing each key/value pair at the peer to which that key maps. Inserting a new key-value pair into the hash table and also to find the key in the overlay network, mapping function is being used. This function uses the key itself to determine the peer which will store a pair [8]. When a peer receives a request with the hashed key identifier, it will use the map function to check how close it is to the key place on the identifier circle. In figure 2 when N8 wants to put a K17 value. It will use the same procedure to query the K29 in Chord identifier circle that are described in previous section. When the query reaches N19, it will verify how close the hashed key is placed in identifier circle and decide that it is the best place to store the key. Then peer N19 will send its IP address to N8 peer, which will utilize this IP address to insert the key in this peer.

3.6 Bootstrapping

Join and leave protocols is dynamic in P2P overlay, neighbors can update their routing state for efficient lookup. To join a peer in the Chord identifier circle, a peer needs to be discovered from overlay that is existed in the network. The process of discovering and contacting an existing peer is called bootstrap peer. This bootstrap peer is the entry point of the joining peer [9]. A particular bootstrap process is used to initialize the routing table for newly joining peer. For example, in figure 5 when peer N15 wants to join the overlay, its needs to discover a peer from the network that are the entry point. So that peer N5 is the entry point of N15.

3.7 Placement of the Peers

In dynamic P2P systems, peers may leave and join in the network at any time. The process of peers arriving and leaving the overlay is called churn in P2P system. To simplify these mechanisms, Chord DHT protocol establishes a stabilization protocol to validate and update successor peers [14] and also maintains a predecessor pointer to locate the key efficiently even in highly unstable conditions. Both join and leave take overall time of $O(\log_2 N)^2$. To achieve this goal during peer join and leave, it has to maintain the following invariants [7]:

1) Correctly maintained the successor list in each peer.
2) Initialize the predecessor and fingers table of joining peer.
3) Update the finger tables and predecessor of joining and leaving peer.
4) Peer successor (k) is responsible for every key k.

3.7.1 Joining of the Peers

In order to join a peer in the identifier circle, the successor pointers of a few peers need to be change. To make the lookup process accurate, finger tables are needed to be updated every moment [30]. There are few more steps to do for the joining procedure [14] [50].
1) Using the consistent hashing function to create a peer identifier (P_ID) by using the IP address of joining peer.

2) Creating a link with existing peer in the overlay network that is the entry point of the Chord identifier cycle.

3) Forward the P_ID in the identifier circle of newly joining by existing peer to find the suitable position for this new peer according to Chord DHT.

4) After getting a suitable place, new peer gets an acknowledgement from the existing peer and joins this place.

5) At the same time routing information in the system is to be updated, so that new peer can have a valid successor pointer and finger table.

Figure 5: Chord routing model: After peer 15 joining.
6) Finally, the new peer acquires all the data from the successor peer which held the data previously. The data that is retrieved by the new peer should only be associated with the key it is responsible for.

Figure 5 shows that, peer N15 wishes to join the Chord identifier circle, it also knows an existing peer by using bootstrap method, assume peer N5. Peer N5 is the entry point of peer N15. At the same time peer N15 obtains its peer ID by hashing the IP address of peer N15. The new peer N15 sends a join request to peer N5. Peer N5 uses the algorithm to find the successor of N15 by checking its finger table and calculate that, which successor peer is immediately greater than N15. Peer N5 gets N19, so that N19 is the successor of N15. At that time N15 get a response message from N5 to join the circle before N19. As a result, N15 is the new predecessor of N19 replaced with N14 and a response message, the initialized finger table is sent back to peer N15. Therefore, peer N15 sends request to inform all peers whose finger tables should refer to N15 [49].

3.7.2 Leaving of the Peers

Willingly leave the system, this instance can be treated as a peer failure. In order to leave a peer in the identifier circle, a few peers need to change their successor and predecessor pointer for acceptable lookup. In figure 5, if peer N15 leaves the Chord identifier circle, change occurs only in the successor pointer of N14 and the predecessor pointer of peer N19.

3.7.3 Finger table Update & Transfer Data key

Peers in an overlay network change dynamically. Finger table update and key transfer process occur for peer join or leaves the Chord identifier circle. Stabilization protocol, fix_ finger and check_ predecessor operations are used to stabilize the network. In Chord circle, new peer initiates its finger table and informs other peers to update their finger table. At the same time new peer retrieve all the data on it associated with the key it is responsible for. On the other hand, the leaving peer should transfer the key to its successor peer, and notifies other peer to update the finger table [29]. Figure 5 shows that, after peer N15 joins the identifier circle, it fixes its successor pointer N19 and also informs N19 to change predecessor pointer N15. With the help of stabilization protocol, peer N14 easily detects that peer N15 as its new successor and sets its successor pointer. Hence, changes in finger tables for N9 and N14 are shown. Finally peer N19 transfers key K15 to peer N15 [49].

3.7.4 Redundancy

In traditional Chord DHT, each peer has a redundant connectivity between DHT peers that can help to achieve the system path redundancy [27]. This system, forwards the query to one of the neighbor peer (successor) that is closest to the target peer. To observe the conventional Chord, finger table is one of the significant elements for lookup service that provide $(\log N)$ path length between any two peers, where $N$ is the total peers in the network [27].

For example, the finger table in figure 5 shows that there are two or more object point to the same successor peers. This redundant information misused the limited space in the finger table. As a result, finger table addressing space is reduced of this peer. Lookup effectiveness of Chord ring can be superior if we can eradicate the redundancy entries in finger table and insert new peers on that place [25].

