Southern Illinois University Carbondale
OpenSIUC

Dissertations | Theses and Dissertations

12-1-2017

A NOVEL LINEAR DIOPHANTINE EQUATION-BAESD LOW DIAMETER STRUCTURED PEER-TO-PEER NETWORK

Shahriar Rahimi
Southern Illinois University Carbondale, nickrahimi@gmail.com

Follow this and additional works at: http://opensiuc.lib.siu.edu/dissertations

Recommended Citation
Rahimi, Shahriar, "A NOVEL LINEAR DIOPHANTINE EQUATION-BAESD LOW DIAMETER STRUCTURED PEER-TO-PEER NETWORK" (2017). *Dissertations*. 1462.
http://opensiuc.lib.siu.edu/dissertations/1462

This Open Access Dissertation is brought to you for free and open access by the Theses and Dissertations at OpenSIUC. It has been accepted for inclusion in Dissertations by an authorized administrator of OpenSIUC. For more information, please contact opensiuc@lib.siu.edu.

A NOVEL LINEAR DIOPHANTINE EQUATION-BAESD LOW DIAMETER STRUCTURED PEER-TO-PEER NETWORK

by

Shahriar “Nick” Rahimi

B.S., IAU- TNB University, 2000
B.S., Southern Illinois University, 2009
M.S., Southern Illinois University, 2011

A Dissertation
Submitted in Partial Fulfillment of the Requirements for the
Doctor of Philosophy Degree

Department of Computer Science
in the Graduate School
Southern Illinois University Carbondale
December 2017

DISSERTATION APPROVAL

A NOVEL LINEAR DIOPHANTINE EQUATION-BAESD LOW DIAMETER STRUCTURED PEER-TO-PEER NETWORK

By

Shahriar “Nick” Rahimi

A Dissertation Submitted in Partial

Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in the field of Computer Science

Approved by:

Bidyut Gupta, Chair

Wen-Chi Hou

Henry Hexmoor

Koushik Sinha

Sam Chung

Department of Computer Science
in the Graduate School
Southern Illinois University Carbondale
Oct 30th 2017

AN ABSTRACT OF THE DISSERTATION OF

Shahriar "Nick" Rahimi, for the Doctor of Philosophy degree in COMPUTER SCIENCE, presented on OCTOBER 30, 2017, at Southern Illinois University Carbondale.

TITLE: A NOVEL LINEAR DIOPHANTINE EQUATION-BAESD LOW DIAMETER STRUCTURED PEER-TO-PEER NETWORK

MAJOR PROFESSOR: Dr. Bidyut Gupta

This research focuses on introducing a novel concept to design a scalable, hierarchical interest-based overlay Peer-to-Peer (P2P) system. We have used Linear Diophantine Equation (LDE) as the mathematical base to realize the architecture. Note that all existing structured approaches use Distributed Hash Tables (DHT) and Secure Hash Algorithm (SHA) to realize their architectures. Use of LDE in designing P2P architecture is a completely new idea; it does not exist in the literature to the best of our knowledge. We have shown how the proposed LDE-based architecture outperforms some of the most well established existing architecture.

We have proposed multiple effective data query algorithms considering different circumstances, and their time complexities are bounded by (2+ r/2) only; r is the number of distinct resources. Our alternative lookup scheme needs only constant number of overlay hops and constant number of message exchanges that can outperform DHT-based P2P systems. Moreover, in our architecture, peers are able to possess multiple distinct resources. A convincing solution to handle the problem of churn has been offered. We have shown that our presented approach performs lookup queries efficiently and consistently even in presence of churn. In addition, we have shown that our design is resilient to fault tolerance in the event of peers crashing and

leaving. Furthermore, we have proposed two algorithms to response to one of the principal requests of P2P applications' users, which is to preserve the anonymity and security of the resource requester and the responder while providing the same light-weighted data lookup.

DEDICATION

To: Shahram, Maman, Baba, and Asal.

ACKNOWLEDGEMENTS

I am sincerely and heartily grateful to my advisor Dr. Bidyut Gupta for his excellent support and guidance throughout my PhD journey. I believe that I am tremendously fortunate to have worked with such an outstanding scholar. I have learned so much from him, and without his help, I would not have been able to finish this dissertation.

My sincere thanks also goes to all my committee members, particularly Dr. Sinha for providing me with the expertise and continued encouragement and ideas. Thanks to Dr. Hexmoor for helping me to get started in the Ph.D. program, and to Dr. Hou, for his continued support and help. Thanks to Dr. Sam Chung for spending hours reading my dissertation and providing extensive feedback and comments.

Also, I would like to thank Dr. Shohreh Hemmeti, for her unconditional support, encouragements, and patience with me during this process.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

INTRODUCTION

Peer-to-Peer (P2P) Overlay networks began to gain attention in the world of data communication in late 90s. Fifty million users of Napster [1], the first well-known P2P application, proved the success of this new file sharing technology. Although the court of law shut down Napster due to the unauthorized distribution of copyrighted material [2], their enormous achievement opens a new door for industry and researchers to dig deeper into the domain of P2P communication.

Since then, P2P networking has been a popular way to share data among ordinary Internet users. Consequently, P2P applications have consumed a substantial portion of the network resources and accounts for a major amount of traffic on the Internet. Popular applications such as, BitTorrent, Skype, Team Viewer, Facebook video messaging, Spotify, etc., are just few of the many successful applications that have designed based on the P2P data communication architecture. For years, P2P traffic used to consume more than 70% of Internet bandwidth [3]. According to Global Internet Phenomena Report (Sandvine) [4], currently, pure P2P applications have about the 40% of Internet bandwidth consumption. Today, BitTorrent, which is a P2P file-sharing application, alone, accounts for more than 36% of the Internet daily terrific.

1.1 Problem Statement

P2P overlay networks are widely used in distributed systems. There are two classes of P2P networks: unstructured and structured ones. In unstructured systems [24], peers are organized into arbitrary topology. Flooding is usually used for data lookup. Problem arising due

to frequent peer joining and leaving the system, also known as churn, is handled effectively in unstructured systems. However, it compromises with the efficiency of data query and the much needed flexibility. Unstructured networks have excessive lookup costs and lookups are not guaranteed. On the other hand, structured overlay networks provide deterministic bounds on data discovery. They provide scalable network overlays based on a distributed data structure which actually supports the deterministic behavior for data lookup. Recent trend in designing structured overlay architectures is the use of Distributed Hash Tables (DHTs) [25, 70, 71]. Such overlay architectures can offer efficient, flexible, and robust service [27, 29, 72, 73].

However, maintaining DHTs is a complex task and needs substantial amount of effort to handle the problem of churn. Consequently, the major challenge facing such architectures is how to reduce the amount of effort of handling churn while still providing an efficient data query service.

1.2 Contributions

In this dissertation, we have presented a new hierarchical architecture in which at each level of the hierarchy existing networks are all structured. We have used Linear Diophantine Equation (LDE) as the mathematical base to realize the architecture. Note that most structured approaches use DHTs to realize their architectures. Use of LDE in designing P2P architecture is a completely new idea. We have explored many different possible advantages that can be fetched using LDEs; some of these advantages include efficient handling of data look-up, node (peer) join/leave, anonymity, load balancing among peers, to name a few; besides achieving fault-tolerance is reasonably simple. We have shown that the complexity involved in maintaining different data structures is much less than that involved in the maintenance of DHTs. On several points, LDE-based overlay architecture can outperform DHT-based ones. The proposed

architecture has considered interest-based P2P systems [47, 58, 69]. The rationale behind this choice is that users sharing common interests are likely to share similar contents, and therefore searches for a particular type of content is more efficient if peers likely to store that content type are neighbors [36].

1.3 Document Outline

The rest of the dissertation is organized as follows. Chapter 2 starts with reviewing the basic concepts and terminologies in P2P overlay networks. This chapter provides a summary of the key concepts which constitute the technical background of the dissertation and the main research directions. Chapter 3 presents a brief review of the state of the art and provides a summary of well-established P2P architectures. Chapter 4 reviews the concepts and solutions of LDE and introduces the implementation of hierarchical LDE-based P2P architecture. Chapter 5 presents algorithms for intra-group and inter-group data lookup in LDE-based P2P overlay, and then it proceeds to compare the data lookup complexities of the presented P2P overlay with some major structured P2P systems. Chapter 6 provides algorithms to support anonymity and security in the LDE-based P2P overlay. Chapter 7 presents various methods to handle churns. In addition, approaches on how a new peer is able to join or leave the LDE-based system are presented. In Chapter 8, the generalization of the architecture, which means how a peer can possess multiple distinct resource types, is presented. Finally, Chapter 9 provides various scenarios for performance evaluation of hierarchical LDE-based P2P system.

CHAPTER 2

BACKGROUND

Every day, developers invent a new technology to answer ever-changing needs of their clients in data communication world. The legacy protocols such as Internet Protocol (IP) network layer protocol provide limited functionalities. Some important tasks such as locating a data in a network, finding address of users, and many more are not supported in the old-style protocols. Additional systems are necessary to solve these limitations. Systems such as Virtual Private Networks (VPN) [6], P2P file sharing, Distributed Hash Tables (DHT) [7], Content Distribution Networks (CDN) [8], [9], Wireless Ad hoc Networks [10] , etc., are just few examples of the applications that need to coexist in the Internet and add more features to it. All of the named systems (and many more) are required to integrate into the Internet and implement all standard protocols and rules, so they can be functional. Internet has created through the application programming interfaces (API). Developers are able to utilize these powerful building blocks and construct their new topologies on top of the Internet protocols. Moreover, they do not have to deploy new equipment or modify existing software. However, including another layer to network stack makes the entire data communication system more complicated. For instance, the packet header needs to be modified to fit the new requirements. Even, sometimes, overlays may have behaviors that originate misleads in the network. For example, a corruption drops on CDN links, can be interpreted as congestion drops by Transfer Control Protocol (TCP). All and all, any new network that constructs on top of an underlying network and provides a new service is an overlay network (Figure 2.1).

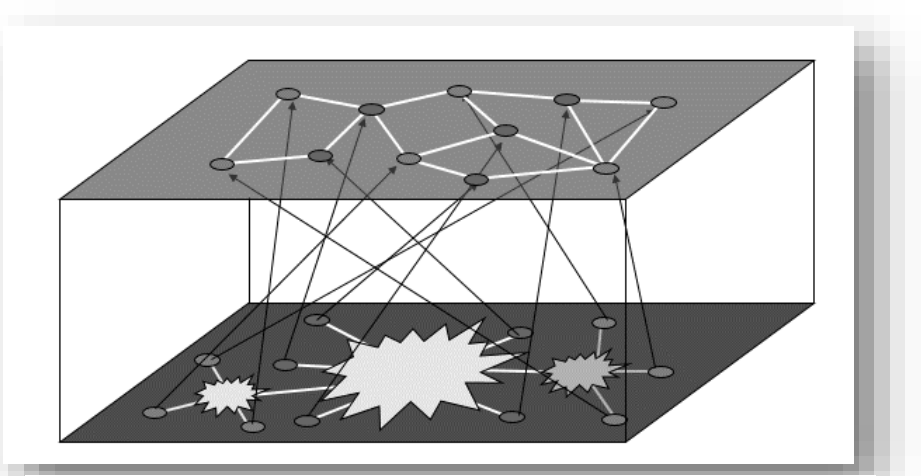


Figure 2.1 Overlay network on top of the Network layer

2.1 Traditional Network Service Models

The Internet has evolved a lot in the past decades. Emerging the Internet of Things (IoT) [11] takes the data communication to the completely new level. The demand for services in the Internet are growing exponentially. The growth of the Internet can be expected to be as large as world's population. Today there are more than a billion websites. In 2016 only, there were 3,434,971,237 Internet users [12]. This number has increased by 7.5 % since 2015. It is interesting to know that the world population's change rate was 1.13% in the same period. Figure 2.2 shows the global Internet users per year since 1993. Traditional services in the Internet are based on the concept of central repository of information. Today, traditional network such as Client-Server or Content-Delivery based networks are not capable to provide the daily or even hourly necessities of the end-users around the world.

In the following sections, we discuss the legacy approaches of data communication and provide reasons to support the need of more efficient data communication systems.

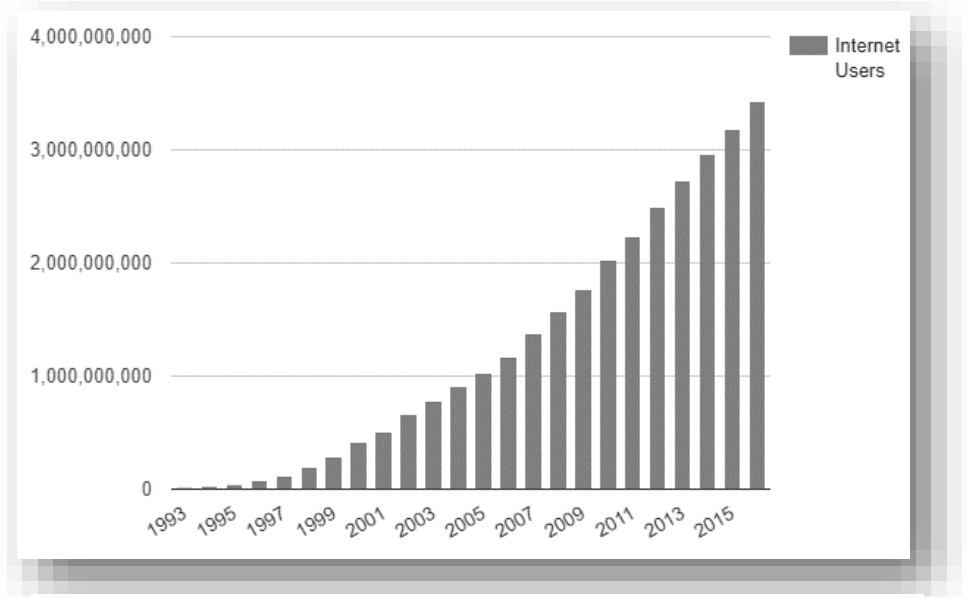


Figure 2.2 Internet Users in the Word per Year [12]

2.1.1 Client-Server Network

Client-Server architecture [9] model has been a fundamental way to implement distributed systems and consequently has been heavily used for web traffic. In this model, a server provides all the resources needed to satisfy clients' requests [5]. However, this approach cannot deliver all the needs and requirements for today's web traffic. More specifically, client-server systems impose some limitations in scalability, reliability and efficiency of distributed systems.

- Scalability is the major necessity to response to the continuous growth of the Internet. The continuous demands for resources such as bandwidth, processing power and storage capacity make it impossible for client-server technology to be the only architecture that being utilized in Internet backbone.
- Reliability is a concern that needs to be addressed in server-client architecture. If the number of servers were small then what would occur when they crash, fail, get

disconnected, or being mismanaged by humans? This type of distributed systems by nature are vulnerable to the single point of failure issue.

- Efficiency in performance is challenging in server-clients practices. Responding to the users' requests that are distributed across the globe in a timely and accurately manner is not as efficient as it should be.

2.1.2 Content Delivery Networks

A Content Delivery Network (CDN) is a geographically distributed network of proxy servers and their data centers. In this model, the service provider places many copies of their content (e.g. web pages) at set of nodes at different locations and directs the clients to use a nearby node as the server. Several traditional web based services such as Domain Name Servers (DNS) and Netnews [13] adopted a CDN based architecture in early days.

The CDN goal is to distribute service spatially relative to end-users to provide high availability and high performance. CDN is mostly used to decrease the time to access to a website content and offers a real time performance for video streaming. In CDN, the traffic comes from the servers to their end-users (Figure 2.3). This means all traffic are loaded through servers. Eventually content delivery model is centralized. Therefore, CDN model is also vulnerable to the limitation that client-server model has.

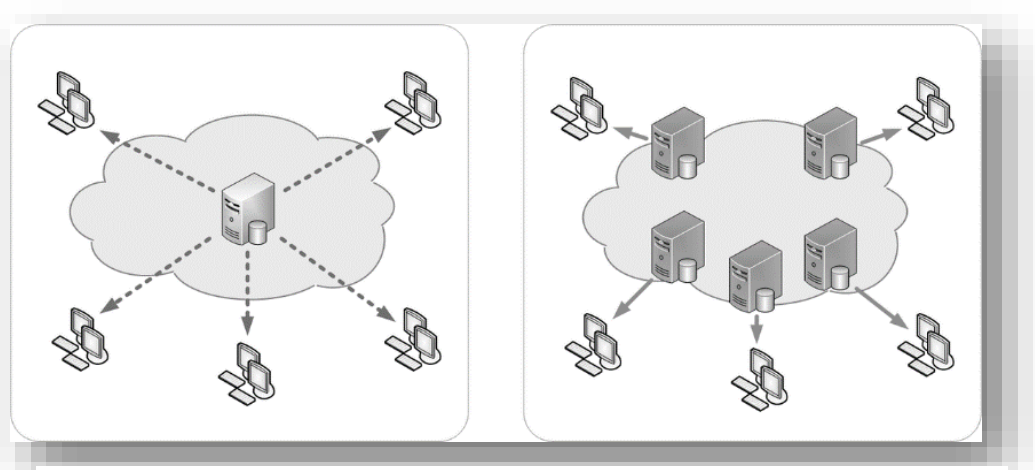


Figure 2.3 Client-server model vs. Content-based model

2.2 Peer-to-Peer Networking

Peer-to-peer systems have been defined in many papers. Here are two definitions that cover the concepts of resource sharing, self-organization, decentralization, and interconnection:

> *"A distributed network architecture may be called a peer-to-peer network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers). These shared resources are necessary to provide the Service and content offered by the network (e.g. file sharing or shared workspaces for collaboration). They are accessible by other peers*."[14]

> *"Peer-to-peer systems are distributed systems consisting of interconnected nodes that are able to self-organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity*

and performance, without requiring the intermediation or support of a global centralized server or authority."[15]

The cooperative model of P2P architecture and the fact that their users (peers) bring their own resource to share, provides a set of remarkable benefits. These types of overlay systems are highly scalable due to the idea that the capacity of their resources such as storage, processors and bandwidth escalates proportionally to the number of users. The load of the network spreads across the peers; consequently, the probability of all nodes are being crashed, would be unprecedented if not impossible. Furthermore, the distribution of the peers delivers a better efficiency. A resource can be located in a nearby peer and this advantage can save a lot of bandwidth and time consumption. Moreover, all the peers in a P2P system are powered with equal abilities, accountabilities, and functionalities, despite their different possessions. In most cases, there is no central system to oversees and controls the peers 'operations. In addition, P2P systems shall provide incentives for fair resource contribution for each of their users. Anonymity support to some extent is another desire of the participants' peers. Recently, some literatures are also referred to P2P systems as BYOD (Bring Your Own Devices) [16].

P2P is a green technology [17]. The idle computers are sharing resources instead of powerful servers in data centers. As a result, electricity is being saved which mean less Carbon emission produced in the environment. Although, this is a valuable characteristic of P2P overlays, does not receive enough credits.

In addition, in structured P2P overlay (which will be discussed later), peers are identified by a unique address. This ID is being used for routing purposes in P2P overlays. All the methods to generate and assign an ID such as Distributed Hash Tables (DHT) and LDE-based (which we

have been presented in this study) are able to provide a very large ID values. It is known that IPv4 is limited to 2^{32} addresses and IPv6 has its own challenges [18]. Hence, P2P routing overlay is able to provide an additional method to decrease the routing load on the Network layer.

2.3 Challenges of P2P Overlay Network

P2P overlay networks encounter the following challenges:

- **Churn:** The main challenge in every P2P overlay is the unpredictability of peers. Dealing with the constant arrival and departure of the nodes is the critical part in every P2P systems. Unlike servers that they are always online, peer uptime is solely based on the behavior of the user that is a member of the system [19].
- **Bootstrap:** How to join a P2P overlay? Which peer is going to provide initial configuration to a newly joining node?
- **Communication:** How to find other peers to communicate? On the other hand, how to find the resource that we are looking for?
- **Security:** Vulnerable to Sybil attack [20]. A security method is required to be implemented in P2P communications.
- **Network reachability:** Most of the computers are connected to the Internet via Network Address Translation (NAT). NAT prevents unknown incoming traffic**. S**ometimes, Internet Service Providers (ISP) block the P2P traffics.
- **Lookup latency:** Lookup cost can be very high in some P2P applications.
- **Application download:** In the world today, people are used to downloading apps from a center such as Apple or Google store, downloading and installing the P2P application can be a little challenging for them.

2.4 Classes of P2P Overlay Networks

A P2P network is a logical overlay network on top of a physical network [21]. Each peer corresponds to a node in the peer-to-peer network and resides in a node (host) in the physical network. All peers are of equal roles. The links between peers are logical links, each of which corresponds to a physical path in the geographical network. The physical path is determined by a routing algorithm and composed of one or more geographical links. Logical links can be added to the P2P network arbitrarily as long as a corresponding physical path can be found, that is, the physical network is connected. There are two fundamentally different types of P2P networks: **unstructured** and **structured** ones.

An unstructured P2P system [22] is composed of peers joining the network with some loose rules, without any prior knowledge of the topology. In their early version, the network uses controlled flooding as the mechanism to send queries across the overlay. When a peer receives the flood query, it sends a list of all contents matching the query to the originating peer. While flooding based techniques are effective for locating highly replicated items, they are poorly suited for locating rare items. Clearly, this approach is not scalable as the load on each peer grows linearly with the total number of queries and the system size. Thus, unstructured P2P networks face one basic problem, peers rapidly become overloaded, and therefore the system does not scale when handling a high rate of queries and sudden increases in system size. Gnutella [23] and Yappers [25] are two examples of unstructured P2P architectures.

In newer versions of unstructured P2P, a few other methods have been introduced to reduce the impact of flooding [26].

- **Expanded ring search**: In this method, the querying node issues a series of searches initially with small number of hops, if no reply is received, then increases the hop limits.
- **Random Walk**: In random walk, the query propagates randomly though out the network.
- **Gossiping**: In this approach, a node issues a lookup query to a neighbor who once it received a packet from it. Neighbor is also sending the request out to other neighbors in a similar manner. This method is comparable to spreading a virus in a community. Sometimes this approaches called epidemic protocol as well.

To join an unstructured system, a new peer initially connects to one of several known hosts that are usually available. Unstructured network handle effectively the problem of churn. As it has been stated before, churn, peer joining and leaving the system, is frequent. However, resource lookup-time complexity in flat unstructured P2P network is *O* (n), *n* being the number of nodes in the P2P network. On other hand, a properly designed structured architecture provide efficient, flexible, and robust services [28], [30].

In structure overlay networks, peers organized into specific topologies. Typically, they utilize a DHTs. Distributed hash table is a decentralized system of hash tables. DHTs are utilized to map resources to an identifier. As a result, it provides functionalities such as data lookup, insertion, deletion, etc. to the system.

By taking advantage of DHTs, a better complexity of *O* (log n) is achievable, in contrast with unstructured networks. However, maintaining DHTs is a complex task and needs huge amount of effort to handle the problem of churn. Hence, the major challenge facing such architectures is to reduce the amount of effort for churn handling while still providing an efficient data query service.

Chord [31] [32], Pastry [33] [34], Tapestry [11] are structured P2P. We will discuss further about each of these architectures.

Additionally, a few literature have considered server-based P2P systems as one category of P2P architecture. They classified P2P systems into two main categories of centralized and decentralized, based on the existent of a server [35]. Centralized P2P systems are a hybrid of client-server and P2P models. Usually, they have one or more servers to coordinates their peer resources. To find a specific resource, a message is sent to the system server by a requesting peer. The server replies by sending the address of the resource holder. Centralized P2P applications are vulnerable to single point of failure and they have limited scalability. Napster is the most famous example of centralized P2Ps. Decentralized P2P systems are divided into structured and unstructured classes. Furthermore, structured P2P can categorized to classes of Flat, Hierarchical, and Hybrid (Figure 2.4).

