

TWO LAYERS PEER-TO-PEER (P2P) DOMAIN NAME SYSTEM
(DNS)

BY

MAHMOUD SALMAN JASSEM AL-SABA

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

1963 ١٣٨٣

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER ENGINEERING DEPARTMENT

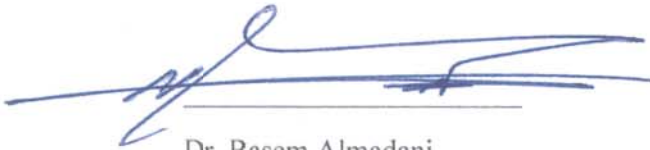
MAY 2013

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS


DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **MAHMOUD SALMAN JASSEM AL-SABA** under the direction his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER NETWORKS**.



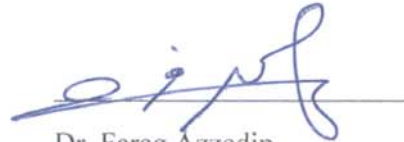
Dr. Basem Almadani
Department Chairman



Dr. Salam A. Zummo
Dean of Graduate Studies



12/9/13
Date



Dr. Farag Azzedin
(Advisor)



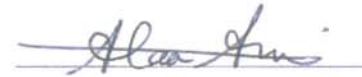
Dr. Marwan Abu-Amara
(Co-Advisor)



Dr. Ashraf Mahmoud
(Member)



Dr. Mohammad Sqalli
(Member)



Dr. Alaaeldin Amin
(Member)

© Mahmoud Salman Jassem AL-Saba

2013

Dedication

ACKNOWLEDGEMENTS

Praise and Thanksgiving to Allah for the strengths and his blessing to complete this thesis. Second, I would like to thank all those who helped me throughout my work, specially the thesis committee for their guidance and help. I would like also to express my thanks to the thesis advisor Dr. Farag Azzedin for his time and effort. Not to forget my family for the support and encouragement they provided me with all the time.

I also would like to thank The King Fahd University of Petroleum & Minerals (KFUPM) for supporting my study and King Abdul-Aziz City for Science and Technology (KACST) for funding the project under the National Science and Technology Plan (project number 08-INF97-4).

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	V
TABLE OF CONTENTS.....	VI
LIST OF TABLES	IX
LIST OF FIGURES	X
LIST OF ABBREVIATIONS.....	XIV
ABSTRACT.....	XVI
ملخص الرسالة	XVII
CHAPTER 1 INTRODUCTION	1
1.1. DOMAIN NAME SYSTEM EVOLUTION	4
1.2. CURRENT DOMAIN NAME SYSTEM STRUCTURE.....	6
1.3. THESIS STRUCTURE	9
CHAPTER 2 PROBLEM STATEMENT.....	10
2.1. THESIS CONTRIBUTION	12
CHAPTER 3 RELATED WORK.....	13
3.1. DNS WEAKNESSES.....	13
3.2. SOLUTIONS TO DNS PROBLEMS.....	17
3.2.1. DNS Tuning.....	17
3.2.2. Restructuring DNS Solutions	19
3.2.3. DNS Protection with DoS Defense Strategies.....	25

3.3.	CHORD: A SCALABLE PEER-TO-PEER LOOKUP PROTOCOL FOR INTERNET APPLICATIONS	27
3.3.1.	Enhance Chord Performance	30
3.3.2.	Balance Load among Nodes	33
CHAPTER 4 TWO LAYERS P2P SOLUTION		35
4.1.	STRUCTURE OF THE PROPOSED SOLUTION	35
4.1.1.	Local Overlay Network (LON)	37
4.1.2.	Global Overlay Network (GON)	38
4.1.3.	Cached Records Expiration	40
4.1.4.	Request Scenario	41
4.2.	ADVANTAGES	42
4.3.	LIMITATIONS	45
4.4.	ASSUMPTIONS	46
CHAPTER 5 SIMULATOR		47
5.1.	SIMULATOR'S DESIGN	47
5.2.	SIMULATOR'S ASSUMPTION	49
5.3.	SIMULATION MEASURES	50
5.3.1.	Load Balance	50
5.3.2.	Path Length	50
5.4.	PERFORMANCE METRICS	53
5.5.	SIMULATOR'S VALIDATION	55
5.5.1.	Load Balance	55
5.5.2.	Path Length	56
5.5.3.	Simultaneous Node Failures	59
5.5.4.	Lookups during Stabilization	59
CHAPTER 6 SIMULATION AND RESULTS ANALYSIS		63

6.1.	SIMULATION SCENARIOS FOR PROPOSED STRUCTURE	63
6.2.	SIMULATION RESULTS	69
6.2.1.	Worst Case Scenario with no Request Repetition	69
6.2.2.	Worst Case Scenario with Zipf Distributed Requests	77
6.2.3.	Best Case Scenario with Zipf Distributed Requests.....	86
6.3.	RESULTS COMPARISON	95
6.3.1.	Load Balance	95
6.3.2.	Path Length.....	96
6.3.3.	Simultaneous Node Failures	97
6.3.4.	Lookups during Stabilization	98
6.3.5.	Node Blockage	100
	CHAPTER 7 FUTURE WORK AND CONCLUSION.....	103
7.1.	CONCLUSION	103
7.2.	FUTURE WORK.....	104
	REFERENCES	105
	VITAE.....	109

LIST OF TABLES

Table 3-1: Domain name samples percentage in TLD	15
Table 5-1: Finger table for node n.....	57
Table 6-1: Simulation configuration	68
Table 6-2: Predicted and simulated results for the average load for normal Chord simulation.....	80
Table 6-3: Predicted and simulated average load for LON and GON in the case of Zipf distributed requests. Each of LON and GON contains 50% of total TLP2P nodes.	80
Table 6-4: Simulated and predicted values for the path length for TLP2P	82
Table 6-5: Predicted and simulated results of the average load for LON network which is 5% of TLP2P total network size.	88
Table 6-6: Predicted and simulated path length for TLP2P (5% LON – 95% GON) with Zipf distributed requests.	89

LIST OF FIGURES

Figure 1.1: An example of resolving the host name “www.google.com” with recursive DNS lookup.....	4
Figure 1.2: Iterative DNS lookup with first query being recursive while resolving the host name “www.google.com”.....	4
Figure 1.3: Manual installation of the mapping file on hosts.....	5
Figure 1.4: Host to IP mapping file is hosted on one server and clients download it locally to resolve hostnames.....	6
Figure 1.5: DNS tree.....	7
Figure 3.1: Impact of TTL on hit rate with different number of clients sharing the cache.....	16
Figure 3.2: Routing table for node in Chord network.....	29
Figure 3.3: An example of a Chord network with three nodes (A, B, C), showing the difference in the identifier space.....	34
Figure 4.1: An example of 4 local overlay networks served by a global overlay network.....	36
Figure 4.2: Local Overlay Network Flowchart.....	38
Figure 4.3: Global Overlay Network Flowchart.....	40
Figure 5.1: Simulator Design Layers. The two columns at the middle of the layer stack is for the two different structure of simulated networks (proposed solution and normal Chord).....	48
Figure 5.2: The mean, 1 st and 99 th percentiles of the number of keys stored per node in 10,000 nodes network.....	55
Figure 5.3: Average path length, 1 st and 99 th percentiles of lookups with varying number of nodes.....	56
Figure 5.4: Average path length, 1 st and 99 th percentiles of lookups with random node failed before start resolving lookups.....	59

Figure 5.5: Average path length, 1 st and 99 th percentiles of lookups while node joining and leaving the network.....	61
Figure 5.6: Average, 1 st and 99 th percentiles of timed out requests while nodes leaving and joining the Chord network	61
Figure 5.7: Average, minimum and maximum of failed lookups while nodes leaving and joining the Chord network.....	62
Figure 6.1: Path length in term of network size, $p = 12 \log_2(n) + 12 \log_2(N)$. The total number of nodes is 100 and the point shown represents the highest path length when GON and LON size is 100/2 and lowest path length when either network having only one node.	67
Figure 6.2: The plot of the average load with fix number of nodes and varying number of queries. The plot represents for each query load three results which are pure node, LON and GON.	70
Figure 6.3: Load fairness results for the three networks normal Chord, GON and LON.....	71
Figure 6.4: Path length measures for normal Chord simulation and LON and GON as well as the total in TLP2P simulation with different network size and query load. .	72
Figure 6.5: Path length measures for node failures scenario. The plot has four results normal Chord, LON, GON and the total for the TLP2P.	73
Figure 6.6: Path length for stabilization scenario comparing normal Chord and TLP2P (LON and GON).	74
Figure 6.7: Time out experienced by the lookup during the stabilization scenario for the three networks normal Chord, LON and GON in addition to TLP2P total.....	75
Figure 6.8: Number of failed requests in 10,000 requests within stabilization scenario for the three networks normal Chord, LON and GON in addition to TLP2P total.	75
Figure 6.9: Path length for the case of blocked nodes scenarios for Chord network and TLP2P (GON and LON) with uniform requests.....	77
Figure 6.10: Average load between nodes for the case of blocked nodes scenarios for Chord network, GON and LON with uniform requests.	77

Figure 6.11: Average keys per node for normal Chord and TLP2P networks with Zipf distributed requests for networks with 10,000 peers. 81

Figure 6.12: Path length results for normal Chord and TLP2P (50% LON – 50% GON) with the two overlay networks. Request distribution is Zipf. 82

Figure 6.13: Path length of the two architectures while the networks are experiencing node failures. Networks received a Zipf distributed requests. 83

Figure 6.14: Stabilization path length for the two designs normal Chord and TLP2P (50% LON – 50% GON) with Zipf distributed requests. 84

Figure 6.15: Lookups' time out for the stabilization case of the two designs normal Chord and TLP2P (50% LON – 50% GON) simulated with Zipf distributed requests. 84

Figure 6.16: number of failed request's within 10,000 Zipf distributed requests for the stabilization case. The plot represents the two designs results normal Chord and TLP2P (50% LON – 50% GON). 85

Figure 6.17: Path length, 1st and 99th percentile for Chord network and worst case of TLP2P (including GON and LON) with Zipf distributed requests. 86

Figure 6.18: Average keys per node, 1st and 99th percentile for Chord network, GON and LON in worst case TLP2P and Zipf distributed requests. 86

Figure 6.19: Average load of Zipf distributed requests simulations for the two design normal Chord and TLP2P (5% LON – 95% GON). 88

Figure 6.20: Path length for normal Chord and TLP2P (5% LON - 95% GON) with Zipf distributed requests. 90

Figure 6.21: Path length for Zipf distributed requests to the two designs normal Chord and TLP2P (5% LON - 95% GON) after node failure. 91

Figure 6.22: Simulation result of the path length for the two designs normal Chord and TLP2P (5% LON - 95% GON) receiving Zipf distributed request. 92

Figure 6.23: Simulation result of the time out for the two designs normal Chord and TLP2P (5% LON - 95% GON) receiving Zipf distributed request. 92

Figure 6.24: The results for number of failed lookups out of the 10,000 Zipf distributed requests for normal Chord and TLP2P (5% LON - 95% GON). 93

Figure 6.25: Path length, 1st and 99th percentile for Chord network and best case of TLP2P (including GON and LON) with Zipf distributed requests..... 94

Figure 6.26: Average load, 1st and 99th percentile for Chord network, GON and LON in best case TLP2P with Zipf distributed requests. 94

Figure 6.27: Number of keys per node simulation for the case of 500,000 Zipf distributed requests (a) Presents record distribution for LON network (5% of TLP2P nodes) (b) Presents pure Chord and GON (95% of TLP2P nodes)..... 96

Figure 6.28: Average path length for the simulated scenario..... 97

Figure 6.29: Average path length for node failure scenario..... 98

Figure 6.30: Average path length for stabilization scenario. 99

Figure 6.31: Average time out for stabilization scenario. 99

Figure 6.32: Average number of lookup failures for stabilization scenario. 100

Figure 6.33: Average path length of the node blockage scenario..... 101

Figure 6.34: Number of keys stored per node in the case of node blockage simulation for the case of 50% of nodes are blocked with Zipf distributed requests (a) Presents record distribution for LON network (5% of TLP2P nodes) (b) Presents pure Chord and GON (95% of TLP2P nodes) 102

LIST OF ABBREVIATIONS

ccTLD	:	country code TLD
com	:	commercial
DDoS	:	Distributed Denial of Service
DHT	:	Distributed Hash Tables
DNS	:	Domain Name System
DoS	:	Denial of Service
edu	:	education
GNP	:	Global Network Positioning
GON	:	Global Overlay Network
gTLD	:	generic TLD
ISP	:	Internet Service Providers
LFU	:	Least Frequently Used
LLCHORD	:	Low Latency Chord
LON	:	Local Overlay Network
LRU	:	Least Recently Used
net	:	network

P2P	:	Peer to Peer
RTT	:	Round Trip Time
SHA	:	Secure Hash Algorithm
TLP2P	:	Two Layers Peer-to-Peer
TTL	:	Time to Live
USDC	:	United States Department of Commerce

ABSTRACT

Full Name: MAHMOUD SALAMAN JASSEM AL-SABA
Thesis Title: TWO LAYERS PEER-TO-PEER (P2P) DOMAIN NAME SYSTEM (DNS)
Major Field: COMPUTER NETWORKS
Date of Degree: 29/5/2013

The domain name system (DNS) is a critical service for survival of all other hosted services on the Internet. Some incidents show that the whole Internet can be halted by defecting DNS functionality. The incidents were caused by exploiting the weaknesses in the current DNS structure. The main weakness that lead to these incidents is the root DNS servers, where all DNS queries will start with to be resolved if the answer is not cached in clients' local DNS servers. These root servers are the targeted point by attackers. In addition, the servers are owned and controlled by one committee which has the authority to stop serving queries coming from any region for political reasons. The aim of this research is to present the new Two Layers Peer-to-Peer (TLP2P) DNS structure which is based on Peer-to-Peer (P2P) model. While there are other solutions that developed structures resistant to denial of service attacks, the new solution targets the problem of ownership of the root DNS servers by one committee. The simulation and analysis results have proven that the proposed design tackled the ownership problem of root DNS servers with lower number of hops than Chord network. Also, the results show that it is a flexible structure with many advantages.

ملخص الرسالة

الاسم الكامل: محمود سلمان جاسم السبع
عنوان الرسالة: حجب الوصول للإنترنت من قبل مزود خدمة الإنترنت الدولية
التخصص: هندسة شبكات
تاريخ الدرجة العلمية: ٢٠١٣/٥/٢٩

نظام أسماء النطاقات (DNS) عبارة عن خدمة أساسية للوصول للخدمات المتواجده على الإنترنت. حيث ان بعض الاعطال التي حصلت في الماضي لهذا النظام اثبتت انها قادرة على ايقاف وظيفة شبكة الإنترنت وبالكامل. و لتعطيل هذا النظام يقوم المهاجمون بأستغلال نقاط الضعف في بنية النظام الحاليه. وأساسا هذا الضعف هو جذر ملقمات نظام أسماء النطاقات حيث كافة الاستعلامات تبدأ بهذه الجذور إذا لم تحل عن طريق الخوادم المحليه. هذه الخوادم الجذرية هي النقطة المستهدفة من قبل المهاجمين، وهي كذلك مملوكة و تسيطر عليها لجنة واحدة والتي لديها السلطة لوقف الخدمة عن اي عملاء من أي منطقة. الهدف من هذا البحث هو تقديم هيكل جديد لنظام أسماء النطاقات (TLP2P) والذي يتكون من طبقتان من الند لند P2P. في حين أن هناك حلول أخرى وضعت هيكل مقاومة لهجوم حجب الخدمة، الحل الجديد يستهدف مشكلة تملك الخوادم الجذرية لنظام أسماء النطاقات. اثبتت نتائج المحاكاة والتحليل في هذا البحث أن التصميم يستطيع مقاومة الهجمات وبأداء جيد وانه مرن مع العديد من المزايا.

CHAPTER 1

INTRODUCTION

The evolution of the Internet has brought a global expansion to the communication era. In this globalization advent, the importance of the Internet has grown tremendously. This growth has continued and driven by the ever great online piles of information, electronic commerce, entertainment and social networking. The Internet basically creates a new layer of functionality and usage in the era of technology. This motivated world-class leading organizations such as Google, Yahoo, eBay, and Amazon to utilize the Internet in providing their day-to-day services [7] [44] [47].

The domain name system (DNS) plays a vital role in the Internet connectivity by converting domain names into IP addresses that can be used by network devices. The domain name space consists of a tree of domain names. Each node or leaf in the tree has zero or more resource records, which hold information associated with the domain name. The tree sub-divides into zones beginning at the root zone. The DNS consists of a hierarchy of DNS servers with 13 root nameservers at the top of the hierarchy. The IP addresses of the 13 root nameservers are hard coded in root hints file in every recursive or caching DNS server [37] [22]. The second important type of name servers is the Top Level Domain (TLD) which is divided into two categories: the generic TLDs (gTLDs) and the country code TLDs (ccTLDs). From TLDs, the rest of the DNS servers extend in

tree-structure into multiple zones [22] [2]. The DNS server that is responsible for adding, removing, or updating the resource records that belong to its zone is called an authoritative name server.

DNS provides a critical and a core service and its absence has a severe impact on other application-layer protocols such as HTTP, FTP, and SMTP. Therefore, for any organization, DNS provides a vital service without which the organization is deemed isolated and hence blocked from using Internet services.

With the current DNS structure, there is no centralized source of data containing all DNS information; rather it is distributed over the authoritative servers in each domain. The authoritative servers for each level domain are required to store the information about the next level domain. For example, root servers store the information about all gTLD and cc-TLD servers and so on. Resolving queries require traversing through multiple servers to reach the authoritative server of the queried host name which can answer with the IP address. The design of the DNS is hierarchal and queries follow the hierarchal path starting from the root. Usually at the Internet Service Providers (ISPs), there are caching DNS servers to cache frequently queried host names to reduce the traffic going to the Internet and also to enhance the response time. When a client requests to resolve a host name, it forwards the request to the local ISP caching DNS. If the answer is already cached, then the local DNS server answers directly from the cache. Otherwise, the local DNS server fetches the answer from the authoritative DNS server. To know the authoritative DNS server of the requested host, the local DNS server contacts any root DNS server. The root DNS server will respond with

the TLD server responsible for the TLD domain in the host name. The TLD server will also answer with the responsible of the organization domain or the next TLD authoritative server depending on the requested host name. The process is repeated until the authoritative DNS server is reached and answers with the IP address or a failure response is received from any of the contacted servers. While the local ISP DNS server uses its cache to answer DNS queries, it is also possible that an intermediate DNS server resolves the query from cache [22].

To illustrate the concept of resolving DNS queries, we provide the following example as shown in Figure 1.1 and Figure 1.2. In this example, we assume that the local caching DNS server is contacting all authoritative DNS servers for each domain to resolve the host name. However, in reality the communication by the caching servers is not always carried out that way, because there are two types of DNS queries namely, recursive and iterative. Typically, clients submit recursive queries and depending on the configuration of the authoritative DNS servers, the query might be forwarded iteratively or recursively. The difference between recursive and iterative queries is that servers accepting recursive requests should either answer with the IP address or a failure. While using iterative queries, the initiator will carry the subsequent interactions with all authoritative DNS servers based on the information it receives. Figure 1.1 shows an example of resolving the host name “www.google.com” with recursive DNS lookup, while Figure 1.2 shows the same example with iterative DNS lookup [23].

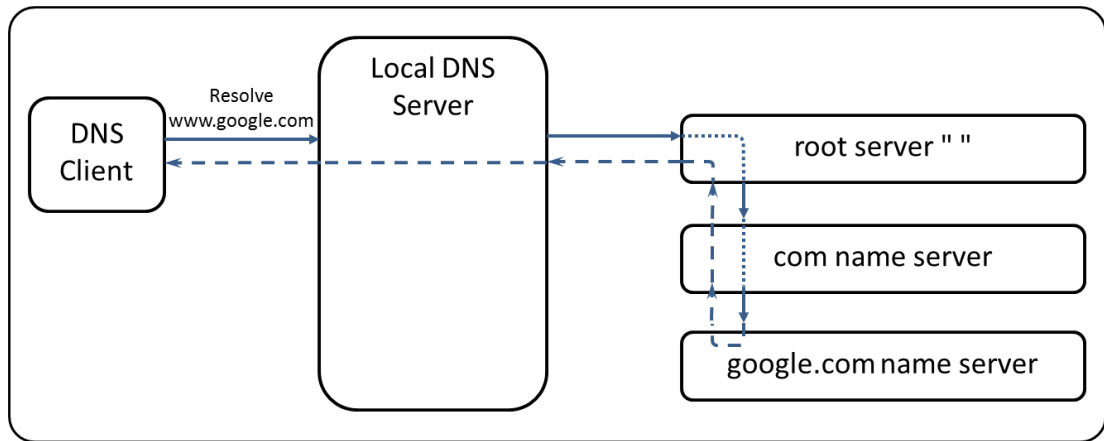


Figure 1.1: An example of resolving the host name “www.google.com” with recursive DNS lookup.

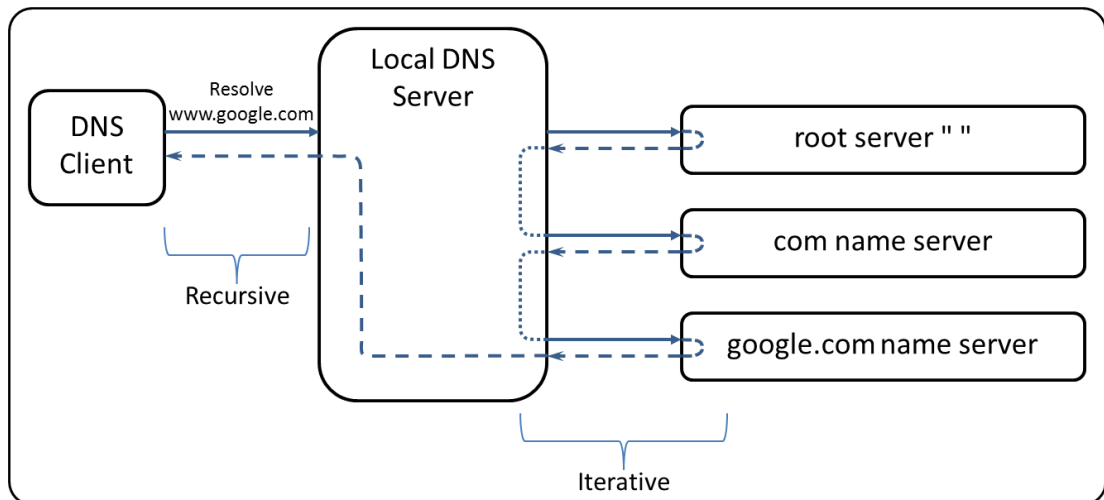


Figure 1.2: Iterative DNS lookup with first query being recursive while resolving the host name “www.google.com”.

1.1. Domain Name System Evolution

At the start of the Internet, the size of the network was limited and a small database can contain the table of all hostnames to IP entries. To resolve a hostname, a simple solution was implemented to satisfy the needs at that time. It

was simply storing the database on each host. Whenever there is a request to initiate a communication with a remote host, the name is first resolved by matching the host name with the entries in the local database and fetching the IP address as shown in Figure 1.3. But with the growth of the network, this solution did not scale. Large network size indicates more possibility of nodes updating their IP addresses and hence increased number of obsolete records in the database. Consequently, this results in frequent manual updates to the database. To eliminate this tremendous overhead of frequent and repetitive manual process, a centralized server was hosting the DNS database and clients download the database whenever they need as shown in Figure 1.4 [28] [2].

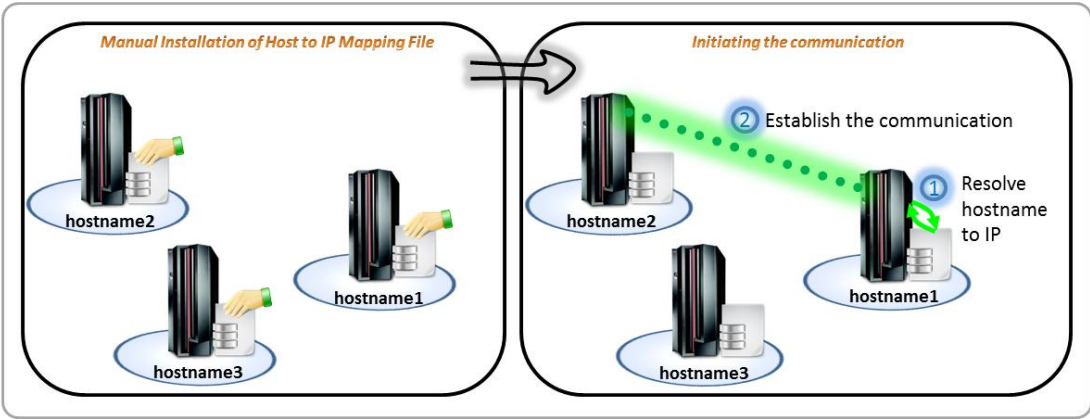


Figure 1.3: Manual installation of the mapping file on hosts.

With further increase in the network size, other problems start to surface. First, the rate of hosts updating their IP addresses was large, resulting in many obsolete entries in the database in a short period of time. Second, the database size started to increase rapidly and that negatively affected the time it takes the clients to download the database. Third, the server hosting the database got overloaded

causing increased response time. Fourth, the downloaded database consumed large storage, at the client side, while only small part of the database is being utilized. [2]

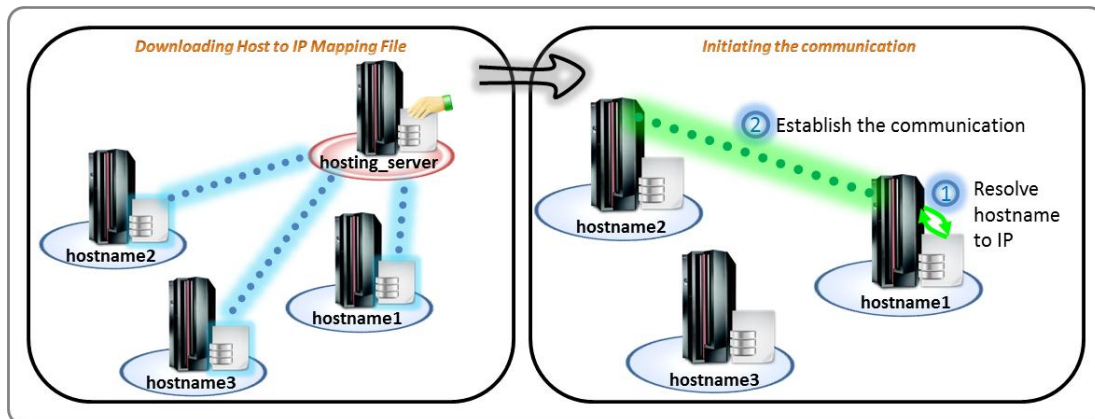


Figure 1.4: Host to IP mapping file is hosted on one server and clients download it locally to resolve hostnames.