3.7.4.1 Optimized Finger Table

On behalf of efficient lookup process, it is needed to optimize the finger table. For optimizing the finger table, first to check the numbers of redundant successor entries in the table during update time.
Traditional finger table in Chord DHT has a border line, which is half of the identifier circle to forward the query request to the successor peer. In an attempt to extend the lookup border line, we can insert a live successor peer from the other half of the identifier circle according to the proportion of X and Y with adding Z. where, \(X = (2^m - 1 + S - L)\), \(Y = (\text{count} + 1)\) and Z is the last entry successor in this finger table [25]. Finally, we can rewrite the equation:

\[\text{New entry Peer ID} = Z + I \times \frac{X}{Y} = Z + I \times \left\{\frac{(2^m - 1 + S - L)}{(\text{count} + 1)}\right\}\]

Here, \(2^m\) is the number of peers in the identifier circle, source peer is “S” and L is the large successor peer ID in the table. And “\text{count}” represents the degree of redundant entries information in finger table. Furthermore, initial value of “I” is 1, which range is 1 to “\text{count}” i.e. I=1; < (count + 1).

Figure 6: Chord routing model: path of query for key 29 starting at peer 5 with update the finger table.

Figure 6 illustrates that, there is no redundant entries in the finger table. In traditional Chord ring, finger table of N5 has redundant information. All are pointing the same peer N9. As a result, new finger tables are optimized. Based on the optimized finger table method, 2nd and 3rd N9 pointer information will be removed from conventional finger table. At that time the value of the count is set to 2 and adds new
successor entries in the optimized finger table according to the above function. The new successor peer is N30 and N1. As a result, the optimized finger table has no redundant information that can reduce not only the routing time but also routing hops [25].

3.7.4.2 Optimized Lookup Procedure

Figure 6 also demonstrates the lookup procedure, when N5 gets a query request from X for K29. In this lookup process, initially N5 searches the K29 in its own database. If not found and then checks its finger table. In the optimized finger table, Peer N5 finds that the N28 is the immediate closest peer ID before 29. So that, N5 forwards the query request to the N28. Similarly, N28 follows the same procedure as N5 and finds that N30 is the live immediate large peer ID for 29 i.e. N28 announces that identifier K29 is between peer ID 28 and its successor peer N30. N30 is returned as the final result of the lookup process, since the relevant identifier information is stored in the successor of this peer. Finally, N28 forwards the query K29 with an IP address of the requested peer X to N30 and instantly initiates the link between these peers [25] [49].

3.8 Analytical Result

Routing of a query request is based on a routing table and the table is called a finger table. In traditional Chord protocol, to identify the accurate peer which holds the desired data item in the overlay is required the average $\frac{1}{2} \log_2 N$ numbers of hops, where N is the number of peers in the Chord ring. We analyze that optimized finger table not only reduces the routing overhead but also decreases the number of hops for each query operation than conventional Chord finger table. As well as it increases the lookup efficiency.

![Lookup Latency Analysis](image)

**Figure 7: Analytical result for lookup latency.**

Figure 7 depicts the analytical study in the traditional Chord and optimized Chord based lookup result. In this graph, we observe that optimized Chord lookup process is more efficient than the traditional Chord. The main element to create a different line graphs is the redundant information entries in the finger table.
These values on the graph is not a precise value, it is the analytical value to plot the graph to show the understandable difference between the two procedures.

3.9 Conclusion

Peer-to-Peer systems are attractive for self-organizing capability and fully distributive characteristics that are differing from traditional Client/Server architecture for some particular application. In this chapter, we discussed about the routing table i.e. finger table which is one of the significant component for Chord lookup service that can reduce the overhead of routing. The main task of P2P is to share the file or data. To discover the file or data in the large scale distribution network, Chord lookup service is preferable. In Chord: given a key, it identifies the responsible peer that is storing the key’s value with the help of finger table (routing table). Routing table on traditional Chord protocol has redundancy information. In this chapter we also discuss to improve the lookup efficiency by removing the redundant information to create optimized finger table. To analyze the lookup efficiency, optimized Chord is more efficient than the traditional Chord.
Chapter 4
Hierarchical P2P with Decentralized Database Framework

Overview
A fundamental task of Peer-to-Peer application is file sharing to locate files from the appropriate peer in the overlay network. For effectiveness and to increase the lookup efficiency of large scale file sharing system, hierarchical DHT architecture is more suitable for better fault isolation, bandwidth utilization and better adaptation to the physical network [3] [5]. The key feature of our design is it uses Superpeer based routing algorithm as well as applies optimized finger table in the DHT Chord that helps the system to decrease the average routing latency. The basic principal for estimating the lookup efficiency is done by calculating the number of hops required to route a message to the responsible peer and the size of the routing table [40]. To achieve the benefit in the network, P2P service is divided into layers forming a hierarchical structure. In this architecture, local tasks are solved inside their own layers hence decreases the system workload in the upper layer which is not possible in the traditional P2P system. It is widely used in Client/Server systems [48]. Moreover, amount of the flexibility and resilience of the network is directly proportional to the number of layers. Due to the heterogeneity of the peers in the overlay, we construct a three-layer hierarchical network that can offer a way to reduce the lookup path length.

4.1 System Model
In DHT based overlay architecture peers are grouped into different layers that create a hierarchical overlay. Participating peers in the P2P overlay are heterogeneous. Based on the availability and capability of peers, they can be classified into three categories: Ultra-superpeers, Superpeers and Ordinary peers. We design a three layer hierarchical model according to their capability and availability.

The upper layer holds the peers which are the most powerful and stable than the other peers. The middle layer holds the peers which are less powerful and stable than the upper layer peers. Finally, Ordinary peers take place in the lower layer in the hierarchy to make groups. The upper layer Ultra-superpeers organize a structured based DHT overlay network that forms a Chord ring to provide efficient routing across the overlay network [5] [35].

In our design, we implemented the optimized finger table in the upper layer P2P ring. Ultra-superpeers in upper layer and Superpeers in middle layer are performing as a server. Additionally, stabilization protocols are running periodically over the Chord identifier circle that can maintain the finger table [5]. Middle layer Superpeer manages a single group of Ordinary peers. The Ordinary peers form the single connection structure where peers are communicating only with their Superpeers [47]. All peers frequently join and leave the lower layer overlay network without affecting the whole network except its associates.