In Flat or single tier P2P architecture, all nodes are only in one overlay and all functions like routing are performed in that single overlay. It turns out that most of the overlays such as Chord, Pastry, and Kademlia [37] are flat.

Hierarchical architecture (Figure 2.5) is a P2P overlay consists of more than one structured overlay. They have different routing mechanisms that are built into their different layers. Generally, routing in one-layer leads to gateway to another layer. Nodes are grouped into different clusters. Some nodes are in one overlay, and some other are participating in more than one. The nodes that are members of more than one layer with some special responsibilities are called Super nodes [38]. Super nodes have a large number of neighbors. They carry out tasks such as handling data flow and connecting the layers together.

Normally, super nodes are the computers with better bandwidth and stronger processor power in comparison to the other nodes. In addition, they have a longer on-line time. One of the best P2P overlay examples for hierarchical architecture is the LDE-based hierarchical P2P that is going to be present in this dissertation.

Hybrid P2P (Figure 2.5) Overlay is an architecture consisting of both structured and unstructured P2P are embedded into different layers. HP2P [39], KaZaA [40], Gnutella 6.0 [41] are few examples of this type of architecture.

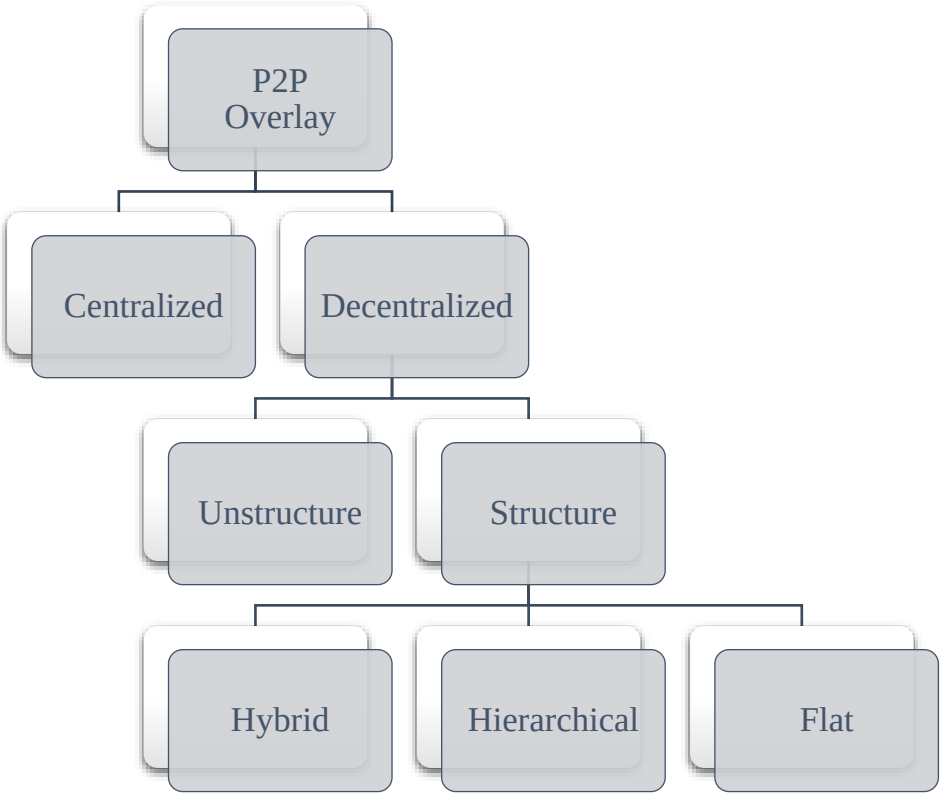
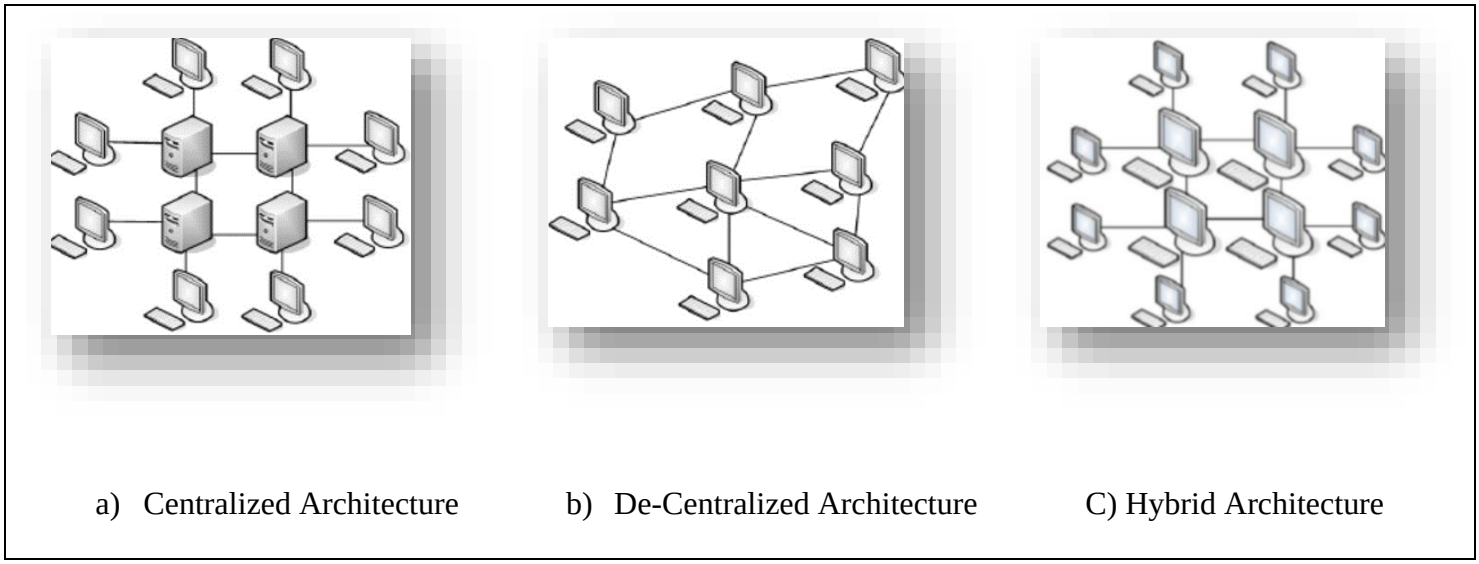


Figure 2. 4 Hierarchy of P2P Overlays



a) Centralized Architecture b) De-Centralized Architecture C) Hybrid Architecture

Figure 2.5 Architecture of P2P

To summarize, as we have discussed above, current unstructured P2P architectures are facing the traffic overload problem and their time complexity for the data lookup is related to the number of their peers, which is substantial. While structured P2P networks typically have a better search complexity but problems arise due to frequent peer joining and leaving which is known as churn. Our interest-based hierarchical P2P architecture attempts to address these issues and provide a better solution for P2P data communication.

As we are going to present our interest-based hierarchical P2P, we would like to refer to a study that have generated the exact number of the distinct resources available in thepiratebay.org website. Pirate Bay is the biggest host of the torrent files, which assists file sharing by BitTorrent protocol. This study reveals that the number of distinct resource types being used is far less than the number of peers in the Internet. In fact, it justifies the use of interest based P2P systems.

In this research [42], about 3.4 million pieces of data, over 680 thousand torrents have been processed from thepiratebay.org. The below dataset is a sample output of distinct resources available in this website. In addition, Figure 2.6 shows Torrent category statistics.

"Total Number of items: 679516; Music: 189278; Movies: 275763; Applications: 51173 Games: 47540; Other: 39005Porn: 63659; Sizes; TotalMusic:43125.1 average 227.84; TotalGames: 64948.06 average1366.177; TotalMovie: 450009.6 average 1631.871; ApplicationTotal: 17735.65 average 346.5822; PornTotal: 40811.07 average 641.0887; OtherTotal: 6865.408 average 176.0135; PERCENTAGES %%%% ;Music: 27.85483; Movies: 40.58227 ;Applications: 7.530801 ;Games: 6.996156; Other: 5.740115 ;Porn: 9.368286 ;Category Ratios ;Music: 3.010126 ;Movies: 1.438316 ;Games: 2.493047 ;Application: 4.493691 ; Other: 3.264016 ;Sizes 1_10KB: 901 %: 0.1325944 ;10_100KB: 1244 %: 0.1830715 ;100_1000KB: 15457 %: 2.274707 ;1MB: 9470 %: 1.393639 ;2_5MB: 21107 %: 3.106181 ;5_25MB: 55221 %: 8.126519 ;25_50MB: 46142 %: 6.790421 ;50_200MB: 170754 %: 25.12877 ;200_500MB: 106744 %: 15.70883 ;500_1,000MB: 124043 %: 18.25461;1_2GB: 53213 %: 7.831015 ;2_4GB: 23008 %: 3.385939 ;4_6GB: 37442 %: 5.510098; 6_10GB: 10772 %: 1.585246 ;10_20GB: 2500 %: 0.3679089 ;20_100GB: 1466 %: 0.2157418 ;100+GB: 31 %: 0.004562071 ;Ratios 1_10KB: 4.647742 ;10_100KB: 4.062142 ;100_1000KB: 3.980501 ;1MB: 4.082843 ;2_5MB: 4.151653;5_25MB: 3.757301 ;25_50MB: 2.848794 ;50_200MB: 2.813746; 200_500MB: 1.888859 ;500_1,000MB: 1.531591 ;1_2GB: 1.231281 ;2_4GB: 1.192464 ;4_6GB: 0.9310662 ;6_10GB: 0.7831408 ;10_20GB: 0.6954312 ;100+GB: 0.7111191 ;Average Number of Seeders/Leechers Ratio 1KB7.284238 …"

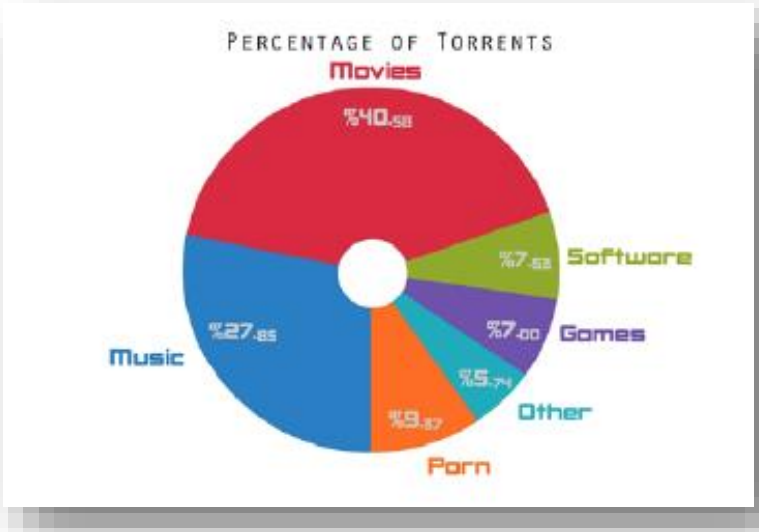


Figure 2.6 Current Torrent Categories [42]

CHAPTER 3

LITERATURE REVIEW

To start reviewing related P2P systems, we begin with a well-known unstructured peer to peer.

- **Gnutella** is the first system that adopted unstructured P2P architecture topology. To join the Gnutella system, a new peer initially connects to one of the several known hosts that are usually available (e.g., list of peers available from http://gnutella.com). For lookup, user forms a query including the search string and floods it out to its neighbors. Recipient peers compare the string with their own resources, if they find a match, query response messages are sent back to the sender containing information on how to download the resource. The peer that requested the file downloads it directly. The updated version of Gnutella 0.6 is not a flat unstructured P2P anymore [43]; it is a hierarchical network made of leaf nodes and super nodes. Typically, three leaf nodes are connected to a super node, and each super node is connected to more than 32 other super nodes. In Gnutella 0.6, the maximum number of hops a query can travel has been lowered to 4. Like all of the unstructured P2P systems, Gnutella's lookup complexity is proportional to the number of available peers, which can be very expensive in a relatively big P2P network.

- **Freenet** [44], [45] is an adaptive P2P that was designed to resist censorship. Freenet is a loosely structured decentralized P2P network that provides anonymity for peers. The nodes' identifier keys are location independent; the system has no central server and administrator. Information stored on Freenet is encrypted first, then distributed around the network and stored on several different nodes. Peers are not aware who provides the data

during the lookup, even have no idea what they stored themselves as the data is encrypted. This keeps the anonymity of their members, and hides the content that any peers hold. Peers only have information of their neighbors. When a controlled lookup query is issued by a node, number of query hops is decremented at each peer to prevent infinite loops. Peers are also able to reject the request that they have received before, due to the use of a unique random identifier for each request. In this case, the preceding peer forwards the request to another peer and the search continues. To connect to the system, a peer is required to know another existing user address. In Freenet, there is no peer with special responsibility, and therefore, no hierarchy exist. As a result, there is no centralized point of failure. Freenet should find the requested file, approximately by

O [log (n)]^2 hops, n in the number of the peers in the system. Nevertheless, the system does not guarantee that data will be found at all.

- **BitTorrent** [46] is designed for fast and efficient content distribution. Its architecture is ideal for large files delivery. BitTorrent is not considered as a pure P2P system due to its centralized server. This server is called Torrent tracker. The tracker keeps the records of all peers who possess the complete or some portion of a file. The system works as follows: a peer known as seeder possesses a file that wants to share it in the BitTorrent system. First, the peer needs to generate a torrent file. The torrent file contains the information about the file, its length, name, identification information, and URL of a tracker (tracker metadata). The torrent file shall be placed on some torrent websites such as Pirate Bay. In order to download a file, a user logs in to the website first and downloads the torrent file. The torrent file is just a metadata that informs the user on how to access the seeking content. By having the metadata, the user is able to learn on how to connect to the tracker server. The tracker

server associates the requester to those who possess the content. The requester (which called leecher) is able to obtain the content from multiple seeders instead of just downloading it from one peer. Increasing the number of leechers is equivalent to more replicas of the content and ends up with faster download. Note that, leechers are able to take some load off the seeders, by sharing the pieces of file that they have just downloaded. In addition, BitTorrent implements a technique for enticing peers to contribute. In this method peer replies with the same action that other collaborating peers previously performed (tit for tat). BitTorrent breaks files into pieces of 64 KB – 1 MB per piece and the torrent file also contains the hash of each file that can be used to perform an integrity check on a downloaded piece. Regarding the routing performance, BitTorrent guarantees to locate data and provides a constant routing state.

- **Content Addressable Network (CAN)** [48] developed at University of California, Berkeley is a distributed decentralized P2P overlay. The architectural design is a virtual multi-dimensional hyperspace with multiple zones (Figure 3.1). The entire space is partitioned among all the nodes and every node possesses a zone in the space. Data is stored at the enclosing zone and a key identifies this data. Each node only maintains state for its immediate

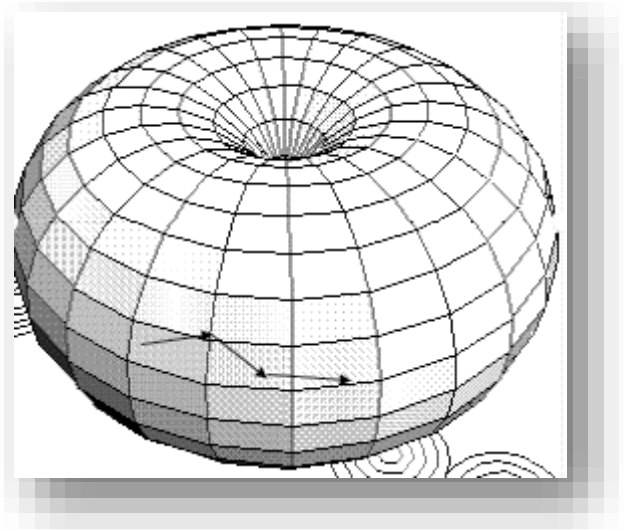


Figure 3.1 CAN Architecture. d-dimensional hyperspace with N zones

neighboring nodes. This state consists of the IP address and the virtual coordinate zone of each of its neighbors. A CAN message includes the destination coordinates. The routing algorithm chooses the nearest neighbor to the destination. These factors make CAN a scalable, fault-tolerant, and self-organizing P2P system. CAN routing performance is $O(d \times N^{1/d})$; N being the number of peers in network and d is the number of dimensions.

- **Chord** developed by Berkeley and MIT researchers uses consistent hashing [49] to assign keys to its peers. In Chord, in order to generate the consistent hashes, IP addresses and port numbers are used as the input of SHA-1 [50] hash function. The calculated message digest is always 160 bits. These hash values are truncated to m bits, where m is a system parameter. This peer ID is an integer between zero and $(2^m - 1)$ bits. The peer IDs then map to one of 2^m logical points on a ring (Figure 3.2). Each node maintains a pointer to its successor and predecessor nodes. Data lookup queries are sent and received through the successor and predecessor nodes. Consistent hashing is designed to let peers enter and leave the network with minimal interruptions. Essentially, Chord uses a technique where each of the nodes in the P2P overlay selects its neighbors in an intelligent fashion. In a

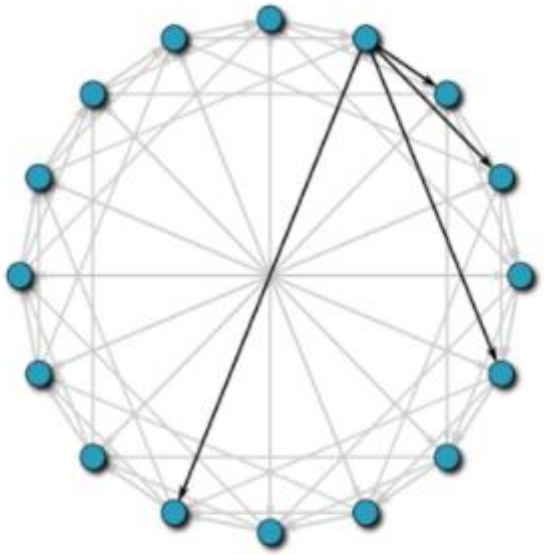


Figure 3.2 A 16-node Chord network

steady state, for a total of N peers in the system, each peer maintains a routing state information for about *O* (log N).

- **Kademlia** [37] is a P2P decentralized structured overlay network. Similar to most of the structured P2P systems, Kademlia assigns a NodeID with the size of 160 bit to each peer. The NodeID and {key, value} pairs are deposited on peers with IDs close to the key. The routing algorithm which is based on the NodeID uses to find peers near a destination key. Kademlia uses the unique idea of exclusive or (XOR) to calculate distance between two nodes. The XOR of the two NodeIDs produces the distance between them. It uses a single routing algorithm to locate peers near a particular ID. The Kademlia routing protocol consists of the following steps:
    - PING: inquiries a peer to verify if the node is active.
    - STORE: store a {key, value} pair for later retrieval.

- FIND_NODE: takes a Node ID, and returns its associated {IP address, UDP port, NodeID} triple.
- FIND_VALUE: is similar to FIND_NODE, send back {IP address, UDP port, NodeID} triples, if the recipient of the request possesses the requested key in its store

Kademlia has a routing performance of *O* (log BN) +c, where c = small constant

N-number of peers in network and b-number of bits ($B = 2^b$) of NodeID

- **Pastry** [34] came out of academia as well. Pastry, just like Chord, assigns Ids to nodes, using a consistent hashing function. The peers are hashed onto a point on a logical circle. In Pastry, Leaf Set is defined as each node including with its successor(s) and predecessor(s). Routing tables are based on prefix matching. The niche marketing of Pastry is emphasizing on locality. It is considering the nodes in underlying network topology and trying to make the neighbor edges to be short. Pastry has two metrics, Id distance and physical distance. A peer with an Id of B bits may have up to B neighbors, one for each of the prefix matches. A node may have many neighbors with matching prefix, however it chooses the one with the shortest round-trip-time. Initially, shorter prefixes are going to be assigned to the peers first. As a result, chances that peers with shorter prefixes are being physically close to each other are very likely. Because of prefix routing method, first a few hops are physically shorter and later ones are longer to reach. The routing performance of Pastry is of *O* (log ($_B$N); *N* is the number of peers and *B* is the number of bits used for the base of the chosen identifier.

- **KaZaA** [40] is a hybrid P2P that supports meta-data searching. It is the cross of Napster and Gnutella. Peers with high bandwidth, stronger processors are selected to act as the

super nodes and impose a hierarchy in overlay. Super nodes are performing as a central-server to a group of peers. Compare to unstructured P2P, search is more efficient in KaZaA. To perform a query, a peer just sends a lookup message to the super node instead of flooding the system with the query. If multiple peers are hosting a queried file, then a user can perform parallel downloading. Usually a KaZaA's super node supports 30-50 ordinary peers. Super nodes are forming a structured P2P on top of the underlay peers. In addition, some of the super nodes are hardcoded into the KaZaA application. To join the system, new peers need to contact the hardcoded super nodes. In order to share any resource with the network, the newly joined peer informs the super node with a list of files. The super node stores the metadata of the shared files. KaZaA offers better scaling properties than Gnutella. It provides some degree of guarantee to locate data, since queries are routed to the super nodes.

- **HP2P** [39] proposes a two-layer hybrid P2P network. The HP2P network combines both unstructured and structured P2P networks. Generally, the upper structured layer is based on Distributed Hash Tables (mostly Chord), and the lower underlay is unstructured. The super nodes are responsible to coordinate the lower layers' peers. Similar to KaZaA, all communications between layers are going through the super nodes. Considering N peers in the entire HP2P system, and the size of each cluster being m nodes, there are at least N/m clusters available in the H2P2. Therefore, the lookup complexity is $O(\log N/m + m)$.

- **Overnet/eDonkey** [51] is another hybrid two layer P2P file sharing system. The architecture of eDonkey is very similar to KaZaA. On upper layer, there are servers for maintaining the metadata of shared files and the second layer is consisting of ordinary

peers. Any request is sent to the local server first. The local sever locates the requested file based on the list of registered peers. eDonkey supports parallel downloading, detection of corrupted files, file sharing, and partial sharing. In order to link to the system, the peer needs to find out the IP address and the port of one of the servers. To share any resource, peers are required to provide the metadata of their files to the server. After registration, peers can either search by querying the metadata or request a particular file through their PeerID. Reporting the availability of the resource is guaranteed through the system. After server locates the requested file, then it informs the peer with the address of the file owner, so the peer is able to download the files directly from the specified locations.

- **YAPPERS** [25], [26] is another hybrid P2P, consisting of both structured and unstructured P2P. The ideas of immediate and extended neighborhood provide a relatively efficient data lookup complexity. In order to issue a data lookup query, the peer first checks its immediate neighbors within a specified h hop distance. If the query was unsuccessful, then the requester issues another lookup message for its extended neighborhood, which consists of peers in 2h + 1 hops of the requester. Then, these nodes will forward the request to the peers in their own immediate neighborhood, along with others. Finally, all the peers in the system will be checked. YAPPERS is also susceptible to the traffic overhead problem, due to the constant message floods through the network.
- **YANG** et al. **[**52**]** introduced another hybrid P2P system which is composed of two parts: a core transit network and many stub networks. Stub networks are attached to a node in the core transit network. The core transit network is a structured P2P overlay, which organizes peers into a ring similar to the Pastry ring. Each peer in a transit network is assigned a peer

ID, which is a positive integer. Peers are inserted to the ring in the order of their IDs. The stub networks are Gnutella-style unstructured P2P networks.