1.2. Current Domain Name System Structure

Newer design proposals were introduced to restructure the DNS. These DNS design proposals, after many enhancements resulted in the current DNS structure [2]. Currently, DNS is structured as a hierarchical tree. Each node of the tree is either a branch to a new level or a leaf node. The root node is the root domain and it is represented by an empty string in the host name. It consists of root DNS servers. There are 13 root DNS servers named with $[a \text{ to } m].root-servers.net$. However there are many replicas which are distributed globally as *anycast* root servers. *Anycast* servers are servers with the same IP address distributed over many locations and the packets targeting the IP address will be directed to the nearest topologically server. The root zone is controlled by United States Department of Commerce (USDC). Any changes to this zone require approval from USDC. The next level in the

DNS hierarchy tree is the top-level domain (TLD). It consists of over 300 domains including country code top-level domains (cc-TLD). Example of general top level domains are *com* (commercial), *net* (network) and *edu* (education). Examples of country code top level domains are *sa* for Saudi Arabia and *ae* for Arab Emirates. The next level is the second level domain which is registered to organizations such as *King Fahd University of Petroleum and Minerals* (KFUPM). The next level is the fourth level and it can be a sub-domain within an organization and can be used to refer to different parts of the organization such as departments. Also this level can be the last part of the hostname string which is the name of the leaf node. For example, *scholar.google.com* is the full host name and *scholar* is the name of the host in Google organization. Figure 1.5 shows an example of the DNS tree. It should be noted that the number of domains in a host name is not limited to 4 domains only, it can have larger number of domains but the host name must not exceed the maximum number of 255 characters including the dots.

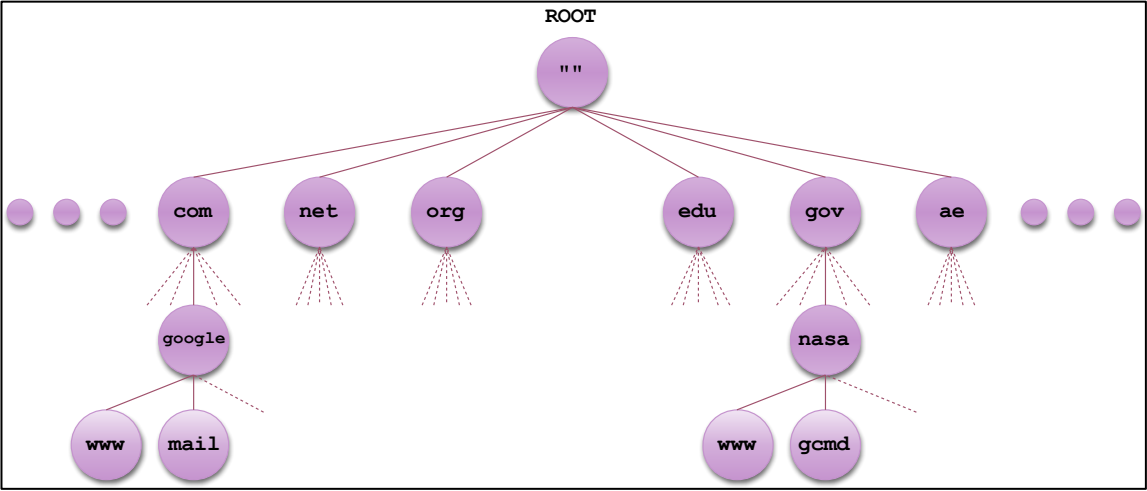


Figure 1.5: DNS tree

While DNS main function is resolving host name to IP address, it is also being utilized to act as a load balancer to share the load between servers. Load balancing is simply achieved by having multiple IP addresses for the same host name. Every time the domain name replies to client requests, it replies with different IP address. However, this is not always an effective way to load balance the requests due to caching [9]. A clients' request goes first to the clients' local DNS. If the request is cached then the local DNS will reply from the cached data. Otherwise, it will fetch the data from the domain server and will respond to the client with the response. Also, a cached copy of the answer will be stored on the caching server. As such, caching DNS servers will have different round robin cycles resulting in more requests targeting some servers than others, ending up with servers not receiving equal loads. Solutions attempted to omit cache effect by specifying zero *time-to-live* (TTL) value. Each entry in the DNS has TTL value which indicates the time limit for the entry to be used before it expires. With zero TTL value, cached entries will be immediately obsolete. But even with this solution, some caching DNS has a predefined minimum TTL value for all records and will not adhere to the specified TTL by the domain server making this solution ineffective.

DNS also requires specifying primary and backup servers for the domain name. In case one fails the other can handle the requests. Web servers for example can take advantage of this by acting as the primary name server for its own hostname. If the web server goes down then the client request for web server's IP address will get no answer. The client will initiate new request to the backup server which will respond with the IP address of the second web server and ending up

with a successful request. All of this process is hidden for the end user in a way that he/she will not notice any change in the service as if the response was fetched from the primary server [9].

1.3. Thesis Structure

The rest of the thesis is structured as follows. Chapter two discusses the motivation of this research and sheds light on the problem statement. Chapter three presents the state-of-the-art related work to the denial of service attacks against DNS and their general solutions. For completeness and clarity purposes, we outline the Chord protocol in chapter four. In chapter five, a detailed explanation of the proposed solution is presented with a discussion of the solution advantages. The structure of the implemented simulation is explained in chapter six. Chapter six also provides verification and correctness studies of our simulator. Chapter seven discusses the simulation results of the proposed solution while chapter eight concludes the research conducted in this thesis and envisions future work.

CHAPTER 2

PROBLEM STATEMENT

In this thesis, we were motivated by the fact that DNS is vulnerable to blockage while it is a critical system for the Internet survival. As discussed in the literature [14] [30] [36] [45], DNS service can be interrupted by a range of attacks such as misconfiguration, denial of service (DoS), cache poisoning, and compromised data. To address such attacks, researches put forward various solutions spanning from using DNS security extensions [24] [35], using “Anycast routing” [37], manipulating the Time-To-Live (TTL) value [34], and increasing the efficiency of DNS caching [33].

The problem of the DNS blockage results in halting the services provided by the DNS system including resolving host names to IP addresses. The blockage can happen in two ways, by DoS attacks or by the DNS higher nameservers. The higher nameservers have the authority to reject queries initiated by clients [37] [38] and with the current DNS design the resolution process can be broken intentionally through those servers for political reasons. The two vulnerabilities exist because DNS structures servers in a hierarchal manner and the job of the root servers is routing DNS requests to the next level which is the top level domain servers. The top level domain servers will also route the requests to the next level and this process is repeated until the requests reach a domain name server which in turn answers the request. If the request is not directed by root DNS servers at the start

of the resolution process, then it will turn out with no answer. There were two incidents where a number of root DNS systems were attacked resulting in an outage of their service [38]. That led to unusable Internet with no accessible service. Prior to these incidents, an incident due to a technical problem in seven of the root DNS servers resulted in one day Internet outage [38]. In addition to these incidents, the resolvers need to go through the higher nameservers (i.e., root and/or TLDs) to resolve the client's request. This raises the level of risk as these higher nameservers can stop serving specific region for political or malicious reasons [13] [29] [31] [36] [45]. Although the idea of malicious and hence intentional denial of service by higher nameservers seems unlikely at first, there are several reasons that may force a root nameserver to become malicious and perform Internet access denial against a specific organization or country. For example, Internet access denial can be driven by political motivations, as governments may force higher nameservers to block Internet access to a specific region or country in an attempt to establish an Internet embargo on the targeted region. Many large services and networks have been attacked recently for political motivations. On December 2009, Gmail, for example, had many attacks targeting email accounts of Chinese human rights activists [16] [43]. Twitter, a popular social network, has also been attacked during 2009 by hackers from Iran [1]. Another prime example of political motivations to deny Internet access to an organization are the recent attempts by many governments to pressure service providers (e.g., DNS service) to block access to WikiLeaks [3] [4]. Such types of attacks are driven by political motives.

2.1. Thesis Contribution

- Resolve DNS blockage problem caused by the international Internet service provider.
- Restructure DNS to two levels Peer-to-Peer overlay networks.
- Implement a recursive Chord peer-to-peer simulator with Java and comparing the obtained results with the presented results in Chord paper for the iterative simulator.

CHAPTER 3

RELATED WORK

This chapter will explore some of the weaknesses in the current DNS system and some of the proposed solutions to Denial of Service attack against DNS as well as the general architecture to tackle DoS and DDoS attack. For the purpose of clarity and completion, we included in section 3.3 a discussion about Chord peer-to-peer protocol with some of the proposed enhancements because it is related to our work.

3.1. DNS Weaknesses

DoS and DDoS are two forms of attacks that aim to disable services provided on the Internet by saturating a targeted service with huge amount requests so it cannot respond to legitimate requests. With the new technologies, clients are becoming more powerful increasing the threat for all published services [40] [43]. As mentioned, two outage incidents of DNS service were caused by DDoS attacks. To find the threat of this type of attack, several studies and measures have been conducted [38]. In 2001, different measurements in different times have been performed in the 'f' root server 'f.root-servers.net' and the results show surprising numbers. In all of the results, 60-85% of the received queries were repeated from

the same hosts and more than 14% of the queries were violating DNS standards. Many requests were unanswerable, like the requester IP address is in the private address space (ex, 10.*, 192.168.*) or asking for invalid top-level-domain. Also, the DNS servers are utilized to act as a reflector for DoS attacks by spoofing the source IP address to be the victim machine. Also, the study shows some hosts were trying to update the root DNS server IP address and sending up to 15000 update requests within one day. The statistics in one of the measures showed 20% out of 10 millions queries were for invalid TLD and 16.5% of the servers just asking for invalid queries. This illustrates how the DoS and DDoS attack is all the time active and consuming servers' resources [26].

In 2002, different measures were done at the clients' side in different locations. The study focuses on the response time and considers only non-cached names in collected statistics as they experience the longest response time. The results show a wide range in the response time which varies from 0.95 seconds to 2.31 seconds. gTLD servers account for 13.9% to 28.9% of the response time and they are queried at 60% of the lookup at each site. Root DNS servers have a very low response time and account for 1.5% to 3.4% of the response time. They are queried during 7% of the lookups at each site. The response time is found to be effected by two factors, i.e., caching and location. The study also collected a large number of domain names using Larbin crawler with 8 different starting points. The number of collected names is around 100,000 names, and after the filtration of invalid names due to typo mistakes and removing duplicate names based on the second level domain, the number of names became 14,983. The names then were

categorized based on the gTLD and ccTLD and the percentage is shown in Table 3-1. It is clear that the variation is very large between TLDs. [30]

Table 3-1: Domain name samples percentage in TLD

TLD category	Percentage of names in category
com	50%
org	14%
net	9%
edu	6%
de	3%
ru	2%
fr	1%
ca	1%
gov	1%
it	1%
151 others	less than 1% each

Jung, Sit, and Balakrishnan performed different measures by collecting extensive packet traces to find the effectiveness of caching and to infer different DNS usage from the associated TCP packets. The observations found are that one quarter of all DNS lookups do not get any answer and at most it requires 2 to 3 times of retransmission to receive a successful answer. While a lack of answer causes a larger number of retransmissions which results in traversing wide area. The results also show that small TTL values of A-records does not degrade DNS scalability, because the hit rate is almost the same for both small and large TTL value for the same set of clients that share the same cache (See Figure 3.1) [14].

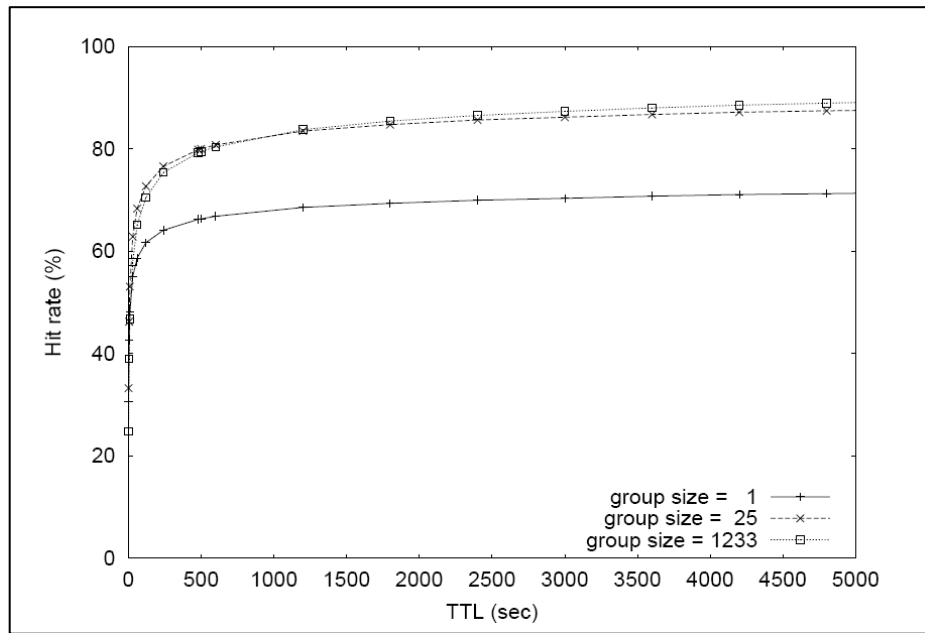


Figure 3.1: Impact of TTL on hit rate with different number of clients sharing the cache

In 2009, RIPE Network Coordination Centre (one of five regional Internet registries) reported a case with increase of query load in the k-root, a root DNS server, which the center supports. The load was not posing a problem to block the traffic and for analysis purposes, all the packets has been collected. However, the load was distributed over other replica of the k-root servers to eliminate drops, as small drops appeared at the start of the queries load. The reason behind the drops is internal bandwidth limitation. Moreover, large amount of queries were originating from large network targeting one anycast server making anycast not effective as usual. The analysis shows that the load was caused by queries for only one hostname which belongs to the *.com* domain. Tracking the source IP address of the queries shows that the top 8 networks querying for the host name are from China. Further analysis shows a low number of queries per IP address, which makes this incident not accurately to be classified as DDoS attack. Measurements

within 30 minutes found that about 60,000 to 65,000 distinct IP addresses are sending queries. With this information, the speculation of the cause of traffic load is that the queries are either for a misconfigured software that are related to the Chinese language or a capability test for a botnet [45].

3.2. Solutions to DNS Problems

DNS attracted many researchers as it is a critical service for the Internet. This section will outline some of the proposed solutions to enhance DNS robustness. The researches are either proposing a totally new DNS structure or proposing some additions to the current DNS system to tackle some of the existing issues. Also, many researches have been conducted to find a solution to DoS and DDoS attacks, because they are the common type of attacks against services published in the Internet.

3.2.1. DNS Tuning

Vasilis, Dan, and Lixia found that the existing DNS system can be enhanced without making significant changes to design architecture. This can be achieved by setting longer time-to-live (TTL) values to some classified DNS records. The classification of DNS records depends on how many zone records are queried. This is used in different techniques for increasing the TTL value. The techniques are TTL Refresh, TTL Renewal, Long TTL, and combination of these. The TTL Refresh

technique refreshes the TTL value every time the record is invoked even if it is not expired. The TTL Renewal tries to keep the popular records cached by refetching and renewing the TTL value just before they expire. The proposed ways of determining which records should be renewed are the least recently used (LRU), least frequently used (LFU), and adaptive of both of these renewal methods. The long TTL technique is making the TTL values long and this can be done by the zone administrator. The last is a combination of more than one method to reduce the overhead that exists in each.

The test was made with different techniques and the results show the following: 1) TTL Refresh has a better resilience than the existing system. 2) A combination of TTL Refresh and Renewal results show that the TTL Refresh with A-LFU (adaptive least frequently used) has the best result compared with other Renewal methods. As an order LRU is the best, then LFU, then A-LRU, and at the last is A-LFU. 3) TTL Refresh and Long-TTL provide almost the same resilience as TTL Refresh and A-LFU. The value for Long-TTL used is 5 and 7 days while this might cause obsolete DNS records. 4) The last technique used in the test is the combination of all, TTL Refresh, Renewal, and Long-TTL. The achieved resilience in the last test was the best. Even though the results show better resilience to DoS attacks than the legacy DNS system, these methods cannot provide resilience like the peer-to-peer networks [34].

Increasing TTL value will help to withstand the service outage in root DNS servers for a short period, but it will also increase the number of obsolete records.

Besides, it will not help in case of long time outage and specially the case of the intentional disabling of root DNS services.

3.2.2. Restructuring DNS Solutions

3.2.2.1. Dynamic Round-Robin Peer-To-Peer (P2P) Domain Name System (DNS)

Fahd [5] developed a solution based on Chord protocol and round robin. The solution depends on the current DNS system to resolve all the requests. It responds from the cache in case the same query was resolved earlier by a peer before the expiration of the record. The query submission to the legacy system is done in round robin to achieve good load balance and with only one hop. First, the network is constructed as specified in the Chord protocol with minor modification to the “find successor” algorithm in order to filter out the blocked peers. Also, the “closest predecessor” and “notify neighbors” algorithms have been changed to consider the blocked peers. The first will account for the new type of peers, the blocked peers, and will deals with them as living peers. The second algorithm will be executed by a node once it gets blocked. The peer with the second function will notify its neighbors that its status has changed from active to blocked or that it wants to leave the network. The round robin has a lookup window to limit the number of trials to resolve a query. Introducing the window meant to resolve the large delays that might be caused by the blocked or dead nodes in the routing table. The size of the window can be ‘one’, which means contacting only the next node to resolve the query; and if it fails, the query fails. The window size can go up to the size of the

routing table and by increasing window size it would increase the likelihood to resolve the query through different node in case the resolution trial with the first node fails. With windows size 'n' the node tries 'n' times to contact different nodes sequentially in the routing table before reporting a failed query. The limitation mentioned in the study is the network cannot be in the consistent state. In other words not all entries in the routing tables will point to the correct nodes and have the right node status. This is due to the natural behavior of the peers as they i.e., join and leave the network and this limitation is inherited from Chord. The simulation results showed that the configuration best performance is to have the window size equals to the size of the routing table and in the simulation case of 4 virtual nodes per real node was the best [5]. It has better performance than the pure Chord, however the solution does not utilize the caches and the dependency increased on the legacy DNS system. Also, hosts published by the authoritative DNS servers in a blocked region are not resolvable outside its region.

3.2.2.2. The Case for Pushing DNS

Mark and Adam [19] construct a DNS structure with a goal to have a robust DNS System invulnerable to attacks. The DNS servers are peers constructing an overlay network distributed over the world at thousands of ISPs distributing DNS name server records. This assumes there is a single master DNS organization that has the authority to access all the name server records existing in the root zone and all the top level domains. This organization should have a public key known by all the DNS servers in the peer-to-peer network. The master site takes all the DNS NS records, and creates a single file. It then signs this file using its private key, and

distributes the signed file to the peers. When the peers receive this file, they use their built-in public key to check the signature on the file. If the signature is valid, then the node caches the file and uses the file as a DNS database to answer DNS requests. It also passes the exact file to other peers. All nodes will do the same processing to end up with the nodes capable of answering the DNS requests. In case there is a node in the network corrupted the file or modified it, other nodes will check the signature and refuse to talk to that peer again and will discard the received file. The updates to records is designed to be incremental, that is the master DNS will regenerate the entire DNS record file weekly and send it to the peers in addition to the updated records that are sent hourly [19].

The result of the experiment shows that the design is robust against attacks and as the number of the learned malicious peers by normal node is less than 30% (compared to non-malicious peers) the infrastructure robustness will not be degraded [19]. The drawbacks are huge data transfer between peers and also the records will be obsolete. Also, having a master DNS organization will introduce the same problem as the one that exist, in the current DNS system with the root server which will make the organization the targeted point for the attack.

3.2.2.3. Overlook: Scalable Name Service on an Overlay Network

Marvin and Michael [21] developed a new DNS system structure utilizing the scalability and the fault tolerance of the overlay (Peer-to-Peer). It can support large number of clients, and allows large numbers of concurrent lookups for the same records while the lookup latencies are measured in seconds. Furthermore, hundreds of updated records can be visible to the public within seconds. The

overlay network (Peer-to-Peer protocol) selected for the implementation was Pastry, while there are others which can be as well used without affecting the performance. The main function of the Pastry protocol is that given a request it routes the request to the destination home node [21].

Pastry network design requires every node in the network to have a unique 128-bit node Id and the set of existing node Ids must be uniformly distributed. The node Id can be generated by hashing the node public key or the IP address. To reach the destination, Pastry uses the targeted node Id as an input for nodes in the route. Each node that receives the message will look for a destination node Id and will send it to the node that is numerically closer to the destination Id. For a node to know which other nodes have ids closer to the destination Id, it should maintain a list of nodes with Ids numerically closer to its Id. The list is called the routing table and should have at least $O(\log N)$ entries, where N is the number of nodes in the network. Each entry is a mapping of a node Id to a node's IP address. With this design, it can cost a message to hop through $O(\log N)$ to reach its destination node which can cause a delay. To reduce the lookup time and the latency resulted due to large number of hops, a replica strategy of records is used [21].

The testing experiment was done in simulation. The number of stations used is 600 in the core network and 60,000 LAN nodes connected to different link capacities. The result shows even with only 100Mbps links capacity and system of about 10,000 servers the network can have message processing service times of 0.5 milliseconds, handling request loads of 560,000 requests per second. However, the aggregate CPU overhead of processing multiple application-level and the

forwarding hops per lookup request makes the contribution of each new server adds no value in reducing the lookup latency. Also if a node with low CPU capacity and connection bandwidth is connected to participate with the peers nodes, it will decrease the performance of the overall system. [21]

3.2.2.4. The Design and Implementation of a Next Generation Name Service for the Internet

Venugopalan and Emin [33] redesigned the DNS system to be of a high performance, resilient to attacks, fast update propagation, and can replace the existing DNS system without making changes to the clients. It is a peer-to-peer solution and it uses distributed hash tables (DHT) structure. The nodes and records are assigned random identifiers and the records are stored at the home node (nearest node in the identifier space). This solution uses Beehive to reduce the latency time of the lookup but requires more space and bandwidth by automatically replicating the DNS mappings throughout the network to match the anticipated demand and provides a strong performance. It is done by locally measuring the access frequency of each record, and periodically aggregating them with other nodes at every aggregation interval. Then, each node aggregates values gathered from nodes one level higher in the routing table till it reaches the home node. The home node computes a final aggregate and propagates it to all replicas in the system. For the replication process, the home node determines the replication level to know which node should have a replica. The record does not have TTL values. If the record has changed, the home node will send the updated records to other nodes which have the replica. Home nodes ensure the correctness of the records

through DNSSEC standard protocol and also through a centralized authority to sign records fetched from the legacy DNS. The test was done with 75 nodes and the average query per second for the system was 6.5. The results show that the average bandwidth was 12.2 KB/s and the average number of records per node was 4127 (13 MB), while the experiment started with no records stored in the nodes. This means that the load is evenly balanced across the nodes. Also, the update propagation is taking less than one second to reach 98% of the replicas and for the worst case scenario it takes less than a minute. So, this solution provides an effective resilience to attacks and has low update latencies [33]. But, this solution does not tackle the intentional disabling of DNS service by root DNS committee.

3.2.2.5. Serving DNS using a Peer-to-Peer Lookup Service

Russ et al. [32] used Chord peer-to-peer network protocol as a design to the DNS system to have a solution that is fault-tolerant, load balanced and eliminates the need of many administrations as it is the case of the existing DNS system. It uses the same resource record sets as the existing DNS system and also the DNSSEC protocol to ensure the authentication of data it receives. In this peer-to-peer structure, the distribution of the records over peer nodes is done using DHash (a hashing algorithm). To verify the effectiveness of the solution, a simulation was done with 1000 nodes and no record replication algorithm is applied and with no node failure. The result shows that the solution did an adequate job in balancing records among the nodes, but the response time was worse than the legacy DNS system. The average response time they had was 350ms while in the legacy DNS is 43ms. [32]

3.2.3. DNS Protection with DoS Defense Strategies

There are many researches that have been made for DoS and DDoS, and there are many proposed defense systems to tackle these two common attacks. In this section, two of the proposed solutions will be discussed. They depend on the middleware software technology which allows the inter-process communication between different systems by hiding the heterogeneity among them. Wei et al. [35] showed a DoS preventing system for a distributed heterogeneous environment. The structure of this solution is to install two components in the network that operate in a virtual private environment. The two components are middleware box and domain agent. They are special devices inserted at various locations in the network. Middleware boxes have generic primitive functionality that can be reprogrammed to change their roles and functionality to suit the network condition in preventing DoS attacks. The domain agent is the controller, which controls the middleware boxes in its domain and also it has the capability to communicate with other domain agents to cooperate in case of DoS attack. Middleware boxes belong to only one domain and each domain is controlled by one domain agent. The middleware boxes will monitor, intercept, and filter the packets. When it detects potential events, it will alert the domain agent with some information about the event. The domain agent will store the events in its database and will check the alert with its intrusion detection system. If the result of this process indicates an attack, then with the information it has about the network and aggregated events it will take the optimal decision. The decision can be changing the role of the middleware boxes or

sending the intrusion report to other domain agents to cooperate and mitigate the effect of a DoS attack [35].

To enable the middleware boxes to communicate with each other, detect the attack, and change their policy and behavior, it needs to have at least these modules as specified in the paper: Attack detection module, signalling module, policy parser module, policy integration module, and traffic processing module. The proposed layers for the domain agent are: (1) Link Layer, which provides the controlling signals. (2) Device Layer, the controller of a single middleware box. (3) Aggregator Layer, which aggregates intrusion reports from different middleware boxes and allocates different middleware boxes to cooperate in case of attack. (4) Feature Services Layer, which provides the virtual private operation environment services like VPN and DDoS prevention. For the signalling protocols, there are four which are: (1) Secured Topology Auto-Discovery, at the start of deployment, the domain agents create the self organized virtual network with middleware boxes. (2) Secured Software Download, which enables the middleware to download required or updated software from the domain agent using this protocol. (3) Run-Time Control Protocol, which specifies the message format between a middleware box and a domain agent. (4) Multiple Domain Support, which allows for the inter-domain communication. This solution has three advantages. First it is transparent: the middleware hides the heterogeneity for the application, providing upward and downward compatibility. Second, the solution is efficient: with the limited defense resources, the system uses all of its capability to prevent the DDoS. Also, it can utilize the resources in different domains for this purpose. The third advantage is

the effectiveness: the solution can detect and prevent attacks without noticeable service degradation by the end users [35].

Mudhakar et al. [24] and Ling developed a middleware protection system for application level DoS. The solution uses middleware on the server side, and its operation is divided to server's firewall and application layer. The proposed solution does not try to detect the DoS attack, but rather it determines the amount of used resources to satisfy a request. The evaluation of the request is used to prioritize clients. When the evaluation indicates high resources consumption, the client priority is decreased, giving the chance for the higher priority clients; thus, controlling the number of requests per unit time a client can issue. If the client exceeds this limit, the requests are filtered [24].

Depending on the size of the network to be protected and the service provided, this defense system can be adapted to mitigate the DoS attack. In this research, no defense strategy is used, but any defense strategy can be used as an extra layer of protection to enhance the DNS system and make it more robust to DDoS attack.

3.3. Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications

Chord is a structured peer-to-peer network protocol that maps keys to nodes. The Chord system is designed to efficiently assign nodes with suitable records of data to simplify the process of finding records among peer nodes. It is

also scalable with no need for centralized management to handle the joining and leaving nodes. The architecture of this protocol depends on the uses of consistent hashing to assign keys to nodes. With consistent hashing, the load (number of records) between nodes tends to be balanced and when nodes join or leave, there would be little movement of data. For a Chord node to perform a lookup to map a key to a node, it needs to communicate with other nodes based on routing information. Each node builds the routing table once it joins the Chord network. The table contains information about the other nodes. Each row in the table stores information about a node, which are node's IP address and its hashed value. Node's hashed value is the hashing result of the node's IP address. In Chord network consisting of N nodes, the number of rows in the table is $O(\log N)$. In Chord paper [10], the routing table is called finger table and the i -th row is called finger i . Each finger has a value which is the result of adding the node's hashed value with 2^i , where i is the finger number. A finger table is populated with nodes' information by finding the home node of each finger. To achieve that, for every finger a lookup is generated to find the home node of the finger value. Once the home node for a finger is found, the home node information is stored within the finger. An example of how the finger table is created is shown in Figure 3.2. The structure of the finger table has two advantages, first it stores information about small number of nodes and the seconds it knows more about the nearer nodes and less about far nodes. To make the system function as desired, the finger table should be updated periodically to avoid inconsistency in the network. Different algorithms need to be

executed regularly to correct the finger table for any new joining or leaving node from the system.