Peers with superior availability and capability may dynamically migrate to the upper layer [23] [36]. Each peer in the network has a peer ID. The peer ID is obtained from IP address/port pair by using the consistent hash function (SHA-1) which is a positive integer [18]. The number of peers in the lower layer group may vary from one group to others.

In our hierarchical architecture, peer’s ID in the lower layer groups must be smaller than the associated Superpeer ID and the Superpeer ID in the middle layer must lie between the Ultra-superpeer ID and predecessor ID in the upper layer. Every Superpeer contains the metadata of the lower layer Ordinary
peers as index pointer. Similarly, Ultra-superpeers also contain the pointer index as Superpeer with Superpeer ID and the finger table. As a result, during peer join, leave and migration, the maintenance cost of overlay structure remains convenient [23].

During the query process, query is always routed through the querying peer’s Superpeer. If querying data ID is not available in the middle layer Superpeer then the query request is forwarded to the associated Ultra-superpeer in the upper layer [36] [48]. It is noticeable that, most of the query process has been done in the lower layer. If the query process is done in the lower layer, rests of the network peers remain free from the responsibility of this query process.

We define our hierarchical DHT as a P2P system that:

- Organizes all participating peers into 3-layer hierarchy,
- Defines the individual responsibilities to the peers in different layers,
• Allows migration/transition between the layers and  
• Uses Superpeer based and DHT Chord as the basic lookup concept.

4.2 Design Issues

To design a hierarchical model for a P2P system, we focused on some objectives which are given below:

1. Easy to maintain the system.
2. Should decrease the lookup path length.
3. Should reduce the effect of the network during peer join and leave.
4. To minimize the fault tolerance compare to the traditional system.
5. Should reduce the routing time to finish the lookup on desire data item with the help of optimized finger table.

4.3 Basic Components of Hierarchical Model

To explain the operation of our hierarchical model, it is necessary to clarify the following parameters.

4.3.1 Routing Table

This hierarchical model uses optimized routing mechanism. At first, routing process is executed in the associated Superpeer in the middle layer. If the requested key ID is not available then the routing procedure is executed inside the ring in the upper layer until the routing message has been forwarded to the expected peer whose peer ID is numerically closest to the requested key ID [48]. For routing, the upper layer Chord ring uses the optimized finger tables that are illustrated in chapter 3. As a result, there is no redundant information in the finger table that can reduce not only the routing time but also the routing hops [25].

4.3.2 Data Structure

Distributed data sharing techniques are applied in the hierarchical P2P model. We choose Chord because of well-organized data structures and efficient routing schemes. A data item is represented by a key-value pair, where key is the data name and value is the content associated with the key [18]. A global consistent hash function is used to map the object key to the peer identifiers. A peer uses a put(key, value) command for insert the data item and get(key) command for obtaining the value of the data item in the overlay [18]. In the 3-layer hierarchical based P2P system, when the lower layer Ordinary peer wants to share the file, it is necessary to send the metadata information to its associated Superpeer in the middle layer.

<table>
<thead>
<tr>
<th>Key_ID</th>
<th>Peer_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>N12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key_ID</th>
<th>Peer_ID</th>
<th>Superpeer_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>N12</td>
<td>N1</td>
</tr>
</tbody>
</table>

Figure 2(a): Index of Superpeer.  (b) Index of Ultra-superpeer.

The Superpeer keeps the metadata in the index. The pointer index has two field, key_ID and peer_ID. Where key_ID is the file sharing identity and peer_ID is the Ordinary peer identity. However, it is necessary to further declare the metadata to the associated Ultra-superpeer in extended format that are
kept in the index of the Ultra-superpeer. The extended index has three field, key_ID, peer_ID and Superpeer_ID. The Superpeer_ID is the associated Superpeer of the Ordinary peer. At the query process, the metadata index provides the sufficient information [15].

### 4.3.3 Ordinary/Lower Layer

In our hierarchical system, we consider the Ordinary peers that have joined the system recently. The Ordinary peers are placed in the lower layer and do not have their own routing table. Basically unstable peers are the member of the lower layer hierarchy and may be disconnected after sharing the resource from P2P network. The Ordinary peer in the lower layer cannot directly connect with each other because of their single-connection structure. The Ordinary peer communicates with the other Ordinary peers through the associated Superpeer in the second layer hierarchy. As a result, at least two hops are needed to communicate with each other [46] [47]. In the single-connection structure, every Ordinary peer uses the PING/PONG algorithm periodically to check the connectivity with the Superpeer. The Ordinary peers send the PING message to its Superpeer and the Superpeer responds with a PONG message [5] [47]. Additionally, an Ordinary peer becomes Superpeer by sending an upgrade (capability and availability) level request to the associated Superpeer. The current Superpeer creates a list of Ordinary peer request which may become a Superpeer after leaving the current Superpeer [23].

### 4.3.4 Super/Middle Layer

The peer is called a Superpeer which is more stable and capable than the Ordinary peers which is placed in the middle layer of the hierarchical system. In this layer, Superpeers are also organized by the single-connection structure. So, the middle layer Superpeers checks their connectivity with the upper layer via PING/PONG messages. Moreover, the Superpeers act as a server to maintain the index pointer information of Ordinary peers and are also responsible for querying an object in the overlay network [5] [46]. Finally, Superpeers execute multiple tasks.

### 4.3.5 Ultra-super/Upper Layer

The Ultra-superpeer is also a peer which has an ideal characteristic, higher availability and predicted to be available in future for long a period compared to the middle layer Superpeer. In this layer, Ultra-superpeers are organized in a circle where peers are connected with each other similar to the Chord network. Moreover, the Ultra-superpeer is the gateway or path to communicate among the Superpeers in the middle layer. In the Chord ring, each Ultra-superpeers apply the stabilization protocol periodically to update the successor/predecessor pointer and the optimized finger table in the event of peer’s failure or migration from middle layer [5] [46] [47].