- **Thau Loo** et al. [53] proposed a hybrid P2P system, which treats rare and popular data items differently. As it has been discussed before, flooding-based lookup methods are unable to locate rare items in the network. They proposed system, attempts to solve this issue. Their hybrid P2P overlay consists of unstructured networks on the lower layer and a structured one on top. The super nodes form the structured layer. Multiple ordinary peers are connected to a super node. Data lookup is first performed through the regular flooding method. If unsuccessful, the query is sent to the connected super node. At that step, DHT are utilized in structured network for searching rare items.
- **SKYPE** [54] is a P2P VoIP service based on KaZaA network structure, an overlay P2P network consisting of ordinary and super nodes. In SKYPE an ordinary host must connect to a super node and must authenticate itself with the SKYPE login server. Node's anonymity has not been considered in its design.
- **Garces** et al. [55] have proposed a hierarchical, fully structured P2P. The architecture has been influenced by KaZaA. However, unlike KaZaA, a DHT-based P2P such as, Chord, or Pastry has been implemented into both layers. The lookup complexity in this system is the summation of the complexity of both layers. Since both layers are DHT based, the lookup complexity in the lower layer is O (log N), where N is the number of the peers in the system, and O (log M), is the upper layer complexity, where M is the number of the lower layer clusters.
- **HIERAS** [56] is a hierarchical DHT based P2P routing algorithm. In this system, multiple lower level rings underneath of a highest-level ring. Locality has been considered in this

system. The peers in lower level ring are physically closer to each other; therefore, the latency is shorter than the layer above. In HIERAS, a lookup query first performed inside the lower level rings. Higher level routing will be executed, if the lower level ring lookup is unsuccessful. The lookup complexity is O (log M) + O (log N), where N is the number of peers in the system and M is the number of the rings.

CHAPTER 4

ARCHITECTURE DESIGN

In this chapter, a structured architecture for hierarchical interest-based P2P system and the required mathematical basis supporting the architecture are discussed. The following notations along with their interpretations will be used while the architecture is defined.

We define a resource as a tuple $<R_i, V>$, where R_i denotes the type of a resource and V is the value of the resource. A resource can have many values. For example, let R_i denote the resource type 'songs' and V' denote a particular singer. Thus $<R_i, V'>$ represents songs (some or all) sung by a particular singer V'. In the our model for interest-based P2P systems, we assume that no two peers with the same resource type R_i can have the same tuple; that is, two peers with the same resource type R_i must have tuples $<R_i, V'>$ and $<R_i, V''>$ such that $V' \neq V''$, as shown in Table 4.1. Let S be the set of all peers in a P2P system. Then $S = \{P^{Ri}\}, 0 \leq i \leq r-1$. Here P^{Ri} denotes the subset consisting of all peers with the same resource type R_i and no two peers in P^{Ri} have the same value for R_i and the number of distinct resource types present in the system is r. Also for each subset P^{Ri}, P_i is the first peer among the peers in P^{Ri} to join the system.

Table 4.1:
No peers with the same resource type R_i can have the same value ($V' \neq V''$)

Peer (P)	Resource (R)	Value (V)
$\mathbf{P_i}$	R_i	V'
$\mathbf{P_i}$	R_i	V''
$\mathbf{P_j}$	R_i	V'

We now introduce the following architecture suitable for interest-based peer-to-peer system. It has been assumed that no peer can have more than one resource type. Generalization of the architecture will be considered in Chapter 8.

4.1 Two Level Hierarchy

We present a two level overlay architecture and at each level, structured networks of peers exist. It is explained in detail below.

a. At level 1, there is a ring network consisting of the peers P_i $(0 \leq r \leq d\text{-}1)$. Therefore, the number of the current peers on the ring is r, and d is the maximum number of the peers that can be present. This ring network is used for efficient data lookup and so it has been named as transit ring network.

b. At level 2, there are r numbers of completely connected networks of peers. Each such group, $G_{i,}$ is formed by the peers of the subset P^{Ri}, $(0 \leq i \leq d\text{-}1)$, such that all peers (ϵP^{Ri}) are directly connected (logically) to each other, resulting in the network diameter of 1. Each such G_i is connected to the transit ring network via the peer P_i. We name such a peer P_i as the group-head of network G_i.

c. Each node in the transit network maintains a Global Resource Table (GRT) that consists of tuples of the form, <*Resource Type, Resource Code, Group Head Logical Addres*s>, where Group Head Logical Address refers to the logical address assigned to a node by our overlay P2P architecture.

d. Any communication between a node $p_i \in G_i$ and $p_j \in G_j$ takes place only via the respective group-heads P_i and P_j.

The proposed architecture is depicted in Figure 4.1.

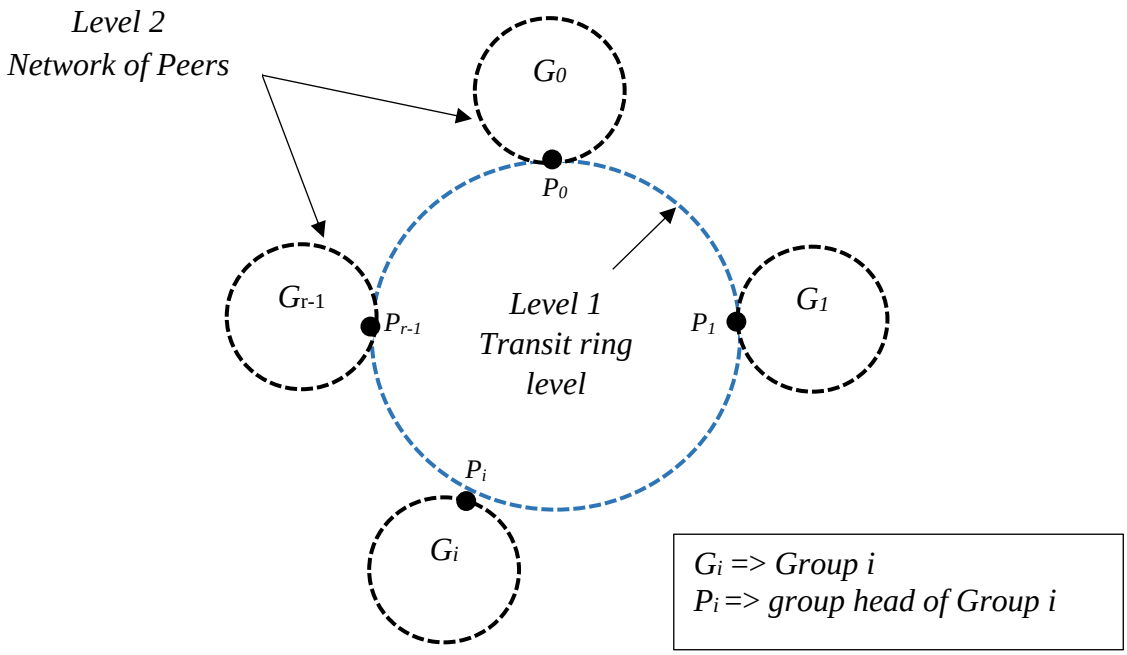


Figure 4.1. A two-level structured architecture with distinct resource types

We shall use solutions of a given Linear Diophantine Equation (LDE) to realize the architecture. The solutions are used to determine the following.

a. Logical addresses of peers in a subnet P^{Ri} (i.e. group G_i). Use of these addresses will be shown to justify that all peers in G_i are directly connected to each other (logically) forming an overlay network of diameter 1. In graph theoretic term, each G_i is a complete graph.

b. Identifying peers that are neighbors to each other on the transit ring network.

c. Codes of distinct resource types.

An overview of LDEs is provided below, which will offer the mathematical foundation of the presented architecture.

4.2 Linear Diophantine Equation (LDE) and Its Solutions

By assumption that a, b, and c are integers, the equation:

$$an + ck = b \tag{1}$$

is called a LDE in geometry [57], the objective of solving the Diophantine equation is to find all the lattice points, if there exist any. Lattice points are the integer coordinates that intersect with equation (1). The existence of the lattice points is based on the a, b, and c. In order that there exist integers n and k that satisfying the equation (1), it is necessary and sufficient that

$$d \mid b, \text{ where } d = gcd\,(a, c) \tag{2}$$

To proof, let consider

$$a=e.d,\ c=f.d. \tag{3}$$

Then we can rewrite the equation (1):

$$b = edn + fdk = d(en + fk). \tag{4}$$

On the other hand, if $d \mid b$, let $k.\ d = c$. Therefore, we can find n' and k' such that:

$$an' + ck' = b \tag{5}$$

Thus

$$a(n't) + c(k't) = bt = b \tag{6}$$

Hence $n = n't$ and $k = k't$ provide a solution for (1).

Consider for equation (1) there are w_0 and z_0 such that:

$$aw_0 + cz_0 = d \tag{7}$$

Then, an integer h can be found such that $b = dh$; and we let $n_0 = w_0 h$ and $k_0 = z_0 h$. For equation (1), (n_0, k_0) is a solution. Suppose another solution is also (n', k'), therefore:

$$an' + ck' = b = an_0 + ck_0 \tag{8}$$

Then,

$$\frac{a}{d}n' + \frac{b}{d}k' = \frac{a}{d}n_0 + \frac{b}{d}k_0 \tag{9}$$

Hence,

$$\frac{a}{d}(n' - n_0) = \frac{b}{d}(k_0 - k') \tag{10}$$

Since $gcd\left(\frac{a}{d}, \frac{b}{d}\right) = 1$, so

$$\frac{b}{d} | (n' - n_0) \tag{11}$$

Therefore, there exists an integer t such that

$$(n' - n_0) = \frac{tb}{d} \tag{12}$$

That is,

$$n' = n_0 + c\left(\frac{t}{d}\right) \tag{13}$$

It can be determined that, for each solution (n', k') of equation (1). There exists an integer t such that:

$$n = n_0 + c\left(\frac{t}{d}\right) \quad (14)$$

$$k = k_0 + a\left(\frac{t}{d}\right) \quad (15)$$

The equation (1) can also be stated as,

$$a.n \equiv b \pmod{c}, \quad a, b, \text{ and } c \text{ are integers.} \quad (16)$$

$$\text{Let } d \mid b, \text{ where } d = \gcd(a, c) \quad (17)$$

Note that if *n* satisfies this

Each solution of equation (16) (and hence of (1) as well) has also the form

$$n = n_0 + c\left(\frac{t}{d}\right) \qquad k = k_0 + a\left(\frac{t}{d}\right) \quad (18)$$

Where n_0 and k_0 constitute one specific solution and *t* is any integer.

Among the different values of n described by $n_0 + c\left(\frac{t}{d}\right)$, note that the *d* values:

$$n_0,\ n_0 + \frac{c}{d},\ n_0 + 2\left(\frac{c}{d}\right),\ \ldots,\ n_0 + (d-1)\left(\frac{c}{d}\right)$$

are all mutually incongruent modulo *c*, because the absolute difference between any two of them is less than *c*.

Also the values of *a, b,* and *c* can be chosen as to make *d* very large. Similarly, note that there are infinite other solutions which are congruent to each of the d solutions. For example, all solutions of the form:

$$(n_0 + mc),\ m \text{ is an integer},$$

are mutually congruent. Similarly, all solutions of the form:

$$n_0 + t\left(\frac{c}{d}\right) + mc$$

are mutually congruent.

LDE Examples:

a. Consider the congruence $240.n \equiv 60 \pmod{180}$, $a = 240$, $b = 60$, $c = 180$

Here, $d = gcd(240,180) = 60$, and $d \mid b$. So there exist $60\ (= d)$ mutually incongruent solutions. These are:

$n = n_0 + ct/d$ for $t = 0, 1, 2, ---, 59$ and $n_o = 1$ is a solution

All solutions of the form $n_0 + mc$ (m is any integer), i.e. in this example $(1 + m.\ 180)$ are congruent to $n_0 = 1$.

Similarly all solutions of the form $(n_0 + c/d + mc)$, i.e. $(1 + 180/60 + m.180) = (4 + m.180)$ are congruent *to* $(n_0 + c/d)$, i.e. *4*

b. Consider the LDE $1000n \equiv 250 \pmod{750}$

$d = gcd(1000, 750) = 250$ and $d \mid b$. Therefore, there are 250 mutually incongruent solutions.

Thus, it is shown that by appropriately selecting the values of the integers *a, b,* and *c* of LDE, number of mutually incongruent solutions can be made very large.

Congruence Properties:

If *a, b, c,* and *d* are any integers, we can declare that:

$$a \equiv a \pmod{c}; \tag{19}$$

$$\text{If } a \equiv b \pmod{c}, \text{ then } b \equiv a \pmod{c}; \text{ and} \tag{20}$$

$$\text{If } a \equiv b \pmod{c}, \text{ and } b \equiv d \pmod{c}; \text{ then } a \equiv d \pmod{c} \tag{21}$$

Above statements (19), (20), and (21) are the reflexive, symmetric, and the transitive properties of congruence's respectively [57].

4.3 Implementation of the Architecture

By assumption, that in an interest-based P2P system there are r distinct resource types $(r \leq d)$. That is, a maximum of d resource types can be present. Note that this is not a restriction, because d can be set to an extremely large value by choosing an appropriate LDE. The set of all peers in the system can be given as

$$S = \{P^{Ri}\}, 0 \leq i \leq r\text{-}1. \tag{22}$$

Also as mentioned earlier, for each subset P^{Ri} (i.e. group G_i) peer P_i is the first peer with resource type R_i to join the system. Now the mutually incongruent solutions of a given LDE is used to solutions of a given LDE to define the architecture as follows.

The transit ring network (Figure.4.1) at level 1 will consist of all such P_i's, for $0 \leq i \leq r-1$, and $r \leq d$, such that:

i) Each P_i will be assigned the logical address $(n_0 + i\left(\frac{c}{d}\right))$. Note that $(n_0 + i\left(\frac{c}{d}\right))$ is the i^{th} mutually incongruent solution where $0 \leq i \leq d-1$.

ii) Two peers in the ring network are neighbors if their assigned addresses differ by $(\frac{c}{d})$, with the exception that the first peer P_0 and the last peer P_{l-1} will be considered as neighbors even though their addresses differ by $(r-1)\frac{c}{d}$. Such an exception is required for forming the ring. This exception makes the joining of new peers having new resource types very simple.

iii) Resource type R_i possessed by peers in G_i is assigned the code $(n_0 + i\left(\frac{c}{d}\right))$, which is also the logical address of the group-head P_i of group G_i.

iv) Diameter of the ring network can be at most $\frac{d}{2}$

At level 2, all peers having the same resource type R_i will form the group G_i (i.e. the subset P^{Ri}). Only the group-head P_i is connected to the transit ring network. Observe that any communication between any two groups G_i and Gj takes place via the respective group-heads P_i and P_j. Peers in G_i will be assigned with the addresses

$$[n_0 + i\left(\frac{c}{d}\right) + mc] \text{ , for } m = 0, 1, 2 \ldots \qquad (23)$$

Where $m = 0$ corresponds to the address of group-head P_i of G_i.

It is observed from equation (23) that all addresses in G_i are, in fact, mutually congruent solutions for a given i. In addition, "congruence relation" is reflexive, symmetric, and transitive. Therefore, it can be concluded that all peers in a group G_i are directly connected (logically) to each other forming a network of diameter 1 only.

CHAPTER 5

DATA LOOKUP

In this section, it is proved that how the properties of LDE-based P2P architecture will be very helpful for efficient resource queries - both for intra-group as well as inter-group resource lookups.

5.1 Intra-Group Data Lookup

Without any loss of generality, let us consider data lookup in group G_i by a peer p_a possessing $< R_i, V_a >$ and requesting for resource $< R_i, V_b >$. The algorithm for intragroup data lookup is presented in algorithm Intra-Group-Lookup (Figure 5. 1 Algorithm1).

```
1 node pa (ϵ Gi) broadcasts in Gi for < Ri, Vb >
  // one-hop communication since Gi is a complete graph
2 if pb with < Ri, Vb > then
3       node pb unicasts < Ri, Vb > to node pa
4 else
5       search for <Ri, Vb > fails
6 end
```

Figure 5.1 Algorithm 1: Intra-Group-Lookup

5.2 Inter-Group Data Lookup

In introduced architecture, any communication between a node $p_i \in G_i$ and $p_j \in G_j$ takes place only via the respective group-heads P_i and P_j. Without any loss of generality let a peer $p_i \in G_i$ request for a resource $< R_j, V^* >$; where R_j denote a resource type and V^* denotes a value. The following steps are executed to answer the query:

Peer p_i knows that $R_j \notin G_i$. .Assume that there are r distinct resource types and $r \leq d$. Then,

in order to locate resource R_j, a search along the transit ring network is required. We call this method as algorithm Inter-Group-Lookup (Figure 5.2 Algorithm 2).

1 Node p_i (ϵ G_i) unicasts request for < R_j, V*> to group-head P_i

2 P_i determines resource <R_j, V^*> group-head P_j's address code from Global Resource Table (GRT)

// address code of P_j = resource code of R_j= n_0 + j (c/d)

3 P_i computes $h \leftarrow$ | (n_0 + i (c/d)) – (n_0 + j (c/d)) |

// looking for minimum no. of hops along the transit ring

4 **if** $h > r/2$ **then**

5 P_i forwards the request along with the IP address of p_i to its predecessor P_{i-1}

6 **else**

7 P_i forwards the request along with the IP address of p_i to its successor P_{i+1}

8 **end**

9 Each intermediate group-head P_k forwards the request until the request arrives at P_j

10 **if** P_j possesses < R_j, V*> **then**

11 P_j unicasts < R_j, V*> to p_i

12 **else**

13 Pj broadcasts the request for<R_j, V*> in group G_j

14 **if** P_j possesses <R_j, V*> **then**

15 P_j unicasts <R_j, V*> to p_i

16 **else**

17 P_j unicasts search failed to p_i

18 **end**

19 **end**

Figure 5.2 Algorithm 2: Inter-Group-Lookup

5.3 Data Lookup Complexity

In Chord and other DHT based structured P2P networks, search along the chord is not followed, because it would be very inefficient in a large peer to peer system as the maximum number of hops required per search will be *n/2*, where n is the number of peers in the system. In the introduced architecture, the use of LDE and the same logical address to denote a resource type R_i and the corresponding group-head P_i has not only made the search process simple and efficient, it has also made it feasible for every group-head P_i to maintain the address of every other group-head in the transit network. This has two significant advantages:

- The maximum number of hops required per any resource search is *r/2*, where *r* is the number of distinct resource types
- As an alternative resource lookup process, a group head P_i can directly unicast a message to any other P_j, without having to route through the other group-heads in the transit network. This would allow our resource lookup process to work with a constant number of message exchanges.

Thus, the time complexity for data lookup in presented architecture is bounded by $\left(1+\frac{r}{2}\right)$, *r* being the number of distinct resource types. It has been observed in most P2P networks that the number of peers is much larger than the number of distinct resource types. Thus, the search along transit ring network is very efficient as it is independent of the number of peers *n* in the P2P system. In contrast, for Chord and other structured P2P systems the complexity involved in data lookup is a function of the number of nodes (peers) *n* in the system. In the following table, the complexity of the introduced data lookup approach along with those of some other noteworthy structured approaches is presented.

Table 5.1 Data Lookup Complexity Comparison

	CAN	**Chord**	**Pastry**	**Our Work**
Architecture	DHT-based	DHT-based	DHT-based	LDE-based
Lookup Protocol	{Key, value} pairs to map a point P in the coordinate space using uniform hash function.	Matching key and NodeID.	Matching key and prefix in NodeID.	Inter-Group: Routing through Group-heads Intra-group: Complete Graph
Parameters	*N*-number of peers in network *d*-number of dimensions.	*N*-number of peers in network.	N-number of peers in network *b*-number of bits $(B = 2^b)$ used for the base of the chosen identifier.	*r* - Number of distinct resource types. N-number of peers in network. $r << N$
Lookup Performance	$O(d\ N^{1/d})$	$O(\log N)$	$O(\log_B N)$	Inter-Group: $O(r)$ Intra-group: $O(1)$

CHAPTER 6

ANONYMITY AND SECURITY CONSIDERATION

Anonymity and security concerns are important aspects in the design of P2P networks. In this chapter, a new data lookup algorithm considering anonymity is presented. In addition, a secure data lookup algorithm will be present in the following section.

6.1 Anonymity Consideration

The goal is to maintain the same efficient data lookup as in the Inter-Group-Lookup (Figure 5.2, Algorithm 2), while supporting the anonymity of the resource requester and the responder. We consider the following problem: how can a data lookup scheme can assure that two peers $p_i \neq Pi \in G_i$ and $p_j \neq P_i \in G_j$, where one is the requesting peer, and the other is the responding peer can hide their identities (i.e. IP addresses) from each other?

Firstly, it should be noted that it is not possible to maintain anonymity if both peers belong to the same group since all peers are directly connected to each other and each peer maintains a list of all its neighbors' addresses in its group. However, for the case of inter-group data lookup, the IP addresses of the requester and responder can be hidden from each other. The modified version of algorithm Inter-Group-Lookup is presented as the algorithm Inter-Group-Anonymous (Figure 6.1 Algorithm 1).

1 Node p_i sends lookup request for < R_j, V^*> to its group-head P_i

// one-hop communication

2 P_i determines resource < R_j, V^*> group-head P_j 's address code from GRT

// address code of P_j = resource code of R_j = n_0 + j

3 P_i replaces the IP address of p_i with its own

// IP address of P_i will be forwarded instead of the p_i's

4 P_i sends the request packet to P_j

5 **if** P_j possesses < Rj, V^* > **then**

6 P_j sends < Rj, V^*> to P_i

7 P_i replaces the IP address of Pj with its own

8 P_i unicasts the response packet to p_i

9 **else**

10 P_j replaces P_i's IP address with its own in the request packet

11 P_j broadcasts the request for < Rj, V^*> in group G_j

// one-hop communication in G_j

12 **if** $\exists\, p_j \in G_j$ with < Rj, V^*> **then**

13 p_j unicasts < Rj, V^*> to P_j

// resource found in group G_j

14 P_j replaces the IP address of p_j with its own

15 P_j sends < Rj, V^*> to P_i

16 P_i replaces the IP address of P_j with its own

17 P_i unicasts the response packet to p_i

18 **else**

19 P_j sends search failed to P_i

20 P_i replaces the IP address of P_j with its own

21 P_i unicasts the response packet to p_i

22 **end**

23 **end**

Figure 6.1 Algorithm 1. Inter-Group-Anonymous

6.2 Security Consideration

To achieve security from the viewpoints of authentication and confidentiality, we apply symmetric cryptography [59] to the intra-group data communication and asymmetric cryptography for inter-group communication. Symmetric key technique uses the same key to the ciphering and deciphering. In symmetric cryptography, generating strong keys for the ciphers are relatively easier compared to its asymmetric counterpart. The encryption and decryption computations are faster since we use one key for both operations. In addition, in general it is more difficult to break symmetric keys compared to asymmetric keys. However, it requires a secure way to distribute the shared keys among the peers. In the introduced P2P architecture, the use of symmetric keys for intra-group communication appears to be suitable since all peers in a group form a complete graph and hence they all are one hop away from the group-head and from each other. In LDE-based P2P system, it is assumed that group-heads are trustworthy peers and they act as trusted key distributed centers [60]. In addition, when a group-head crashes or leaves, the new group-head acts as a trusted center as well.