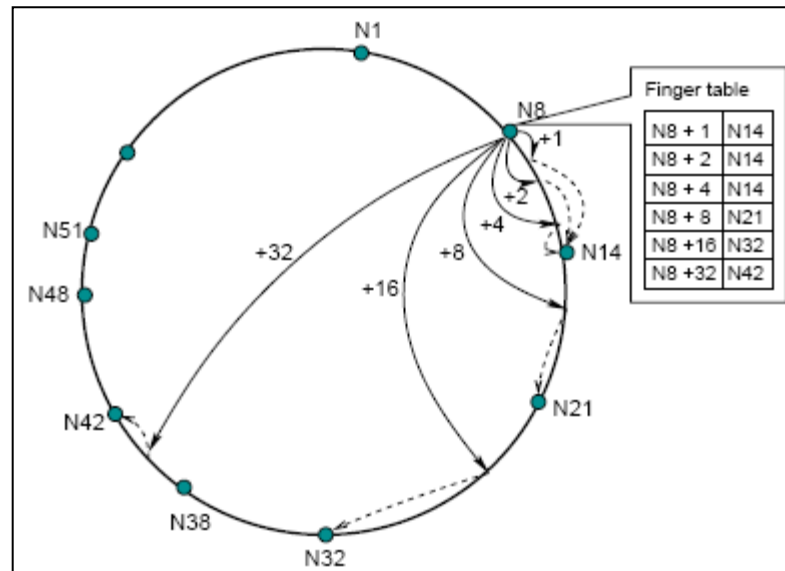


Figure 3.2: Routing table for node in Chord network.

To locate the home node of a request, any node receiving a request will forward it to the node in its finger table that satisfies this condition: its hashed value is the largest among the others and smaller than the request hashed value in the identifier space. This process is repeated at each node till the request reaches the home node which responds to the requester directly. Locating the home node requires passing through intermediate nodes each having latency. That is, each node will introduce more delay resulting in larger response time. The response time will differ from one request to another depending on the number of nodes the requests will pass through. In the worst case, it is $O(\log N)$. If the network size is very large, then it can lead to a large response time. Chord can maintain additional

routing information to reduce the number of hopping, but at the expense of more complexity.

A simulation has been performed and the results show good performance even in the face of concurrent node arrivals and departures and it continues to function correctly, although at a degraded performance. At this performance condition, the nodes' finger table is not fully update to point to the right nodes, which yield to larger number of timeouts and number of hops. The test also shows that the best frequency of running the stabilization algorithm is the same as the frequency of leaving and joining nodes [10].

3.3.1. Enhance Chord Performance

Chord performance is directly affected by the size of the network. As the network size increases, the path length for the lookup increases which in turn increases the latency for the lookups. Yi and Jinyuan [46] worked on creating improved Chord protocol, which is a low latency Chord (LLCHORD). LLCHORD reduces the query latency while retaining the same number of hops for the lookup. To reduce the latency, LLCHORD looked to two concerns in Chord, i.e., number of hops to resolve the lookup and query latency between peers. For the first concern, LLCHORD reduces the average number of hops by maintaining the nodes' information in the counter clockwise in addition to clockwise that is maintained by Chord. The finger table in this case is populated with clockwise and counter clockwise nodes, and the information gathered for both is the same except for the use of subtraction instead of addition in the case of counter clockwise. This

additional information will make it possible for a lookup to be routed in the counter clockwise direction too; and in this case, the path to the destination will require fewer hops. For the other concern which deals with query latency between peers, it is caused by the mismatch between nodes' physical locations and the peer-to-peer overlay network. There are many solutions to localize nodes on the Internet among the other nodes. One solution is the IDMap which uses the trace to find the latency between the peers and then advertise the results to them. Another solution is the Global Network Positioning (GNP) model which uses Landmark nodes to measure the round trip time (RTT) between nodes to create a 2D space of their locations. With this model peers can find their locations in the 2D space by measuring the RTT to the landmark nodes. The two mentioned solutions can be used in LLCHORD, but usually these kinds of services are not easy to get in peer-to-peer environments since they requires either a centralized node or third party nodes to perform the measurements. LLCHORD can depend on simple measurements such as finding only the RTT between the node and nodes in its finger table. After the collection of these measurements, they are stored in another table called the neighbor table. With the new design, the routing is no longer the same as Chord, but it requires extra steps. The routing starts by finding the successor of the key in clockwise or counter clockwise directions. Then, find the nearest node in the neighbor table to the successor node. That node should also be nearer in key space to the destination than the current node. The process is repeated till the destination node is reached. The ideal case is finding the destination node in the neighbor table and the query can be routed directly. While this approach will reduce query latency, it will

eventually result in larger number of lookup hops; and with the bidirectional query, the lookup hops will tend to be the same as in Chord. A simulation is made for this study and the results indicate larger number of hops in LLCHORD than in Chord. In a Chord network, the average path length is 5.62 whereas in LLCHORD it is 6.94. On the other hand the average lookup latency in LLCHORD is 79.87 ms and in Chord it is 85.2 ms [46].

Yu et al. [48] reformat Chord link to enhance the lookup performance and also minimize the influence of the dynamic network. Besides this, the load balancing can be obtained as well. The proposed structure makes a group of nodes as the basic unit in the Chord ring. Each group consists of one or more nodes, and lookups will be traversing groups instead of the nodes which is the case in Chord. Grouping will scale down the network size to result in shorter path length. Nodes with similar bits in the range (25, 32) of their IP addresses will belong to the same group and the group identifier will be the result of hashing those bits. Keys are distributed over groups; and in a group with multiple nodes, the key is assigned to a node with good capability. Also, in each group there is a representative node responsible for storing and updating the group table. The group table is constructed like the finger table in Chord, but each row has a group identifier and the IP address of the representative node. To make it possible for a lookup to locate a key, another table is required which is GroupIN table. It contains the information about the nodes in the group, such as node identifier, node IP address, key location, and the representative node. All the nodes will communicate with each other from time to time to have the updated GroupIN table. Simulation runs have been

performed with maximum group size is 4 nodes and the results show improvement in the average path length compared to the original Chord. The difference is about 0.9 in all scenarios of different network sizes [48].

3.3.2. Balance Load among Nodes

Chord simulation outputs, for the case of load balance, as shown in [10] Figure 8, shows large variation in the number of keys assigned to nodes. In these simulation runs there were nodes storing no key while other nodes store more than 450 keys. The explanation to this phenomenon is explained clearly in [10]. For the purpose of completion and clarification, we re-iterate this explanation here. Nodes position within Chord is based on the output of the hashing algorithm SHA-1 which is random. This randomness of the nodes distribution over the identifier space causes ununiformed gaps between nodes. The consequence is that nodes far from the predecessor in the identifier space will have large key assignments; on the other hand, nodes nearer to their predecessors will result in less key assignments. Figure 3.3 illustrate the inconsistency of key assignment with an example of a Chord network consisting of three nodes (A, B, C). In this example the difference between node B and its predecessor (A) is 10, while the difference between C and B is 72. Again, as the hashing used (SHA-1) for keys is uniform the probability of keys assigned to C will be higher and that indicates more loads targeting C than B.

Chord proposed a solution to the unbalanced load problem by introducing a mechanism to re-map the identifier space to the nodes. Virtual nodes are used here to map the identifier space to the real nodes. Each of the virtual nodes will be

assigned a random key which is independent and unrelated to each other. As in the case of real node each of the virtual nodes will be responsible for the keys in its range. Then randomly map multiple of those virtual nodes to each real node. The simulation results showed a better balance as the number of virtual nodes per real node increased. With 20 virtual nodes mapped to one real node, the range of keys per node is [50, 160] whereas it is [0, 480] without virtual nodes. [10]

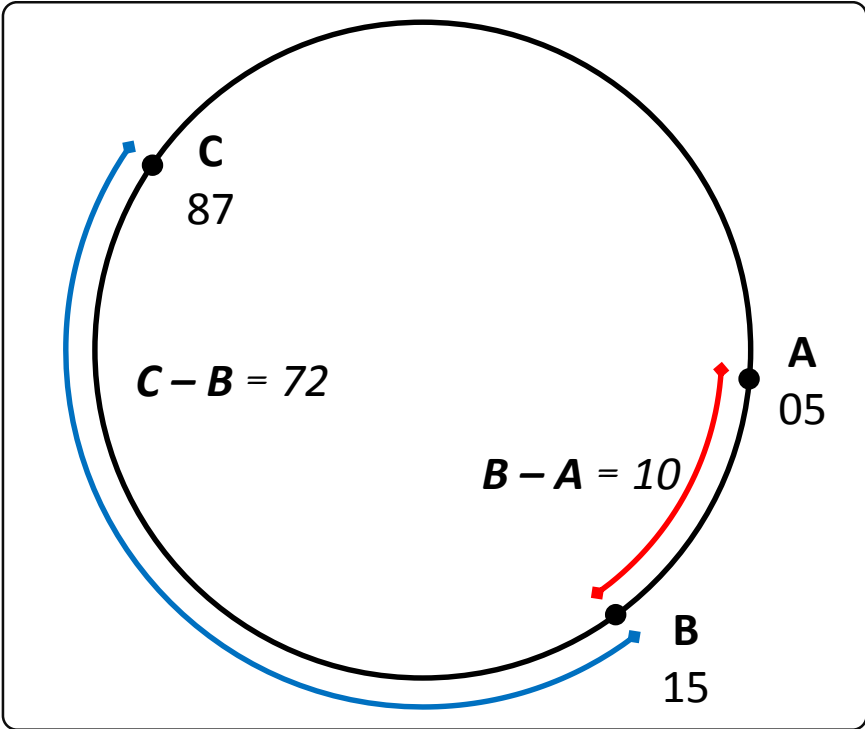


Figure 3.3: An example of a Chord network with three nodes (A, B, C), showing the difference in the identifier space.

CHAPTER 4

TWO LAYERS P2P SOLUTION

This chapter describes our proposed Two Layers P2P (TLP2P) solution and its structure. This chapter also includes the advantages and limitations of the solution with the assumptions for the design to function as required.

4.1. Structure of the Proposed Solution

Structured peer-to-peer networks are self managed, fault tolerant and robust against the DoS attack [19] [21]. The proposed design uses a structured peer-to-peer network to inherit its advantages. By making the design based on peer-to-peer, it will be self managed eliminating the need for a centralized control over the system which is the main issue we want to tackle. The new structure consists of two levels of overlay networks and both are based on Chord protocol. The selection of Chord was based on the simplicity of its design, provable correctness and provable performance. Also by comparing Chord with Pastry which is another P2P protocol, it shows that Chord has better resistance against DoS and location hiding. However, the Content-Addressable Network (CAN) P2P protocol is better in these two criteria compared to the other two P2P protocols, but CAN requires more computation and complexity [8]. In addition, in Chord the load

balance and lookup path length are more consistent than CAN which makes Chord a good selection for this problem [10].

The new structure consists of two overlay networks namely local overlay network (LON) and global overlay network (GON). A blocked region will construct a LON to serve client requests while GON will server LON to answer the request through the legacy system. Figure 4.1 explains the concept which shows multiple LONs consisting of different number of nodes and GON serving all LONs. The next sections will have further explanation of the two overlay networks along with the difference between the nodes.

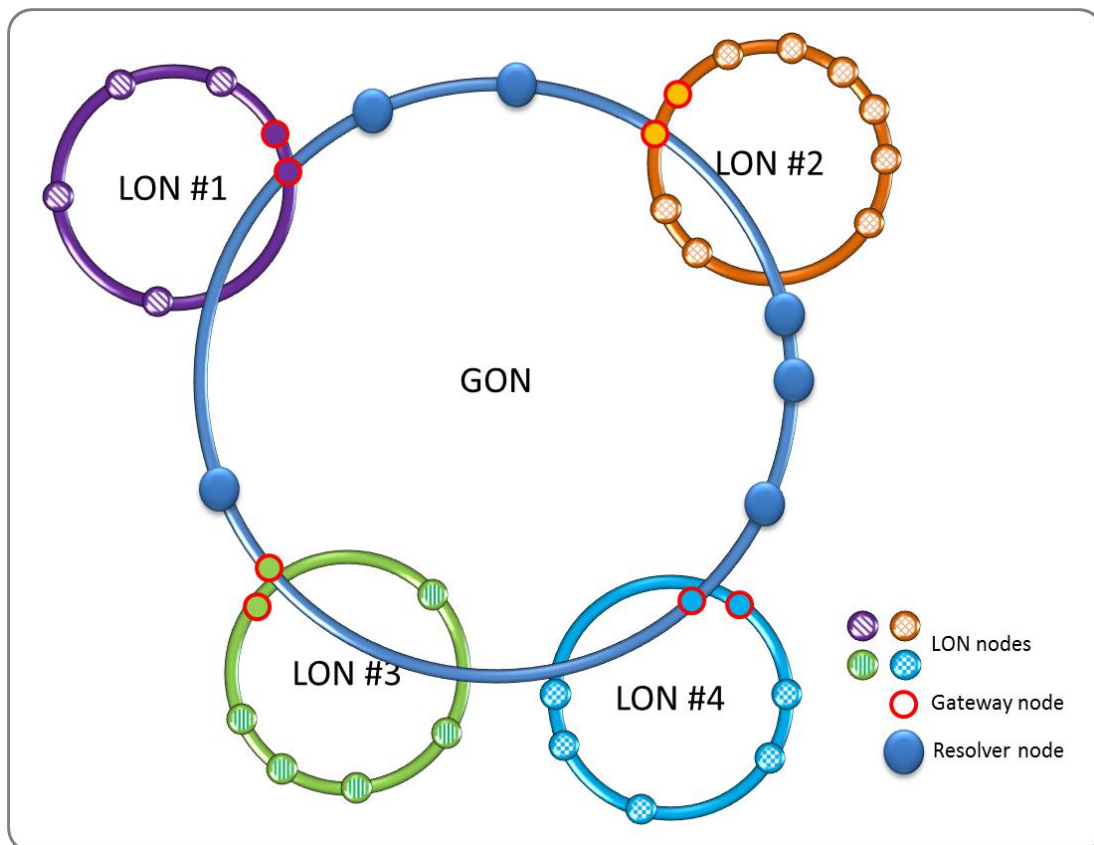


Figure 4.1: An example of 4 local overlay networks served by a global overlay network

4.1.1. Local Overlay Network (LON)

LON consists of nodes belonging to the blocked region and they are not expected to be able to resolve DNS queries from the legacy DNS system. The function of those nodes is to answer DNS requests coming from clients in the blocked region. To answer DNS requests, they need to have the answer in the cache or direct the request to the second layer which is GON. When the request is directed to the second layer, the answer comes back to LON in which the home node for the request will cache the answer and send a copy to the client. Figure 4.2 shows the flowchart of the mentioned lookup process in LON.

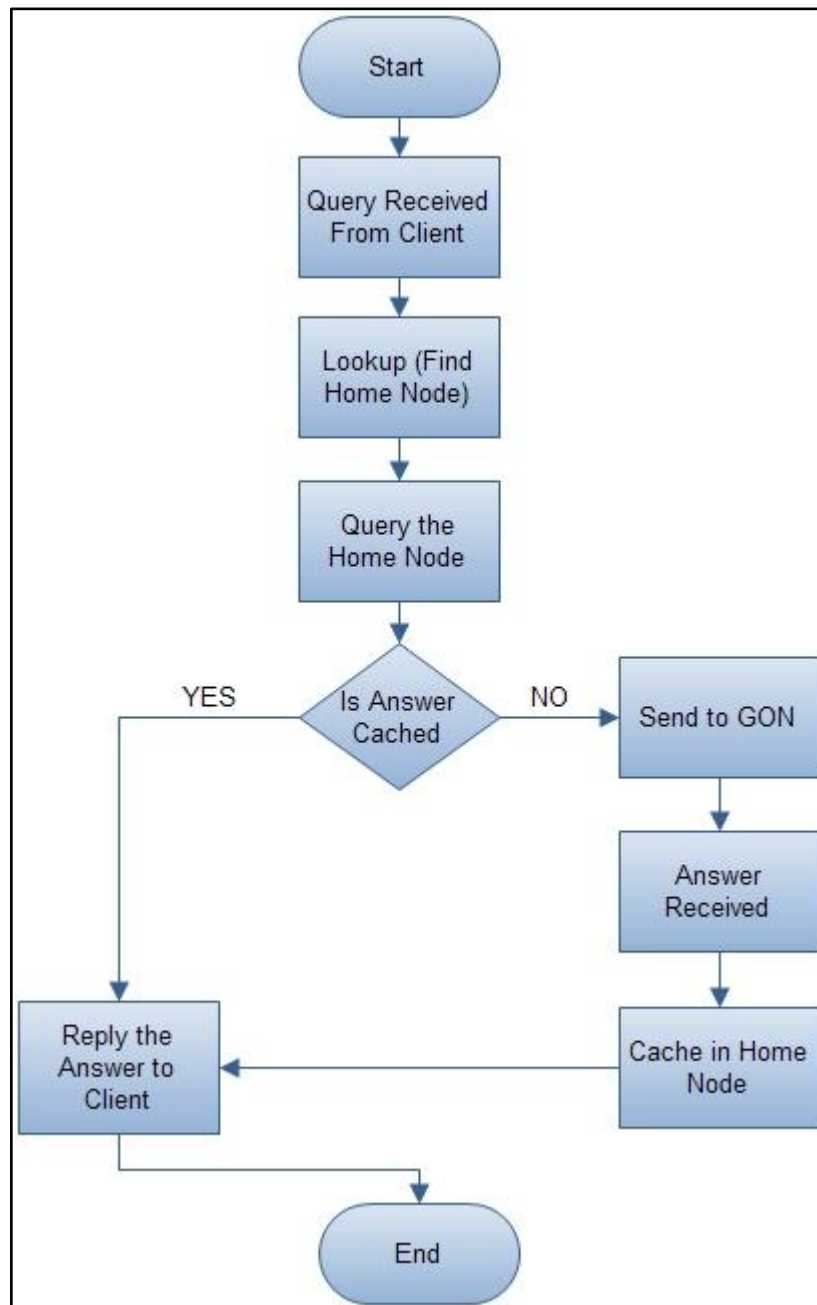


Figure 4.2: Local Overlay Network Flowchart

4.1.2. Global Overlay Network (GON)

The nodes in this layer are of two types, resolver nodes and gateway nodes. Resolver nodes' main job is to resolve DNS queries coming from LON by forwarding

the requests to the legacy DNS system. They are expected to be able to resolve DNS queries through the existing DNS system. The other type of nodes is the gateway nodes which participate in both layers and as they are part of LON then they are not expected to be resolving DNS queries from legacy DNS system. They work as a gateway between the two layers and their existence in GON is mandatory to direct unresolved queries from LON to GON. Their other function is acting as the authoritative DNS servers for all authoritative DNS servers in their region. This will enable clients to resolve DNS names for authoritative servers in the blocked region. GON serves multiple LONs and has at least one gateway node from each LON. When a query comes from LON, it will be passed through the gateway node for that LON. A lookup process will find the home node for the request which will answer either from its cache or the legacy DNS system. The home node in this case has to be a resolver node, otherwise the request needs to be routed to the first successor resolver node, see Figure 4.3 for the steps as a flowchart to resolve query coming from LON.

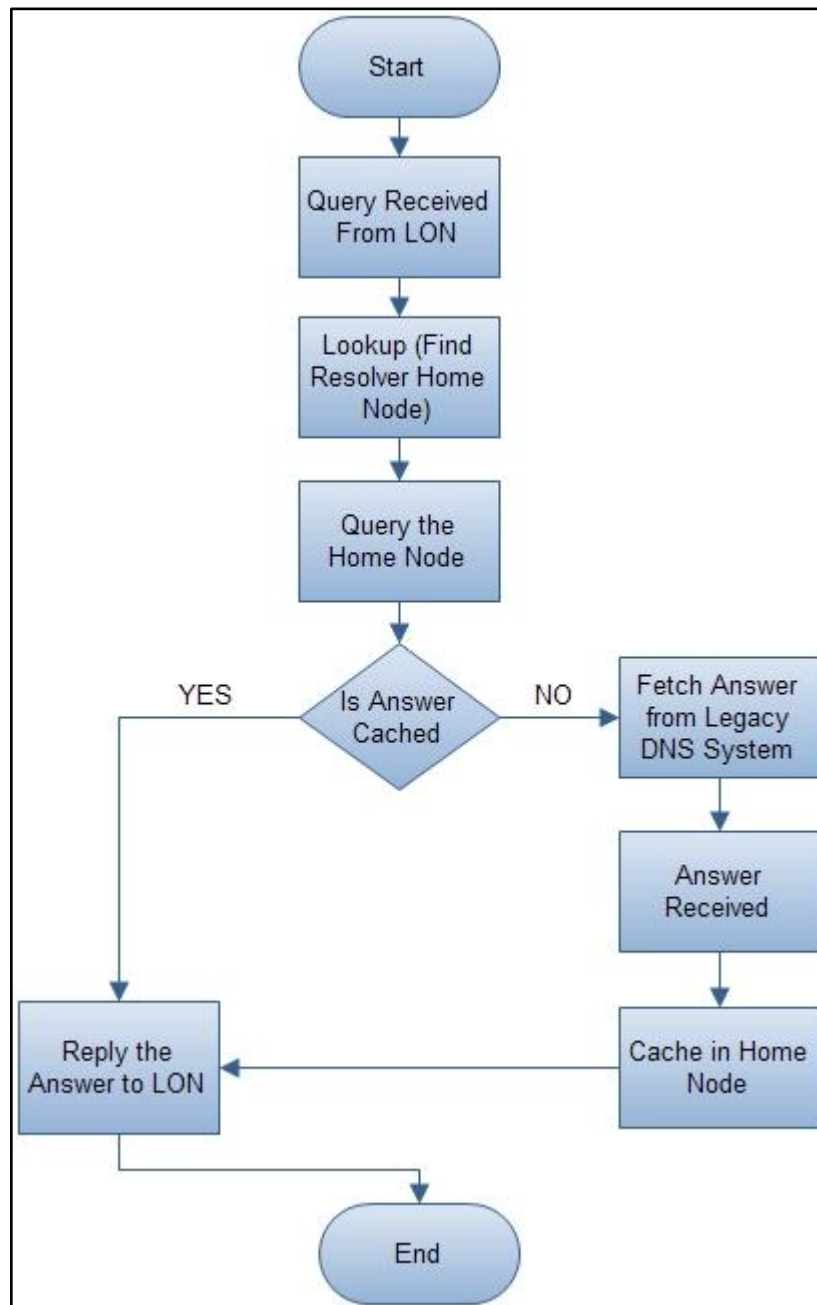


Figure 4.3: Global Overlay Network Flowchart

4.1.3. Cached Records Expiration

To ensure the availability of the answers to clients' request, the behavior of peers in GON with respect to DNS records will depend on the source of the answer.

The DNS answers can be coming from the legacy DNS system or from an authoritative DNS server in GON or LON. When answers are received from the legacy DNS system, GON caches them with their TTL values. While if the authoritative DNS server is part of GON or LON, then all its records are stored in GON without their TTL value. With this way, any changes to a record will trigger the authoritative DNS server to publish the changes to the home node of the record which will make the necessary actions to reflect the update. LON peers always store the TTL value unless the authoritative DNS server is in its LON. In the case the peer in LON stores the TTL value, it will prioritize the records based on the number of requests to determine the updating mechanism for cached records. Cached records with high priority will be updated with a proactive cache which refreshes the records whenever they expire. On the other hand, lower priority cached records will be in the passive cache, in which an expired record will not be updated unless a client requests it. This will ensure faster update propagation of obsolete records to LON and GON, where only two peers need to be updated, and will reduce the traffic that can be caused with the expiration of the TTL while no change has occurred. Also, the proactive caching will enhance the response time for highly queried records.

4.1.4. Request Scenario

A scenario of a client who wants to resolve a name to IP will start by sending the DNS query to LON. Any peer can serve the request and start the process of the lookup. When the request reaches the home node, it will check the cache and if it

has an answer it will reply with the cached answer. Otherwise, the node will forward the request to the gateway node. The gateway node will start the lookup process to find the resolver home node for the request in the GON. Once the home node is found it will either reply with an answer from the cache if it exists or will direct the request to the legacy DNS system. For the second case, the home node would receive an answer from the legacy DNS system in the normal case. Then it will reply to the home node in LON which in turn will cache the answer and will forward a copy to the client.

It might be thought that when LON has only one node, TLP2P will behave like pure Chord. However, when LON has one node the lookup to find the home node is zero and the cache result is found immediately, whereas in the case of the Chord the lookup will be on average $\frac{1}{2} \log_2(N)$ to find the home nodes.

4.2. Advantages

Building the structure as two overlay networks has many advantages over the existing system. One of the gained advantages is more effective caching. It has been achieved by splitting the blocked areas by their geographical location into overlay networks, since people belonging to the same area will more likely have the same interest which will increase the possibility of accessing the same records. It is supported by the fact that members of the same culture or subculture will have similar behavior and activity in the Internet [11] [20]. As a result, the cache will have a higher hit rate and the effect over the whole system is better response time.

Also the peers with this design will be near each other which will have less communication latency between them. In addition, LON can increase cache utilization by implementing a replication algorithm to replicate DNS records on multiple peers, but it will not be discussed in this research.

It is also good to mention that splitting culture will limit the set size of DNS records allowing a small size memory to cache all favorite records. In other words if Least Recently Used caching algorithm is used then the algorithm most likely will not discard a frequently requested record with nowadays memory size in case of full cache. Along with these mentioned advantages LON size can be elastic based on the demand. For example, we can increase the number of peers when performance starts degrading or reduce the number of peers if the utilization is low.

In summary, our proposed solution has the following advantages:

1. **Self-managing structure:** the structure is based on peer-to-peer. The system automatically manages the joining and leaving nodes as well as the allocation of DNS records and locating them.
2. **Harder to detect the cooperating nodes by enemies:** the structure requires cooperating nodes to be known only by the GON peers which are gateway nodes and cooperating nodes. To identify the peers, the enemy must be in GON. In addition, Chord protocol has a good location hiding [8].
3. **More resilient to DoS attack:** in this design, there is no single point of failure. To halt the system, all the peers in the network need to be attacked and that will require large amount of resources to make the attack successful. In addition, to start the attack, the peers need to be known first which will not be an easy task.

If in any case the first layer has been attacked and the network became not accessible, the second layer can serve from the cache.

4. **Faster update to obsolete records:** records are cached in two locations which are GON and LON. Any update to the records will need to be reflected in these two locations only, which makes it easier and faster to propagate the update.
5. **All peers in TLP2P will be able to resolve hostnames belonging to the served blocked regions:** in case a region is blocked by the international Internet service provider, any DNS queries will not be processed by higher DNS servers. This includes resolution queries, registration of new DNS domain, and update requests. By that, all hostnames for the hosts in the blocked regions will be unresolvable. However, if the clients join TLP2P, then they will be able to resolve all the hostnames, because the authoritative DNS servers' will publish the updates within TLP2P as discussed earlier, and clients in TLP2P will be able to resolve the hostnames normally.
6. **Better utilization of the cache in LON which results in lower response time:** LONs are created based on the geographical locations and users belonging to the same location will more likely share the same interests, which increases the likelihood of using the same services on the Internet. This will limit the number of popular records and will increase cache hit rate resulting in lower response time.
7. **Different replication algorithm of popular records can be used in LON based on the culture activity:** a replication algorithm can be applied to

increase the hit rate and reduce the response time. The replication algorithm can be different from one LON to another to suit the needs.