### 4.3.6 Superpeer Selection

In Peer-to-Peer network, we focus on the peer heterogeneity that may affect the scalability and efficiency of the P2P overlays. Peer heterogeneity can be classified by the physical capability and stability of the peers. Physical capability can be measured by bandwidth, CPU power and storage capacity, while stability can be measured by the duration of availability of the peers in the P2P network [5] [36] [37]. The Superpeer can be selected from the Ordinary peers based on the following factors:

- NAT support or firewall
- Bandwidth
- Speed of CPU
- Storage space
- Number of TCP connection supported
- Stability or availability

Similarly, Ultra-superpeers can be selected from other Superpeers by comparing the above mentioned factors with other Superpeers.

### 4.3.7 Superpeer Redundancy

In Peer-to-Peer network, peers can join and leave the system dynamically. After peer’s joining, it registers its shared file’s metadata with its associated Superpeers. The Superpeer keeps the metadata in the index pointer. All the Superpeers in the middle and upper layers also contain the physical strength information of immediate lower layer peers that helps to select the candidate Superpeer. Moreover, the selected candidate Superpeer also periodically copies the index of the connected Superpeer [15]. When the Superpeer fails or leaves the system, its corresponding Ordinary peer is detached from the network until the candidate Superpeer is found [15]. The candidate Superpeer is an Ordinary peer which is the member of this group and migrates in the middle layer to perform as Superpeer. But the upper layer backups the metadata index in another way. They apply the DHT Chord protocol and use the successor as a backup of Ultra-superpeer’s object or index [37]. In our 3-layer hierarchical model, we apply the redundancy technique that influences the system to tolerate the failure of the Superpeer or Ultra-superpeer which helps the system to be said fault-tolerance capable. In addition, the upper layer Ultra-superpeer also keeps the metadata index. As a result, the object index or metadata index also has three backups in our design. One backup is maintained by the candidate Superpeer of the Ordinary peer group and the rest of the backups are maintained by the Ultra-superpeer and its successor. For example, if the Superpeer X fails, its candidate Superpeer Y will act as a temporary Superpeer of this group’s Ordinary peers to keep all objects that were handled by X. So that Ordinary peers do not need to re-transmit their metadata to their new Superpeer. In the hierarchical structure of the Superpeer based system, redundancy is an important factor to provide reliability to the system and to balance the load of the Superpeers and also to avoid the single point of failure [15].

### 4.3.8 Superpeer Maintenance

Because of dynamic joining and leaving of the peers in the P2P network, some of the Superpeers manage the huge number of Ordinary peers. As a result, some Superpeers are overloaded. If we split the overloaded Superpeers into two Superpeers and merge the low loaded Superpeers, then the load of the system will be balanced [3].

### 4.3.9 Migration of Peers

Migration is another focusing objective in the P2P hierarchy. When a Superpeer in the middle layer fails or leaves, a new Superpeer will be needed for the query process and to make a link between the disconnected Ordinary peers and the rest of the P2P network. The selection of the new Superpeers is based on the firewall support, availability, bandwidth and capacity etc. which are the basis of selecting an Ordinary peer as a candidate Superpeer [36]. After finding the candidate Superpeer, it will migrate to the middle layer in the place of the leaving Superpeer and performs as other Superpeers. Same procedure is also applied between the middle and upper layer in the hierarchy when an Ultra-superpeer fails or leaves. As a result, the number of Superpeers in the middle layer and Ultra-superpeers in the upper layer remains possibly unchanged.
4.3.10 Hierarchical Depth

For achieving better routing performance, we consider the hierarchy depth of the system. If the hierarchy depth increases, the number of routing hops will decrease in the same requested query process. Finally, we assume that, the hierarchy depth will also influence the lookup efficiency [48]. Traditional Chord ring needs $O(\log N)$ hops but in hierarchical Chord ring based query process only needs $O(\log S)$, where $N$ is the number of normal peers and $S$ is the number of Ultra-superpeers in the system.

Figure 3: Lookup process comparison between traditional and 3 layer hierarchical system.

Figure 3 shows that, peer N1 wants to lookup the peer N17 which takes 4 hops in traditional Chord. If we use 2 layer hierarchies, it needs only 2 hops plus the number of hops required to communicate between peer N1 and its Superpeer. Moreover, if we use 3 layer hierarchies, it will require only 1 hop plus the number of hops required to communicate among the peer N1, its Superpeer and Ultra-superpeer.
4.4 Lookup Procedure

![Query flow chart in 3-layer hierarchical P2P network.]

Figure 4: Query flow chart in 3-layer hierarchical P2P network.

In the beginning of lookup process, data ID of the data item is needed to look up the peer. The data is obtained by hashing the data key [18]. Here we consider the 3-layer hierarchy scheme, where the upper layer manages the Ultra-superpeers, middle layer manages the Superpeers and the lower layer is organized by the Ordinary peers. The query message from the Ordinary peer is forwarded first to the associated Superpeer. After getting the query request, Superpeer checks its database for the data item. If
the data ID is found in the Ordinary peer groups, lookup process is done, otherwise the query request is forwarded to the connected Ultra-superpeer in the ring [40] [47]. When the query request arrives at associated Ultra-superpeer in the upper layer Chord ring, it checks its own database. If the data item is not found, it forwards the query to its successor based on the optimized finger table similar to Chord ring and finds the data item. Ultra-superpeers in upper layer Chord ring collectively contain all the overlay peers metadata information. Thus the data item is found efficiently if it is available in the overlay.