However, for inter-group communication, we take advantage of asymmetric cryptography. In asymmetric cryptography [61], the keys are not identical. For each secure communication, there is a pair of keys for encoding and decoding interchangeably. The key in the pair that can be shared openly is called the public key. The matching key, which is kept secret, is called the private key. Both keys can be used to encrypt a message; the other key can act in reverse.

Furthermore, to be able to support the use of asymmetric cryptography, we do a minor modification of GRT. A new entry is used in the GRT to represent the public key of each group-head. Therefore, the new GRT consists of tuples of the form which was introduced in section-

4.1: <*Resource Type, Resource Code, Group Head Logical Address, and Group Head Public-Key*>. Group-head G_0 is responsible for updating the GRTs to reflect the effect of churn caused by group-heads leaving / joining the P2P system. In addition, it is assumed that in each group, its members share a unique master key each with the group-head for secure intra-group communication.

6.3 Secure Intra-Group Data Lookup

For Intra-Group data lookup, without any loss of generality, let us consider that in group G_i, peer p_a possesses $<R_i, Va>$ and requests for resource $<R_i, V_b>$. Notation K_{mn} denotes the master key shared only by a peer p_n (ϵG_n) and the corresponding group-head P_m of group G_m. Thus, p_a has the master key, K_{ia}, known only to itself and the group-head P_i. For secure intra-group data lookup the following steps are followed (Figure 6.2, Algorithm 2):

1. p_a issues an encrypted request for resource $<R_i, V_b>$ to the group-head P_i.
   // This requested message is encrypted by the shared key K_{ia} of P_i and p_a. Thus, P_i is the only one who can successfully read the message and P_i knows that it has originated at peer p_a
2. Group-head P_i decrypts the message with K_{ia}
3. Group-head P_i broadcasts in G_i for $<R_i,V_b>$
4. If peer p_b possesses $<R_i,V_b>$, it encrypts $<R_i,V_b>$ with K_{ib} and sends it to P_i
5. P_i decrypts the message with K_{ib}
6. P_i encrypts the message $<R_i,V_b>$ with K_{ia} and sends it to the requesting peer p_a
7. p_a decrypts the received message with K_{ia} and now has the resource $<R_i,V_b>$

Figure 6.2 Algorithm 2: Secure-Intra-Group- Lookup

6.4 Secure Inter-Group Data Lookup

In the architecture, as we have discussed before, any communication between two peers p_i $(\in G_i)$ and p_j $(\in G_j)$ takes place only via the respective group-heads P_i and P_j. We use the notations Pu_m and Pr_m to denote respectively the public and private keys of group-head P_m. Without any loss of generality, let a peer $p_i \in G_i$ request for a resource $<R_j,V^*>$. Peer p_i knows that $R_j \notin G_i$. Assume that there are r distinct resource types and $r \leq d$.

The following steps are executed to answer the query (Figure 6.3 Algorithm 3):

1. Peer p_i (ϵG_i) encrypts the request for $<R_j,V^*>$ with K_{ii}
2. P_i dycrypts the message with K_{ii} and finds group-head P_j's address code from GRT
   // address code of $P_j = n_0 + j\ (c/d)$
3. P_i computes $h \leftarrow\ |\ (n_0 + i\ (c/d)) - (n_0 + j\ (c/d))\ |$
   // looks for minimum no. of hops along the transit ring to reach P_j
4. **if** $h > r/2$ then
   P_i encrypts the message with Pu_j and forwards the request to its predecessor P_{i-1}
5. **else**
   P_i encrypts the message with Pu_j and forwards the request to its successor P_{i+1}
6. **end**
7. Each intermediate group-head P_k forwards the request until the request arrives at P_j
8. P_j decrypts the message with its own private key Pr_j
9. **if** P_j possesses $<R_j,V^*>$
10. P_j encrypts the message with the public key Pu_i of P_i and unicasts it to P_i
11. **else**
12. P_j broadcasts the request for $<R_j,V^*>$ in group G_j
13. **if** $\exists\ p_k\ (\epsilon G_i)$ which possesses $<R_j,V^*>$

14. p_k encrypts the request message with K_{jk}

15. P_j decrypts the message with K_{jk}

16. P_j encrypts the decrypted message with the public key Pu_i of P_i and sends it to P_i

17. P_i decrypts the message with its own private key Pr_i

18. P_i encrypts the message $<R_i, V_b>$ with K_{ii} and sends it to the requesting peer p_i

19. p_i decrypts the received message with K_{ii}

20. **else**

21. P_j unicasts 'search failed' to p_i

22. **end**

23. **End**

Figure 6.3. Algorithm 3: Secure-Inter-Group-Lookup

CHAPTER 7

METHODS TO HANDLE CHURNS

Churn is frequent arrivals and departure of the peers in the system. In this chapter, first, an efficient algorithm on new peers joining to the system is presented, and subsequently methods on handling the departure of peers from the system are discussed.

7.1 Peers Joining the System

For joining a new peer to the system, two possible situations are considered:

1. A new node possessing an existing resource type wishes to join.
2. A new node with a new resource type wishes to join.

It is assumed that any new node p wishing to join the system contacts a well-known server that sends the IP address of the group-head P_0 of the first group G_0 in the P2P transit network. In fact, this IP address is also the address of the first peer to join the system. The IP address of the server can be obtained by a DNS-like public service. New node p then sends a join request to P_0. All join requests are processed sequentially by P_0 by putting arriving requests in a queue. After a requesting new node p joins the P2P network successfully, it sends an ACK to P_0. P_0 then starts processing the next request from its queue.

In the LDE-based P2P scheme, for the case of new peers joining with existing resource type, all that is needed is every group member adds the new peer in its list of neighbors in the group. In case of peers joining with new resource type, only three group-heads P_{s-1}, P_s and P_0 need to update their neighboring pointers and the GRT of the transit network nodes is updated with only the information for the new resource R_s.

7.1.1 New Peer with Existing Resource Type

The method of joining for a new node with an existing resource type is quite simple and is described in Algorithm 1 (Figure 7.1), JoinExisting. In algorithm JoinExisting, p is a new node having an existing resource type R_k. p is assumed to have already obtained the IP address of P_0.

1 New peer p with resource type R_k unicasts its join request to P_0

2 P_0 determines the group G_k for p from its GRT

3 P_0 unicasts IP address of p to P_k

4 P_k assigns p with the next available address $[n_0 + k\left(\frac{c}{d}\right) + qc]$

5 P_k includes p in its list of neighbors in G_k

6 P_k asks all members of G_k to include p in their lists

7 P_k sends the updated list of neighbors in G_k to p

8 p establishes direct logical link to all members of G_k

Figure 7.1 Algorithm 1. JoinExisting

7.1.2 New Peer with New Resource Type

Let p be a new host, which wishes to join the overlay P2P network with a new resource type R_s. Let $S_R = R_i$, $0 \leq i \leq s < d$, be the set of the existing resource types in the system with the current last node on the transit network being P_{s-1}. It is important to note that d can be suitably set at the initial design phase by selecting an appropriate LDE in order

to accommodate all possible new resource types in the system. In LDE-based overlay P2P architecture, the location of a new peer joining with a new resource type is always predetermined unlike the existing DHT-based architectures and the rule for insertion of a new node/resource type is as follows:

> Any insertion of a peer with a new resource type always takes place between the existing last group-head P_{s-1} and the first group-head P_0 on the ring.

By virtue of the logical address assignment process the code for the resource type R_S will be $(n_0 + s \cdot c/d)$ and this code will also be the new logical address of the joining node p, also the group-head (P_s) of a new group G_s in the system. The address of P_{s-1} on the ring network differs from that of P_{s-2} and P_s by $\pm c/d$. Figure 7.2 shows how the joining of a new node with a new resource type happens while the join process is described in details in Algorithm 2, *JoinNew* (Figure 7.3)

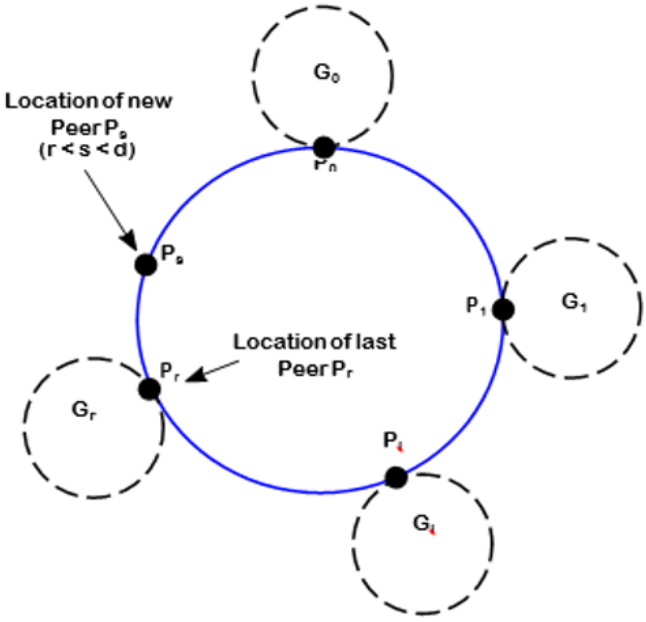


Figure 7.2 Joining of new host with new resource type

1 New peer p unicasts its join-request to P_0

2 P_0 assigns p with logical address $n_0 + s\left(\frac{c}{d}\right)$

3 p becomes the group-head P_s of a new group G_s and logical address of P_s = code (R_S)

4 P_0 unicasts P_s's IP address to P_{s-1}

5 P_0 unicasts P_{s-1}'s IP address to P_s

6 P_0 includes code of R_s in GRT

7 P_0 sends a copy of its GRT to P_s

8 P_0 asks all group-heads P_i, $0 \leq i \leq s-1$, to include code (R_s) in their GRTs

9 Do in Parallel

10 begin

11 begin

12 P_0 updates its pointers to its neighbors

//IP addresses of P_s and P_1 are the pointer values

13 P_0 saves IP addresses of P_s and P_1 in $p_0 \in G0$ with address $(n_0 + c)$

14 end

15 begin

16 P_{s-1} updates its pointers to its neighbors

// IP addresses of P_{s-2} and P_S are the pointer values

17 P_{s-1} saves IP addresses of P_{s-2} and P_S in $P_{s-1} \in G_{s-1}$ with address $n_0 + (s-1)\frac{c}{d} + c$

18 end

19 begin

20 P_S sets its pointers to $P_S{-}1$ and P_0

// IP addresses of $P_S{-}1$ and P_0 are the pointer values

21 P_S sends join-completion message to P_0

22 end

23 end

Figure 7.3 Algorithm 2: JoinNew

7.2 Fault Tolerance - Peer Crash or Leave

Following two possible situations are considered:

1. Any group member $p \in G_k$ and $p \neq P_k$ crashes or leaves.
2. Any group-head P_i crashes or leaves.

7.2.1 Group Member Crashes or Leaves

Let a node $p \in G_k$ crash or leave and let $p \neq P_k$. If p crashes, its neighbors in G_k will know about it and each group member in G_k will simply delete the entry for p from its list of neighbors. The direct connectivity among the rest of the group members remains intact, because congruence relation is symmetric as well as transitive.

7.2.2 Group-Head Crashes or Leaves

The procedure to handle the case of a group-head crashing or leaving the network can be achieved easily with a small overhead of saving pointer values present in a group-head P_i in a peer $p^* \in G_i$. An update from P_i to p_i^* is triggered whenever P_i detects a change in the transit network. In order to guard against any loss of information due to group-head P_i's crash/leave, P_i also sends a snapshot of its request queue to p^*each time the content of the queue is updated. The selection of p_i and the procedure to setup a new group-head are described in details below.

Step 1: Let the group-head $\boldsymbol{P_i}$ of group $\boldsymbol{G_i}$ crash or leave. If $\boldsymbol{P_i}$ crashes, its neighbors on the transit network - P_{i+1} and P_{i-1} as well as those in $\boldsymbol{G_i}$ learn about it via the periodic hello packet exchanges. If $\boldsymbol{P_i}$ leaves, it informs its neighbors on the ring as well as those in $\boldsymbol{G_i}$ via a broadcast, prior to leaving. In the presented scheme, we use $p^* = [(n_0 + \boldsymbol{i} \cdot c/d) + c]$ as the choice of our

replacement node for P_i in G_i. As mentioned earlier, peer p^* already has the IP addresses of P_{i-1} and P_{i+1} the neighbors of G_i on the transit ring network.

Step 2: A new successor of p^* is chosen in G_i using the same rule that was applied for choosing p^*. A new p^{**}in G_i with the logical address $[(n_0 + i.\ c/d) + 2c]$ is set as the successor of $p^* \in G_i$. The IP addresses of P_{i-1} and P_{i+1} in the transit network as well as the GRT table from p^* are copied to p^{**}.

Step 3: To make sure that the GRTs remain unchanged, the new group-head $p*$ is now designated as P_i and its logical address is changed from $[(n_0 + i \cdot c/d) + c]$ to $(n_0 + i \cdot c/d)$. This change is broadcast to all other group members of G_i only. Observe that it will not affect the direct connectivity relation among the neighbors in G_i, because congruence relation is symmetric as well as transitive. Effect wise, the news of group-head crash/leave does not propagate in the system.

The above procedure leads to the following important observation. Effect of a group-head P_i's crash/leave is restricted only to its group G_i.

CHAPTER 8

GENERALIZATION OF THE ARCHITECTURE

Thus far in the architecture, we have assumed that no peer can have more than one resource type. It may become a hard restriction in practice. Therefore, to overcome this restriction, in this chapter, we have considered Generalization of the architecture; that is, a peer can have multiple different resource types.

8.1 Peer with Multiple Existing Resource Types

To describe the situation, it is considered that in $group_i$ the group-head P_i or a peer p (ϵ G_i) wants data insertion in the system of another existing resource type R_k; note that R_k exists in $group_k$ and P_i /p already possesses R_i.

The solution works as follows. Peer P_i /p will become a member of $group_k$ as well. That is, the members of both $group_i$ and $group_k$ will know the IP address of P_i /p. Logically, it means that in the overlay network, P_i /p will be directly connected to all members of both $group_i$ and $group_k$. Algorithm 1 (Figure 8.1) states its implementation.

Time complexity of Algorithm 1 is bounded by $(1+ r/2)$, r being the number of distinct resource types. Data insertion for more existing resource types can be done similarly.

1 Data insertion request for R_k from P_i/p is forwarded along the transit ring from group-head P_i to P_k *// maximum r/2 hops*

2 P_k assigns to P_i/p the next available address, not yet assigned in $group_k$.

// the address is of the form $[(n_0 + kc/d) + yc]$, y is an integer

3a P_k broadcasts the address of P_i/p in $group_k$

// P_i/p is the new member of $group_k$; 1-hop communication

3b each $group_k$ member updates its list of neighbors

4 P_k unicasts a copy of neighbor list to P_i/p

// P_i/p is now a member of G_k

Figure 8.1 Algorithm 1 Data insertion for multiple existing resource types

8.2 Existing Peers Declaring New Resource Types

To start with, it is assumed that the P2P system has S number of distinct resource types, viz., $R_0, R_1, R_2 \ldots R_{s-1}$.

Without any loss of generality, let peer P_i/p in $group_i$ wants a data insertion for a new resource type R_s. Then following the way the transit ring is constructed, peer P_i/p will become the group-head of the newly created $group_s$ possessing resource type R_s. As the recent group-head P_s, location of P_i/p on the ring is now between P_{s-1} and P_0. Therefore, if it is P_i, peer P_i will appear (logically) twice as group-heads on the ring for $group_i$ and $group_s$. If it is peer p, it will appear once as the group-head of $group_s$ and once as a member of $group_i$. Note that R_s will have the code $(n_0 + s.c/d)$, and it will also be another logical address for P_i/p. Now, P_i/p will ask the group-heads to update

their global resource tables by including R_s and its code $(n_0 + sc/d)$ along with the IP address of P_i/p. For implementation, P_i/p will now have another set of pointers pointing to its new neighbors, P_{s-1} and P_0. Group-heads P_{s-1} now changes its right neighbor from P_0 to P_s and group-head P_0 changes its left neighbor from P_{s-1} to P_s; they adjust their pointers accordingly.

To guard against group-head crash or leave, later when more peers join this group, P_i/p will store the IP addresses of P_{s-1} and P_0 in the peer with the next address $[(n_0 + sc/d) + c]$. We will elaborate further on fault-tolerance in Section 8.5

8.3 Data Lookup Considering Generalization of the Architecture

It is considered that a peer P_i is also the group-head P_s of group$_s$. The proposed approach works as well if P_i possesses any number of distinct resource types. It is assumed that the system has r distinct resource types.

8.3.1 Intra Group Lookup

Generalization of the architecture has no effect on the Intra group lookup. Algorithm1 (Figure 5.1) is still applicable, considering the generalization of the architecture.

8.3.2 Inter Group Lookup

In the architecture, any inter group communication involves travelling along the transit ring. Without any loss of generality let a peer p_a in G_i request for a resource $< R_j, V^* >$. The following algorithm answers the query. In order to locate resource R_j, a search along the transit ring network is required. The algorithm for inter group lookup considering the generalization of the architecture is presented in Algorithm 2 (Figure 8.2).

However, in the introduced architecture, number of peers on the ring is the number of distinct resource types *r* and it has been observed that the number of peers in most P2P networks is too large compared to the number of distinct resource types. Therefore, such search on the ring in the introduced architecture appears to be quite practical.

Another point to note is that use of the same logical address to denote a resource type and the corresponding group-head has not only made the search process simple and efficient, it also has made it feasible for every group-head to maintain the address of every other group-head in the transit ring network. This has two significant advantages:

1. Following Algorithm 2, (Figure 8.2) the time complexity is bounded by *(2 + r/2)*, because maximum number of hops required per any resource search is *(2 + r/2*), where r is the number of distinct resource types. Note that *r « n*, where n is the total number of peers in the system.
2. As an alternative resource lookup process, using the GRT a group-head P_i can directly unicast a message to any other group-head P_j avoiding any communication along the transit ring network. In this way, the lookup process will need a constant number of hops and a constant number of message exchanges.

1 p_a (ϵ Gi) unicasts request for < Rj, V*> to group-head P_i

2 **if** P_i is also the group-head (P_j) for resource type R_j

3 **if** P_i (as P_j) possesses < R_j, V*> **then**

4 P_i (as P_j) unicasts < R_j, V*> to p_a

5 **else**

6 P_i (as P_j) executes Intra-Group-Lookup in G_j Algorithm 1, (Figure 5.1)

7 **else**

8 P_i determines resource <R_j, V^*> group-head P_j's address code from GRT
// address code of P_j = resource code of $R_j = n_0 + j\ (c/d)$

9 P_i computes $h \leftarrow\ |\ (n_0 + i\ (c/d)) - (n_0 + j\ (c/d))\ |$
// looking for minimum no. of hops along the transit ring

10 **if** $h > r/2$ **then**

11 P_i forwards the request along with the IP address of p_a to its predecessor P_{i-1}

12 **else**

13 P_i forwards the request along with the IP address of p_a to its successor P_{i+1}

14 **end**

15 **if** an intermediate group-head P_k is also the group-head for resource type R_j **then**

16 **if** P_k (as P_j) possesses < R_j, V*> **then**

17 P_k (as P_j) unicasts < R_j, V*> to p_a

18 **else**

19 P_k (as P_j) executes Algorithm 1, (Figure 5.1) in G_j as its group-head P_j

20 **else**

21 each intermediate group-head P_k forwards the request until the request arrives at P_j

22 **if** P_j possesses <R_j, V*> **then**

23 P_j unicasts <R_j, V*> to p_a

24 **else**

25 P_j executes Intra-Group-Lookup Algorithm 1, (Figure 5.1) in G_j

26 **end**

Figure 8.2 Algorithm 2: Inter-Group-Lookup

8.4 Joins and Leaves

It is assumed that a well-known server keeps a copy of the GRT. When a new node (peer) wishes to join the system, it contacts the server. If the request to join is for an existing resource type, say R_i, the server sends the IP address of the group-head P_i to the node. If the request is for a new resource type, the server sends the IP address of the group-head P_0. Therefore, in introduced design the server plays a small but very important role related to load sharing by group-heads. All that is needed is when the GRT is updated by the group-heads, a copy is sent to the server. By virtue of its construction, the GRT remains sorted by default and in an ascending order of the group-heads' logical addresses; so determining the exact group-head is O (log r).

The different possible situations of joining and leaving of peers is now presented.

8.4.1 Concurrent Joins

As pointed out earlier, a peer p either can join an existing group, or can form a new group with the group-head being the peer itself. In the former case, since nodes in a group are directly connected to each other, hence joining a group means forming a logical link between the peer p and each node in the group. The procedure is presented in Section 7.1. If multiple peers join the same group, say G_i, the join requests are queued at the group head P_i and are served on FCFS basis. Observe that joining multiple groups can take place concurrently, because joining one group is unrelated to joining other groups.

In case it is a new resource type R_s, the joining peer contacts P_0 that is the group-head of the very first group formed in the system. Multiple such requests eventually arrive at P_0 and P_0 serves the requests on FCFS basis. We can handle insertion of multiple new resource types by the

same peer in a similar way. Note that in our architecture joining of any new resource type always takes place between the recent and the first groups; this feature makes such joining localized to a single position on the ring; thereby making the joining process much simpler compared to existing related approaches. It is obvious that the above-mentioned two kinds of joins can take place simultaneously, because one involves existing groups and the other is about the formation of new groups.

8.4.2 Concurrent Leaves

It is assumed that any two directly connected peers in a group or along the transit ring exchange periodic hello packets. Whether it is a graceful leaving or abrupt leaving (crash), absence of a hello packet from a neighboring peer is interpreted as the peer being unreachable (not alive). That is, we do not differentiate between the above-mentioned two types of leaving. In effect, the logical link information about the leaving peer is deleted from the routing table of each peer not receiving the hello packet. Therefore, concurrent such leavings whether taking place in the same group or in multiple groups amounts to the deletion of the corresponding link information in the routing tables of the concerned non-leaving peers only. Of course, a non-leaving peer sequentially deletes multiple link information in case multiple peers leave the same group. Note that handling of single group-head crash has been discussed in Section 7.2. Multiple group heads' leaving is considered in the following section.

8.4.3 Concurrent Joins and Leaves

Observe that concurrent joins and leaves means that addition and deletion of logical links taking place concurrently. If a peer is involved in both actions, it will do so sequentially on FCFS basis; otherwise, different peers can execute these two operations concurrently in the system.

8.5 Ring Maintenance

In Section 7.2, an approach is presented to handle single group-head crash or leave. To generalize the architecture, it is considered multiple group-heads leaving simultaneously and it is shown that the ring will remain connected in such situations. The approach works as follows. Let us consider the peer P_r, the group-head of group G_r. The logical address of P_r is $(n_0 + r.c/d)$. Assume that peers p^r_1 and p^r_2 in G_r have the next two addresses followed by the group-head's address and these are $[(n_0 + rc/d) + c]$ and $[(n_0 + rc/d) + 2c]$, respectively. p^r_1 acts as the secondary group-head for group G_r to guard against the primary group-head leaving and it has considered that during the formation of this group, P_r stores in p^r_1 the addresses of its neighboring group-heads P_{r-1} and P_{r+1} along with a copy of the GRT. In the event of P_r leaving, p^r_1 becomes the new primary group-head and its communication connectivity with P_{r-1} and P_{r+1} remains intact. It also means that p^r_2 now act as the new secondary group-head for group G_r. The new primary group-head p^r_1 will save the neighbors addresses, i.e. the addresses of P_{r-1} and P_{r+1} in p^r_2 and broadcasts to other group-heads to update their GRTs to reflect that p^r_1 is now the group-head of G_r. One noteworthy point is that peer p^r_1 does not need to inform the other peers in G_r about itself being the new group-head. The reason is simple and interesting. The way of construction of the routing table of a peer as a new peer joins group G_r ensures that the routing-table remains sorted by default and in an ascending order of the peers' logical addresses in the group. Therefore, the entries are same in each routing table and each peer knows that the peer with the lowest logical address is the current group-head.