8. **Virtual nodes can be used to distribute Loads evenly:** the new design uses Chord protocol which can achieve good load balance by introducing the virtual nodes in the Chord network, as discussed in Chord paper [10].
9. **No change to client side:** the client will use the same DNS protocol to resolve hostnames. The difference is in the resolution of the query which is transparent to the client.
10. **Advantages of the current DNS are inherited by our design:** there are many advantages in the current DNS system like the security layer DNSSEC and multiple IPs as a load balance approach to service providers. Any advantage that is not related to the design structure can be implemented in the new design with no or minimal modification.

Advantages 1 to 4 are inherited from Chord and the rest are based on the proposed solution.

4.3. Limitations

The main concern with the new design is the number of hops to resolve a query. Chord network has an average of $1/2 \log_2 N$ hops for lookups where N is the number of nodes in the network. However, the proposed solution consists of two overlay networks which can result in larger number of hops. To have control over

the number of hops, LON size should be kept small and this cannot be achieved without a full control over LON. Also, as the load increases, new peers need to join the network. As a solution, LON can be split into two smaller LONs, but it is not a simple task especially in an active system.

4.4. Assumptions

The assumptions for the proposed solution to function correctly are:

- There are some cooperating nodes which can resolve a DNS query on behalf of other nodes. In the simulation, the system was functioning in a good condition with only 100 resolvers out of 1000 nodes, that is 10% of the nodes were resolvers.
- Resolver nodes are trusted and are not behaving maliciously.

CHAPTER 5

SIMULATOR

This chapter discusses the implementation of the simulator to evaluate the proposed structure. It also includes simulator results of Chord protocol to verify its correctness by comparing results obtained with results presented in Chord paper [10].

5.1. Simulator's Design

The simulator is developed with Java programming language. The approach used for lookup is recursive lookup in which a received lookup will be forwarded to the next node based on the information in the finger table and the process is repeated till the home node is found. On the other hand, the simulator used in the Chord paper [10] is based on an iterative lookup. In the iterative lookup, when a node receives a request, it will communicate with other nodes to retrieve information from their finger table to find the home node. Each of the two methods should have the same result and the simulators should generate the same output as we don't consider the network latency in the simulation. [10]

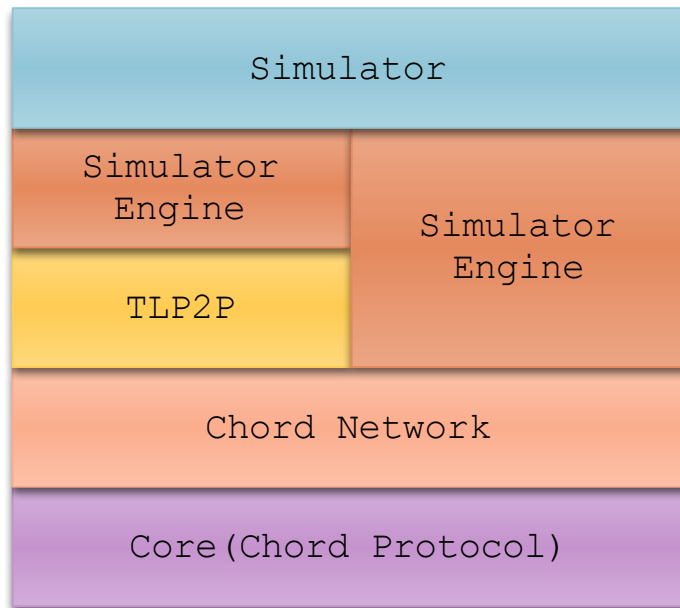


Figure 5.1: Simulator Design Layers. The two columns at the middle of the layer stack is for the two different structure of simulated networks (proposed solution and normal Chord)

The design of the developed simulator is made of five different layers as shown in the Figure 5.1 and these layers are composed of multiple classes. The core layer consists of the Chord protocol and part of it is the Chord algorithms like the stabilization algorithm and the lookup algorithm. The second layer creates the Chord network by initializing the Chord nodes first and then linking them by building the finger and successor tables as well as the other parameters. The third layer extends the simulator capability to have another overlay network acting as the global overlay network when simulating the proposed solution. However, when simulating the normal Chord network, this layer is skipped to the next layer. The next layer is the simulator engine which consists of random generator, query generator, queue manager, and the processor. The random generators follow different distributions and there are three different generators for each, i.e.,

uniform, Poisson, and Zipf. The query generator part generates random queries using a different random generator depending on the simulation scenario. When queries are generated, they are filled in a queue and only one global queue for all nodes exists in the simulator. The queue is managed by the queue manager which also manages the stabilization queue. The stabilization queue is for stabilization algorithms and it is invoked at every stabilization period which is stored in the request state of the stabilization request. The last part of this layer deals with request processing, like lookup request and stabilization requests. The top layer in the design is the simulator layer and it is the layer responsible to start the simulation runs. It initializes the other layers based on the scenario. Once the simulator is started, collecting measures starts till the completion of the simulation.

5.2. Simulator's Assumption

There are three assumptions while designing the simulator and they are:

- No TTL value for DNS names.
- No query processing time.
- Only one query is resolved at a time.

5.3. Simulation Measures

5.3.1. Load Balance

To study how keys are distributed over nodes with a consistent hashing (SHA), a number of simulations are carried out with different number of keys while having the same number of nodes. The number of nodes in the network is 10,000 and the number of keys varied starting from 10,000 to 1000,000 with an increment of 10,000. The size of bit identifier m is 160 and it is same for all scenarios. As the simulation in Chord [10], for each simulation setting, the simulation was repeated 19 times generating a total of 200 outputs. The measure collected were the number of keys resolved by each node.

5.3.2. Path Length

The number of hops is one of the main factors affecting Chord protocol performance. Resolving queries in Chord network depends on distributed hashing tables. This requires query lookup to traverse through multiple nodes and depending on 1) the starting node, 2) the state of the intermediate nodes and their hashing table, and 3) the location of the home node in the Chord network, the number of hops differs from one lookup to another. To find the effect of each of network size, node failures, and node join/leave on path length, multiple of simulations are performed.

5.3.2.1. Varying Network Size

For the case of network size, the simulation starts with a network consisting of 8 nodes and 800 lookups were generated. Doubling for each simulation the number of nodes and lookups to reach to 2^{14} and 100×2^{14} lookups. For each value, a separate simulation was implemented and repeated for 19 times. Simulation inputs are the number of nodes which equals to 2^k and the number of keys 100×2^k , where $k = [3, 14]$. The output of those simulations is the path lengths of the lookups.

5.3.2.2. Simulation with Nodes Failure

Failure simulation is performed to experiment the effect of changing the node status from active to failed on the path length. The simulations start with a network consisting of 1000 nodes in a stable state. That is, all nodes are active and the finger tables point to the correct nodes as well as the successor list which is of size 20. Then, random nodes failed simultaneously with a probability of 10% with increment of 10% in the rest of the simulations up to 50% for the last. To evaluate the impact of massive node failures, the stabilization algorithms are disabled and 10000 lookups were issued randomly. At the start of node failures, none of the active nodes knows about any failed node. But while resolving lookups active nodes will try to forward the request to the next nodes based on their finger table. If the next node is a failed node, then the communication will time out indicating a failed node and resulting in removing the failed node from the node's finger table. The

output of these simulations is the path length which will include the time out as well. For each of the 5 failing probabilities scenarios, a separate simulation was performed and was repeated for 4 times.

5.3.2.3. Simulation with Nodes Joining and Leaving

The last case of simulation is the excessive joining and leaving of nodes while resolving lookups. The simulator configuration is as follows:

- Network size is 1000.
- Successor list size is 20.
- Lookups generated with Poisson process of a rate one per second.
- Joining and leaving rate follow Poisson distribution.
- 8 different simulations with different rates. The simulations start with 0.05 'join' and 'leave' per second and the last simulation has a 'join' and 'leave' rate of 0.40. The increment in each simulation is 0.05.
- Nodes run the stabilization algorithm at a uniform time interval [15, 45].

Collected outputs are path length (hops + timeout), timeouts, and failed lookups. The simulation scenarios have been repeated for 4 times.

5.3.2.4. Simulation with Node Blockage

For the purpose of testing the proposed solution, an additional scenario will be simulated to find the effectiveness of TLP2P in the case of resolver nodes in GON got blocked. The scenario will measure the path length and the load balance while a

portion of the nodes are blocked at the start of the simulation. The number of blocked nodes will vary in each simulation by 10%, starting in the first simulation with 10% of the nodes blocked and ending with 90%. The number of nodes in the network is 1000 and the number of lookups generated is 10,000.

5.4. Performance Metrics

The simulation results will discuss five different measures that will be explained in this section. Some of these measures are calculated on GON and LON level as well as the total of the two whenever applicable. In all measures the 1st and 99th percentiles are calculated too. The pth percentile is the value that falls at the p% of the result set.

1. **Load Balance:** has two measures

- Number of records stored per node:

$$\text{Average Load} = \frac{\sum_{i=1}^n \text{number of resolved requests } i}{n}, \text{ where } n =$$

number of nodes and i represents nodes.

The 1st and 99th percentiles presented with the average load in the simulation results, will indicate how balanced the load is.

- Load fairness between nodes. The resulted value by evaluating the equation can be in the range of 0 to 1. Closer value to 1 indicates better load fairness among the nodes.

$$\text{Fairness} = \frac{(\sum N_i)^2}{\text{count} * \sum (N_i^2)}, \text{ where } N_i \text{ is the number of keys resolved}$$

by the i^{th} node and 'count' is the number of nodes.

2. **Path length:** is the number of nodes hopped for the request to find the home node.

$$\text{Path length} = \frac{\sum_{i=1}^n \text{number of hops } i}{n}, \text{ where } n = \text{Total number of}$$

requests and i represents requests.

For the scenarios where there are failing or departing nodes, the path length accounts for timeout requests as well. So, it will be:

$$\text{Path length} = \frac{\sum_{i=1}^n (\text{number of hops} + \text{number of timeout})_i}{n}$$

3. **Timeout:** is the number of times a node is trying to connect to a non-existing node (failed or left the network) while resolving a request.

$$\text{Timeout} = \frac{\sum_{i=1}^n \text{number of timeouts } i}{n}, \text{ where } n = \text{Total number of}$$

requests and i represents requests.

4. **Lookup Failure:** is the number of mapped lookups to an incorrect home node.

$$\text{Lookup Failure} = \frac{\text{Total number of failed lookups}}{10,000}.$$

5. **Hit Count:** is the number of resolved lookups from the cache. When the lookup is resolved at LON without going to GON, it is considered as a hit.

Hit Count = Total number of hits.

5.5. Simulator's Validation

The four simulation scenarios in Chord paper will be simulated using the implemented simulator. Each of the following subsections will discuss and compare results obtained with results presented in Chord [10].

5.5.1. Load Balance

Figure 5.2 shows the comparison of the two results, the Chord paper and implemented simulator. It is clear from the graph that both simulators have the same results.

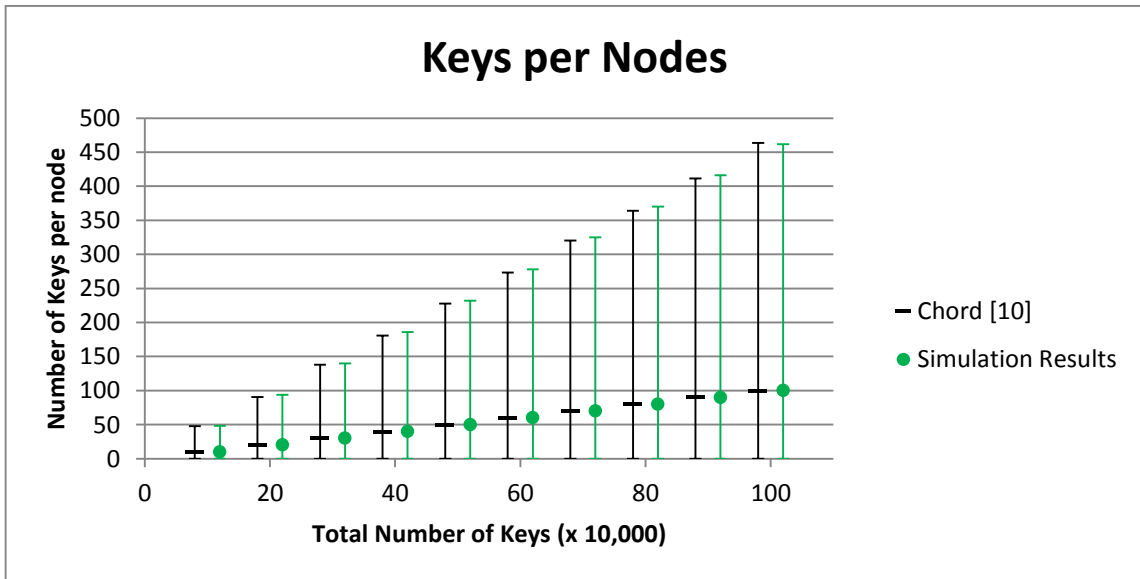


Figure 5.2: The mean, 1st and 99th percentiles of the number of keys stored per node in 10,000 nodes network

5.5.2. Path Length

In path length scenario, the outputs are identical as shown in Figure 5.3. The difference in percentiles is normal and it is due to the randomness of inputs to the simulations.

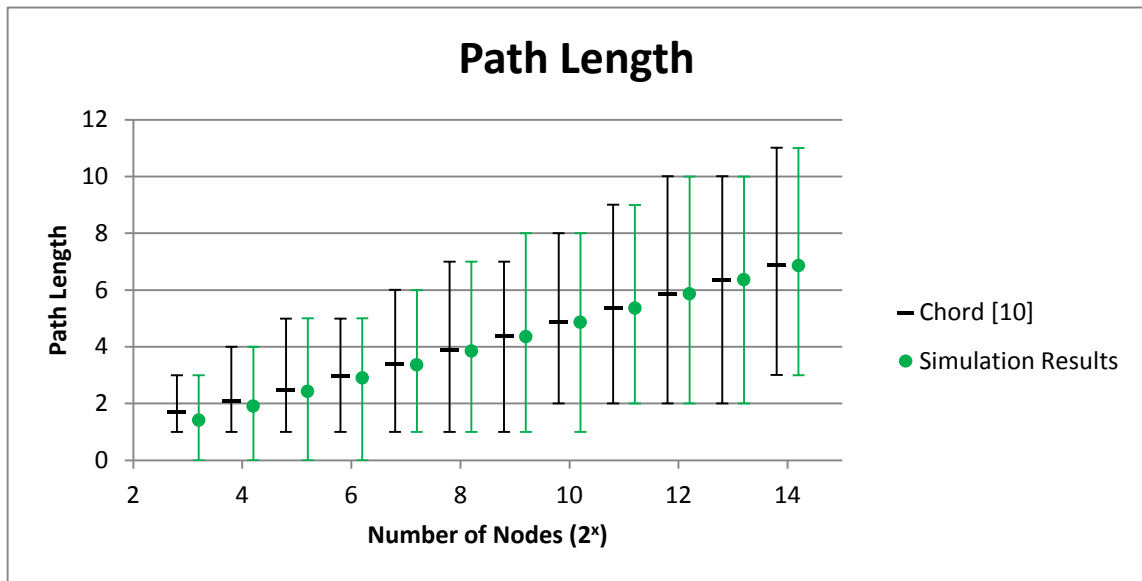


Figure 5.3: Average path length, 1st and 99th percentiles of lookups with varying number of nodes

The proof of the average path length has been explained in Chord [10], but a detailed explanation will be presented here as well since in the next chapter, we will depend on this result to create the simulation scenarios.

Proof of average path length

Let node n generate a random lookup for a key k . Recall that each node has a finger table and each entry i stores the home node of $n + 2^i$.

Table 5-1: Finger table for node n

i	Home node of	Home node must be in
1	$n + 2^0$	$[n + 2^0, n)$
2	$n + 2^1$	$[n + 2^1, n)$
3	$n + 2^2$	$[n + 2^2, n)$
⋮	⋮	⋮
⋮	⋮	⋮
m	$n + 2^{m-1}$	$[n + 2^{m-1}, n)$

With this information, let us find the worst case where $k \in (n + 2^{m-1}, n)$. With this condition, n will forward the request to the home of $(n + 2^{m-1})$ and let us call it n_1 . As $k \in (n + 2^{m-1}, n)$, node n_1 will not forward the request to the home node of $(n_1 + 2^{m-1})$, which is finger number m , because finger m will cover the interval $[n, n_1)$. In other words, it covers the interval $[n, n + 2^{m-1})$ and this interval does not overlap with key interval in the first assumption which is $(n + 2^{m-1}, n)$. So, for the worst case scenario, node n_1 will forward the query to the home node of finger number $(m - 1)$. The process is repeated by forwarding the received query to the next node, which is the finger $m - 2$ of the receiving node. In summary, the request will be forwarded as follows: node n (finger m) \rightarrow node n_1 (finger $m - 1$) \rightarrow node n_2 (finger $m - 2$) \rightarrow node n_{m-1} (finger 1 = successor) \Rightarrow the number of

hops for worst case scenario is $m - 1$. From this result, we can say that, after x hops, there are at most $2^m/2^x - 1$ nodes to reach the home node. In case of a network with N uniformly distributed random nodes, there should be only one node after hopping $2^m/N - 1$ nodes with a high probability. In other words, after $\log_2(N)$ hops, there is only one node to reach the home node with a high probability. For the average path length, it equals to $(\text{maximum} + \text{minimum})/2$, because the nodes are distributed over Chord ring based on SHA algorithm which follows uniform distribution as well as the key and assigned node are uniformly random. The minimum path length is for the case where the home node is the successor node of the requester which does not require any hop and the maximum as found is $\log_2(N)$. So, the average path length = $\frac{1}{2} \log_2(N)$. The results in Figure 5.3 show that the average path length is the same as the proofed theoretical average.

Another proof of the average path length is using the difference in bits between node hash identifier and key identifier. Let k be the key that is queried to find its home node. Depending on the difference between the node and the key in binary, the node will determine which finger to follow. Let us assume the most significant bit is i . If i is 1, then the lookup will be forwarded to the home node of the i^{th} finger and by that the 1 is flipped to 0. The other case is where the i^{th} bit is 0, then no forward is performed at this stage, rather the next bit is examined. The process goes again for the next bit and the same process is repeated till the home node is reached. This means that the number of hops (path length) is the number of ones in binary that differ between home node identifier and the key. Since the node identifiers are distributed randomly, then the expected number of ones are half of

the bit on average \Rightarrow average path length = $\frac{1}{2} \log_2(N)$. The part $\log_2(N)$ came from the number of bits to test.

5.5.3. Simultaneous Node Failures

In Figure 5.4, the path length is shown for both simulators for the node failure scenario. Again, both simulators have the same average path length and the difference in percentile is due to randomness of the inputs.

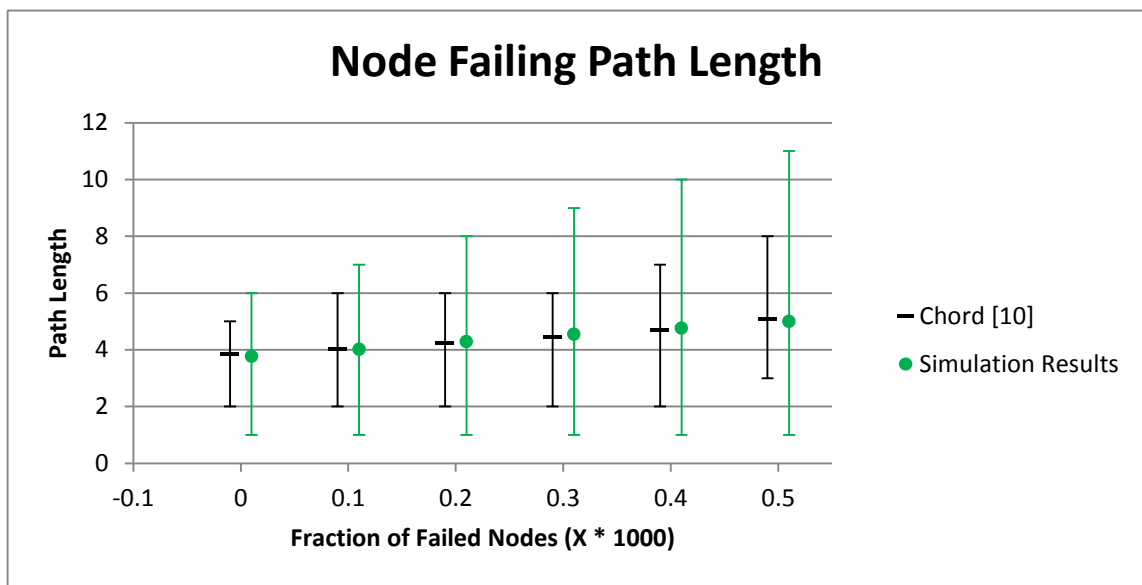


Figure 5.4: Average path length, 1st and 99th percentiles of lookups with random node failed before start resolving lookups

5.5.4. Lookups during Stabilization

In this scenario, the measures have three results which are the path length, timeout, and failed lookups. In Figure 5.5 and Figure 5.6, the outputs look the same

for the two simulators and the minor difference in 99th percentile is expected and it is as stated before because of the random query and nodes. The output presented in Figure 5.7 is for the number of failed lookups. It is unlike the other figures, and it presents the average number of failed requests with the minimum and maximum values for the different runs. The results shows a difference between the two simulators outputs, but also the same difference appears in the simulator developed in [5] which is presented in [5] Figure 6.10. The reason of the dissimilarity is the randomness of four random variables that account into the lookup failure. To know these random variables, it would be better to understand first how a lookup is resolved to an incorrect home node. A failed lookup occurs when the predecessor (n_p) of the home node (n_h) receive the request and n_h has just joined while n_p is not aware of the newly joining home node. In such a case, n_p will map falsely to the successor of n_h as the home node of the request, which results in a failed lookup. The four random variables in this case are the request, time of the request, the home node, and the time of joining. The inconsistency in the failed lookup outputs can also be noticed in Chord simulator output, the rate 0.40 has a lower number of failed lookups than the rate 0.35, while the expectation of a higher join and leave rate is a higher number of failed requests.

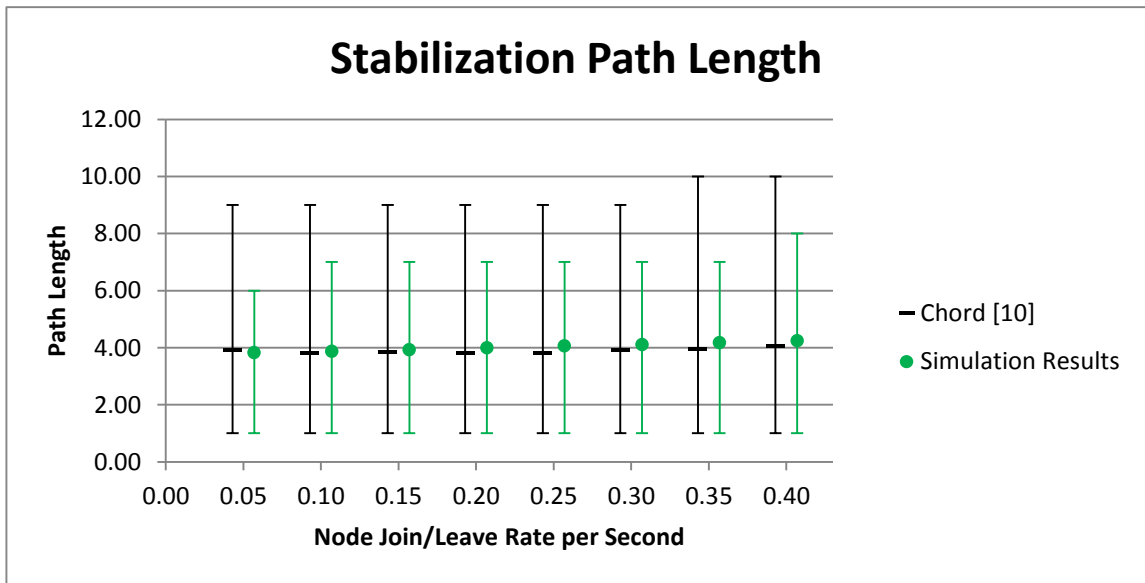


Figure 5.5: Average path length, 1st and 99th percentiles of lookups while node joining and leaving the network

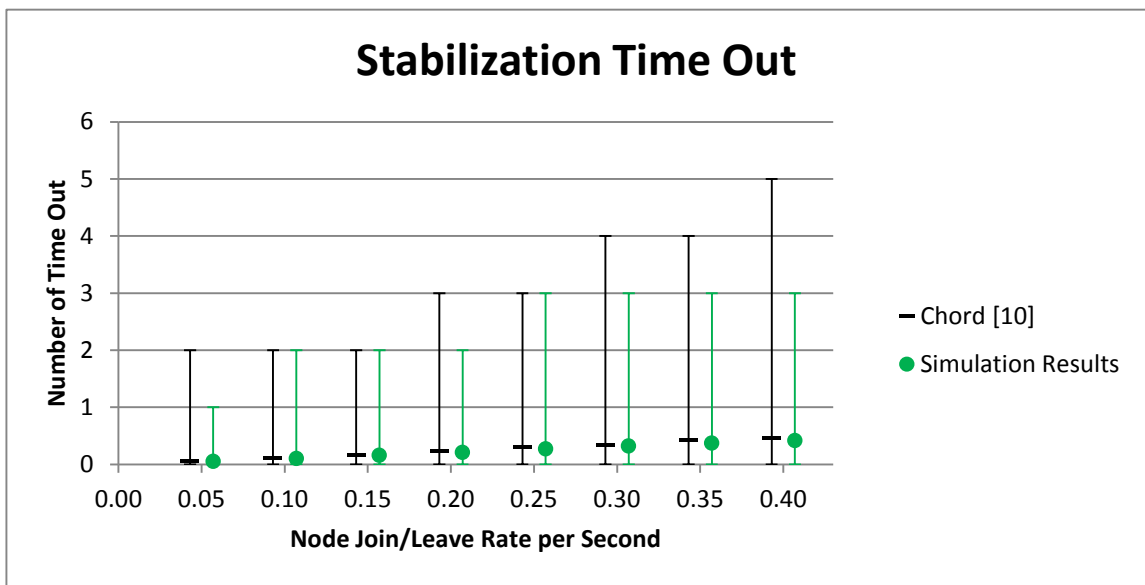


Figure 5.6: Average, 1st and 99th percentiles of timed out requests while nodes leaving and joining the Chord network

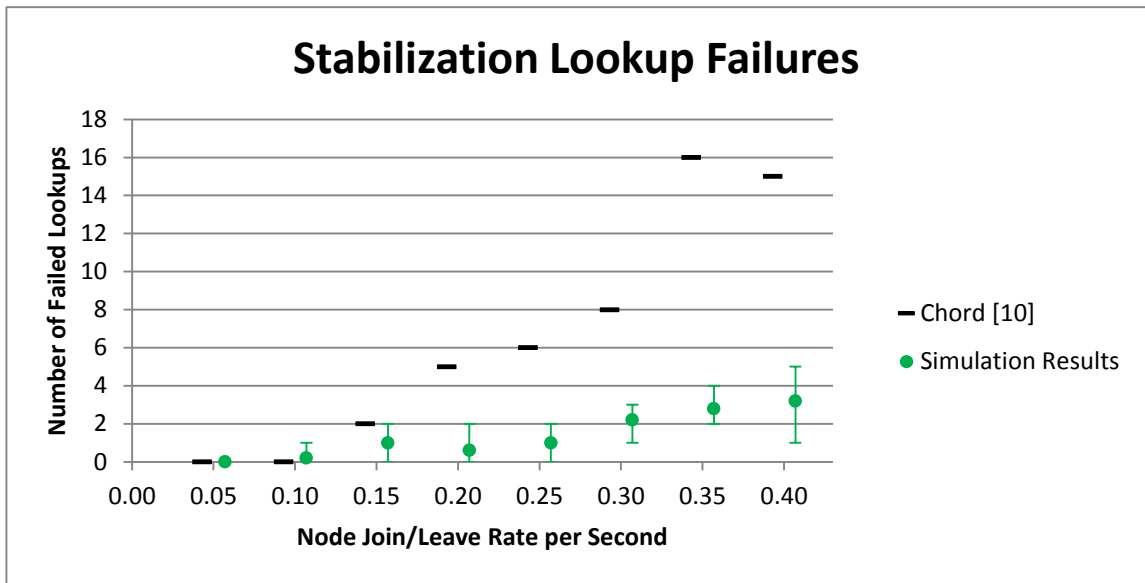


Figure 5.7: Average, minimum and maximum of failed lookups while nodes leaving and joining the Chord network

The implemented simulator has almost an exact output compared to the Chord simulator in all cases, except for some difference in one output which is acceptable due to the nature of the inputs as explained in the previous paragraph. The simulator will be used to implement the proposed solution to study the behavior of the new design with the same simulation scenario ran in this chapter.