Figure 5 shows the lookup process for three different identifiers from different peers that are indicated in variation of colored line arrows. Green arrow line illustrates that, Ordinary peer N25 wants to find the key 35. Peer N25 sends this query request to its Superpeer N40. Superpeer N40 checks its database and obtains key 35 under its control. As a consequence, Superpeer N40 finishes the query request and returns peer N35 to peer N25. This query process has been done in the middle layer. As a result, rest of the
overlay peers remains free from taking any responsibility to a response to the query from N25. Similarly, Peer N45 sends the query request to its Superpeer N57 to find the key 78. Since the key 78 does not exist in the Superpeer N57, Superpeer N57 forwards the request to associated Ultra-superpeer N80. After getting the query request, it checks that key 78 fall under the Ultra-superpeer. Then Ultra-superpeer completes the query process by instantly initiating the link between the peer N78 and N35. This query process has been done with the help of the upper layer peers that are indicated by the blue line in the figure. Moreover, another query process has been done with the cooperation of entire overlay peers that are denoted by the red dotted line where N60 sends a request for key 25. N60 directly forwards the query to its successor N79. N79 forwards the request to its connected Ultra-superpeer N80 because, the key 25 is not present in N79. Similarly, N80 is also not found the key in its database. So, N80 forwards the query request to its successor peer according to the optimized finger table. Finally, the key 25 is found which is under control of Ultra-superpeer N42 and creates a link between the peer N60 and N25 [25] [49].

4.5 Placement of Peer

In P2P overlay system peers may leave and join significantly. Different types of peers in separate layer are designed in this hierarchy. In 3-layer hierarchy, peers apply different methods to join and leave the systems that are described below:

4.5.1 Joining of Peers

The Peer that wishes to join the P2P overlay network will become an Ordinary peer by taking place lower layer first. It cannot be directly recommend for becoming Superpeer or Ultra-superpeer in middle or upper layer in the hierarchy. The Ordinary peers can only migrate to the immediate upper layer based on their physical strength (bandwidth, stability, storage space etc). The joining peer sends the joining request to any existing known Superpeer. That Superpeer then examines the hashed ID (peer ID) of the joining peer and finds out the place of the joining peer in the overlay network. After finding the place, the Superpeer sends a response message to the joining peer with the information of joining place which is suitable for that particular peer. Then the Ordinary peer joins the overlay and sends the metadata to its associated Superpeer. The Superpeer also informs the connected Ultra-superpeer about new peer joining [23] [37].

4.5.2 Leaving of Peers

Peers are classified into three types and are placed at three different layers. In this system, peers can leave any of the layers. The process of leaving of the peers is discussed below:

4.5.2.1 Leaving of Ordinary Peers

The Superpeer detects the leaving peer by PING/PONG messages in the lower layer. When Superpeer identifies the leaving peer, it changes the status of the leaving peer. The Superpeers delete the metadata of leaving peers after a few moments and also inform the associated Ultra-superpeer in upper layer to delete the metadata of this Ordinary peer. The Ordinary peers may leave the hierarchy without informing any other peers because it does not maintain the routing table.

4.5.2.2 Leaving of Superpeers

Leaving of a Superpeer from the middle layer is little bit complex than the Ordinary peer leaves. Before leaving the system willingly, it must select a candidate Superpeer from the Ordinary peers in the group. After selecting a new Superpeer, all metadata is transferred to the new Superpeer. The new Superpeer migrates to the middle layer and performs as a leaving Superpeer. Migrated candidate Superpeer also
informs its connected Ultra-superpeer to update the metadata and its associated Ordinary peers about its arrival. When the Superpeer leaves without prior notice, all the metadata of Ordinary peer are lost but in our design, Ordinary peers group have a candidate Superpeer that is the backup of the metadata. Because of metadata backup, Ordinary peers need not to send their metadata to its Superpeer. Leaving Superpeer substitutes with candidate Superpeer, as a result, number of Superpeer in middle layer will be unchanged [18].

4.5.2.3 Leaving of Ultra-superpeers

When the Ultra-superpeer leaves the system, candidate Ultra-superpeer selection, migration all are similar to the Superpeer’s leaving process except the backup of metadata. Upper layer Ultra-superpeers uses the DHT Chord protocol and the successor is the backup of the metadata. The new Ultra-superpeer also informs its successor. So that, number of Ultra-superpeer in the upper layer remains the same. When the candidate Ultra-superpeer migrates to the upper layer, at the same time candidate Superpeer also migrates to the middle layer to fix the link among the three layers. It is exceptional to leave the Ultra-superpeer because of its stability.

4.6 Advantage of Hierarchy

The basic advantage of the hierarchical P2P structure [36] [40] [48]:

1) DHTs network is well organized. Because of peers heterogeneity, the upper layer DHT overlay is more stable, second layer is just less stable and the lower layer Ordinary peers are unstable. In this system peer newly joins only at the lower layer and become a supper or Ultra-superpeer based on processing power, storage capability and stability. During peer’s join or leave, it does not affect the other group peers, only corresponding peers are affected. On the other hand, in flat overlay peer joins or leaves the Chord ring in any time but it is not possible in hierarchical overlay network. Therefore, flat overlay is not stable well and hierarchical overlay is well organized than that of flat.

2) In hierarchical model, upper layer use the DHT Chord protocol. All the peers in upper layer are stable and maintain the finger table. Moreover, peer does not join or leave frequently, so that, the finger table will be changed infrequently. But in flat overlay, peer joins or leaves frequently, as a result finger table will be change frequently.

3) We consider the large-scale overlay lookup process. To find out any data item from overlay, traditional Chord requires average $\frac{1}{2} \log_2 N$ number of hops. Our 3-layer hierarchical model requires average $(\frac{1}{2} \log_2 U + 2)$ number of hops, where N and U are the number of peers in the Chord ring and also N is much larger than U. After putting the value of N and U, we analyze that, the average number of hops is significantly reduced for hierarchical lookup than conventional Chord.