However how can the connectivity along the ring be maintained if multiple group-heads leave simultaneously? It is proposed that each group-head P_r and its secondary one, p^r_1 store the tuple, $[P_{r-1}, p^{r-1}_1, P_{r+1}, p^{r+1}_1]$. The following example explains the idea.

Let P_{i-1}, P_i, and P_{i+1} be the group-heads of three consecutive groups on the ring. We also call them as primary group-heads. The resource types corresponding to the group-heads are R_{i-1}, R_i, and R_{i+1} respectively. Let in P_i, the secondary group-head be p^i_1; similarly, the respective secondary group-heads in P_{i-1} and P_{i+1} are p^{i-1}_1 and p^{i+1}_1.

Therefore, both P_{i-1}, and p^{i-1}_1 have the tuple $[P_{i-2}, p^{i-2}_1, P_i, p^i_1]$; similarly, both P_i and p^i_1 have the tuple $[P_{i-1}, p^{i-1}_1, P_{i+1}, p^{i+1}_1]$; and both P_{i+1} and p^{i+1}_1 have the tuple $[P_i, p^i_1, P_{i+2}, p^{i+2}_1]$. Now, let us consider the worst-case scenario of all three primary group-heads, i.e. P_{i-1}, P_i, and P_{i+1} leaving at the same time. It is observed that in G_i the new primary group-head p^i_1 has the IP addresses of the new primary group-heads p^{i-1}_1 and p^{i+1}_1 of the groups G_{i-1} and G_{i+1}. Therefore, group G_i remains connected to its neighboring groups G_{i-1} and G_{i+1}. In addition, in group G_{i-1}, its new primary group-head p^{i-1}_1 uses the IP address of the group-head P_{i-2} to communicate with this group along the ring network; if P_{i-2} leaves, new primary group-head p^{i-1}_1 can communicate with this group along the ring network via the IP address of p^{i-2}_1. Similarly, it is observed that group G_{i+1} can communicate with its neighboring groups as well. Observe that to enhance the degree of fault-tolerance, tuple-size can be increased to include more members of a group. The above discussion leads to the following observation.

Transit ring network remains connected even if consecutive primary group-heads leave the system.

CHAPTER 9

PERFORMANCE EVALUATION

A new low diameter structured P2P overlay that can significantly enhance the efficiency of data communications has been presented. Our claims have been proved analytically in previous chapters. However, before any attempt to implement this new P2P architecture occurs, first it must be analyzed and evaluated. Scalability challenge of any new P2P overlay network makes it almost impossible to being analyzed on a real network environment. To test and analyze our system, we have chosen the comparative evaluation approach [61]. Comparative evaluation is one practical method to examine and judge any new P2P architecture. However, in order to be able to make use of this method, first it is essential to select a suitable simulator.

9.1 Characteristics of P2P Simulators

The principles on which P2P simulators are able be compared and contrast the P2P architectures are as follows: [62, 63, 64]:

1. **Simulator architecture:** specifies the types of P2P topologies and characteristics that simulator can perform. Besides, it indicates whether the simulator supports discrete event simulation engine, cycle based engine, or both. In addition, underlying networks and protocols are being defined in the simulator. Ability to test the architecture with churn effect is very crucial for P2P simulators.
2. **Usability:** The simulator documents must be user friendly, clearly defined and easy to understand.

3. **Scalability:** By nature, P2P overlays are designed to be a solution for data communication's scalability issues. Hence, scalability is one significant factor on which a simulator can be selected.
4. **Interactive visualizer:** A Graphical Unit Interface (GUI) to assist users to validate, debug and assist the user to obtain the results is a big advantage for any simulator.

9.2 PeerfactSim.KOM

Based on all we have discussed in Section 9.1, PeerfactSim.KOM [65] is selected to implement our comparative evaluation environment. It is a simulation framework with event-driven model and we use it to evaluate our approach. PeerfactSim.KOM is an open source, java-based simulator designed for large-scale P2P applications. An XML-based configuration file (Figure 9.2b) is used to begin the simulation that denotes the layers included in Figure 9.1. The functional layers of PeerfactSim.KOM, as shown in Figure 9.1, are as follows.

- **User Layer**: To define strategies to be performed on the application layer by user.
- **Application Layer**: Hosts P2P application. Currently, PeerfactSim.KOM provides a file-sharing application and benchmarking Workloads.
- **Service Layer:** Provide additional services such as application layer multicast or monitoring and management features to an application or to the whole system.
- **Overlay Layer:** Various structured and unstructured P2P architecture models are built into this layer. Moreover, a class hierarchy has been provided in this layer in which it gives the opportunity to developers to choose from these functionalities and develop a new overlay. In other words, an application program interface (API) is integrated into this layer that

developer can take advantage of it and create a new P2P overlay in PeerfactSim.KOM

- **Transport Layer:** Supports the transmission of both TCP and UDP messages.
- **Network Layer:** While PeerfactSim.KOM supports static and simple network models, In addition, it provides advanced models, as Global Network Positioning (GNP) [66], which is based on measurements from the PingER project.

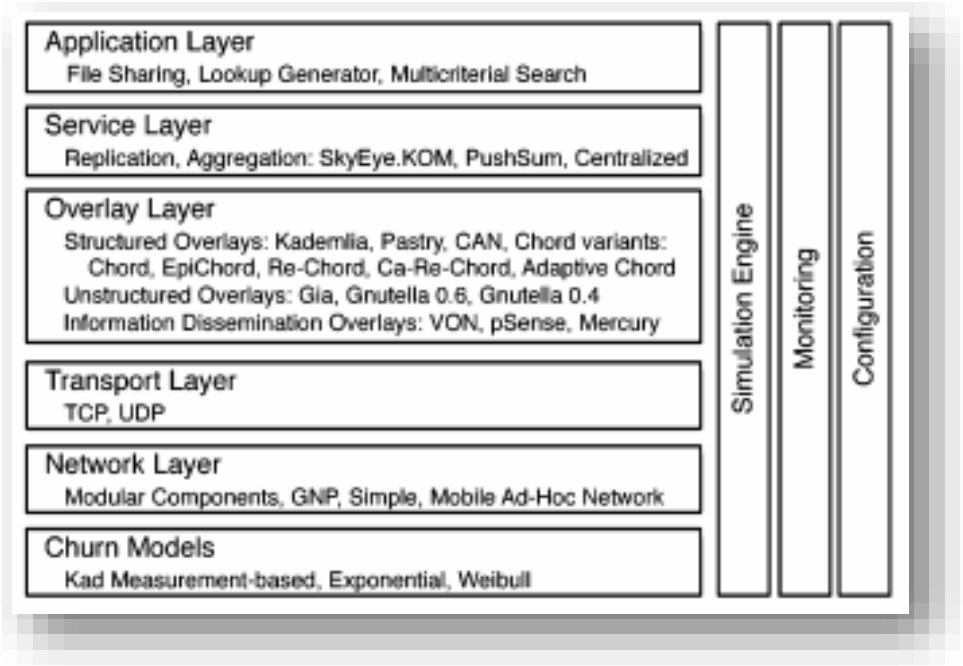
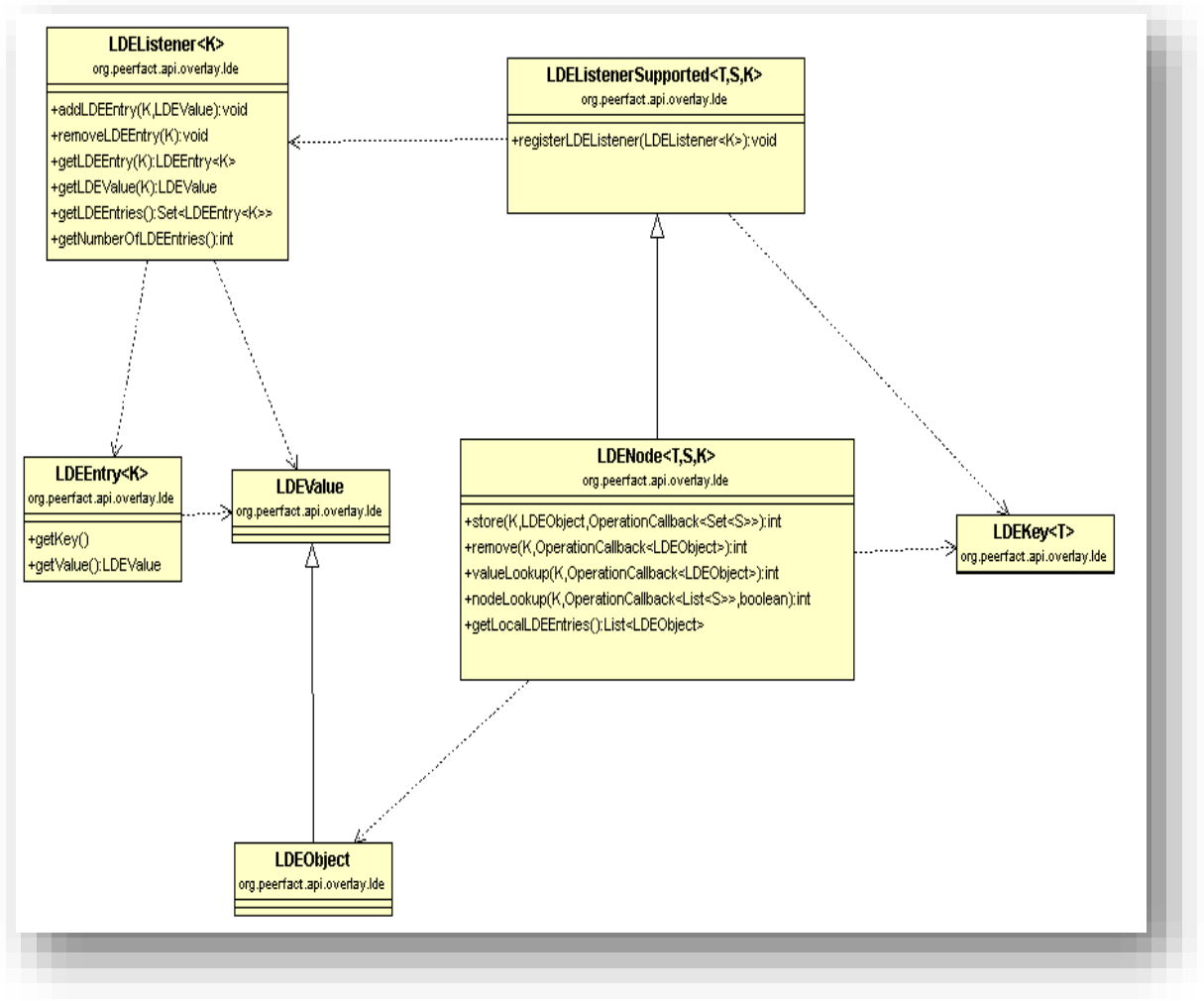


Figure 9. 1 Layard architecture of PeerfactSim.KOM [67]

9.3 Implementing LDE-Based Overlay for PeerfactSim.KOM

PeerfactSim.Kom simulation has provided an API between the layers. Therefore, users are able to configure their desired elements and still can use all the functions and utilities of other layers. For our simulation, we implement our work based on the provided API. In our implementation, operations such as store, remove, lookups are executing in *LDENodeinterface* (Figure 9.2a). This interface extends *LDEListenerSupported* interface. *LDEListenerSupported*

interface registers a LDE listener .*LDEObject* interface, which extends *LDEValue*, represents an object, which can be associated to a LDE assigned node. *LDEValue* interface is providing a unified LDE service. On *LDEEntry*, we assign the LDE key to the object that contains a value, according to the node based on the type of the resource. *LDEListener* interface is responsible to add a new peer to associated group, get the value of the node and keep track of number of entries in a group.



.*Figure 9. 2a. LDE-based Overlay Implementation UML.*

```
<?xml version='1.0' encoding='utf-8'?>
<Configuration>

    <!-- In the default section define variables to be used throughout your config l
    <Default>
        <Variable name="seed" value="0" />
        <Variable name="finishTime" value="60m" />
        <Variable name="actions" value="config/education/LDE-actions.dat" />
        <Variable name="churn" value="true" />
        <Variable name="gnpDataFile" value="config/data/mod_measured_data.xml"/>
    </Default>

    <Description>
        check implementation of LDE-Based
    </Description>


    <!-- Load the simulator and provide it with the seed and the duration. -->
    <SimulatorCore class="org.peerfact.impl.simengine.Simulator"
        static="getInstance" seed="$seed" finishAt="$finishTime">
    </SimulatorCore>


    <!-- Configure the NetLayer (in this case the ModularNetLayer). For a list of av
    <NetLayer
        class="org.peerfact.impl.network.modular.ModularNetLayerFactory"
        downBandwidth="122880" upBandwidth="32768" useRegionGroups="false"
        useInOrderDelivery="false" preset="Fundamental">
        <!-- Loads a XML-File with measurement-data for latency etc. -->
        <MeasurementDB
            class="org.peerfact.impl.network.modular.db.NetMeasurementDB"
            file="$gnpDataFile" />
        <PacketSizing
            class="org.peerfact.impl.network.modular.st.packetSizing.IPv4Header" />
        <Fragmenting
```

Figure 9.2b. A part of XML configuration file.

9.4 Experimental Environment

The core objective of the evaluation of LDE-based structured P2P is to demonstrate its highly efficient data lookup complexity. Better lookup complexities means less communication hops in an overlay. In our assessment, we have mainly focused on measuring the average of hop count in various scenarios. We compute the results in different set-ups and compare them with the performance of two of the most well established P2P networks, viz., Chord and Pastry. To begin, we setup a simple scenario. In our first environment, we consider a stable network (i.e. with few random churn). With the use of simulation, we compute the average of hops in our topology and compare it with the results of Chord and Pastry in the same environment. We repeat the test by

changing the number of clusters in LDE-based P2P system.

In our second simulation environment, the churn model is based on KAD measurements [68]. In PeerfactSim.KOM, each simulation consists of three phases. In the first phase, peers are joining the system in a uniformly distributed manner in 120 minutes. Publishing phase of 60 minutes is the second phase and lastly lookup phase takes 120 minutes.

9.4.1 Results in Stable Network

The metrics that we have used for the evaluation are the hop count, operation duration, and the number of clusters in LED-based overlay. The operation duration for all the results have been set to 300 minutes. We compute the average number of hops in very small networks (100 and 200 peers), small-to-medium sized systems (2000 peers), and lastly for medium-to-large systems (10000 peers). For midsize and large LDE-based networks, we run the simulation three times using 5, 10, and 15 clusters respectively. Figure 9. 3 shows that the performance of LDE-based P2P system is better than those of Pastry and Chord, in a network of 100 peers.

Small-Sized Network- 100 Peers

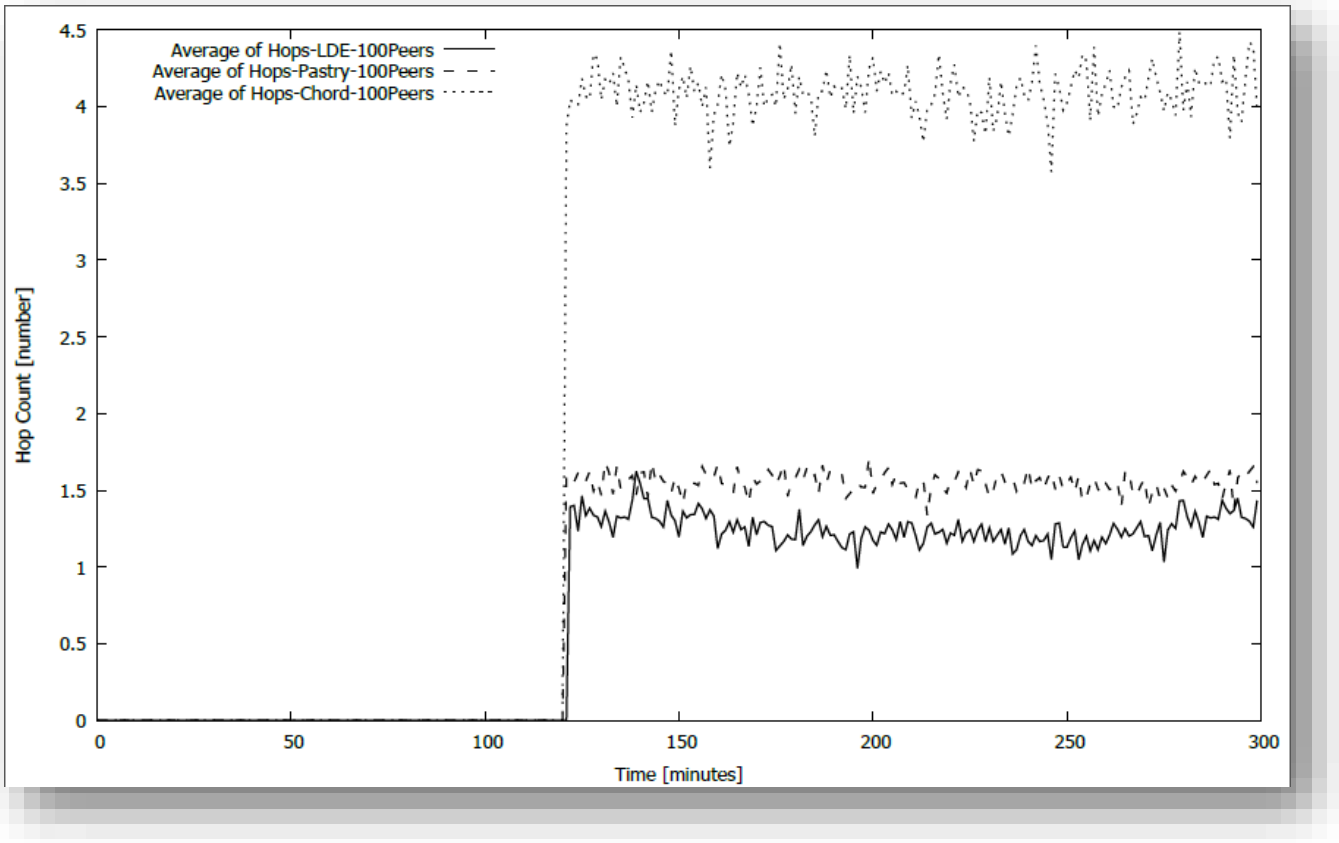


Figure 9.3. Average of hops for 100 Peers in 300 minutes
Number of Clusters in LED-Based System: 5

Simulation results reveal the following:

- Average of the hops for 100 peers in 5 clusters LDE-based system: 1.1847 hop/min
- Average of the hops for 100 peers in Pastry: 1.544 hop/min
- Average of the hops for 100 peers in Chord: 4.0946 hop/min

.

Small-Sized Network- 200 Peers

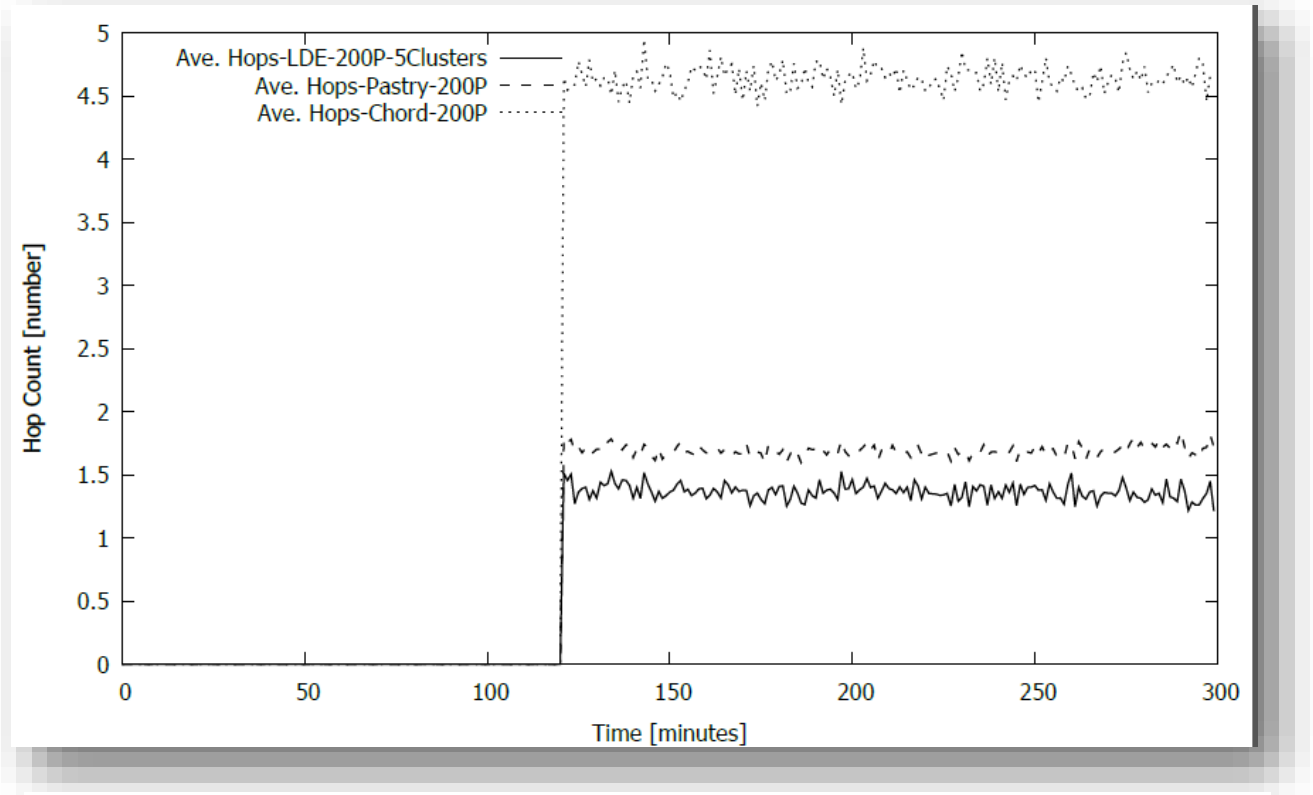


Figure 9. 4. Average of hops for 200 Peers in 300 minutes
Number of Clusters in LED-Based System: 5.

Simulation results reveal the following:

- Average of the hops for 200 peers in 5 clusters LDE-based system: 1. 3629 hop/min
- Average of the hops for 200 peers in Pastry: 1.7570 hop/min
- Average of the hops for 200 peers in Chord: 4.6521 hop/min

Next, we evaluate the Overlay in a small-to-medium size network of 2000 nodes. We repeat the test three times with 5, 10, and 15 clusters respectively.