CHAPTER 6

SIMULATION AND RESULTS ANALYSIS

Simulation results will be presented in this chapter for the proposed solution. Three different scenarios are simulated with different requests distributions. Out of the three, there are two worst cases, one with request repetition and the other with no repetition. The third case scenario is for the best case with request repetition.

6.1. Simulation Scenarios for Proposed Structure

In the designed structure, a GON can serve multiple LONs, however in all simulation scenarios there will be one GON serving only one LON. This is a limitation due to the time and memory required for simulation. To study the effectiveness of the solution, it will be compared with normal Chord network. The same nodes number in normal Chord network simulations will be distributed over GON and LON in TLP2P simulation. The aim is to have a fair comparison between the two designs. In the nodes distribution the consideration is to form a worst case scenario while the second modeling the best case scenario. As stated before, the path length is the main factor determining the performance of the design and hence the node distribution will be evaluated to find the highest path length.

To find the highest path length lets define some variables:

- n : number of nodes in LON
- N : number of nodes in GON
- t : total number of nodes in LON and GON
- p : average path length. Path length for LON and GON

With this information

$$\therefore t = n + N$$

From last chapter, the path length for TLP2P should be:

$$\begin{aligned} p &= \frac{1}{2} \log_2(n) + \frac{1}{2} \log_2(N) \\ &= \frac{1}{2} \log_2(n) + \frac{1}{2} \log_2(t-n) \\ &= \frac{1}{2} \frac{\ln(n)}{\ln 2} + \frac{1}{2} \frac{\ln(t-n)}{\ln 2} \end{aligned}$$

The derivative of p :

$$\begin{aligned} \frac{dp}{dn} &= \frac{1}{2} \frac{1}{n} + \frac{1}{2} \frac{-1}{t-n} \\ &= \frac{1}{2 \ln(2)n} - \frac{1}{2 \ln(2)(t-n)} \end{aligned}$$

To find the global maxima, first find the critical point by solving the derivative for zero.

$$\begin{aligned} \frac{dp}{dn} &= 0 \\ \frac{1}{2 \ln(2)n} - \frac{1}{2 \ln(2)(t-n)} &= 0 \\ \frac{1}{n} &= \frac{1}{t-n} \Rightarrow n = t - n \end{aligned}$$

$$\therefore n = \frac{t}{2} \quad (\text{critical point})$$

Now the critical point needs to be examined to find out whether it is global maxima or global minima. For that the second derivative need to be found.

$$\begin{aligned} \frac{d^2p}{dn^2} &= \frac{-n^{-2}}{2 \ln 2} - \frac{(t-n)^{-2}}{2 \ln 2} \\ &= \frac{-1}{2 \ln 2} (n^{-2} + (t-n)^{-2}) \end{aligned}$$

Evaluate the critical point in the second derivative

$$\begin{aligned} \frac{d^2p}{dn^2} \left(\frac{t}{n} \right) &= \frac{-1}{2 \ln 2} \left(\left(\frac{t}{2} \right)^{-2} + \left(t - \frac{t}{2} \right)^{-2} \right) \\ &= \frac{-1}{2 \ln 2} \left(\left(\frac{t}{2} \right)^{-2} + \left(\frac{t}{2} \right)^{-2} \right) \\ &= \frac{-1}{\ln 2} \left(\frac{t}{2} \right)^{-2} \end{aligned}$$

The result is negative which indicates it is a global maxima. So the highest path length occurs when LON and GON size is $t/2$. Now let us find the local minima. First, the critical point is found as a global maxima. The other points to check are the boundary points. The interval of the network size for GON and LON is $[1, t - 1]$.

$$\begin{aligned} p \left(\frac{t}{2} \right) &= \frac{1}{2} \frac{\ln \left(\frac{t}{2} \right)}{\ln 2} + \frac{1}{2} \frac{\ln \left(t - \frac{t}{2} \right)}{\ln 2} \\ &= \frac{\ln \left(\frac{t}{2} \right)}{\ln 2} = \frac{\ln(t) - \ln(2)}{\ln 2} \end{aligned}$$

$$\begin{aligned} p(1) &= \frac{1}{2} \frac{\ln(1)}{\ln 2} + \frac{1}{2} \frac{\ln(t-1)}{\ln 2} \\ &= \frac{1}{2} \frac{\ln(t-1)}{\ln 2} \end{aligned}$$

$$\begin{aligned}
p(t-1) &= \frac{1}{2} \frac{\ln(t-1)}{\ln 2} + \frac{1}{2} \frac{\ln(t-(t-1))}{\ln 2} \\
&= \frac{1}{2} \frac{\ln(t-1)}{\ln 2}
\end{aligned}$$

$$\therefore p\left(\frac{t}{2}\right) > p(1) = p(t-1)$$

By this result, the local minima is at 1 and $(t - 1)$. In other words, the minimum path length will occur in case of number of LON's nodes is 1 or $t - 1$.

To ensure the results are as found, Figure 6.1 shows the plot of path length which is in the Z-axis and LON and GON size in X and Y-axis respectively. The plot illustrates the points and gives a view of how the path length changes as the number of nodes changes in LON and GON. The shape of the plot is almost half a circle having the highest point in the middle and decreases toward the edges of the interval. In this plot, the number of nodes as total is 100 and the interval for both networks is $[1, 99]$. The plot supports the results obtained for the global maxima and local minima.

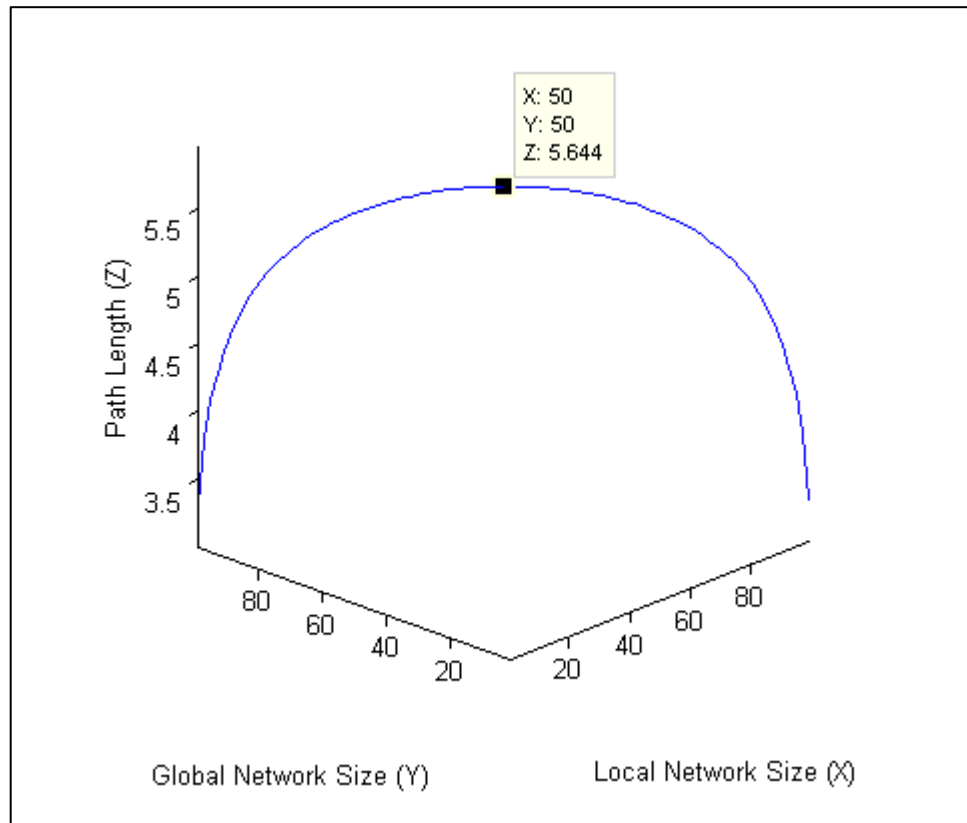


Figure 6.1: Path length in term of network size, $p = \frac{1}{2} \log_2(n) + \frac{1}{2} \log_2(N)$. The total number of nodes is 100 and the point shown represents the heighest path length when GON and LON size is 100/2 and lowest path length when either network having only one node.

The simulation scenarios are summarized in Table 6-1 and they are:

- Scenario#1: 50% of total nodes will be distributed over the two overlay networks and it will be compared to normal Chord simulation. The request distribution will follow the uniform distribution and no request repetition.
- Scenario#2: Same as the first scenario but the request distribution will follow Zipf distribution which has a request repetition. The results will be compared with normal Chord simulation but with a Zipf distributed requests.
- Scenario#3: LON will contain 5% of the total number of nodes while GON will have 95%. The simulation again will be compared with normal Chord simulation that had Zipf distributed request as in the previous scenario.

For each of the simulation configuration the five scenarios stated earlier will be simulated and they are:

- Load Balance
- Path Length
- Node Failure
- Join & Leave
- Node Blockage

Table 6-1: Simulation configuration

Configuration	Request Distribution	
	<i>Uniform</i>	<i>Zipf</i>
<i>Chord</i>	✓	✓
<i>TLP2P Worst Case</i>	✓	✓
<i>TLP2P Best Case</i>		✓

The intention of simulating the TLP2P solution with uniform distributed request is to study how the TLP2P can utilize the cache to reduce the path length and thus the dependency on the legacy DNS system. For this reason, the simulations were run for only one simulations configuration that is uniform worst case.

The selection of Zipf distribution for the simulation was based on many researches. It has been proven by many measurements of real life traffic that human behaviors follow Zipf distribution. However, the number of users' communities and the size can make a difference in the distribution. Smaller communities follow Zipf-like distribution while if the communities are very large the distribution will be Zipf. In the simulated scenarios, the assumption is that the

system is serving a very large number of communities, and thus the Zipf distribution is the right one for these simulations [18].

The reason of simulating the best case scenario with 5% of nodes in LON and 95% in GON is due to the join & leave scenario where the number of leaving nodes is 12 per stabilization period. If the calculated best case is used in the simulation which is LON will have only one node then LON will be totally empty. For that, the size of LON is increased to meet the requirement for the join & leave scenario to run.

While the solution has mentioned the possibility of having a replication algorithm to enhance the lookup hops and also having virtual peers to increase the load balance between nodes, neither of these enhancements were implemented. Also, in all cases of the simulations, the nodes start with empty caching regardless of request distribution and requests repetition.

6.2. Simulation Results

6.2.1. Worst Case Scenario with no Request Repetition

6.2.1.1. Load Balance

The load is doubling by cutting the network size by half, in general the average load is represented by dividing the number of requests by the number of nodes in Chord network. Figure 6.2 also shows that the load in 99th percentile is doubling by dividing the size of the network by half. The results shown for LON and

GON in the figure have the same average number of keys per node since they have the exact setting and they are receiving the same query load. However, the average load in LON and GON networks differ from normal Chord network because Chord network has twice the size of GON and LON network. Figure 6.3 presents the load fairness between nodes for each of the three networks. The fairness is the same for all and it is almost 0.5. These results indicate that, pure Chord and TLP2P are identical in term of load fairness. For the coming two scenarios, the fairness results are not presented because all fairness results have almost the same output as Figure 6.3.

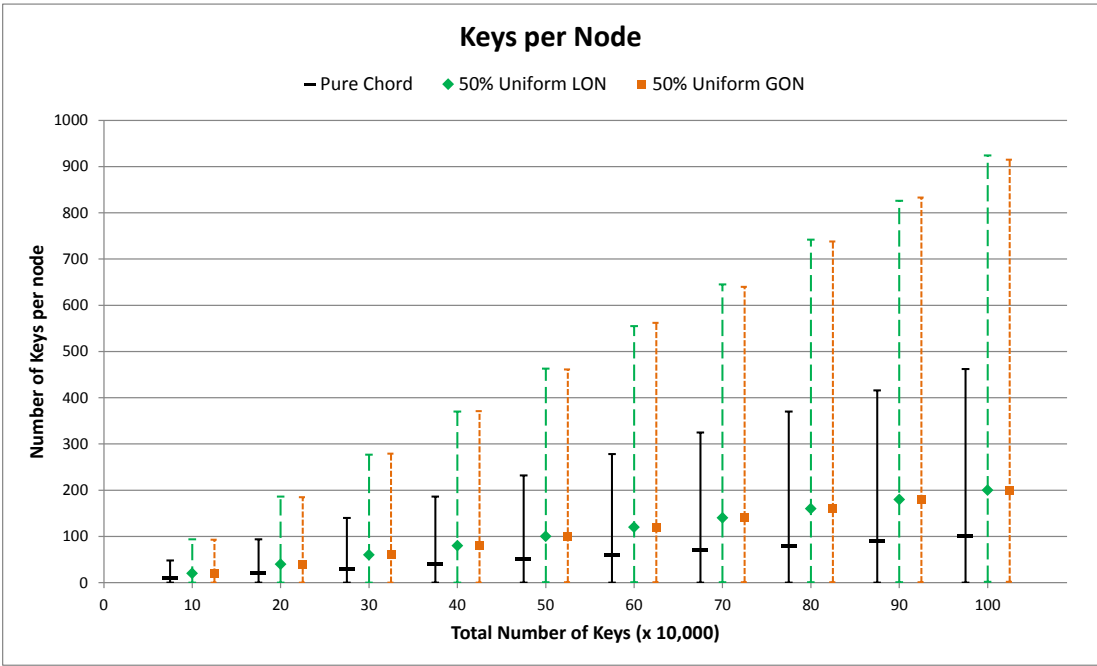


Figure 6.2: The plot of the average load with fix number of nodes and varying number of queries. The plot represents for each query load three results which are pure node, LON and GON.

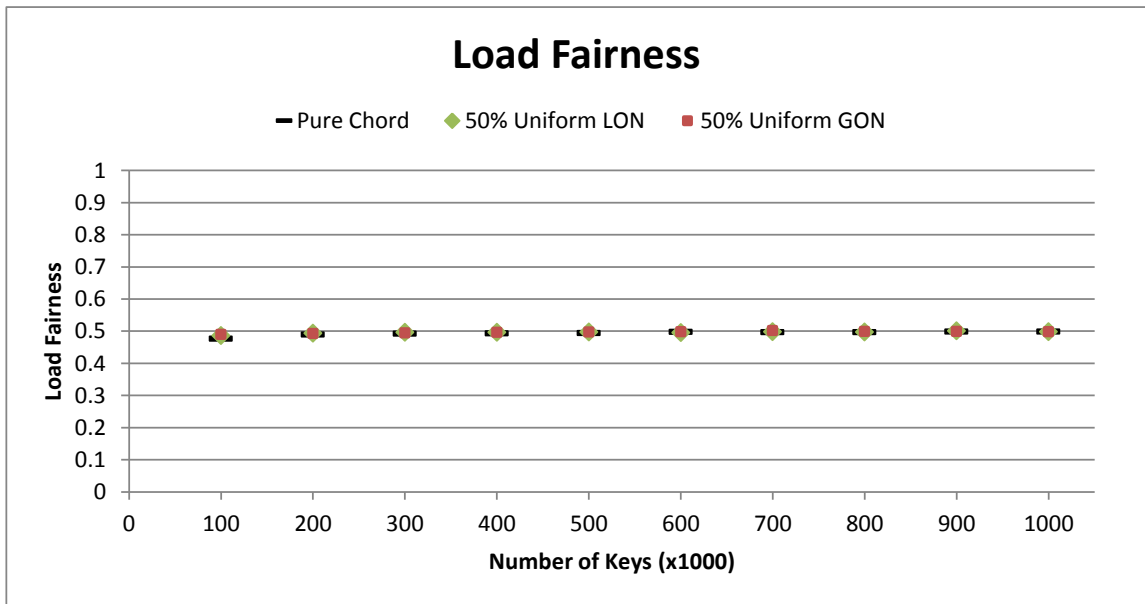


Figure 6.3: Load fairness results for the three networks normal Chord, GON and LON.

6.2.1.2. Path Length

Figure 6.4 displays the simulation results which show that the path length is also affected by the network size as discussed and proven earlier in this chapter. The results for LON and GON show both networks have equal average path length. This is due to the same configuration is being set for both networks and they have the same network size. Also, the same queries are passed from LON to GON which yield to same average path length. However, the difference occurs between the overlays networks in TLP2P and normal Chord which can be explained by the path length formula. The difference basically equals to $\frac{1}{2} \log_2(\text{Size of Normal Chord Network}) - \frac{1}{2} \log_2(\text{Size of LON})$, since the size of the network in normal Chord network is double the size of LON, the formula can be rewritten in this way $\frac{1}{2} \log_2(2N) - \frac{1}{2} \log_2(N)$ which equals to $\frac{1}{2} \log_2(2)$ which is 0.5. So the difference in path length between LON or GON and normal Chord scenario is 0.5, which is as the

case obtained in the simulation results. Of course, the total path length for the TLP2P network is the addition of the path length of LON and GON.

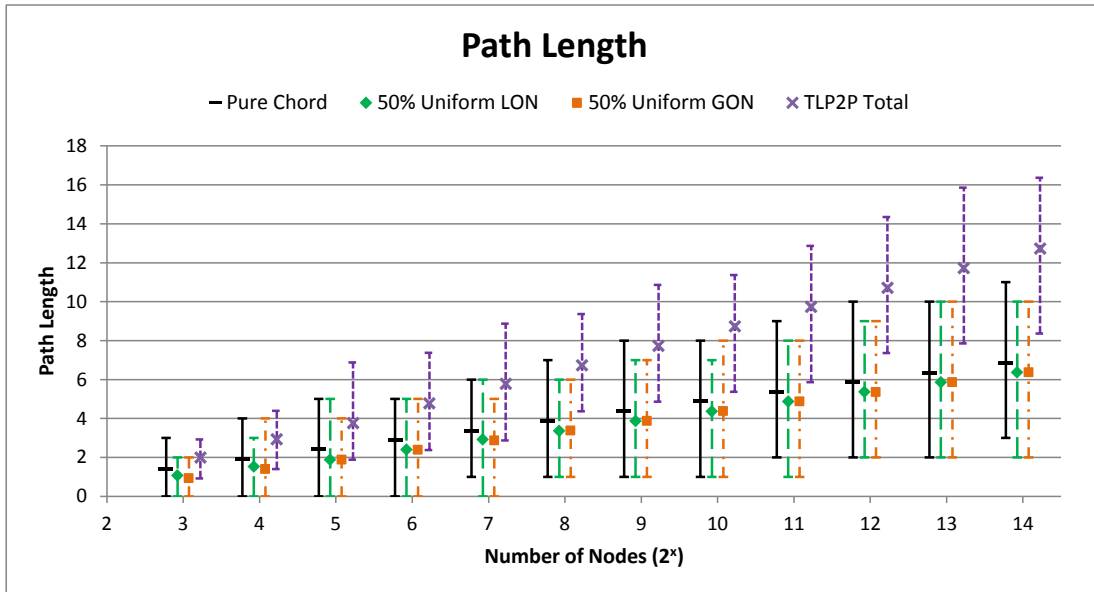


Figure 6.4: Path length measures for normal Chord simulation and LON and GON as well as the total in TLP2P simulation with different network size and query load.

6.2.1.3. Simultaneous Node Failures

In this scenario, the path length increases as the failure fraction is increased. The increase in both LON and GON is almost the same as the normal Chord network (see results in Figure 6.5).

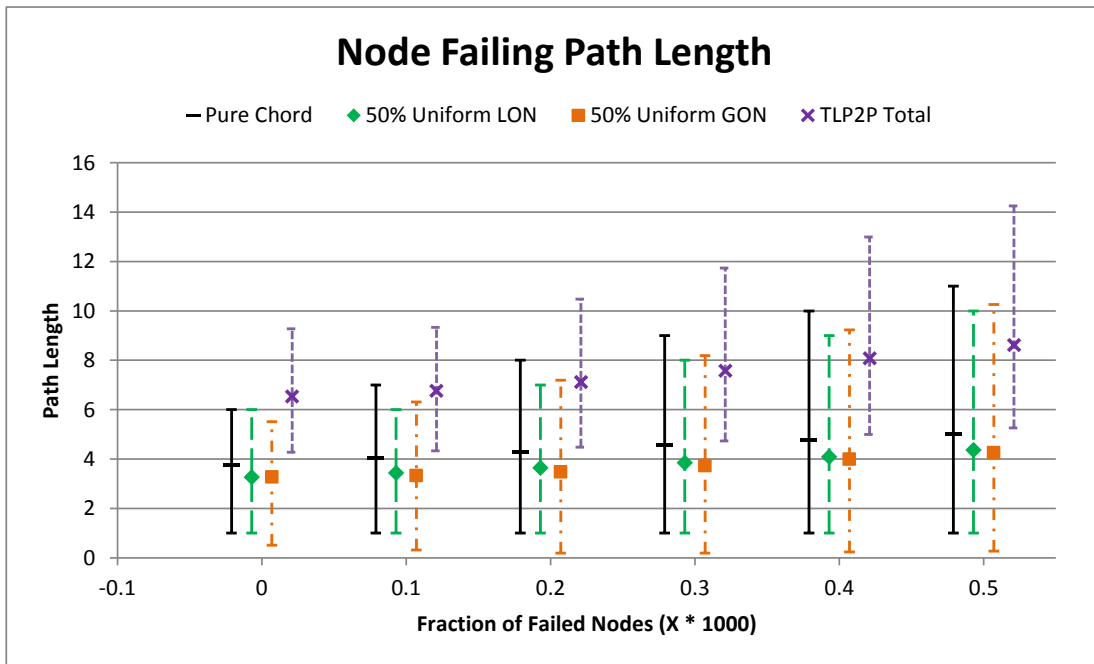


Figure 6.5: Path length measures for node failures scenario. The plot has four results normal Chord, LON, GON and the total for the TLP2P.

6.2.1.4. Lookups during Stabilization

The path length has a minor variation with increasing the 'join' and 'leave' rate. This is occurring for the two simulations and with about 0.2 higher in LON and GON than in the normal Chord network (see Figure 6.6). The average query failure in this scenario has different results for GON and LON as seen in Figure 6.7, which we would expect it to have the same results since they use the same setting and configuration. This can be explained by knowing the query lookup path. In LON, the queries are submitted randomly to any node in the network which makes queries travel with equal possibility through any node. While when the query cannot be resolved in LON, it is sent to GON through the gateway node. That is, the starting node in GON is always the gateway node for all queries, which makes any updates occurring in gateway's successor list or finger table to be reflected quickly. On the

other hands, other nodes have lower probability to update their finger and successor tables. This reduces the number of timeouts, but the consequent of un-updated finger and successor tables is larger failing requests as shown in Figure 6.8. This has been examined by re-simulating the same case but with submitting all the queries to only one starting node in LON and the results for all of the three measures were the same for GON and LON.

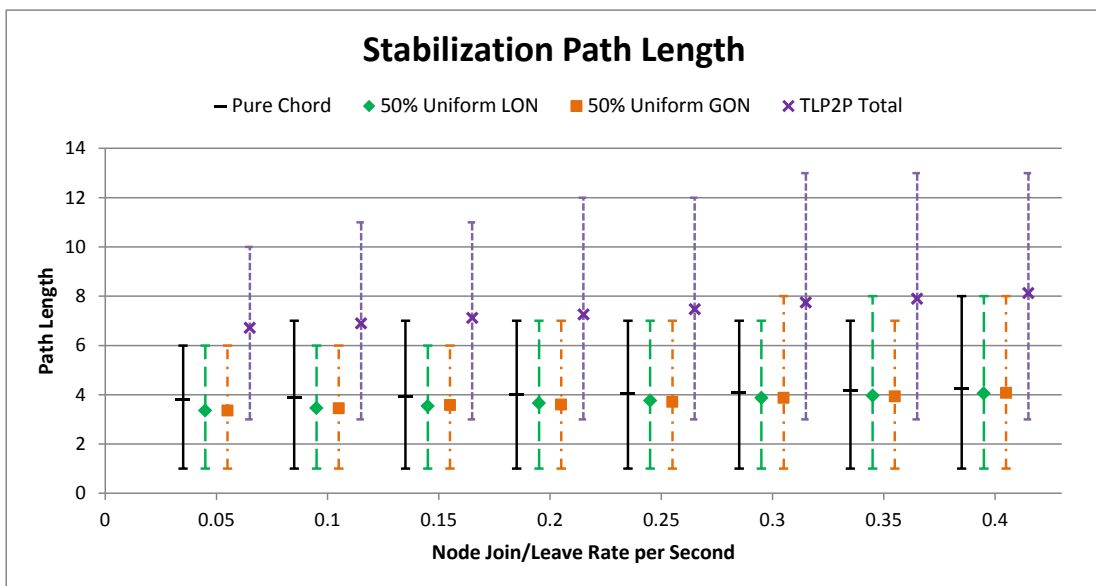


Figure 6.6: Path length for stabilization scenario comparing normal Chord and TLP2P (LON and GON).

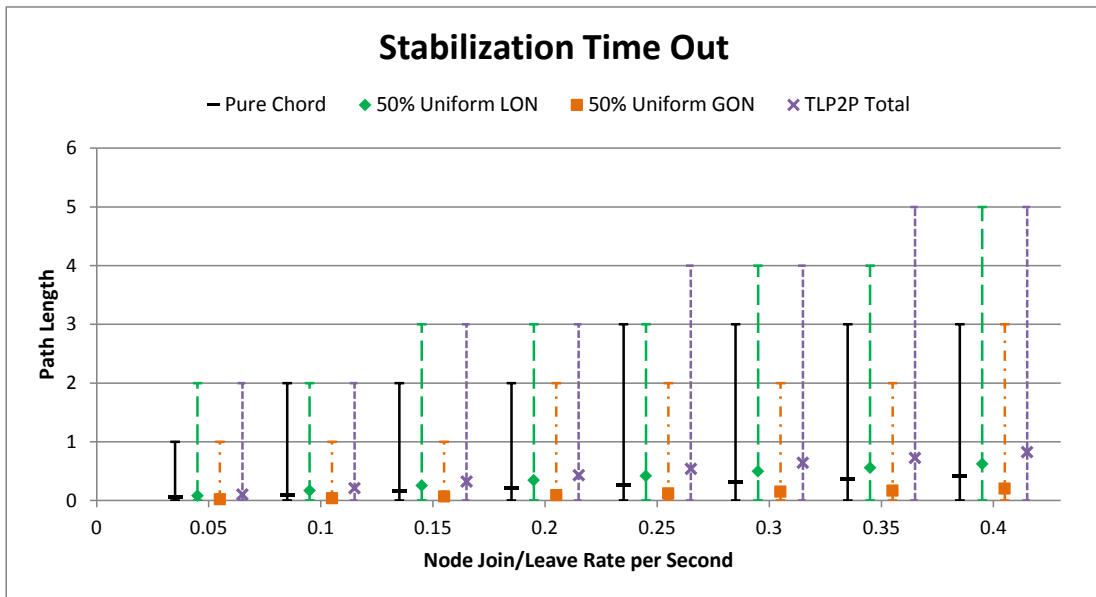


Figure 6.7: Time out experienced by the lookup during the stabilization scenario for the three networks normal Chord, LON and GON in addition to TLP2P total.