4) The lookup latency is also significantly decreased, because the average number of hops is reduced significantly in the hierarchical model. Assume that, there are 50000 peers in the overlay and the link latency is 1ms/hop for Ordinary peer and 0.5ms/hops for superpeer or ultra-superpeer. So, for conventional Chord, average number of hops is 7 and for 3-layer hierarchy is 4. As a result, average lookup latency is $(7*1) = 7$ms for traditional Chord and $(4*0.5) = 2$ms for hierarchical model.
Chapter 5

Analytical Evaluation of Hierarchical P2P Model

5.1 System Performance

Performance of the system can be measured based on the lookup efficiency and the lookup success rate. Moreover, lookup efficiency of the system depends on the number of peer hops required to route the message to the responsible peer and forwarding of the routing message depends on the routing table.

5.2 Analytical Evaluation

We present the performance analysis of 3-layer overlay with conventional Chord on the basis of number of lookup hops and the latency/delay of these two models. For traditional Chord, we assume that [23] [47]:

- Peer and key identifier are random
- Number of peers = N
- Number of entries in the Finger table = $O(\log N)$
- Probability of the number of forwarding = $O(\log N)$

Because of using the old finger table in conventional Chord which is half of the identifier circle to forward the query request to the successor peer. So that, the average hops required for lookup process is $\frac{1}{2} \log_2 N$ to find the desired peer [7].

For 3-layer hierarchy model, we used Chord in the upper layer and in the middle and lower layer we used single connection structure. For this purpose, we take the following assumptions for comparison [23] [47] [55].

- Total number of peers in overlay = N
- Number of Ultra-superpeers = U
- Number of Superpeers = S
- Number of Ordinary peers = $N - (S + U)$
- Number of peers in a group in lower layer = $\frac{N - (S + U)}{S}$ (Assuming all group size to be equal)
- Probability of finding data item at super node = Q
- Number of entries in Finger table = $O(\log U)$
- Time complexity is $O(\log U + 1 + 1)$

Time complexity is the least amount of time to execute the process or program. When the number of peers is N, the typical time complexity of searching is $O(N)$ for unstructured P2P network and $O(\log N)$ for structured P2P network [55]. In our proposed model, we used unstructured topology in two layers and structured topology in one layer. Note that, if the group size is little larger, associated Superpeer could track all the peers in the group. In this case, Ordinary peer forwards the query request to the associated Superpeer. As a result, local lookup process required $O(1)$ step [40]. In the lower and middle layer, we apply the single connection intra group structure with Superpeer based query. For that reason, the typical time complexity of searching is $O(1)$. In upper layer, the typical time complexity of searching is $O(\log U)$.
because we apply structured based Chord protocol. Finally, total time complexity of the model is $O(\log U + 1 + 1)$.

In this hierarchy model, three cases occur during lookup process:

1) Average number of hops is required 1, if the query process will be done between lower layer and the middle layer because of forwarding a query to the associated Superpeer.

2) Average number of hops is required $(1+1) = 2$, if the query process will be done among lower layer, middle layer and associated Ultra-superpeer (without using Chord ring) because of single connection intra group structure.

3) Average number of hops is required $\frac{1}{2} \log_2 U$, if the query process will be done in upper layer Chord ring.

Note that, case 3 is the special case which will happen occasionally because of more Ordinary peers than the Ultra-superpeers.

Thus, the total average number of hops required for looking up any key in 3-layer overlay is given by [23]:

$$\text{Lookup 3-layer} = 1 + 1 + \frac{1}{2} \log_2 U$$
$$= 2 + \frac{1}{2} \log_2 U$$

hops on average, given that $U$ is the number of Ultra-superpeers in the system. The expression for 3-layer hierarchy reflects the first hop from the Ordinary peer that initiates the lookup to its Superpeer, the second hop from the Superpeer that initiates the lookup to its associated Ultra-superpeer and the subsequent $\frac{1}{2} \log_2 U$ hops to resolve the responsible Ultra-superpeer.

Here,

$\text{Required number of hops} = \frac{1}{2} \log_2 U$ in upper layer, because we apply Chord protocol.
$\text{Required number of hop} = 1$ in middle layer, because single connection intra group structure.
$\text{Required number of hop} = 1$ in lower layer, because single connection intra group structure.

When the system is comparatively stable, the expected number of hops is $2 + \frac{1}{2} \log_2 U$.

Moreover, the system lookup latency depends on the peer’s physical capability (response time) and the number of peers required to complete the lookup process. If the link latency of the conventional Chord peer is 1ms then the link latency of our 3-layer hierarchy is less than 1ms because, in hierarchical model, Superpeer and Ultra-superpeer perform the lookup process and have more physical capability of the Superpeer and Ultra-superpeer. As a reason, we assume that the average link latency of a peer in hierarchical system is 50% less than the traditional Chord peer i.e. 0.5ms/hops [48].

Lookup latency = peers link latency multiplied by the numbers of hops (unit time).

Furthermore, in our 3-layer hierarchy model, we analyze that Ultra-superpeer handles the total lookup process as they participate in the upper layer DHT and at the same time, manages the associated Superpeer groups and these Superpeer groups also manage their group of Ordinary peers i.e. Ultra-
superpeers face the whole system load. If the number of the Ultra-superpeers is too low, Ultra-superpeers are overloaded. As a result, system may face the stability problem. But in our proposed hierarchical model, when the load exceeds the capacity of the Superpeer or Ultra-superpeer, we split the overloaded Superpeer or Ultra-superpeer into two Superpeers or Ultra-superpeers to balance the load [3].

5.3 Analytical Result

To analyze the system performance, we assume lot of parameters:

- Total number of peers in overlay $= N = 1024$
- Number of Ultra-superpeers $= U = 8$
- Number of Superpeers $= S = 32$
- Number of Ordinary peers $= \{N-(S+U)\} = 984$
- Number of peers in a group in lower layer $= \frac{N-(S+U)}{S} = 31$

(To calculate how many peers will be there in a group held by a Superpeer, we divided the total number of Ordinary peers by the number of Superpeers as we assumed the group size formed by each Superpeer to be equal.)