Medium-Sized Network- 2000 Peers

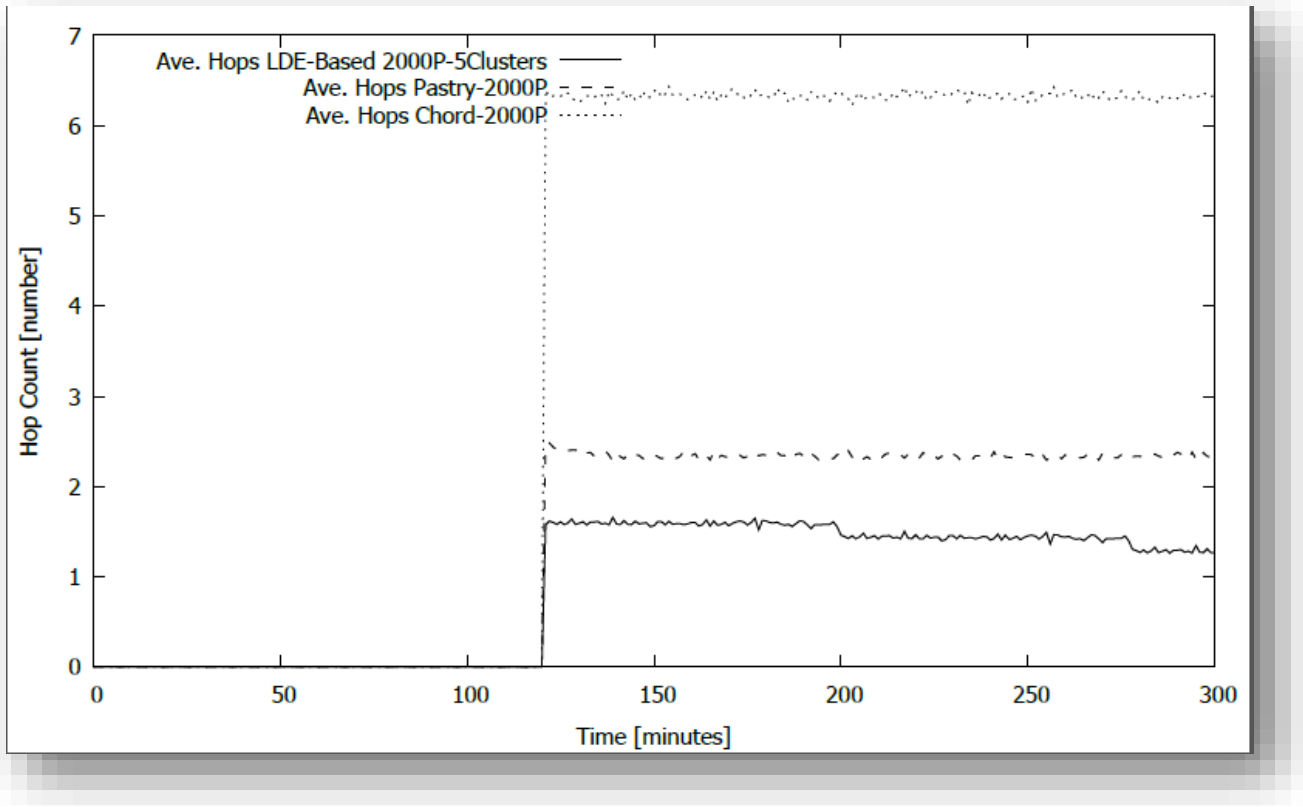


Figure 9.5. Average of hops for 2000 Peers in 300 minutes;
Number of Clusters in LED-Based System: 5.

Simulation results reveal the following:

- Average of the hops for 2000 peers in 5 clusters LDE-based system: 1.4847 hop/min
- Average of the hops for 2000 peers in Pastry: 2.3403 hop/min
- Average of the hops for 2000 peers in Chord: 6.3294 hop/min

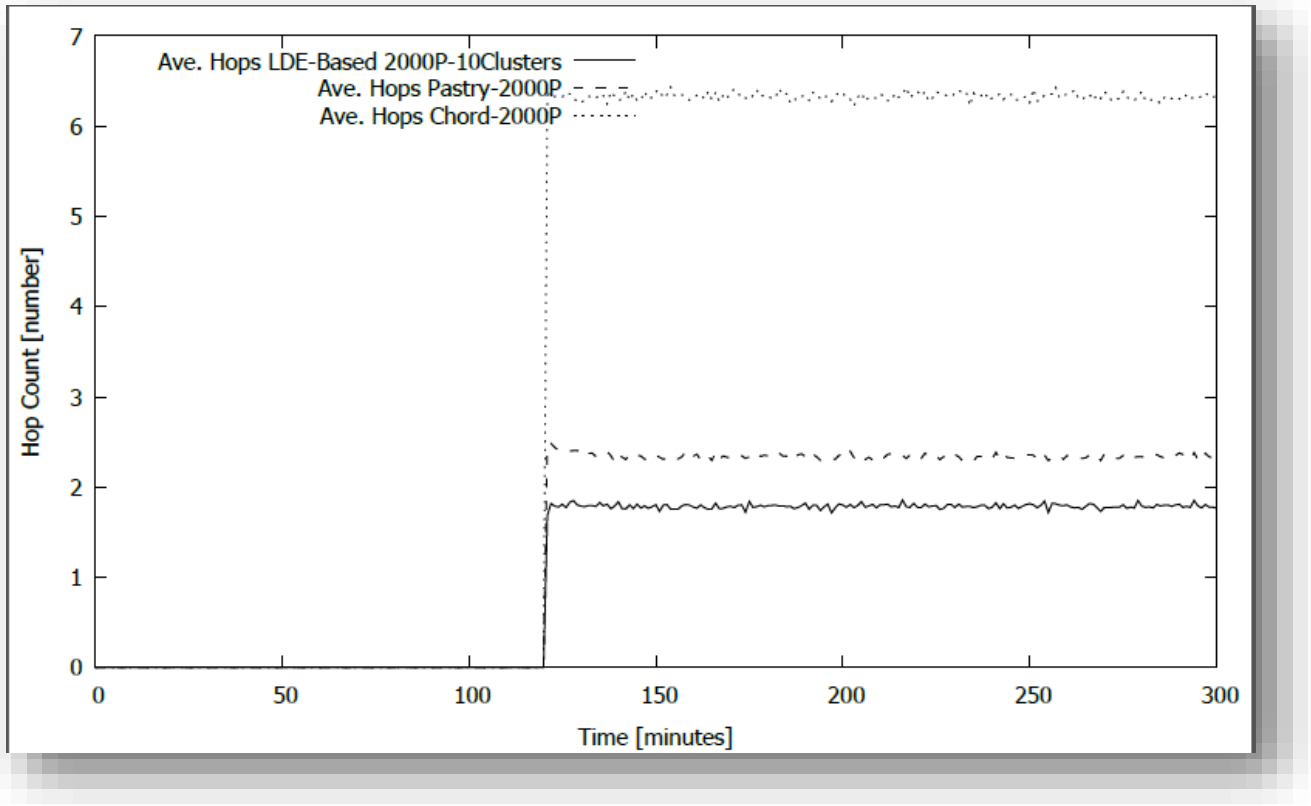


Figure 9.6. Average of hops for 2000 Peers in 300 minutes
Number of Clusters in LDE-Based System: 10

Simulation results reveal the following:

- Average of the hops for 2000 peers in 10 clusters LDE-based system: 1.7647 hop/min
- Average of the hops for 2000 peers in Pastry: 2.3403 hop/min
- Average of the hops for 2000 peers in Chord: 6.3294 hop/min

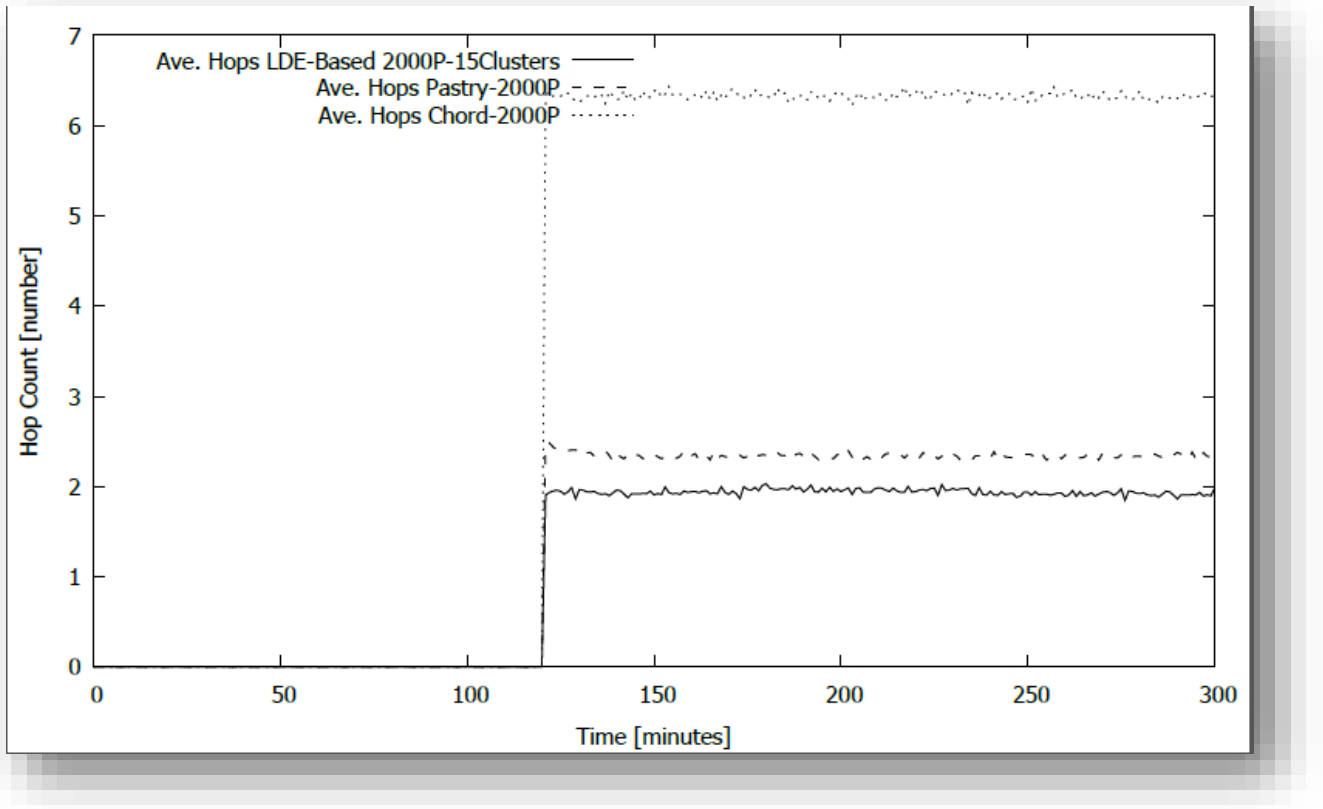


Figure 9.7 Average of hops for 2000 Peers in 300 minutes
Number of Clusters in LDE-Based System: 15.

Simulation results reveal the following:

- Average of the hops for 2000 peers in 15 clusters LDE-based system: 1.917 hop/min
- Average of the hops for 2000 peers in Pastry: 2.3403 hop/min
- Average of the hops for 2000 peers in Chord: 6.3294 hop/min

Large-Sized Network- 10000 Peers

Finally, in stable environment, we analyze the P2P overlays in a large network of 10000 peers.

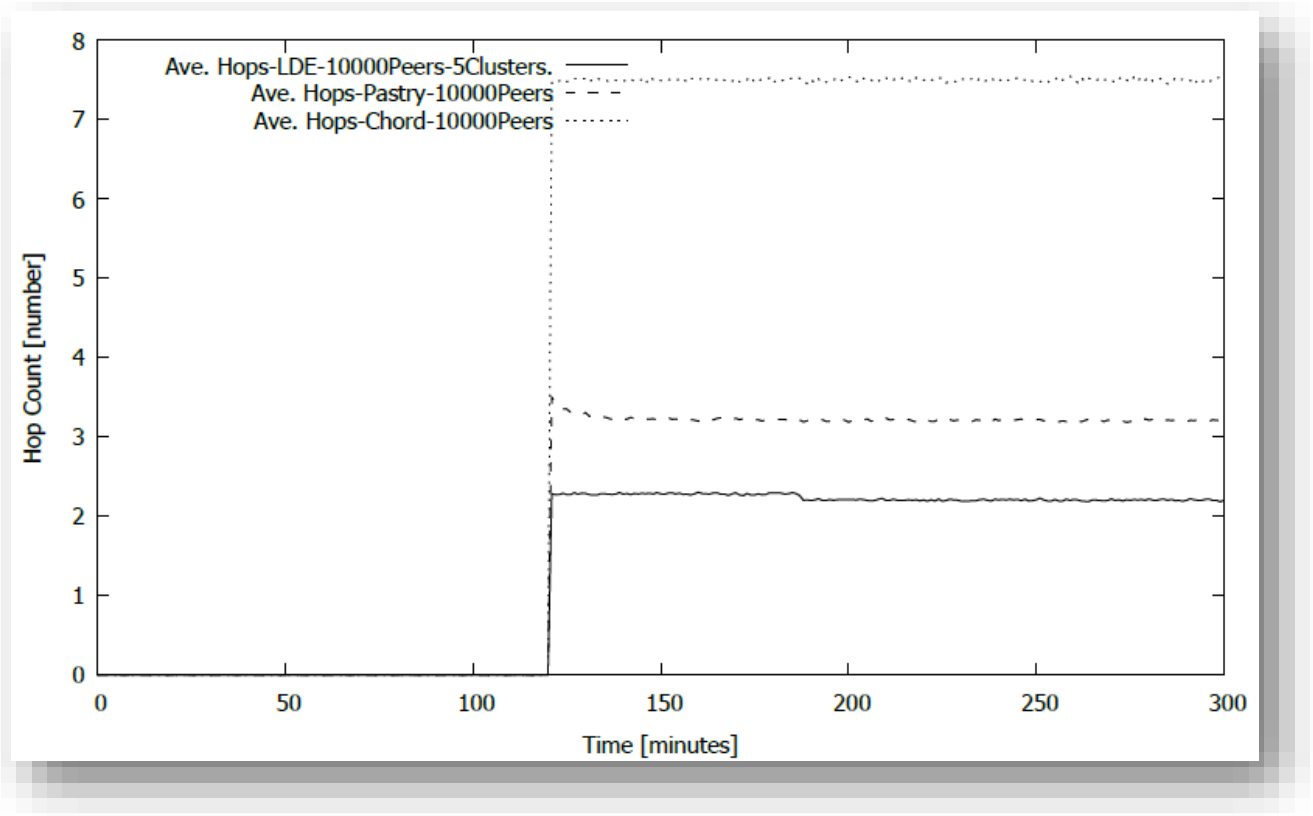


Figure 9.8. Average of hops for 10000 Peers in 300 minutes
Number of Clusters in LED-Based System: 5.

Simulation results reveal the following:

- Average of the hops for 10000 peers in 5 clusters LDE-based system: 2.2354 hop/min
- Average of the hop count for 10000 peers in Pastry: 3.224 hop/min
- Average of the hop count for 10000 peers in Chord; 7.503 hop/min

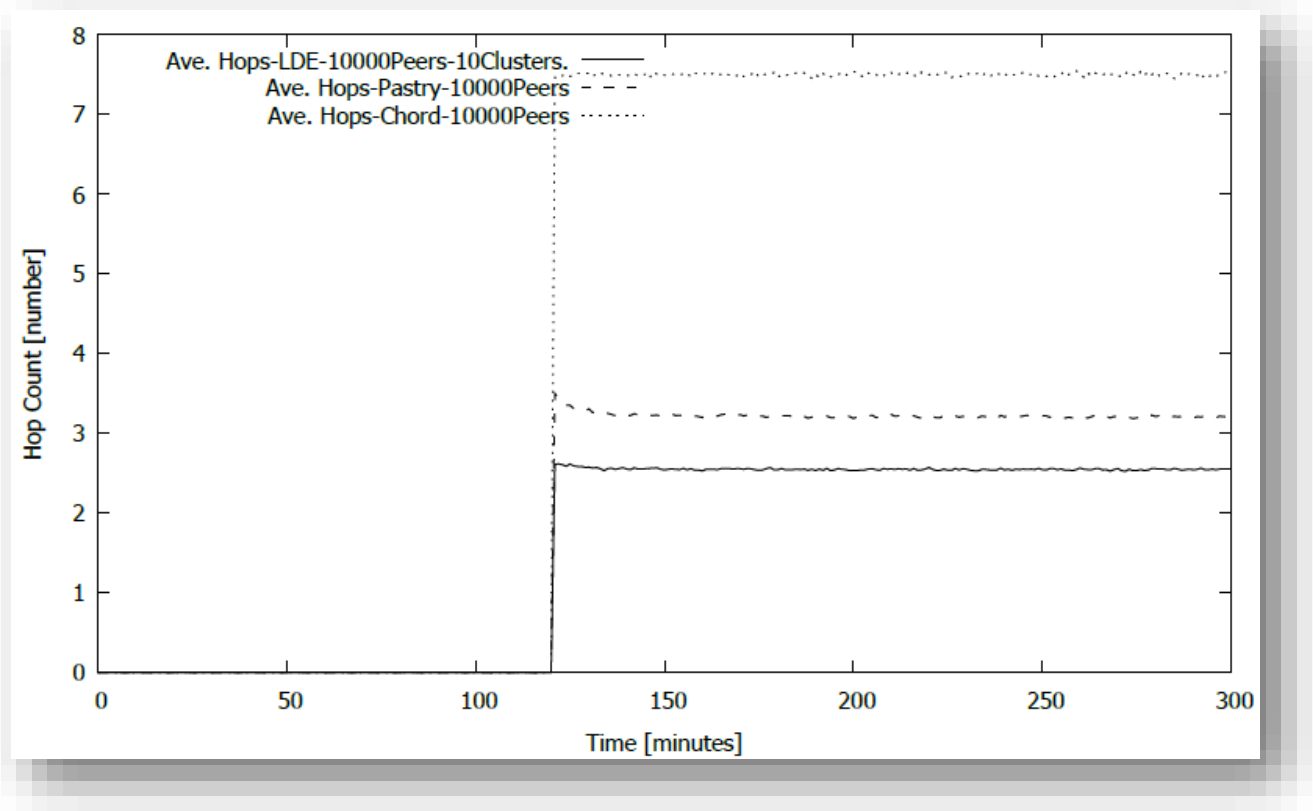


Figure 9.9. Average of hops for 10000 Peers in 300 minutes
Number of Clusters in LDE-Based System: 10

Simulation results reveal the following:

- Average of the hop count for 10000 peers in 10 clusters LDE-based system: 2.552 hop/min
- Average of the hop count for 10000 peers in Pastry: 3.224 hop/min
- Average of the hop counts for 10000 peers in Chord; 7.503 hop/min

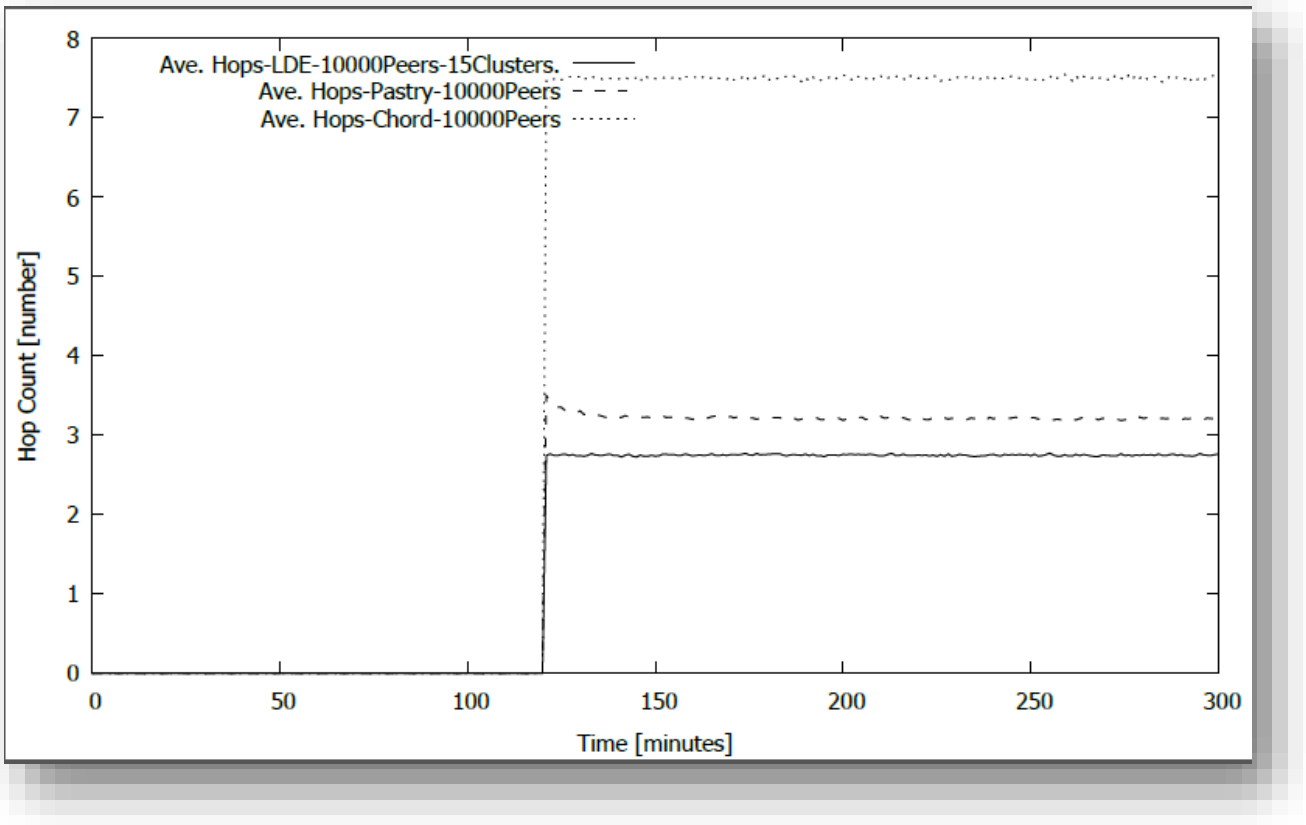


Figure 9.10. Average of hop for 10000 Peers in 300 minutes
Number of Clusters in LDE LDE-Based System: 15

Simulation results reveal the following:

- Average of the hop count for 10000 peers in 15 clusters LDE-based system: 2.7511 hop/min
- Average of the hop count for 10000 peers in Pastry: 3.224 hop/min
- Average of the hop counts for 10000 peers in Chord; 7.503 hop/min

9.4.2 Results in Unstable Network

In this set of simulation, we want to investigate the behavior of our system under more realistic factors, for example, very frequent churn. For this reason, this time, our network is unstable due to the usage of the Kad measurement based churn [68]. A group of scientists has studied the behavior of peers in terms of geographical distribution, their uptime and data usage, every five minutes, for six months. They have successfully collected 51,552 snapshots. Since then, the results of this research have been used in many studies to simulate and evaluate the overlays network. This study is recognized as Kad measurement based study.

In our unstable environment, we compare our topology's hop counts with Pastry and Chord, in very small, mid-size and large networks. For very small 100 and 200 peers have participated in two different simulations, mid-sized networks, it is 2000, and large one, 10000 peers are contributing. As in the case of stable environment, simulation has been performed three times with 5, 10, and 15 LDE-based clusters respectively for both 2000 and 10000 peer-networks.

Note that, in our second evaluation, the packet loss is significant due to the use of Kad measurement churns.