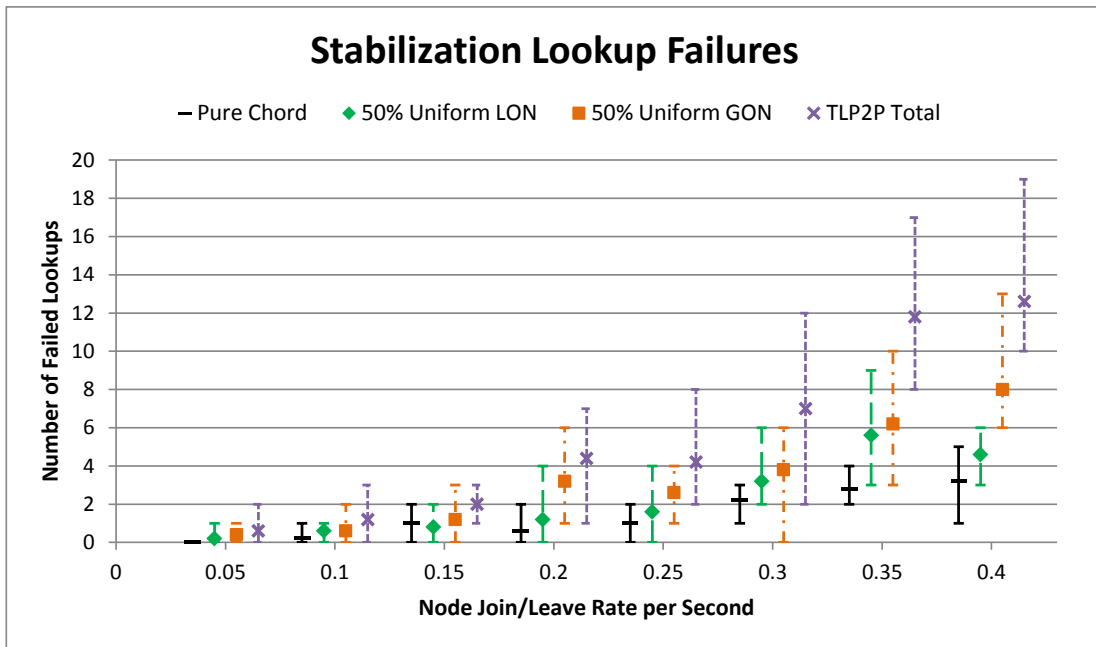


Figure 6.8: Number of failed requests in 10,000 requests within stabilization scenario for the three networks normal Chord, LON and GON in addition to TLP2P total.

6.2.1.5. Node Blockage

In this scenario, the path length has a minor effect as the number of blocked nodes increases, which can be seen in the two network simulation outputs in Figure 6.9. This is due to nodes at the beginning of the simulation are not aware of the blocked nodes, thus nodes' successor lists are not updated with the blocked nodes. When a node receives a request, it will try to forward the request to the nodes in its successor list assuming they are able to answer the query through the legacy DNS. If a node in the successor list is blocked, then it will create another lookup request yielding a longer lookup process and affecting the average path length. But over time the nodes will know which of the nodes are blocked and will update their successor list accordingly. That will reduce the unneeded lookup generated to blocked nodes and will reduce the impact on the average path length.

For the average load outputs, the difference appears in Chord and GON with the increase of blocked nodes, however there is no difference between simulation results for LON as seen in Figure 6.10. This is due to LON nodes not resolving DNS queries and they are utilized for caching only, thus there is no difference for LON in case of blockage or not. The average keys per node in this scenario is simply dividing the number of unblocked nodes by the number of keys and in the case of LON is just dividing the number of LON nodes by the number of keys. For the whole simulation runs in this scenario there were no request timeouts or failures.

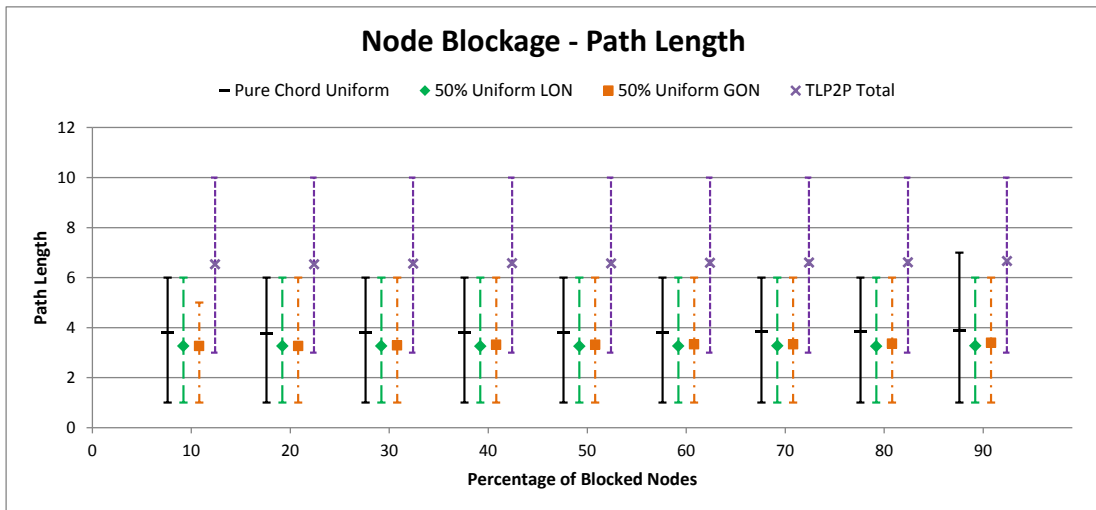


Figure 6.9: Path length for the case of blocked nodes scenarios for Chord network and TLP2P (GON and LON) with uniform requests.

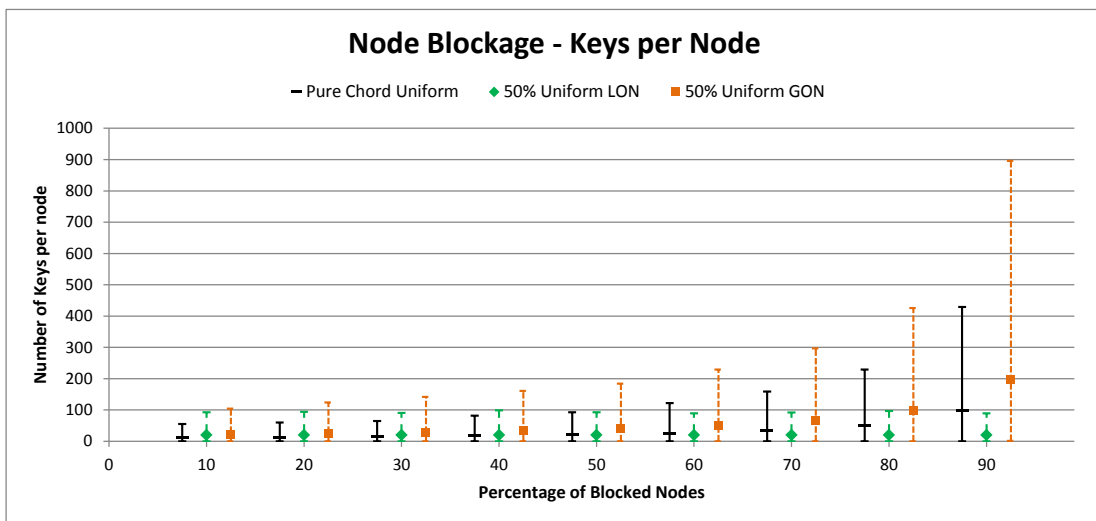


Figure 6.10: Average load between nodes for the case of blocked nodes scenarios for Chord network, GON and LON with uniform requests.

6.2.2. Worst Case Scenario with Zipf Distributed Requests

The normal Chord is re-simulated with Zipf distributed requests. The simulation results are almost similar to the uniform distribution except for the load balance which is explained in the following sections.

6.2.2.1. Load Balance

In this scenario, the average load between peers differs from the uniform scenario. As stated before, the average load is simply dividing the number of requests over the number of nodes and since Zipf distribution has requests repetition, it is expected to have a lower load among the peers than the uniform distribution. The difference can be seen in Figure 6.2 and Figure 6.11. So, to predict the average number of keys per node for this scenario, the estimate of the number of unique requests in Zipf distribution needs to be found first.

Zipf's law representing a frequency of elements is:

$$f(k; s, N) = \frac{1/k^s}{\sum_{n=1}^N (\frac{1}{n^s})}$$

Where k is the element rank, s is the exponent to characterize the distribution, and N is the number of elements. As the used distribution in the simulation is pure Zipf, then $s = 1$ and the law can be rewritten as:

$$f(k, N) = \frac{1/k}{\sum_{n=1}^N (\frac{1}{n})}$$

To find the number of unique requests, there will be two steps. First, we need to find the rank which will have the frequency of one or more. Second, we need to add the rank to the sum of frequencies that are lower than one.

Step one, finding the rank with frequency of 1 or more:

$$N * f(k, N) > 1 \quad (\text{solving for } k)$$

$$N * \frac{1}{k * \sum_{n=1}^N (\frac{1}{n})} > 1$$

$$k < \frac{N}{\sum_{n=1}^N (\frac{1}{n})}$$

Let $R = N / \sum_{n=1}^N (\frac{1}{n})$, in other words the elements from 1 to R occur 1 or more times.

Step two, the sum of frequency that is lower than 1 = $N * \sum_{k=R}^N f(k, N)$

So, the estimated number of unique requests = $R + N * \sum_{k=R}^N f(k, N)$

Table 6-2 lists the predicted and simulated results for normal Chord simulation. The difference is small as shown in the last column. The average load results for LON and GON are the same, but it is twice as the normal Chord load. Likewise LON and GON have close results compared to the predicted values, please refer to Table 6-3.

Table 6-2: Predicted and simulated results for the average load for normal Chord simulation

Number of Requests	Predicted Average Load	Simulated Average Load	Difference in Percentage
100000	2.88869976	2.71038	1.78%
200000	5.55130991	5.21814	1.67%
300000	8.14197427	7.66918	1.58%
400000	10.68821262	10.06792	1.55%
500000	13.20256797	12.46334	1.48%
600000	15.69215393	14.81138	1.47%
700000	18.1614886	17.15196	1.44%
800000	20.61386921	19.4699	1.43%
900000	23.05152615	21.79372	1.40%
1000000	25.4763291	24.08972	1.39%

Table 6-3: Predicted and simulated average load for LON and GON in the case of Zipf distributed requests. Each of LON and GON contains 50% of total TLP2P nodes.

Number of Requests	Predicted Average Load	Simulated Average Load for LON. (LON has same results as GON)	Difference in Percentage
100000	5.77739952	5.41892	1.79%
200000	11.10261982	10.43836	1.66%
300000	16.28394854	15.33208	1.59%
400000	21.37642524	20.14212	1.54%
500000	26.40513594	24.90532	1.50%
600000	31.38430786	29.63576	1.46%
700000	36.3229772	34.29892	1.45%
800000	41.22773842	38.95204	1.42%
900000	46.1030523	43.56248	1.41%
1000000	50.9526582	48.15928	1.40%

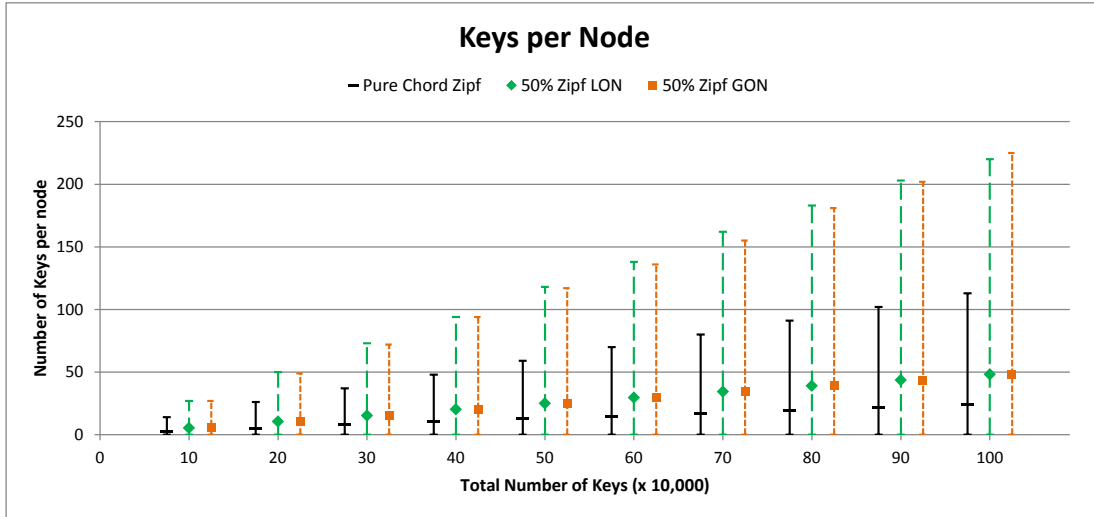


Figure 6.11: Average keys per node for normal Chord and TLP2P networks with Zipf distributed requests for networks with 10,000 peers.

6.2.2.2. Path Length

While the results are the same for normal Chord for the two scenarios, uniform and Zipf, it is different for TLP2P simulation. The two overlay networks in TLP2P experienced the same query path length, but the difference is the overall path length in TLP2P, as seen in Figure 6.12. This makes sense as repeated requests are resolved in LON while non-cached requests in LON are resolved through GON. To predict the path length for TLP2P for this scenario, other calculations need to be performed. From last subsection,

$$u = \text{number of unique requests} = R + N * \sum_{k=R}^N f(k, N)$$

$$r = \text{number of repeated request} = N - \text{unique requests}$$

$$\therefore \text{path length} = \frac{u}{N} * [0.5 * \log_2(LON) + 0.5 * \log_2(GON)] + \frac{r}{N} * 0.5 * \log_2(LON)$$

Table 6-4: Simulated and predicted values for the path length for TLP2P

Network Size 2^x	Predicted Path Length	Simulated Path Length	Difference in Percentage
3	1.4116	1.38825	1.65%
4	2.0802	2.164625	4.06%
5	2.7305	2.547563	6.70%
6	3.3659	3.214313	4.50%
7	3.9887	3.712609	6.92%
8	4.6005	4.472234	2.79%
9	5.2033	5.033566	3.26%
10	5.7981	5.671568	2.18%
11	6.386	6.240034	2.29%
12	6.9678	6.836242	1.89%
13	7.5442	7.367047	2.35%
14	8.1158	7.905904	2.59%

Table 6-4 lists the predicted values for the path length in TLP2P design and the simulation results. It is clear that the values are close to each other and that gives a good indication of the validity of the results obtained from the simulator.

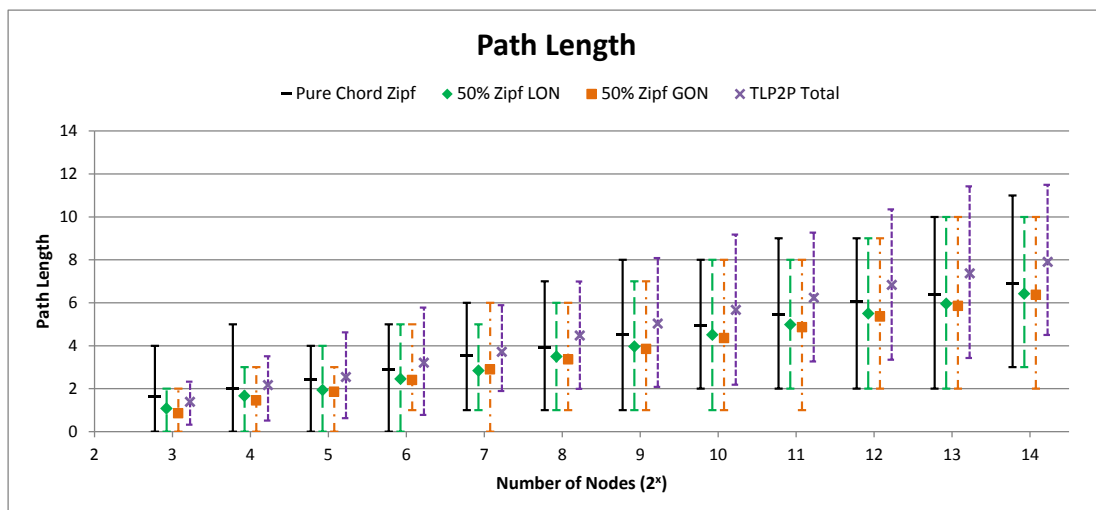


Figure 6.12: Path length results for normal Chord and TLP2P (50% LON – 50% GON) with the two overlay networks. Request distribution is Zipf.

6.2.2.3. Simultaneous Node Failures

Nodes failing affect the path length of the two designs and it can be noticed in the presented Figure 6.13 that the two designs have almost the same degradation of performance. The two overlay networks in TLP2P are almost identical.

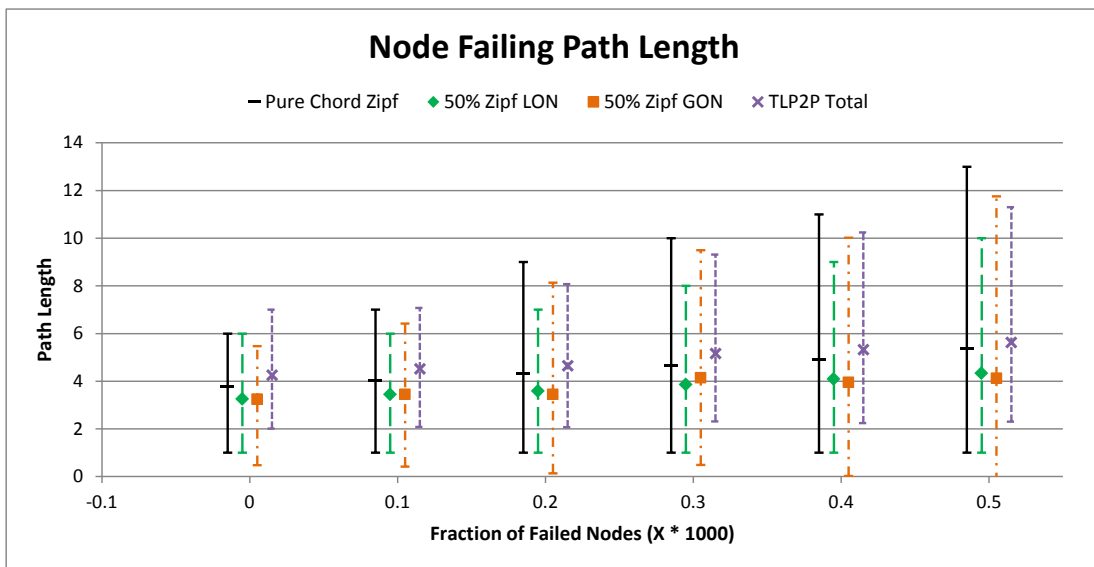


Figure 6.13: Path length of the two architectures while the networks are experiencing node failures. Networks received a Zipf distributed requests.

6.2.2.4. Lookups during Stabilization

As expected, the path length is increasing as the rate of joining and leaving nodes increases. Also, this is the case for the number of timeouts and failed requests as shown in Figure 6.14, Figure 6.15, and Figure 6.16. Unlike the uniform scenario, GON is having lower timeout and requests failure than LON, because the number of received requests is lower as all repeated requests were resolved in

LON. Overall, the performance of TLP2P in this scenario is better than the uniform scenario.

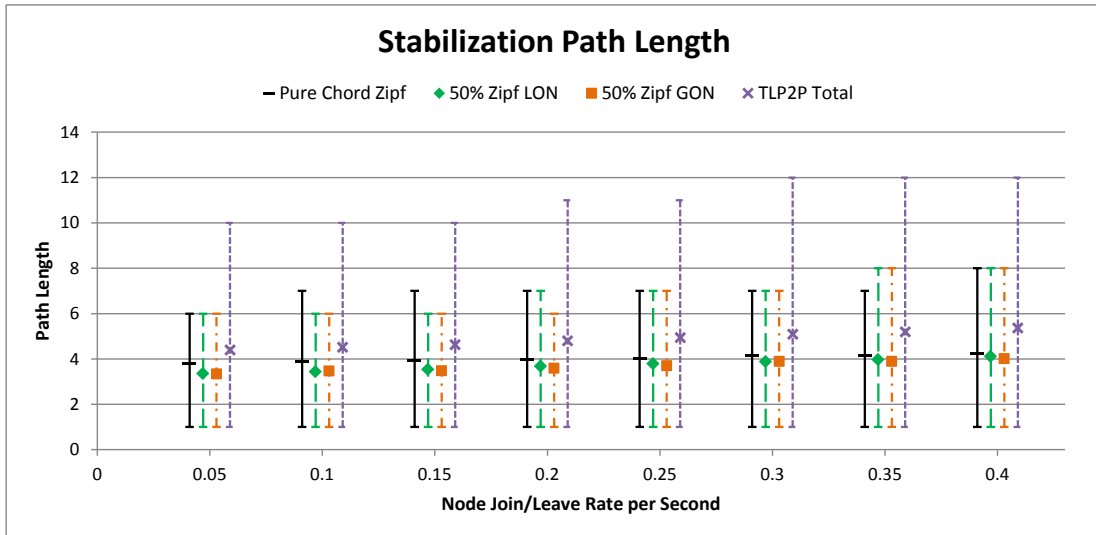


Figure 6.14: Stabilization path length for the two designs normal Chord and TLP2P (50% LON – 50% GON) with Zipf distributed requests.

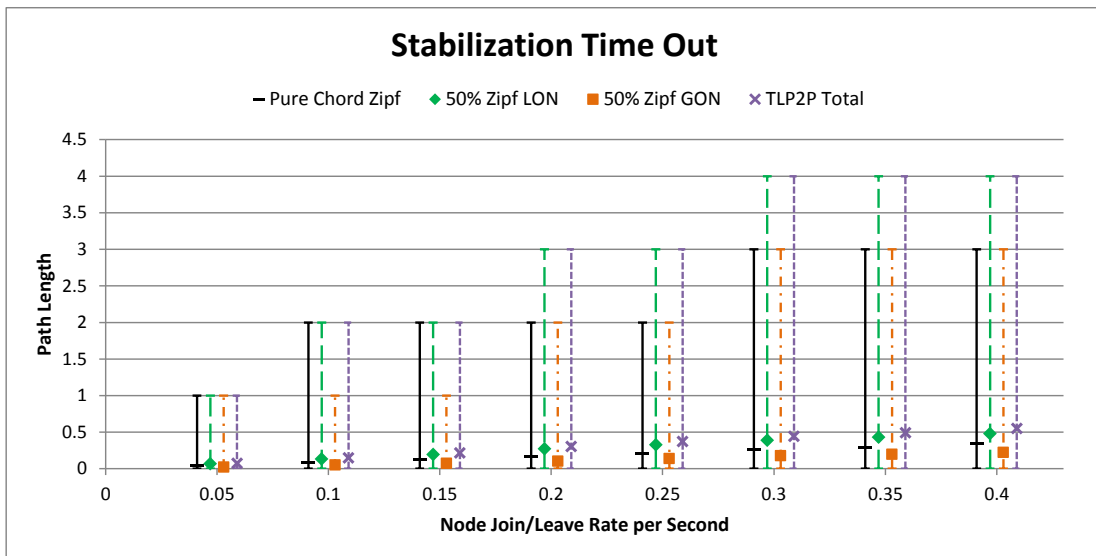


Figure 6.15: Lookups' time out for the stabilization case of the two designs normal Chord and TLP2P (50% LON – 50% GON) simulated with Zipf distributed requests.

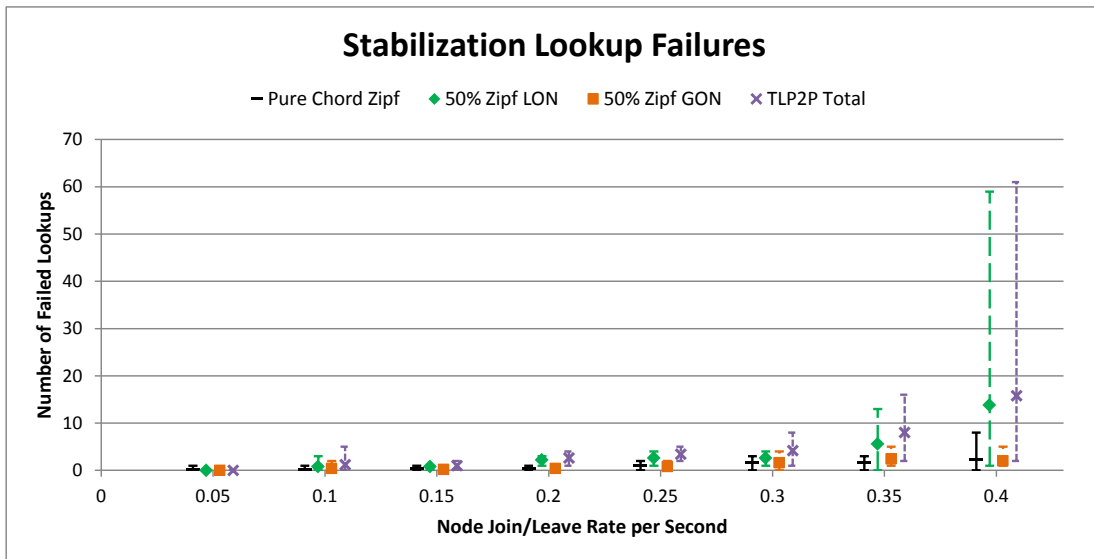


Figure 6.16: number of failed request's within 10,000 Zipf distributed requests for the stablization case. The plot represents the two designs results normal Chord and TLP2P (50% LON – 50% GON).

6.2.2.5. Node Blockage

The effect of the blockage nodes on the path length is the same as the case of the uniform distribution. Figure 6.17 shows small changes in the path length as the node blockage increases. However, the utilization of the cache in LON drops the average path length, unlike the simulation with uniform distributed request where there are no cache hits. Also, the behavior for the average load per node is identical to the uniform simulation. The average number of keys per nodes increases as the number of blocked nodes increases (see Figure 6.18). Again, the number of timeouts and failed requests for all blockage cases are zero.

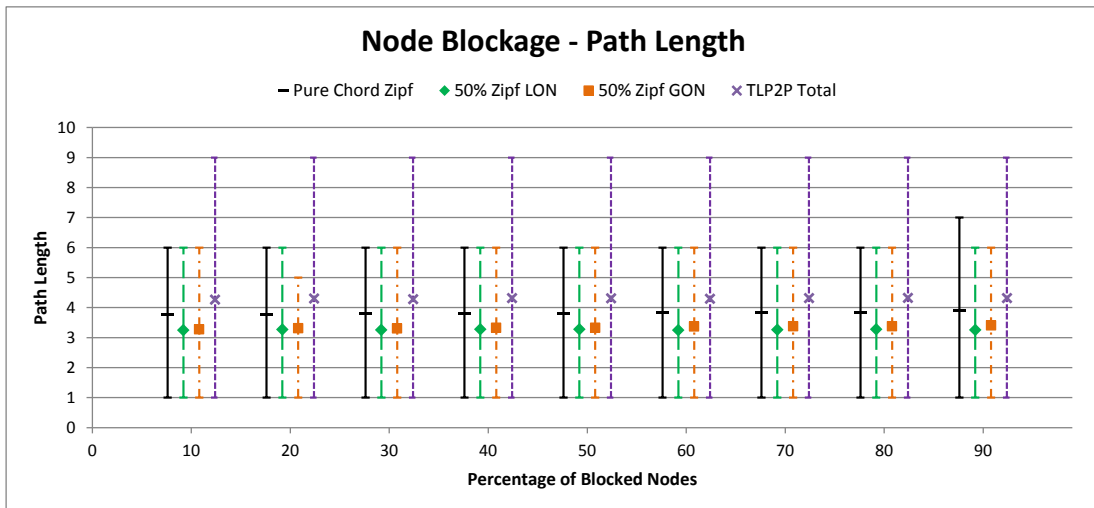


Figure 6.17: Path length, 1st and 99th percentile for Chord network and worst case of TLP2P (including GON and LON) with Zipf distributed requests.