- Average link latency $= 1$ ms/peer for traditional Chord.
- Average link latency $= 0.5$ ms/peer for hierarchical P2P.

Number of hops calculation:

Traditional Chord $= \frac{1}{2} \log_2 N$

$$= \frac{1}{2} \log_2 1024$$

$$= \frac{1}{2} \log_2 2^{10}$$

$$= 5$$

$$3\text{-layer hierarchy} = \frac{1}{2} \log_2 U + 2$$

$$= \frac{1}{2} \log_2 8 + 2$$

$$= \frac{1}{2} \log_2 2^3 + 2$$

$$= \frac{1}{2} 3 + 2$$

$$= 3$$

In the above expression, we assume that, total participating peer in the system is $N= 1024$ and also know that, required no of hops for any query in conventional Chord is $\frac{1}{2} \log_2 N$ [7]. After putting the value of $N$, we get the required no of hops to complete the query process.

$$3\text{-layer hierarchy} = \frac{1}{2} \log_2 U + 2$$

$$= \frac{1}{2} \log_2 8 + 2$$

$$= \frac{1}{2} \log_2 2^3 + 2$$

$$= \frac{1}{2} 3 + 2$$

$$= 3$$

In the above expression, we assume that total participating peer in the system is $N = 1024$. In 3-layer hierarchy, we rearrange the total participating peer in three layers. We also assume, no of Ultra-superpeer in the upper layer is $U=8$ and Superpeer is $S=32$. So that, no of Ordinary peers is 984. If we distribute the equal no of peers in a group in the lower layer then we divide the total no of Ordinary peers by the no of Superpeers which is 31 in this case. From analytical description of 3-layer hierarchy which already is mentioned in section 5.2, we know that required no of hops for any query is $\frac{1}{2} \log_2 U +2$. After putting the value of $U$, we get the required no of hops to complete the query process. Finally, we realize that required no of hops in the hierarchical model is fully depended on the no of Ultra-superpeer in the upper layer Chord ring.

Lookup latency:

Traditional Chord $= 5*1$ ms

$= 5$ ms

3-layer hierarchy $= 3*0.5$ ms

$= 1.5$ ms
To analyze the average number of hops and the lookup latency of the system, we can modify the parameters, assuming that:

- Total number of peers in overlay = \( N = 65536 \)
- Number of Ultra-superpeers = \( U = 8 \)
- Number of Superpeers = \( S = 64 \)
- Number of Ordinary peers = \( N-(S+U) = 65464 \)
- Number of peers in a group in lower layer = \( \frac{N-(S+U)}{S} = 1023 \)
- Average link latency = 1 ms/peer for traditional Chord.
- Average link latency = 0.5 ms/peer for hierarchical P2P.

**Number of hops calculation:**

- Traditional Chord = \( \frac{1}{2} \log_2 65536 = \frac{1}{2} \log_2 65536 = \frac{1}{2} \log_2 2^{16} = 8 \)
- 3-layer hierarchy = \( \frac{1}{2} \log_2 U + 2 = \frac{1}{2} \log_2 8 + 2 = \frac{1}{2} \log_2 2^3 + 2 = \frac{1}{2} 3 + 2 = 4 \)

**Lookup latency:**

- Traditional Chord = \( 8 \times 1 \text{ ms} = 8 \text{ ms} \)
- 3-layer hierarchy = \( 4 \times 0.5 \text{ ms} = 2 \text{ ms} \)

When we fix the total number of peers and only change the number of Ultra-superpeers to 32 in the upper layer Chord ring then the number of required hops is:

- Traditional Chord = \( \frac{1}{2} \log_2 65536 = \frac{1}{2} \log_2 65536 = \frac{1}{2} \log_2 2^{16} = 8 \)
- 3-layer hierarchy = \( \frac{1}{2} \log_2 U + 2 = \frac{1}{2} \log_2 32 + 2 = \frac{1}{2} \log_2 2^5 + 2 = \frac{1}{2} 5 + 2 = 5 \)

And the lookup latency is:

- Traditional Chord = \( 8 \times 1 \text{ ms} = 8 \text{ ms} \)
- 3-layer hierarchy = \( 5 \times 0.5 \text{ ms} = 2.5 \text{ ms} \)

To calculate the lookup hops and the lookup latency ratio of the two systems, we assume the number of participating peers \( N = 65536 \), no of Superpeer \( S = 64 \) and the Ultra-superpeer \( U = 8 \), and the link latency for traditional Chord peer is 1ms and 0.5ms for hierarchical peers. From the above expressions we get the required no of hops and the lookup latency for traditional Chord is 8 and 8ms and for our 3-layer hierarchical system is 4 and 2.5ms respectively.
Lookup hops ratio = \( \frac{\text{no of hops in the hierarchical module}}{\text{no of hops in the traditional chord module}} \)

\[ = \frac{4}{8} = 0.5 = 50\% \]

Lookup latency ratio = \( \frac{\text{latency of the hierarchical module}}{\text{latency of the traditional chord module}} \)

\[ = \frac{2.5}{8} = 0.3125 = 31\% \]

Thus we can say that, our hierarchical model is able to reduce the lookup hops by 50% and lookup latency is reduced by 31% compared to traditional Chord.

In the above analytical result of traditional Chord and 3-layer hierarchy shows that, the average number of hops required for lookup process of 3-layer overlay is less than the traditional Chord. Since the number of hops is decreased, lookup latency also decreases in 3-layer hierarchy. We also analyzed that, if the number of Ultra-superpeers in the upper layer is increased, then the average number of hops is increased gradually. As a result, the lookup latency is also increased at the same manner. Moreover, optimized routing table is exploiting in the upper layer in the overlay that can also influence the system to reduce the latency. If we use the large-scale overlay network, than we achieve the large amount of benefit from the 3-layer hierarchical P2P system. As a result, in our 3-layer hierarchical overlay design, the lookup efficiency is more increased than the conventional Chord. Finally, we thought that, performance of hierarchy model is more efficient than the flat one.