Small-Sized Network- 100 Peers

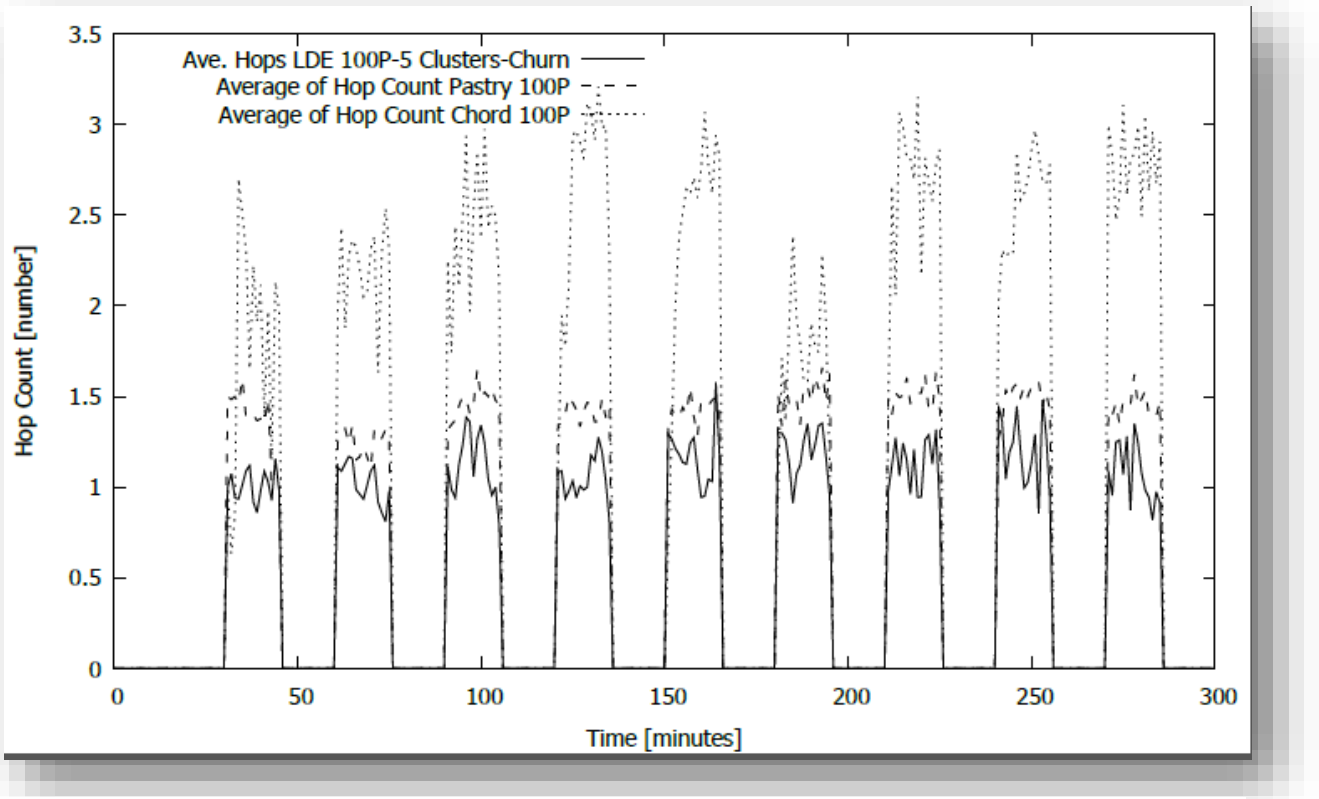


Figure 9.11. Average of hops for 100 Peers in 300 minutes
Number of Clusters in LDE-Based System: 5

Simulation results reveal the following:

- Average of the hops for 100 peers in 5 clusters LDE-based system: 1.3063 hop/min
- Average of the hops for 100 peers in Pastry: 1.6332 hop/min
- Average of the hops for 100 peers in Chord: 3.7325 hop/min

Small-Sized Network- 200 Peers

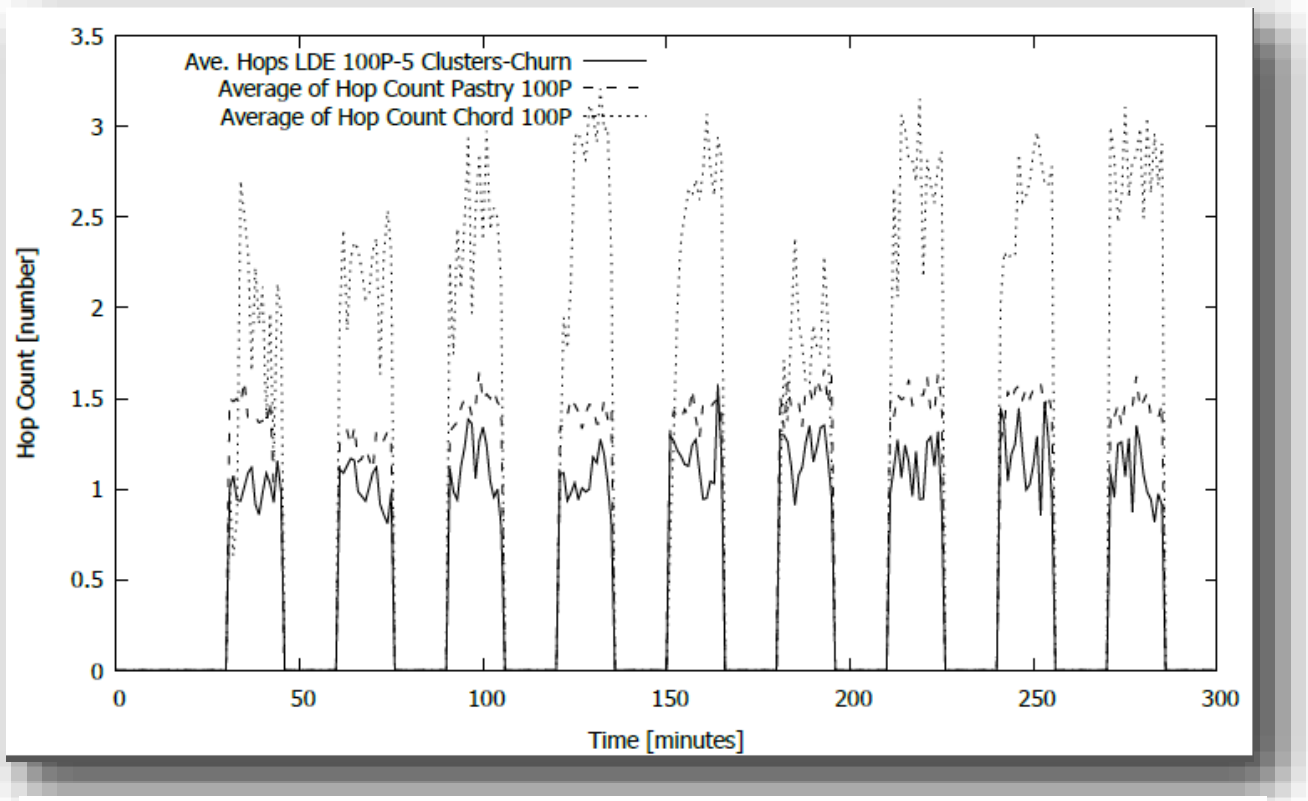


Figure 9.12. Average of hops for 100 Peers in 300 minutes
Number of Clusters in LDE-Based System: 5

Simulation results reveal the following:

- Average of the hops for 200 peers in 5 clusters LDE-based system: 1.5417 hop/min
- Average of the hops for 200 peers in Pastry: 1.8102 hop/min
- Average of the hops for 200 peers in Chord: 4.3321 hop/min

Medium-Sized Network- 2000 Peers

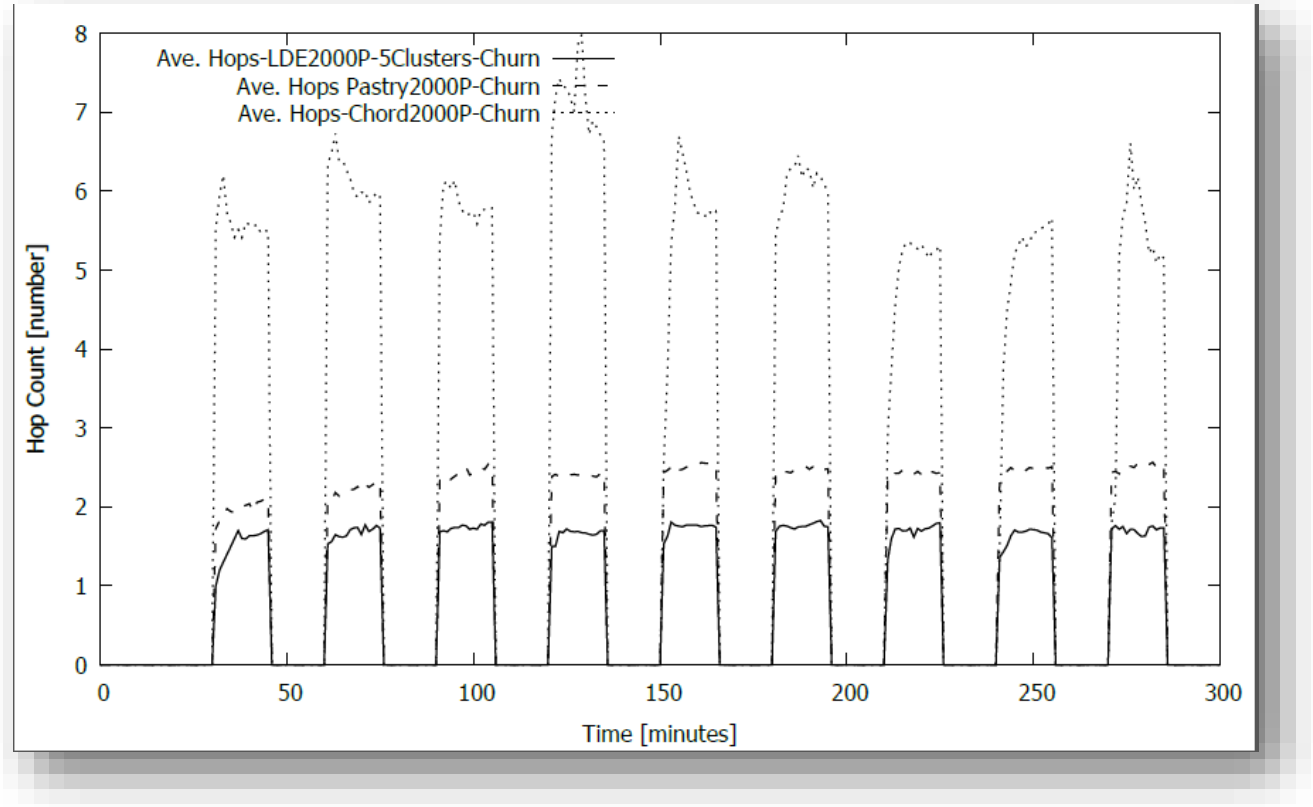


Figure 9.13. Average of hops for 2000 Peers in 300 minutes
Number of Clusters in LDE-Based System: 5

Simulation results reveal the following:

- Average of the hops for 2000 peers in 5 clusters LDE-based system: 1.6803 hop/min
- Average of the hops for 2000 peers in Pastry: 2.377 hop/min
- Average of the hops for 2000 peers in Chord: 5.710 hop/min

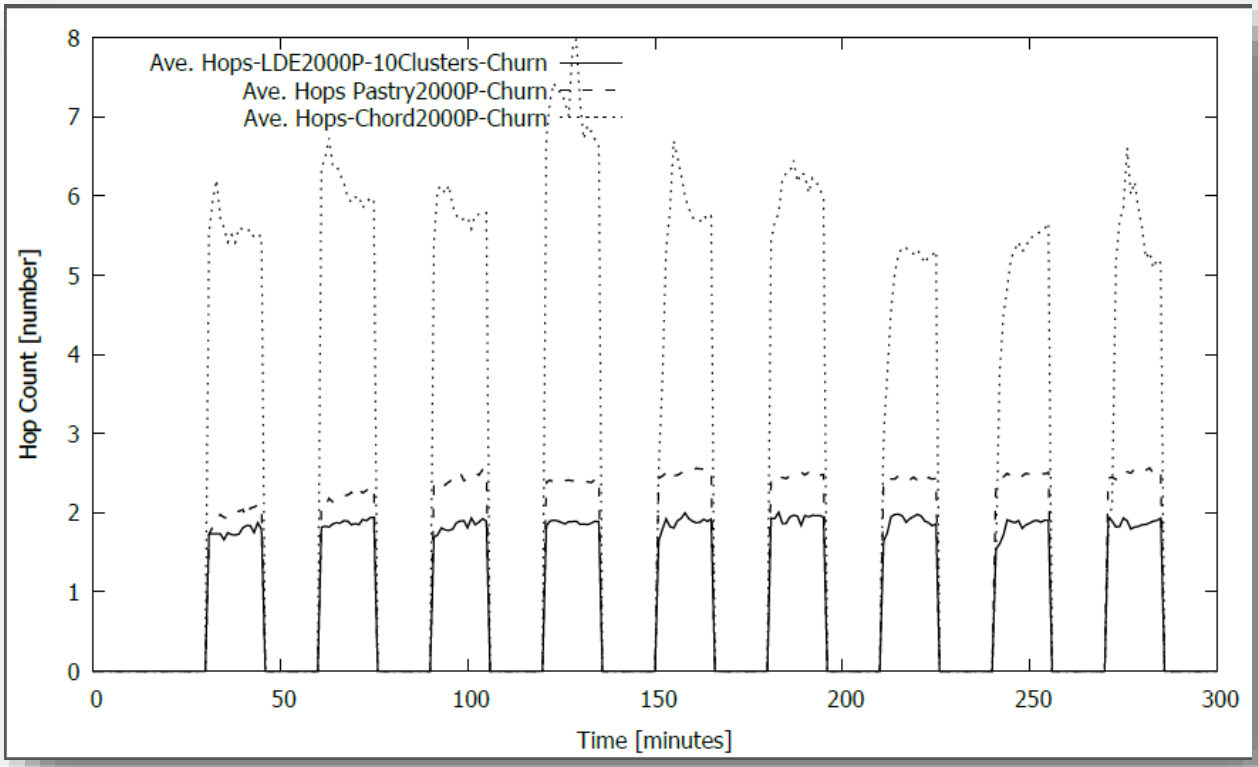


Figure 9.14. Average of hops for 2000 Peers in 300 minutes
Number of Clusters in LDE-Based System: 10

Simulation results reveal the following:

- Average of the hops for 2000 peers in 10 clusters LDE-based system: 1.8610 hop/min
- Average of the hops for 2000 peers in Pastry: 2.377 hop/min
- Average of the hops for 2000 peers in Chord; 5.710 hop/min

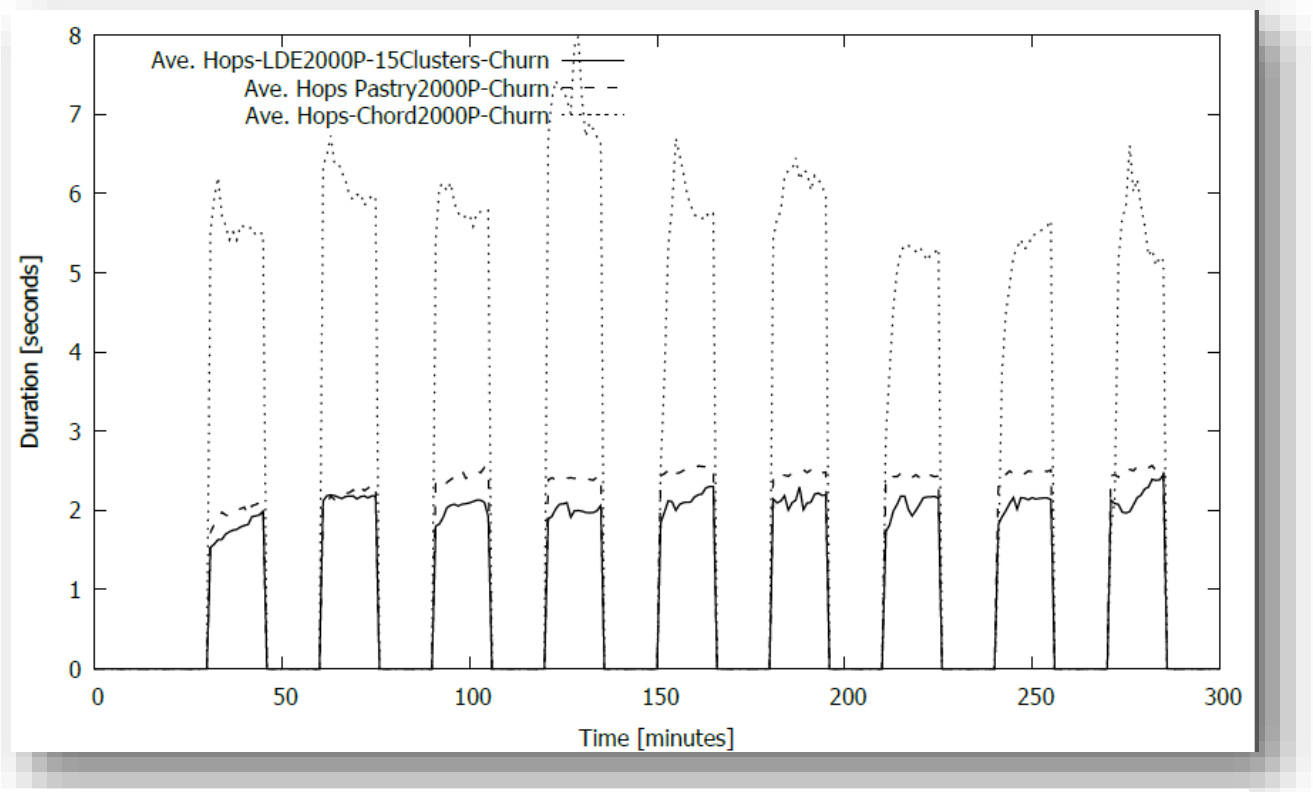


Figure 9.15. Average of hops for 2000 Peers in 300 minutes
Number of Clusters in LDE-Based System: 15

Simulation results reveal the following:

- Average of the hops for 2000 peers in 15 clusters LDE-based is: 2.0618 hop/min
- Average of the hops for 2000 peers in Pastry: 2.377 hop/min
- Average of the hops for 2000 peers in Chord; 5.710 hop/min

Large-Sized Network-10000 Peers

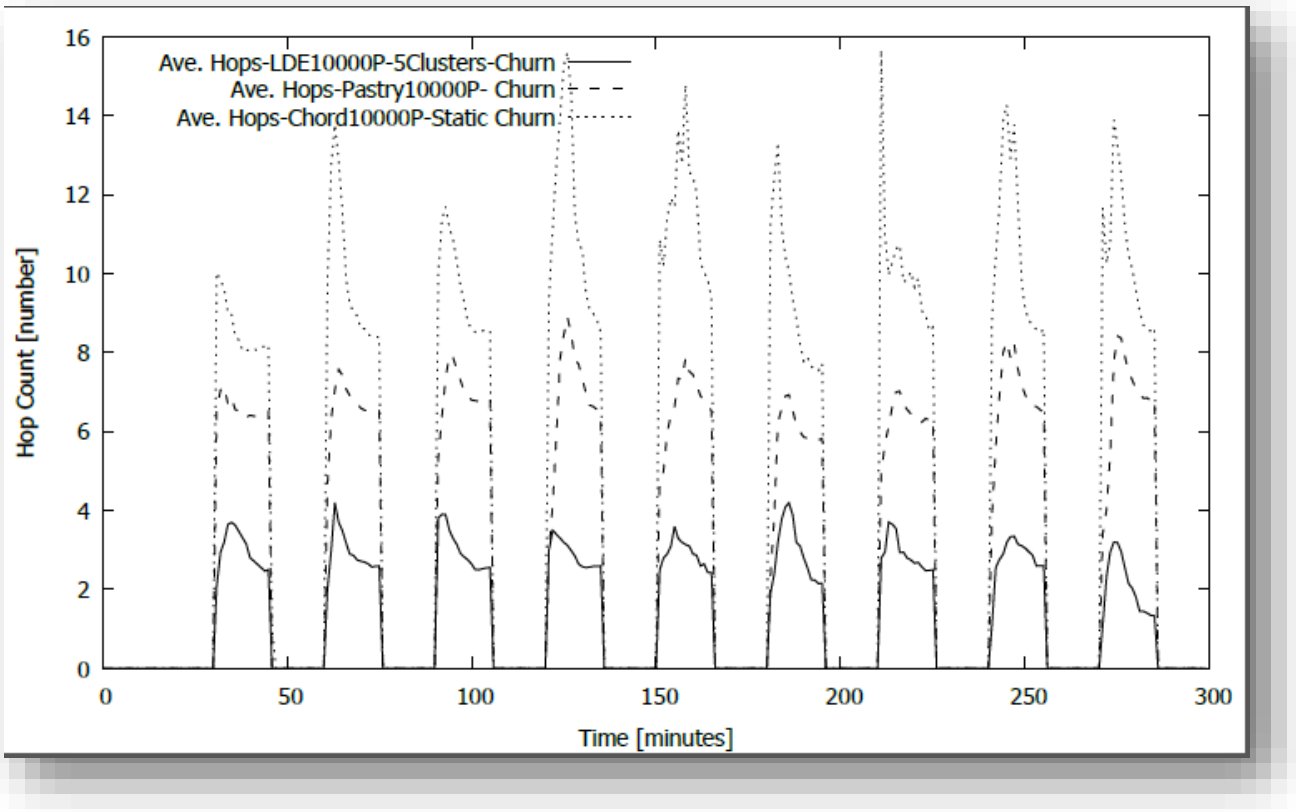


Figure 9.16. Average of hop counts for 10000 Peers in 300 minutes

Number of Clusters in LDE-Based System: 5

Simulation results reveal the following:

- Average of the hops for 10000 peers in 5 clusters LDE-based system: 2.2843 hop/min
- Average of the hops for 10000 peers in Pastry: 6.6174 hop/min
- Average of the hops for 10000 peers in Chord: 10.2083 hop/min

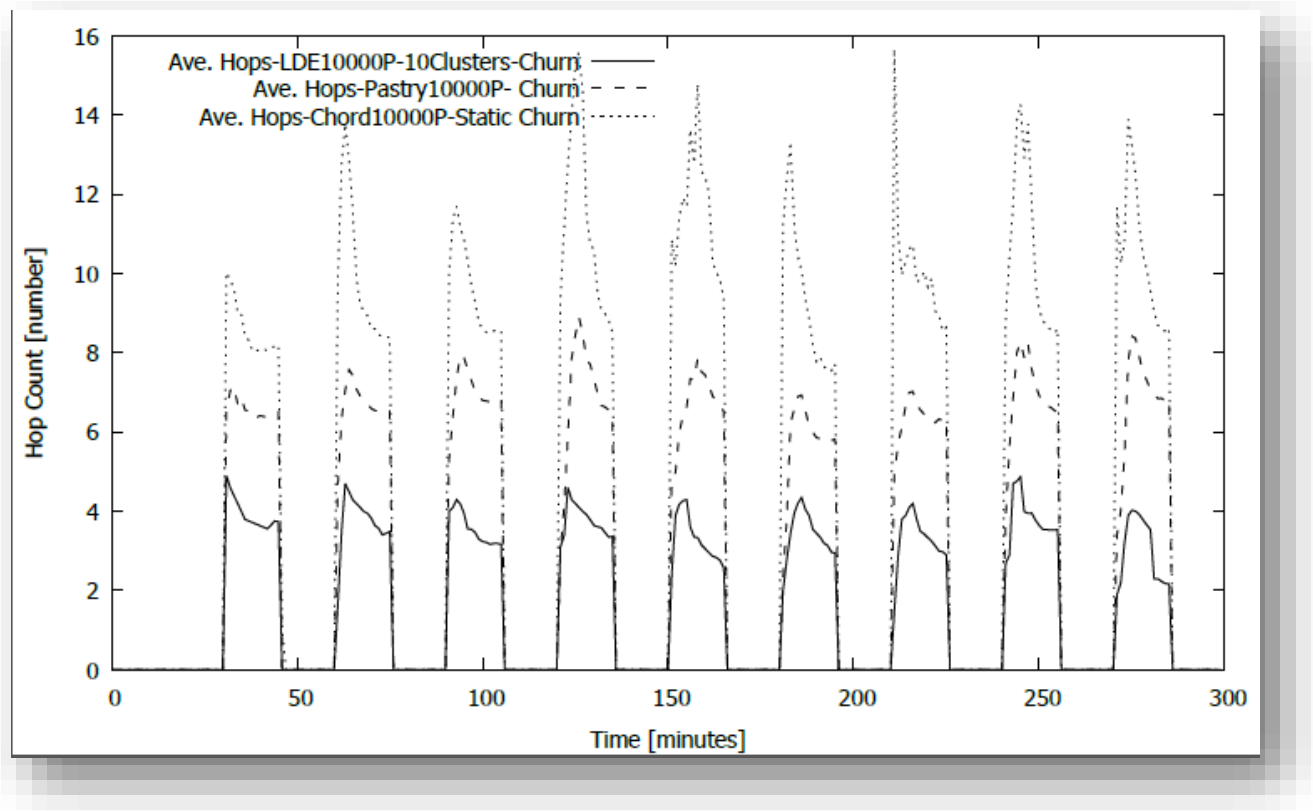


Figure 9.17 Average of hop counts for 10000 Peers in 300 minutes

Number of Clusters in LDE-Based System: 10

Simulation results reveal the following:

- Average of the hops for 10000 peers in 10 clusters LDE-based is: 3.5443 hop/min
- Average of the hops for 10000 peers in Pastry: 6.6174 hop/min
- Average of the hops for 10000 peers in Chord: 10.2083 hop/min

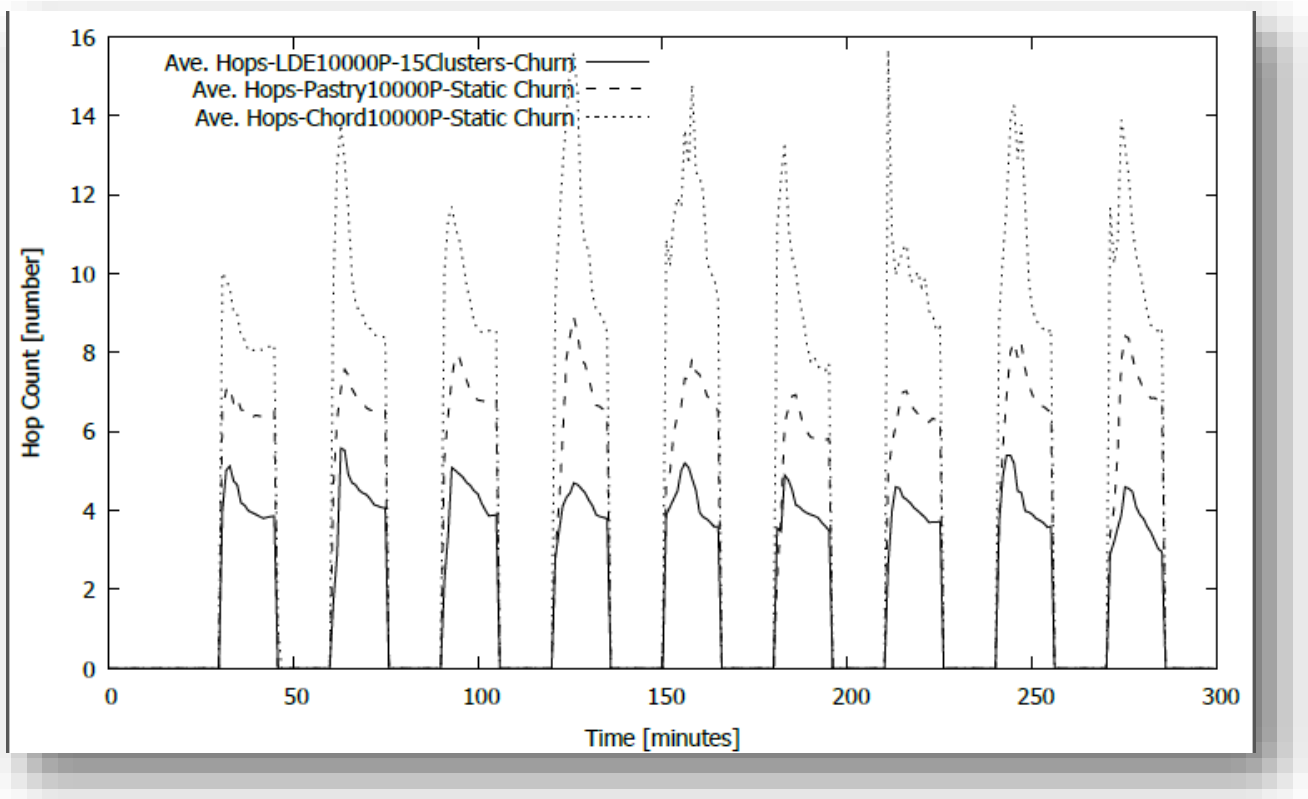


Figure 9.18 Average of hop counts for 10000 Peers in 300 minutes
Number of Clusters in LDE-Based System: 15

Simulation results reveal the following:

- Average of the hop for 10000 peers in 15 clusters LDE-based system: 4.1036 hop/min
- Average of the hop for 10000 peers in Pastry: 6.6174 hop/min
- Average of the hop for 10000 peers in Chord: 10.2083 hop/min

All these simulation results confirm that LDE-based P2P system performs better compared to two of the most well established P2P systems, viz. Chord and Pastry in both stable and unstable environments.

CHAPTER 10

CONCLUSION

At present, most existing structured P2P approaches use Distributed Hash Tables (DHT) to realize their architecture. Use of DHTs guarantees efficient data insertion and data lookup operations in structured P2P systems. However, maintaining DHT-based architecture is a complex task due to random arrivals and departures by peers (churn) at any time. Churn handling is still an open problem in DHT-based P2P networks. To overcome this demerit of DHT-based architecture while improving further the efficiency of data lookup operations, in this research, we have deviated from the existing trend of using distributed hash tables to design structured P2P architecture. To achieve our goal, we have used a number theory based mathematical model, known as 'Linear Diophantine Equation (LDE) and its Mutually Incongruent Solutions' to realize the proposed architecture.

We have shown analytically and through numerous simulations, that LDE-based structured architecture guarantees a significantly lighter weight mechanism to create and maintain the overlay P2P structure as compared to some of the very well established DHT based systems. From the viewpoint of the complexity of data lookup algorithms, the evaluation results have shown that under various environments, the presented LDE-based P2P architecture outperforms Chord and Pastry, two of the very few existing well-established architecture. In addition, we have presented efficient schemes to preserve anonymity, security, and fault-tolerance as well.

To the best of our knowledge, this work is the first to report the use of LDE in designing structured P2P topology. One of the most noteworthy points about the architecture is that

complexity of different data lookup algorithms is a function of the number of distinct resource types only, unlike in other works in which it is a function of the number of peers present in the architecture. In this context, it may be observed that for all practical purposes, the number of distinct resource types is significantly less than the number of peers.

REFERENCES

[1] Sisario, Ben (2011-10-03). "Rhapsody to Acquire Napster in Deal with Best Buy - NYTimes.com". United States: Mediadecoder.blogs.nytimes.com. Retrieved 2013-06-13.

[2] Douglas, G. (2004). Copyright and Peer-To-Peer Music File Sharing: The Napster Case and the Argument Against Legislative Reform. Murdoch University Electronic Journal of Law, Murdoch, Australia, Vol.11, Number 1 (March 2004).

[3] Steinmetz, R., & Wehrle, K. (Eds.). (2005). Peer-to-peer systems and applications (Vol. 3485 .pp 9-16). Springer.

[4] Global Internet Phenomena Report [2016], https://www.sandvine.com/trends/global-Internet-phenomena. Retrieved 2017-9-10.

[5] Coulouris, G. F., Dollimore, J., & Kindberg, T. (2005). Distributed systems: concepts and design. Pearson education. (Vol. 4, pp. 112-120).

[6] Duffield, N. G., Greenberg, A. G., Goyal, P., Mishra, P. P., Ramakrishnan, K. K., & Van der Merwe, J. E. (2005). U.S. Patent No. 6,912,232. Washington, DC: U.S. Patent and Trademark Office.

[7] Li, J., Stribling, J., Gil, T. M., Morris, R., & Kaashoek, M. F. (2004, February). Comparing the Performance of Distributed Hash Tables Under Churn. In Iptps (Vol. 4, pp. 87-99).

[8] Saroiu, S., Gummadi, K. P., Dunn, R. J., Gribble, S. D., & Levy, H. M. (2002). An analysis of Internet content delivery systems. ACM SIGOPS Operating Systems Review, 36(SI), 315-327.

[9] Tanenbaum, A. S., & Van Steen, M. (2007). Distributed systems: principles and

paradigms. Prentice-Hall. (Fifth Edition, pp 740-748)

[10] Haas, Z. J., Deng, J., Liang, B., Papadimitratos, P., & Sajama, S. (2002). Wireless ad hoc networks. Encyclopedia of Telecommunications.

[11] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, 29(7), 1645-1660.

[12] Internet User in the Word. Retrieved 09/30/2017 http://www.Internetlivestats.com/Internet-users/#trend .

[13] Pan, J., Hou, Y. T., & Li, B. (2003). An overview of DNS-based server selections in content distribution networks. Computer Networks, 43(6), 695-711.

[14] Vasilakos, A. V., Zhang, Y., & Spyropoulos, T. (Eds.). (2016). Delay tolerant networks: Protocols and applications. CRC press.

[15] Androutsellis-Theotokis, S., & Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. ACM computing surveys (CSUR), 36(4), 335-371.

[16] Ito, D., Niibori, M., & Kamada, M. (2016, September). A Real-Time Web-Cast System for Classes in the BYOD Style. In Network-Based Information Systems (NBiS), 2016 19th International Conference on (pp. 520-525). IEEE.

[17] Yanovskaya, O., Yanovsky, M., & Kharchenko, V. (2014, September). The concept of green Cloud infrastructure based on distributed computing and hardware accelerator within FPGA as a Service. In Design & Test Symposium (EWDTS), 2014 East-West (pp. 1-4). IEEE.

[18] Caicedo, C. E., Joshi, J. B., & Tuladhar, S. R. (2009). IPv6 security challenges.

Computer, 42(2), 36-42.

[19] Ni, L. M. (2004, December). Challenges in P2P Computing. In ISPA (p. 2).

[20] Dinger, J., & Hartenstein, H. (2006, April). Defending the Sybil attack in p2p networks: Taxonomy, challenges, and a proposal for self-registration. In Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on (pp. 8-pp). IEEE.

[21] Koshy, P., Koshy, D., & McDaniel, P. (2014, March). An analysis of anonymity in bitcoin using p2p network traffic. In International Conference on Financial Cryptography and Data Security (pp. 469-485). Springer, Berlin, Heidelberg.

[22] Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., & Lim, S. (2005). A survey and comparison of peer-to-peer overlay network schemes. IEEE Communications Surveys & Tutorials, 7(2), 72-93.

[23] Ripeanu, M. (2001, August). Peer-to-peer architecture case study: Gnutella network. In Peer-to-Peer Computing, 2001. Proceedings. First International Conference on (pp. 99-100). IEEE.

[24] Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., & Shenker, S. (2003, August). Making gnutella-like p2p systems scalable. In Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (pp. 407-418). ACM..

[25] Ganesan, P., Sun, Q., & Garcia-Molina, H. (2003). Yappers: A peer-to-peer lookup service over arbitrary topology. In INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies (Vol. 2, pp. 1250-1260). IEEE.

[26] P. Ganesan, Q.Sun, and H. Garcia-Molina, “Yappers: A peer-to-peer lookup service over

arbitrary topology," in Proceedings of the IEEE Infocom 2003, San Francisco, USA, March 30 - April 1 2003.

[27] Zhao, B. Y., Huang, L., Stribing, J., Rhea, S. C., Joseph, A. D., & Kubiatowicz, J. D. (2004). Tapestry:"a global-scale overlay for rapid service deployment [J],". IEEE Journal on Selected Areas in Communications, 22(1).

[28] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., & Balakrishnan, H. (2003). Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Transactions on Networking (TON), 11(1), 17-32.

[29] M. Xu, S. Zhou, and J. Guan, "A New and Effective Hierarchical Overlay Structure for Peer-to-Peer Networks", Computer Communications, Elsevier, vol. 34, pp. 862-874, 2011.

[30] Korzun, D., & Gurtov, A. (2014). Hierarchical architectures in structured peer-to-peer overlay networks. Peer-to-Peer Networking and Applications, 7(4), 359-395.

[31] Ratnasamy, S., Francis, P., Handley, M., Karp, R., & Shenker, S. (2001). A scalable content-addressable network (Vol. 31, No. 4, pp. 161-172). ACM.

[32] Klug, J. R., Klug, N. H., & Peterson, T. D. (2015). U.S. Patent No. 8,965,924. Washington, DC: U.S. Patent and Trademark Office.

[33] Rowstron, A., & Druschel, P. (2001, November). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing (pp. 329-350). Springer, Berlin, Heidelberg.

[34] Hoßfeld, T., Oechsner, S., Tutschku, K., Andersen, F. U., & Caviglione, L. (2006, March). Supporting vertical handover by using a pastry peer-to-peer overlay network. In

Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on (pp. 5-pp). IEEE.

[35] Vu, Q. H., Lupu, M., & Ooi, B. C. (2009). Peer-to-peer computing: Principles and applications. Springer Science & Business Media.

[36] Andrea Passarella, "A survey on content-centric technologies for the current internet: cdn and p2p Solutions," Computer Communications, vol. 35, pp. 1-32, 2012.

[37] Maymounkov, P., & Mazieres, D. (2002, March). Kademlia: A peer-to-peer information system based on the xor metric. In International Workshop on Peer-to-Peer Systems (pp. 53-65). Springer, Berlin, Heidelberg.

[38] Hadaller, D., Regan, K., & Russell, T. (2005). Necessity of supernodes survey (Vol. 67, p. 217). Technical report, Technical Report 2005-1, Department of Computer Science, University of Toronto.

[39] Peng, Z., Duan, Z., Qi, J. J., Cao, Y., & Lv, E. (2007, January). HP2P: A hybrid hierarchical P2P network. In Digital Society, 2007. ICDS'07. First International Conference on the (pp. 18-18). IEEE.

[40] Liang, J., Kumar, R., & Ross, K. W. (2004). Understanding KaZaA.

[41] Ebrahimi, M., Bazyar, M. A., Tahmasbi, M., & Boostani, R. (2008, December). Benefiting from data mining techniques in a hybrid Peer-to-Peer network. In Advanced Computer Theory and Engineering, 2008. ICACTE'08. International Conference on (pp. 499-502). IEEE.

[42] Cheng, J., & Donahue, R. (2013). The Pirate Bay Torrent Analysis and Visualization. International Journal of Science, Engineering and Computer Technology, 3(2), 38.

[43] Klingberg, T., & Manfredi, R. (2002). Gnutella 0.6. Network Working Group.

[44] Clarke, I., Sandberg, O., Wiley, B., & Hong, T. W. (2001). Freenet: A distributed anonymous information storage and retrieval system. In Designing privacy enhancing technologies (pp. 46-66). Springer Berlin Heidelberg.

[45] Clarke, I., Miller, S. G., Hong, T. W., Sandberg, O., & Wiley, B. (2002). Protecting free expression online with freenet. IEEE Internet Computing, 6(1), 40-49.

[46] Lawey, A. Q., El-Gorashi, T. E., & Elmirghani, J. M. (2014). BitTorrent content distribution in optical networks. Journal of lightwave technology, 32(21), 3607-3623

[47] E. Cohen, A. Fiat, H. Kaplan, "Associative search in peer-to-peer networks: harnessing latent semantics," vol. 2, pp. 1261-1271, 2003.

[48] Ratnasamy, S., Francis, P., Handley, M., Karp, R., & Shenker, S. (2001). A scalable content-addressable network (Vol. 31, No. 4, pp. 161-172). ACM.

[49] Karger, D., Sherman, A., Berkheimer, A., Bogstad, B., Dhanidina, R., Iwamoto, K., & Yerushalmi, Y. (1999). Web caching with consistent hashing. Computer Networks, 31(11), 1203-1213.

[50] De Canniere, C., & Rechberger, C. (2006, December). Finding SHA-1 characteristics: General results and applications. In ASIACRYPT (Vol. 4284, pp. 1-20).

[51] Tutschku, K. (2004, April). A measurement-based traffic profile of the eDonkey file sharing service. In PAM (Vol. 3015, pp. 12-21).

[52] Yang, B., & Garcia-Molina, H. (2001). Comparing hybrid peer-to-peer systems. In Proceedings of the 27th Intl. Conf. on Very Large Data Bases.

[53] Loo, B. T., Huebsch, R., Stoica, I., & Hellerstein, J. M. (2004, February). The Case for a Hybrid P2P Search Infrastructure. In IPTPS (Vol. 4, pp. 141-150).

[54] Baset, S. A., & Schulzrinne, H. (2004). An analysis of the skype peer-to-peer Internet

telephony protocol. arXiv preprint cs/0412017.

[55] Garces-Erice, L., Biersack, E. W., Ross, K. W., Felber, P. A., & Urvoy-Keller, G. (2003). Hierarchical peer-to-peer systems. Parallel Processing Letters, 13(04), 643-657.

[56] Xu, Z., Min, R., & Hu, Y. (2003, October). HIERAS: a DHT based hierarchical P2P routing algorithm. In Parallel Processing, 2003. Proceedings. 2003 International Conference on (pp. 187-194). IEEE.

[57] Andrews, G. E. (1994). Number theory. Courier Corporation. (pp 32-36)

[58] R. Zhang and Y.C. Hu, "Assisted peer–to-peer search with partial indexing," IEEE Trans. Parallel and Distributed Systems, vol. 18(8), pp. 1146-1158, 2007.

[59] Kartalopoulos, S. V. (2006). A primer on cryptography in communications. IEEE Communications Magazine, 44(4), 146-151.

[60] Ganesan, R. (1996). U.S. Patent No. 5,535,276. Washington, DC: U.S. Patent and Trademark Office.

[61] Stallings, W. (2016). Cryptography and network security: principles and practice. 6th edition Pearson. (pp 417-448)

[62] Naicken, S., Livingston, B., Basu, A., Rodhetbhai, S., Wakeman, I., & Chalmers, D. (2007). The state of peer-to-peer simulators and simulations. ACM SIGCOMM Computer Communication Review, 37(2), 95-98.

[63] Surati, S., Jinwala, D. C., & Garg, S. (2017). A survey of simulators for P2P overlay networks with a case study of the P2P tree overlay using an event-driven simulator. Engineering Science and Technology, an International Journal.

[64] Ebrahim, M., Khan, S., & Mohani, S. S. U. H. (2014). Peer-to-peer network simulators: an analytical review. arXiv preprint arXiv:1405.0400.

[65] Graffi, K. (2011, August). PeerfactSim. KOM: A P2P system simulator—Experiences and lessons learned. In Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on (pp. 154-155). IEEE.

[66] Ng, T. S., & Zhang, H. (2001, November). Towards global network positioning. In Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement (pp. 25-29). ACM.

[67] Feldotto, M., & Graffi, K. (2013, July). Comparative evaluation of peer-to-peer systems using PeerfactSim. KOM. In High Performance Computing and Simulation (HPCS), 2013 International Conference on (pp. 99-106). IEEE.

[68] M. Steiner, T. En-Najjary, and E. Biersack, "Long Term Study of Peer Behavior in the KAD DHT," IEEE/ACM Transactions on Networking, vol. 17, 2009.

[69] Yang, M., & Yang, Y. (2010). An efficient hybrid peer-to-peer system for distributed data sharing. IEEE Transactions on computers, 59(9), 1158-1171.

[70] A. Rowstron and P. Druschel, "Pastry: scalable, distributed object location and routing for large scale peer-to-peer systems," Proc. IFIP/ACM Intl. Conf. Distributed Systems Platforms (Middleware), pp. 329-350, 2001

[71] Ratnasamy, S., Francis, P., Handley, M., Karp, R., & Shenker, S. (2001). A scalable content-addressable network (Vol. 31, No. 4, pp. 161-172). ACM.

[72] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., & Balakrishnan, H. (2003). Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Transactions on Networking (TON), 11(1), 17-32.

[73] D. Korzun and A. Gurtov, "Hierarchical architectures in structured peer-to-peer overlay networks," Peer-to-Peer Networking and Applications, Springer, pp. 1-37, March 2013.

VITA

Graduate School

Southern Illinois University

Shahriar “Nick” Rahimi

NickRahimi@gmail.com

IAU-TNB University

Bachelor of Science, Computer Software Engineering, May 2000

Southern Illinois University

Bachelor of Science, Information Systems Technology, December 2009

Southern Illinois University Carbondale

Master of Science, Computer Science, December 2011

Dissertation Title:

A NOVEL LINEAR DIOPHANTINE EQUATION-BAESD LOW DIAMETER STRUCTURED PEER-TO-PEER NETWORK

Major Professor: Bidyut Gupta

Publications:

B. Gupta, S. Rahimi,, “Efficient Data Lookup in Non-DHT Based Low Diameter Structured P2P Network” To be appeared on: Proc. 2017 IEEE Int. Conf. on Industrial Informatics (INDIN 2017), Emden, Germany, July 2017.

B. Gupta, S. Koneru, A. Alyanbaawi, N.Rahimi; “A Modified Version of DVR-Based Multicasting with Security.” , To be appeared on: Proc. 2017 IEEE Int. Conf. on Industrial Informatics (INDIN 2017), Emden, Germany, July, 2017.

A. Alyanbaawi, B. Gupta, S.Rahimi, and K. Sinha; “An Efficient Approach for Load-Shared and Fault- Tolerant Multicore Shared Tree Multicasting.” , To be appeared on: Proc. 2017 IEEE Int. Conf. on Industrial Informatics (INDIN 2017), Emden, Germany, July, 2017

S. Rahimi, K. Sinha and B. Gupta,, “LDEPTH: A low diameter hierarchical p2p network architecture,” Proc. 2016 IEEE Int. Conf. on Industrial Informatics (INDIN 2016), Poitiers, France, July, 2016.

M. Zhu, S. Rahimi “Experience Towards Integrating Parallel and Distributed Computing Concepts at Different Levels of Undergraduate Courses” IEEE IPDPS conference, Jan 2015

M. Zhu, S. Rahimi, Promoting Teaching Effectiveness and Cultivating Interests in Parallel and Distributed Computing, EduPar workshop in conjunction with IEEE IPDPS 2015, Proceedings of IEEE IPDPS 2015, Hyderabad, India, May 2015

Marhamati, N., Patel, P., Althobaiti, Y., Khorasani, E. S., & Rahimi, S. (2013, May). Revisiting Linguistic Approximation for Computing with Words. In FLAIRS Conference.

Y. Lee, S. Rahimi, S. Harvey, “A Pre-Kernel Agent Platform for Security Assurance,” in proceedings of SSCI 2011 IA - 2011 IEEE Symposium on Intelligent Agents, IEEE Press, pp. 1-7, Paris, France, April 2011.

S. Rahimi, Y. C. Lee, M. Zargham, “An Agent-Based Architecture for High Performance Computing over the Internet,” proceedings of the International Conference on Parallel and Distributed Systems, Rome, Italy, pp.432-444, 2010.