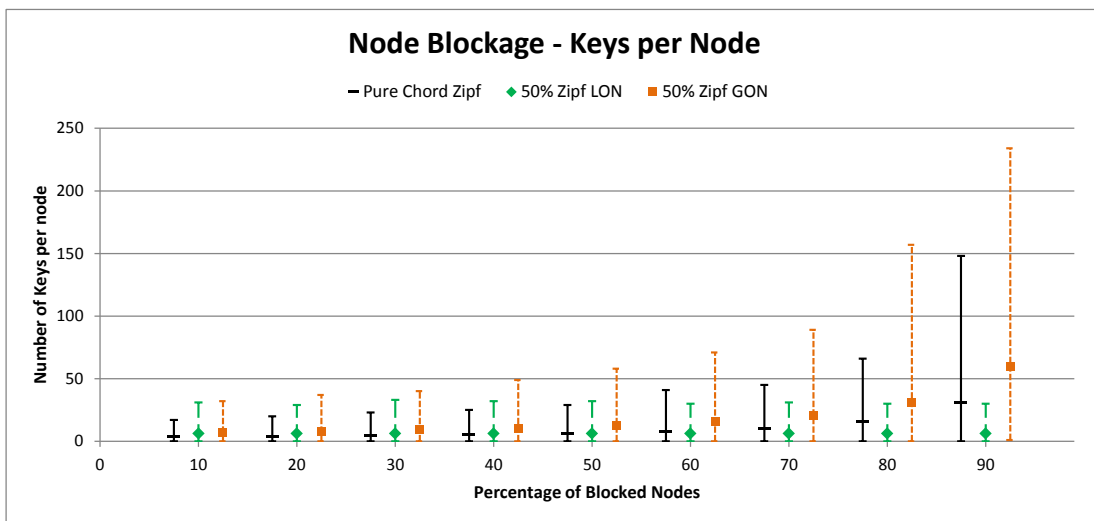


Figure 6.18: Average keys per node, 1st and 99th percentile for Chord network, GON and LON in worst case TLP2P and Zipf distributed requests.

6.2.3. Best Case Scenario with Zipf Distributed Requests

The presented graphs in this section for the normal Chord simulation are the same as the previous section. The simulation is with Zipf distributed requests. The difference in this scenario from the previous one is the size of LON and GON. The

total number of nodes in TLP2P is the same, but the difference is the distribution of the nodes over the two overlay networks. 5% of the total nodes are in LON and the rest (95%) are in GON.

6.2.3.1. Load Balance

Figure 6.19 presents the load distribution of the new rearrangement of nodes in TLP2P and the normal Chord. The results in the figure show a slight variation between normal Chord and GON network as they are almost having the same size. However, they have much lower average load than LON, since its size is much smaller than GON and normal Chord networks. 5% of 10,000 nodes is 500 nodes and that explains the reason behind the large variation. The values can be again predicted with the same way used in the previous section (worst case scenario) and the comparison is presented in Table 6-5 which shows close results too.

Table 6-5: Predicted and simulated results of the average load for LON network which is 5% of TLP2P total network size.

Number of Requests	Predicted Average Load	Simulated Average Load	Difference in Percentage
100000	57.7739952	54.2032	1.79%
200000	111.0261982	104.3956	1.66%
300000	162.8394854	153.2968	1.59%
400000	213.7642524	201.4952	1.53%
500000	264.0513594	249.1736	1.49%
600000	313.8430786	296.23	1.47%
700000	363.229772	343.0744	1.44%
800000	412.2773842	389.4624	1.43%
900000	461.030523	435.7748	1.40%
1000000	509.526582	481.7204	1.39%

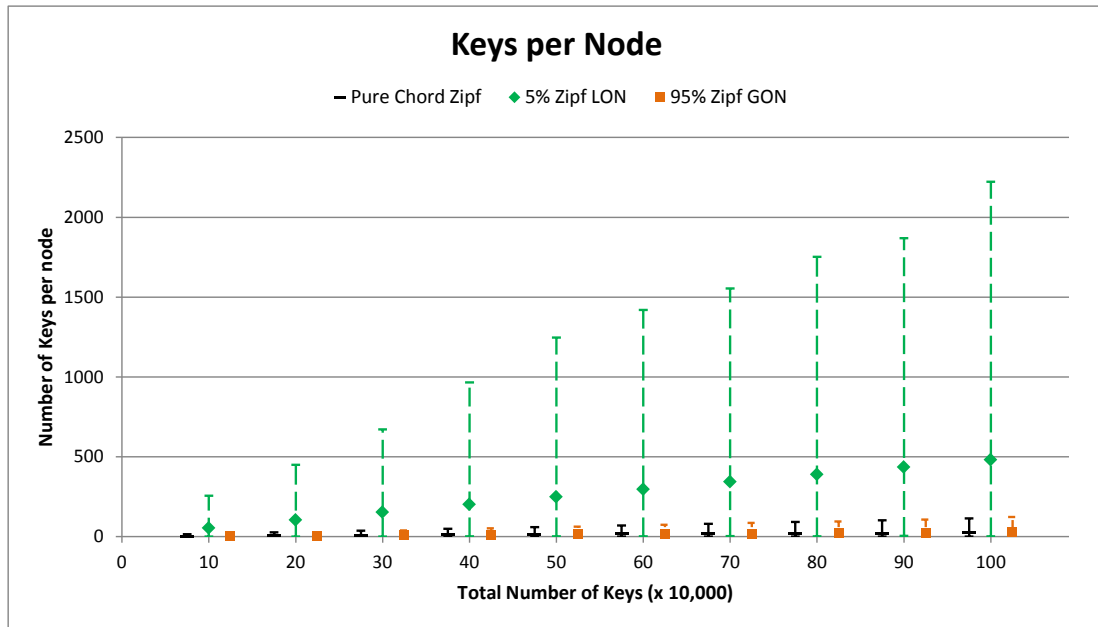


Figure 6.19: Average load of Zipf distributed requests simulations for the two design normal Chord and TLP2P (5% LON - 95% GON).

6.2.3.2. Path Length

For the path length simulations, the network sizes start with a small number of nodes 8, 16, 32 and so on. The size of LON network is 5% of the total nodes,

which means the LON size for the first case is 0.4 nodes. To correct the size, the ceiling is taken for the calculated percentage to find a logical LON size. Figure 6.20 shows the lookup path length for the two designs and it is 0 for the first two cases for LON, this is because there is only one node in LON which is the home node of all requests. The path length increases when LON size increases and thus affecting the TLP2P total path length. To calculate the predicated path length for this scenario, the same equation is applied as the worst case scenario in the previous section. Table 6-6 shows the predicted as well as simulated path length for all network sizes simulated. The results are close and the small variation is expected due to the nature of randomness of the queries and nodes hashed value.

Table 6-6: Predicted and simulated path length for TLP2P (5% LON – 95% GON) with Zipf distributed requests.

Network Size 2^x	Predicted Path Length	Simulated Path Length	Difference in Percentage
3	0.57777	0.5198	10.03%
4	0.75563	0.6723375	11.02%
5	1.3962	1.46398125	4.85%
6	2.0229	1.917469	5.21%
7	2.5438	2.299703	9.60%
8	3.0962	3.041438	1.77%
9	3.6926	3.480258	5.75%
10	4.2817	4.02052	6.10%
11	4.8575	4.694207	3.36%
12	5.4311	5.147848	5.22%
13	6.003	5.769342	3.89%
14	6.5706	6.344525	3.44%

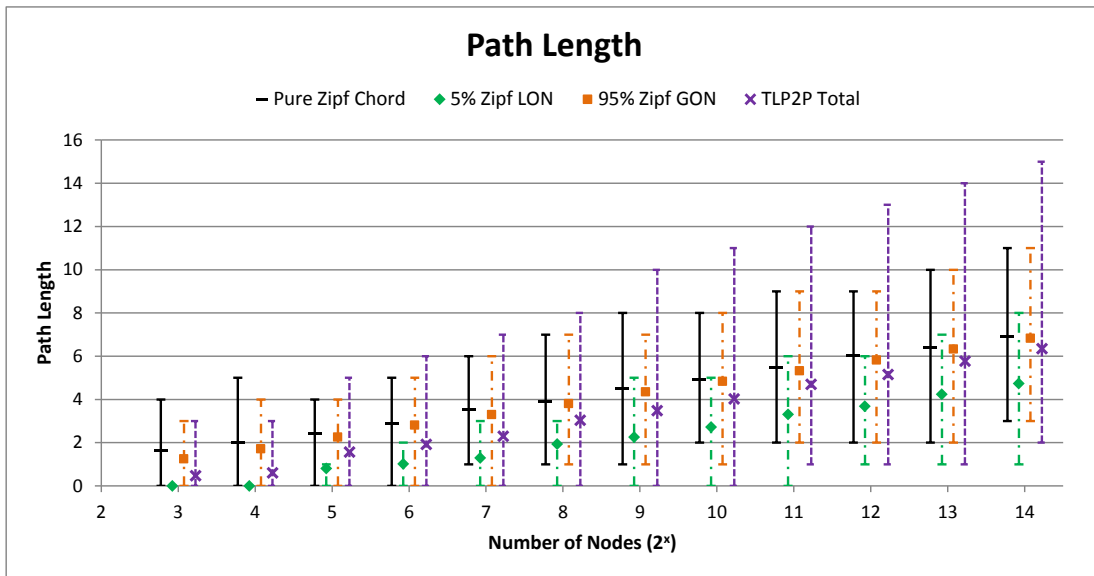


Figure 6.20: Path length for normal Chord and TLP2P (5% LON - 95% GON) with Zipf distributed requests.

6.2.3.3. Simultaneous Node Failures

The LON network size is 50 for all the simulations in this scenario. From Figure 6.21, it can be seen that LON is not affected by node failures. The reason behind it is that the network is small, which means, in the worst case only 25 nodes will fail. In 10,000 requests, this small number of failed node will not make any difference to the lookup path length. Moreover, it might reduce slightly the path length as noticed in the presented figure, because it will reduce the size of the network with negligible effect of failed nodes on the lookups, and the result is a lower path length. On the other hand, GON and normal Chord are having a similar effect by the node failure, since the network size for both is large and almost the same. The total path length of TLP2P is slightly affected by the node failure.

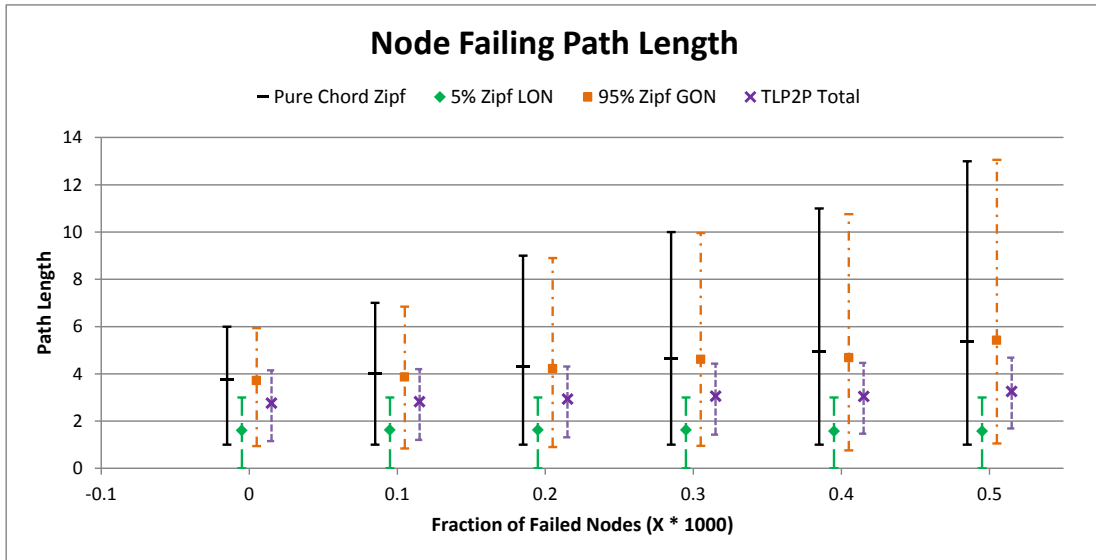


Figure 6.21: Path length for Zipf distributed requests to the two designs normal Chord and TLP2P (5% LON - 95% GON) after node failure.

6.2.3.4. Lookups during Stabilization

All the networks average path lengths are proportionally affected by the increase of nodes' join and departure rate. As seen in Figure 6.22, the path length is increasing with the instability of the nodes. Also, the timeout is increasing as the 'join' and 'leave' increases, and the effect on GON is less in this case since it receives only non-cached requests (see Figure 6.23). In other words, GON receives new requests or the caching node in LON left without transferring its cache to its successor, as explained in Chord paper, resulting in re-fetching the answer from GON. LON is having a larger number of failed requests, that is about 8% in the worst case, which is illustrated in Figure 6.24. Obviously, this is occurring because it is a small size network experiencing frequent joining and leaving nodes and the consequence is incorrect entries in the finger tables and the successor lists. This

also might have led to inconsistency in the network which caused the large number of failing lookups.

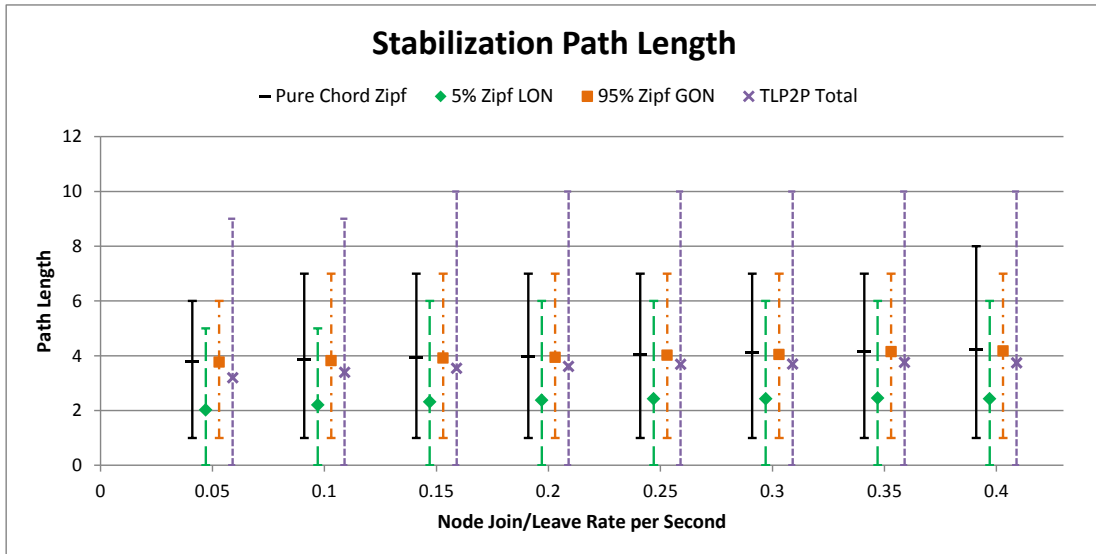


Figure 6.22: Simulation result of the path length for the two designs normal Chord and TLP2P (5% LON - 95% GON) receiving Zipf distributed request.

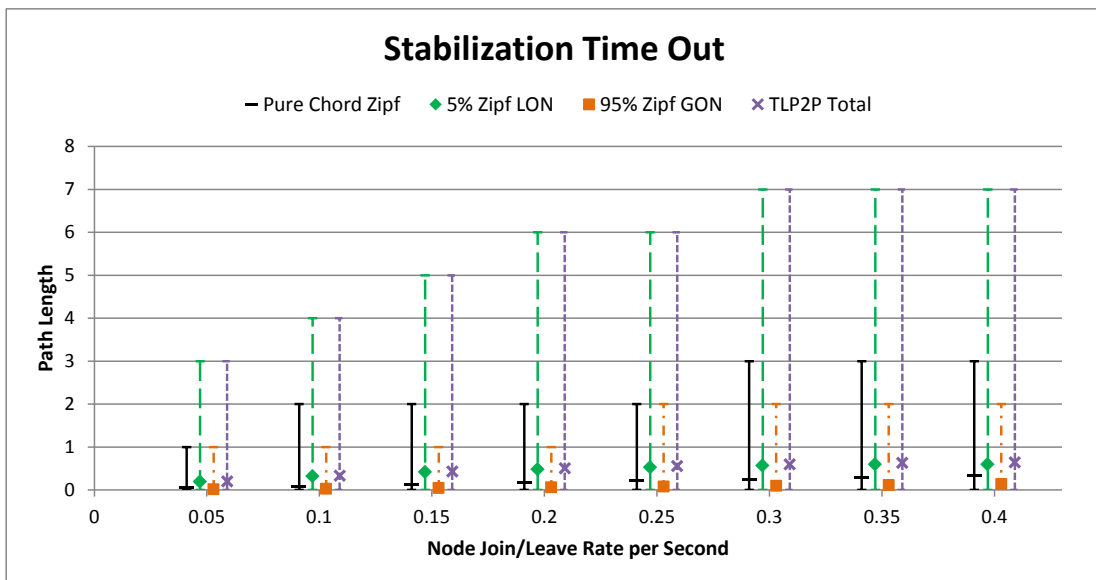


Figure 6.23: Simulation result of the time out for the two designs normal Chord and TLP2P (5% LON - 95% GON) receiving Zipf distributed request.

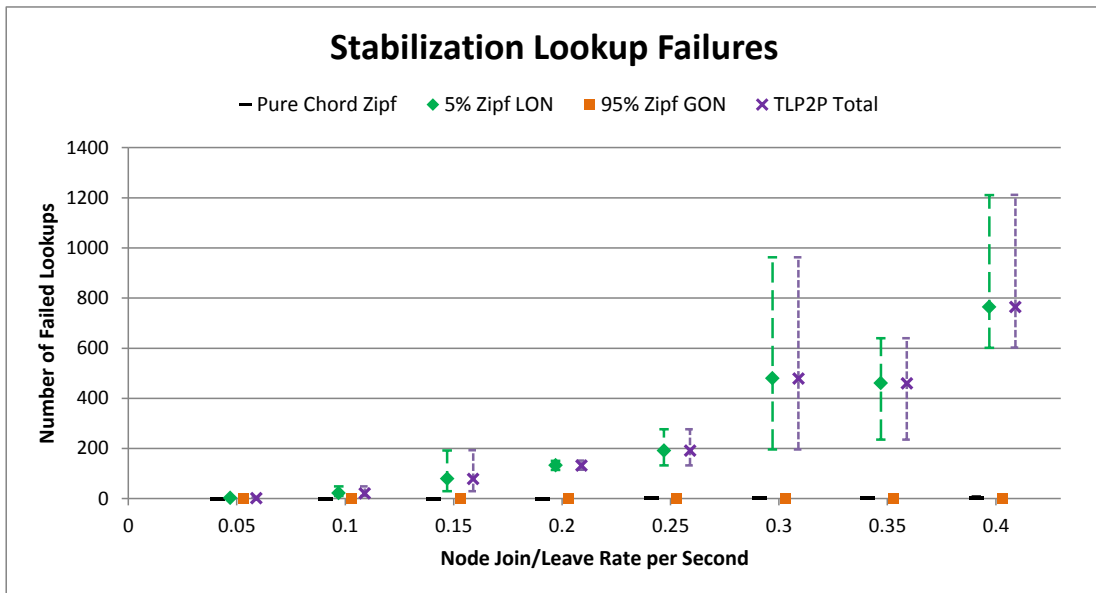


Figure 6.24: The results for number of failed lookups out of the 10,000 Zipf distributed requests for normal Chord and TLP2P (5% LON - 95% GON).

6.2.3.5. Node Blockage

The influence of the node blockage in the best case TLP2P scenario is the same as the other blockage cases. Figure 6.25 shows slight changes in the path length as the node blockage increases, and Figure 6.26 shows the average keys per node with the varied averages in Chord and GON networks.

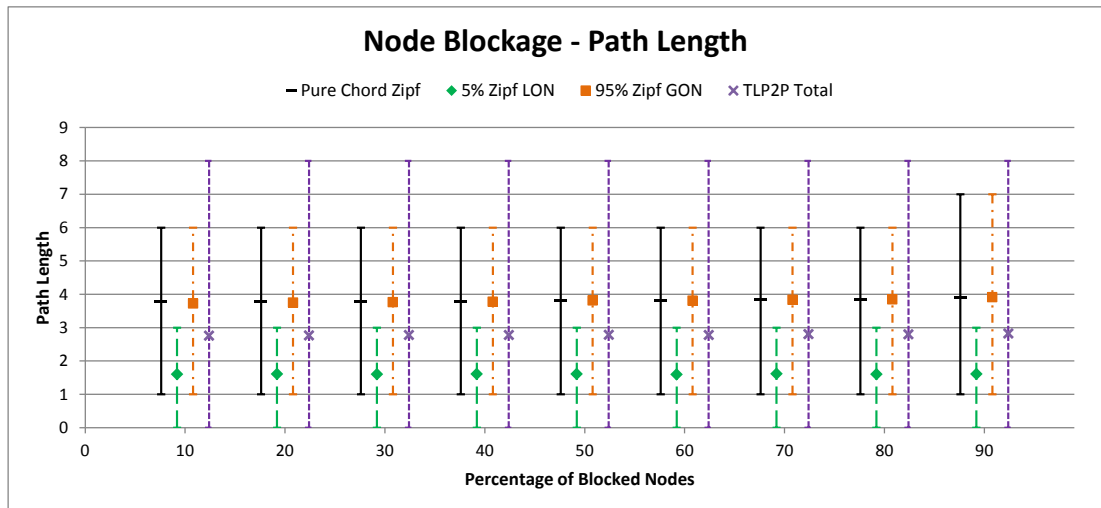


Figure 6.25: Path length, 1st and 99th percentile for Chord network and best case of TLP2P (including GON and LON) with Zipf distributed requests.

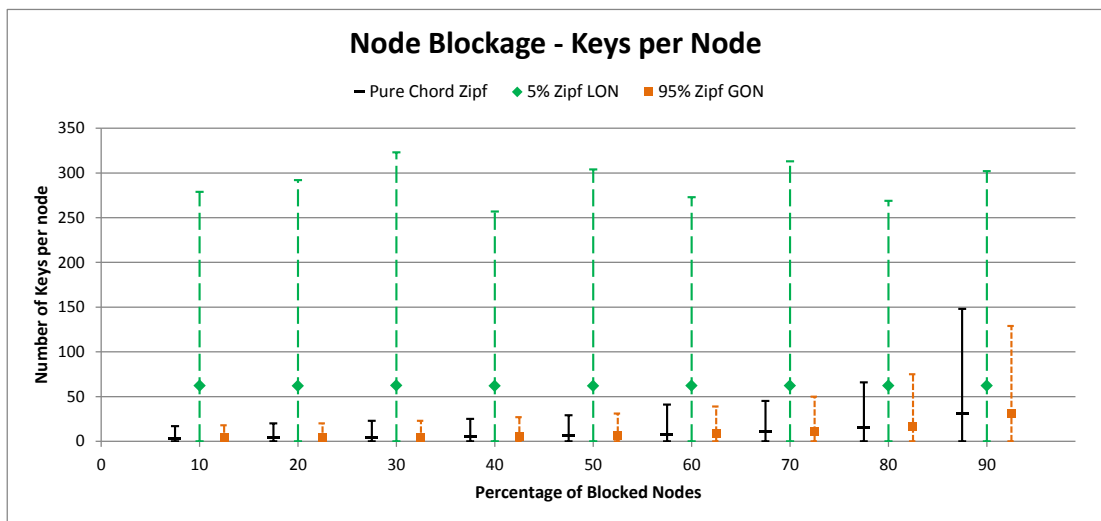


Figure 6.26: Average load, 1st and 99th percentile for Chord network, GON and LON in best case TLP2P with Zipf distributed requests.

6.3. Results Comparison

6.3.1. Load Balance

The results in the load balance simulation have two indications, i.e., how the records are distributed between peers and how the nodes are utilized. The distributions of records among the peers for LON, GON, and pure Chord are plotted in Figure 6.27. The figure only shows the result for the 500,000 Zipf requests case with nodes distribution in TLP2P as 5% in LON and 95% in GON. Clearly, the number of records stored within peers in LON vary more, and the range is [0, 2000] while for the larger networks it is [0, 180]. But more than 80% of nodes are storing 500 records or less. For the utilization concerns, LON has only 0.64% of nodes with no records while it is 7.39% in pure Chord. Also, the results in the figure show that LON is more utilized and this is an advantage in TLP2P over the pure Chord. The LON size can be increased to meet the requirements whenever the load threshold is reached. And, as LON is controlled locally, its size is elastic and it can be scaled up or down to satisfy the requirements. The uniform simulation has the same trend as the pure Chord and GON, but with fewer nodes having no keys, since all the requests are unique in the uniform distribution.

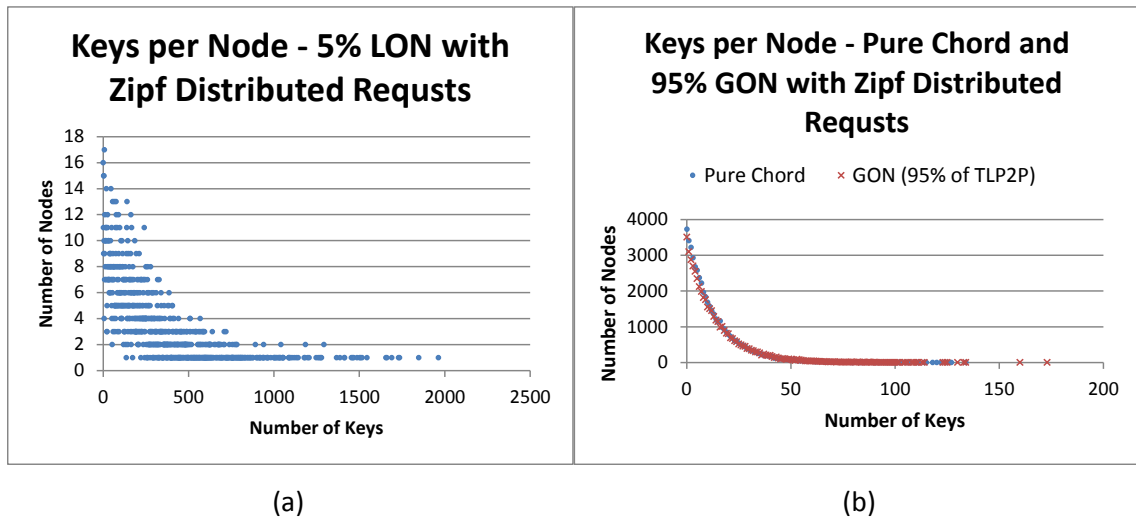


Figure 6.27: Number of keys per node simulation for the case of 500,000 Zipf distributed requests (a) Presents record distribution for LON network (5% of TLP2P nodes) (b) Presents pure Chord and GON (95% of TLP2P nodes).