![Average Lookup Hops Analysis](image)

Figure 1: Average lookup hops considering the fixed participating peers and customized no of Ultra-superpeer.

Figure 1 shows that the traditional Chord and the hierarchical based lookup result. In this figure, we observed that, the participating peers in the overlay network are fixed but the numbers of the Ultra-superpeer are increased gradually at the top layer in the hierarchy. As a result, the average lookup hops for traditional chord is fixed and the number of lookup hops in 3-layer hierarchy is increased slowly.
Figure 2: Average lookup hops considering the fixed Ultra-superpeer.

Figure 2 illustrates the average lookup hops result for traditional Chord and the hierarchical based lookup. In this figure, the participating peers in the overlay network are not fixed but the numbers of the Ultra-superpeer are fixed at the top layer in the hierarchy. Consequently, the average lookup hops for traditional chord is increased sharply and the number of lookup hops in 3-layer hierarchy is fixed.

Figure 3: Average lookup hops considering the all parameter are customized.

Figure 3 depicts the analytical study of traditional Chord and the hierarchical based lookup result. In this figure, we examine that, the number of hops for finding the desired peer which holds the desired data item is depended on the routing tables of the system. It occurs because the query request is forwarded on the basis of the routing table. We also observed that, when the participating peers are large i.e. large-scale overlay, the average number of lookup hops in 3-layer hierarchy is approximately 50% less than the
traditional DHT Chord lookup. The main factor to differ the line graph is the redundant information entries in the finger table in traditional Chord that required more hops to find the data item and the Superpeer based lookup that are applied in the hierarchy model which reduced the lookup hops. To plot the average lookup hops graph, we use the analytical value to show the comprehensible difference among the two different P2P lookup modules.

Figure 4: Average lookup latency when the all parameter are customized.

Figure 4 demonstrates the analytical study of the lookup latency for traditional Chord and the 3-layer hierarchical based lookup process. In this figure, we examine that, lookup latency is depending on the number of hops required for lookup process and the participating peers responding ability. Peers heterogeneity also influences the lookup latency of the system. In traditional Chord, heterogeneous peers are participating in the same Chord ring, so that the average link latency of a peer is approximately 50% larger than the hierarchical model, because in 3-layer hierarchy, more capable and stable peers are executing the lookup process. We analyzed that, average link latency of the 3-layer hierarchical system is approximately 30% to 50% less than the traditional Chord lookup. We finally stood on a conclusion from our above analysis that hierarchical lookup is more efficient than the traditional Chord P2P lookup module. We use the analytical value of average lookup latency to plot the graph that shows the comprehensible difference among the different P2P lookup modules.
Chapter 6

Conclusion and future works

6.1 Conclusion

We have presented a three layer hierarchical overlay architecture that can handle a large number of peers. Peers in such architecture can frequently join or leave the network, whereas migrating to different layers are also showed in our proposed model. This model is able to discriminate among the peers according to their capability and thus classifying them into three categories: Ultra-superpeer, Superpeer and Ordinary Peer. Our hierarchical design offers higher stability by using Ultra-superpeers at the upper layer which are more reliable peers. We also presented an instantiation of our 3-layer hierarchical model using Chord at the upper layer and Superpeer based single connection intra group structure in the middle layer. Since the number of Ultra-superpeer is less than the number of Superpeer and the number of Superpeer is less than the Ordinary peer, the overhead to find a desired data has become less than the 2-layer architecture by keeping metadata at each layer. To reduce the access load of the database, we used decentralized database to distribute the query load. That approach gives a guarantee for the system to say that there will be no single point of failure. Superpeer and DHT Chord based lookup process are used in the hierarchy which also helps to reduce the average number of hops to find the desired peer holding the desired data item. Lookup latency of the model is also decreased by confirming that the average number of required hops is decreased to find the desired peer. This model executes the lookup process with the help of Superpeer and Chord protocol that extensively reduces the query traffic load which is common characteristic of the flooding based query. In our model, we also introduce the candidate superpeer and Ultra-superpeer that also hold the metadata same as Superpeer and Ultra-superpeer. These candidate peers are the backup of the metadata which also help the system to increase the fault-tolerance capability. Overall structure ensures that database load is also minimized as the load of a particular peer is split among the neighbor peers. As of DHT algorithm, we have used optimized finger table for Chord algorithm which minimizes redundancy in the finger table and that is helpful for faster and efficient query procedure. To achieve the benefit from the 3-layer hierarchical P2P model, we utilize the efficiency of the structured P2P network and the flexibility of the unstructured P2P network. We have analyzed the lookup performance of 3-layer hierarchical model that is much better than the traditional Chord for large-scale overlay network.

6.2 Future works

In our future work, we will try to give our three layer architecture a feasible look that is, we will do the simulation based on this framework and will investigate the outcomes. Network lookup efficiency along with high churn rate will be taken under consideration when we will try to simulate our model. We have used DHT Chord algorithm with optimized finger table in our design. So, we will also do simulations using other DHT algorithms like Kademlia, Tapestry and Pastry and will investigate how they will perform. We will also try to investigate which will be the optimized level for hierarchical model considering different DHT algorithm. Due to the unauthenticated nature of the peers in P2P system, each participating peer has to involve in the sharing application without sufficient information about other peers. Therefore, security issues can be another future work in our study to see how to keep security in such 3-layer architecture.
References


[45] C. Miguel, C. Manuel and R. Antony, “Peer-to-Peer overlays: structured, unstructured, or both?”, Microsoft Research, Cambridge, CB3 0FB.