6.3.2. Path Length

TLP2P in the best scenario has the best results, but TLP2P in worst case scenario with uniform requests was having the worst results. In the same scenario, but with Zipf requests, TLP2P performed better and the results was close to pure Chord. TLP2P is structured to utilize the caching layer, however in the case of uniform simulation all requests are unique and no query is resolved from the cache. Also, the node distribution between LON and GON meant to increase the path length which all of these factors contributed to end up with the highest path length in the worst case scenario with uniform requests. In the best case simulation, the increase in path length is due to the increase in LON size. If the LON size was fixed to a low number of nodes, then the path length will even be better than the results obtained as can be inferred from the path length equation found earlier in this chapter.

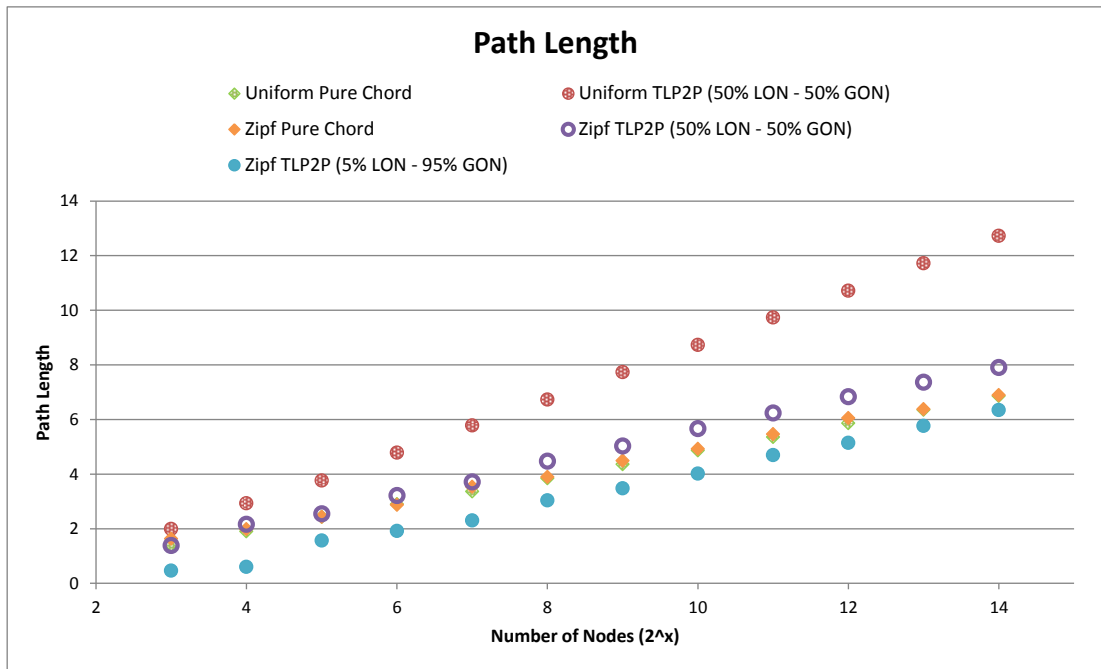


Figure 6.28: Average path length for the simulated scenario.

6.3.3. Simultaneous Node Failures

As in the previous measures, TLP2P in the best case simulation has the best path length in the node failure scenario. Also, unlike the other designs and settings, the effect on path length with the increase in the node failure is minor. TLP2P in the worst case scenario with Zipf requests is similar to normal Chord, while the same scenario with uniform distribution is having the highest path length.

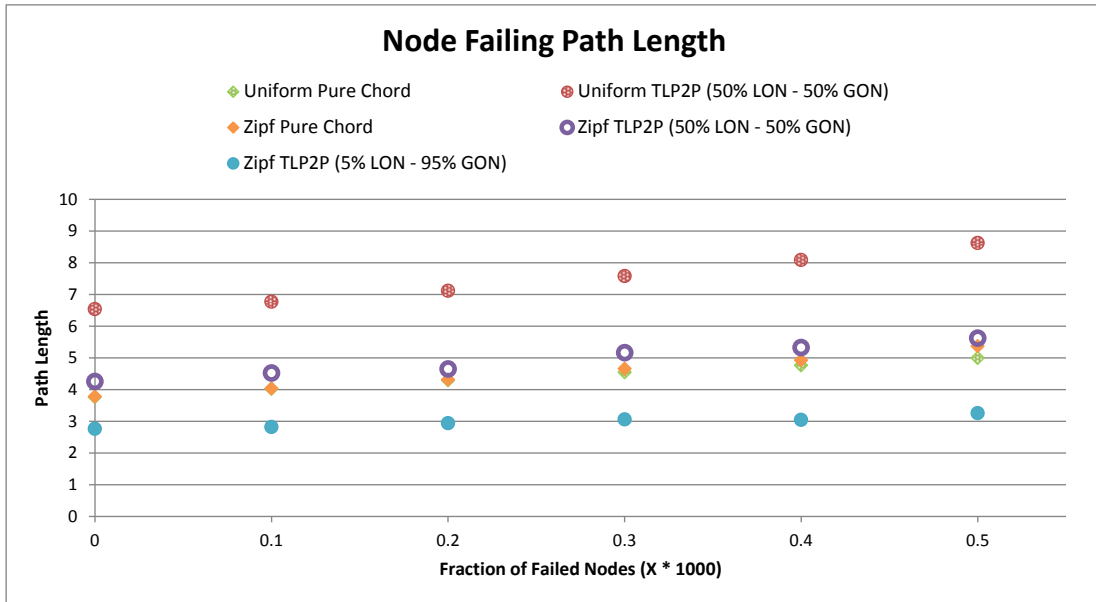


Figure 6.29: Average path length for node failure scenario.

6.3.4. Lookups during Stabilization

TLP2P in the best case simulation experienced network inconsistency as discussed in the last section. Even though it has the lowest path length, the pure Chord performed better in this scenario. This is because it has the lowest number of lookup failures and lower timeout compared to TLP2P results. TLP2P also has good results in term of path length, timeout, and lookup failures when receiving Zipf requests in the worst case scenario. But in the uniform request distribution scenario, TLP2P has the longest path to home nodes and high timeout.

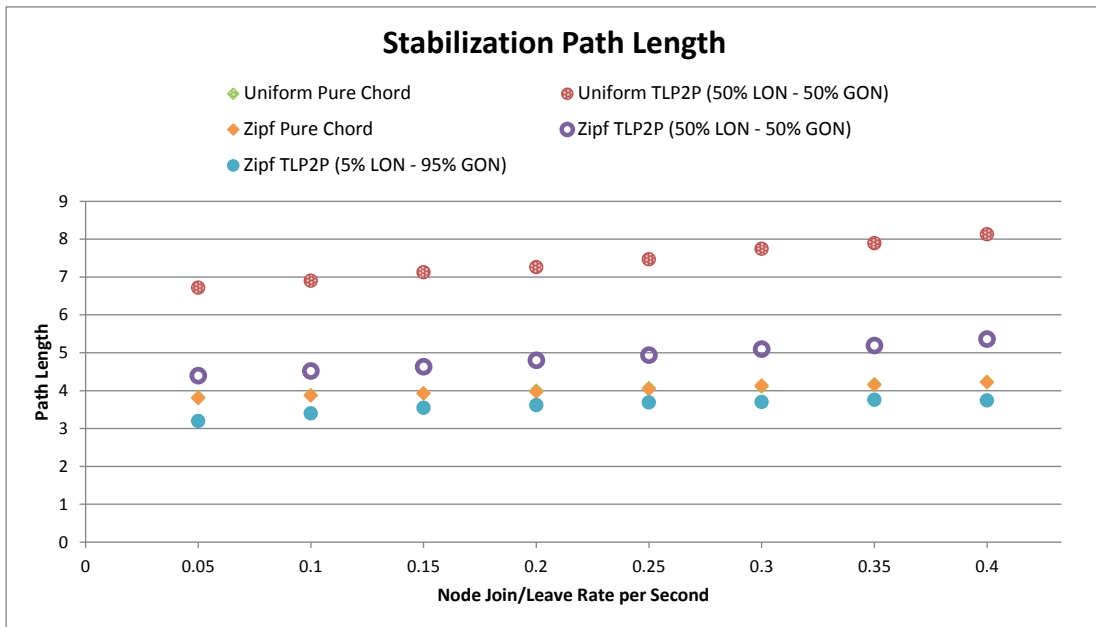


Figure 6.30: Average path length for stabilization scenario.

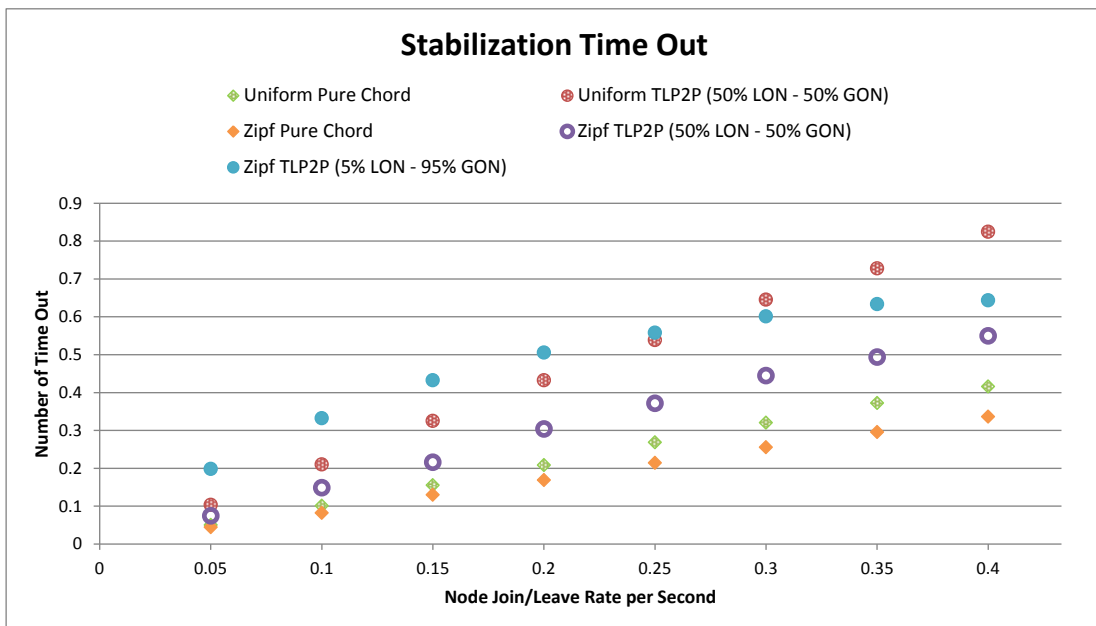


Figure 6.31: Average time out for stabilization scenario.

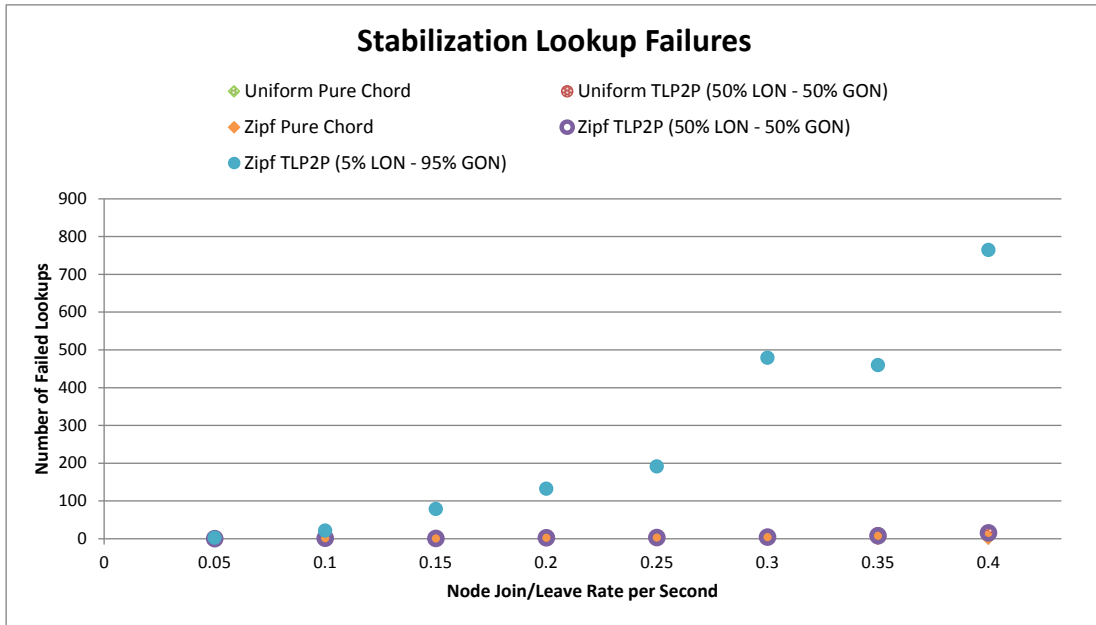


Figure 6.32: Average number of lookup failures for stabilization scenario.

6.3.5. Node Blockage

The results obtained for the three blockage scenarios show small changes in the path length as an effect of the increase in the number of blocked nodes (see Figure 6.33). The reason behind the increase is because nodes' successor lists at the start of the simulation are not updated with blocked nodes which lead into more lookups. Overall, the increase in the path length is not large, please refer to section 6.2.1.5 for more explanation of this behavior. In general, the results give a good indication of the effectiveness of the solution as the increase in path length is minor besides the number of timeouts and request failures are zeros. For the loads on the peers, LON has no impact as the nodes get blocked. However, Chord and GON has almost the same performance in the best case TLP2P scenario (see Figure 6.34). But in the worst case TPL2P scenario, GON nodes store almost double the number of

keys stored in Chord nodes. This is due to the simulation configuration which indicates 50% of TLP2P nodes are in GON and that is half Chord network size. So, the number of resolver nodes is different between the two networks which results in more loads in GON than Chord, but if the number of resolver nodes was the same in both networks then they will end up with the same average load.

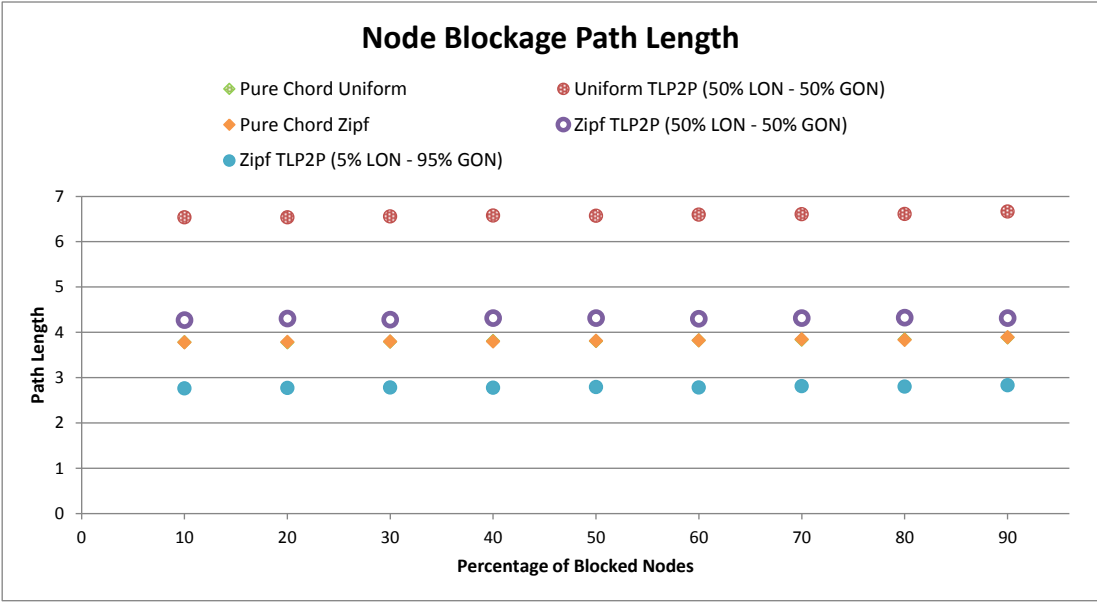
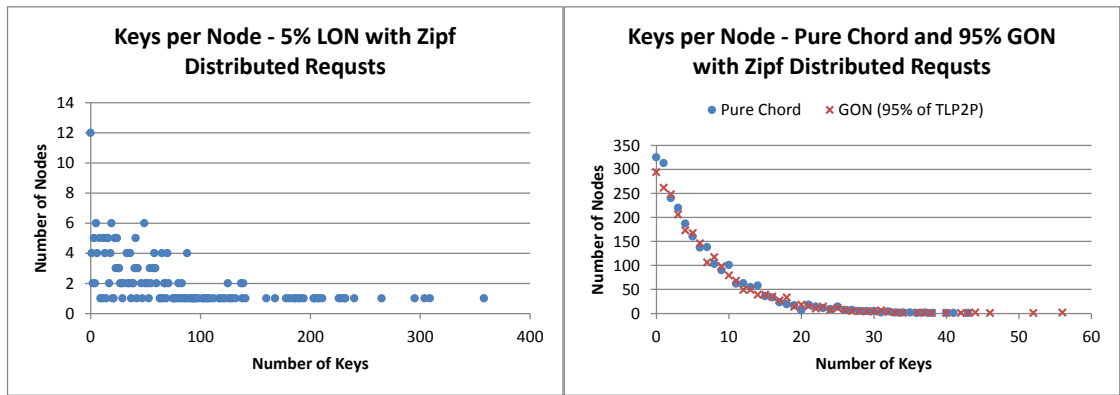


Figure 6.33: Average path length of the node blockage scenario.



(a)

(b)

Figure 6.34: Number of keys stored per node in the case of node blockage simulation for the case of 50% of nodes are blocked with Zipf distributed requests (a) Presents record distribution for LON network (5% of TLP2P nodes) (b) Presents pure Chord and GON (95% of TLP2P nodes)

Overall, TLP2P has the best performance when LON was of small size to satisfy the requirements while being fully utilized. This is a logical view as LON will be locally controlled and elastic based on the goals to be achieved. If each of these simulated scenarios were considered as a real scenario being faced, then LON can be adapted to meet better results than the normal Chord.

CHAPTER 7

FUTURE WORK AND CONCLUSION

7.1. Conclusion

In this thesis, three types of solutions have been presented to mitigate or prevent the DoS attack targeting the DNS system. One of the discussed solutions proposes a new DNS structure (Dynamic Round-Robin P2P) as a countermeasure to the control of root DNS servers by United States Department of Commerce (USDC). However, in this thesis, the proposed design (TLP2P) overcomes the dis-efficiencies and limitations that exist in the Dynamic Round-Robin P2P system. For example, query resolution is not optimized by utilizing the cached answers which implies large dependency on the legacy DNS system. Also, the authoritative domain name server within a blocked region cannot publish hostnames to be served by Dynamic Round-Robin P2P system. The analyzed simulation results showed that TLP2P has less dependency on the legacy DNS system by utilizing the cached responses. Also, the theoretical analysis and simulation results have proven that TLP2P resolved the blockage problem with enhanced performance in terms of path length, even in unstable networks. TLP2P has other advantages too, like its dynamic structure where the local overlay network can be adjusted online to attain user satisfaction while the global overlay network is automated and uncontrolled by an owner. Also, better load balance can be achieved by introducing virtual nodes into the system.

Finally, this solution can be extended by implementing other advanced layers, like record duplication and query prioritization that will be used to enhance the performance.

7.2. Future Work

The future work improvements will look into the following three aspects: reduce the path length, balance the load between nodes, and countermeasure malicious peers. For the first two aspects, there have already been solutions presented in this thesis and they just need to be implemented in the current design to validate the added values. For malicious peers, further investigation and research need to be done to find the threats raised by such peers and how to tackle them.

To enhance the simulation, the simulator should account for the cached records expiration. Each record should have a Time-To-Live (TTL) value, and records should become obsolete when their expiration time is reached. Expired records need to be updated based on the update mechanism as discussed in TLP2P design. Also, query latency should be considered to measure the overall resolution time. Moreover, the solution should be simulated with malicious peers to have a better prediction of TLP2P behavior in real life.

References

- [1] BCC News. (2009, December) [cited Aug. 3, 2012] 'Iranian cyber army' hits Twitter. <http://news.bbc.co.uk/2/hi/technology/8420233.stm>.
- [2] Bill Stewart. (2010) [cited Dec. 4, 2010] Domain Name System (DNS) History. [Online]. http://www.livinginternet.com/i/iw_dns_history.htm.
- [3] Charles Arthur and Josh Halliday. (2010, December) [cited Aug. 3, 2012] WikiLeaks fights to stay online after US company withdraws domain name. [Online]. <http://www.guardian.co.uk/media/blog/2010/dec/03/wikileaks-knocked-off-net-dns-everydns>.
- [4] David Schneider. (2010, December) [cited Aug. 3, 2012] WikiLeaks Demonstrates Web Resiliency. [Online]. <http://spectrum.ieee.org/tech-talk/telecom/internet/wikileaks-demonstrates-web-resiliency>.
- [5] Fahd A. Abdulhameed (2010). Dynamic Round-Robin Peer-To-Peer (P2P) Domain Name System (DNS). King Fahd University Of Petroleum & Minerals, Dhahran, Saudi Arabia.
- [6] Geoff Huston. (2001, May) [cited Sept. 12, 2011] The Unreliable Internet. [Online]. <http://www.potaroo.net/ispcol/2001-05/2001-05-reliable.html>.
- [7] Google. [cited Jun.25, 2012] Company Overview. [Online]. <http://www.google.com/about/company/>.
- [8] Hakem Beitollahi, Geert Deconinck, "Comparing Chord, CAN, and Pastry overlay networks for resistance to DoS attacks," CRISIS 2008: 261-266.
- [9] Helen McDevitt. (2000, November) [cited Dec. 4, 2010] Load Sharing with DNS. [Online]. <http://ntrg.cs.tcd.ie/undergrad/4ba2.01/group8/DNS.html>.
- [10] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications," IEEE/ACM Transactions on Networking, vol. 11, no. 1, pp. 17–32, 2003.
- [11] Ian W. Marshall and Chris Roadknight, "Linking cache performance to user behaviour," Computer Networks and ISDN Systems, 30 (22-23). pp. 2123-2130, 1998.
- [12] IANA. [cited Jan. 2, 2012] Root Zone Database. [Online]. <http://www.iana.org/domains/root/db/>.

- [13] ICANN. (2007, March) [cited Aug. 3, 2012] Factsheet for Root server attack on 6 February 2007. [Online]. <http://www.icann.org/en/about/learning/factsheets/factsheet-dns-attack-08mar07-en.pdf>.
- [14] J. Jung, E. Sit, H. Balakrishnan, "DNS Performance and the Effectiveness of Caching," *IEEE/ACM Transactions on Networking*, V. 10, N. 5, October 2002.
- [15] J. Mirkovic, E. Arikan, S. Wei, S. Fahmy, R. Thomas, and P. Reiher. Benchmarks for DDoS Defense Evaluation. In *MILCOM*, 2006.
- [16] Jeanne Meserve and Mike Ahlers. (2010, January) [cited Aug. 3, 2012] Google reports China-based attack, says pullout possible. [Online]. http://articles.cnn.com/2010-01-12/tech/google.china_1_google-search-engine-david-drummond.
- [17] K. Poulsen. [Cited Jan.19, 2011] FBI busts alleged DDoS mafia. www.securityfocus.com/news/9411, 2004.
- [18] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," *Proc. IEEE INFOCOM*, 1999.
- [19] M. Handley and A. Greenhalgh, "The Case for Pushing DNS," in *Proc. of Hotnets-IV*, 2005.
- [20] M. Recabarren, M. Nussbaum and C. Leiva, "Cultural divide and the Internet," *Computers in Human Behavior*, 24, 2917-2926, 2008.
- [21] Marvin Theimer, Michael B. Jones, "Overlook: Scalable Name Service on an Overlay Network," *ICDCS 2002*: pp. 52-61.
- [22] Microsoft. (2003, March) [cited Oct. 11, 2011] How DNS Works. [Online]. <http://technet.microsoft.com/en-us/library/cc772774%28WS.10%29.aspx>.
- [23] Microsoft. (2005, January) [cited Oct. 18, 2011] How DNS Query Works. [Online]. <http://technet.microsoft.com/en-us/library/cc775637%28WS.10%29.aspx>.
- [24] Mudhakar Srivatsa, Arun Iyengar, Jian Yin, and Ling Liu, "A Middleware System for Protecting Against Application Level Denial of Service Attacks" M. van Steen and M. Henning (Eds.): *Middleware 2006*, LNCS 4290, pp. 260–280, 2006. IFIP International Federation for Information Processing 2006.
- [25] N. Brownlee and I. Ziedins. Response time distributions for global name servers. In *Proceedings of PAM 2002 Workshop*, Mar. 2002.
- [26] N. Brownlee, kc Claffy, and E. Nemeth, "DNS Measurements at a Root Server," In

Globecom, Nov. 2001.

- [27] Nevil Brownlee. (2006, March) [cited Dec. 4, 2010] Root/gTLD DNS Performance Plots. [Online]. http://www.caida.org/cgi-bin/dns_perf/main.pl.
- [28] P.Mockapetris and k. Dunlap. Development of the Domain Name System. In Proc. ACM SIGCOMM, Stanford, CA, 1988.
- [29] Paul Vixie, Gerry Sneeringer and Mark Schleifer. (2002, November) [cited Aug. 3, 2012] Events of 21-Oct-2002. [Online]. <http://d.root-servers.org/october21.txt>.
- [30] R. Liston, S. Srinivasan, and E. Zegura, "Diversity in DNS Performance Measures," In Proceedings of the ACM SIGCOMM Internet Measurement Workshop, 2002.
- [31] Robert Lemos and Jim Hu. (2004, June) [cited Aug. 3, 2012] 'Zombie' PCs caused Web outage, Akamai says. [Online]. http://news.cnet.com/2100-1038_3-5236403.html.
- [32] Russ Cox, Athicha Muthitacharoen, Robert Morris, "Serving DNS Using a Peer-to-Peer Lookup Service," 2002, pp. 155-165.
- [33] V. Ramasubramanian and E. G. Sirer, "The design and implementation of a next generation name service for the Internet," In SIGCOMM, August 2004.
- [34] Vasilis Pappas, Dan Massey, Lixia Zhang, "Enhancing DNS Resilience against Denial of Service Attacks," IEEE/IFIP Dependable Systems and Networks (DSN), June 2007.
- [35] Wei Yu, Dong Xuan and Wei Zhao, "Middleware based Approach for Preventing Distributed Denial of Service Attacks," in Proc. of IEEE Military Communications (MILCOM), October 2002.
- [36] Wikimedia. (2010, September) [cited Dec.4, 2010] Distributed denial of service attacks on root nameservers. [Online]. http://en.wikipedia.org/wiki/Distributed_denial_of_service_attacks_on_root_nameservers.
- [37] Wikipedia. (2010, November) [cited Dec. 4, 2010] Root nameserver. [Online]. http://en.wikipedia.org/wiki/Root_nameserver.
- [38] Wikipedia. (2010, September) [cited Dec.4, 2010] Distributed denial of service attacks on root nameservers. [Online]. http://en.wikipedia.org/wiki/Distributed_denial_of_service_attacks_on_root_nameservers.
- [39] Wikipedia. (2011, November) [cited Dec. 2, 2011] Zipf's law. [Online]. http://en.wikipedia.org/wiki/Zipf's_law.

- [40] Wikipedia. (2011, October) [cited Oct. 18, 2011] Denial-of-Service attack. [Online].
http://en.wikipedia.org/wiki/Denial-of-service_attack.
- [41] Wikipedia. (2011, October) [cited Oct. 18, 2011] History of the Internet. [Online].
http://en.wikipedia.org/wiki/History_of_the_Internet.
- [42] Wikipedia. (2011, September) [cited Oct. 18, 2011] Hostname. [Online].
<http://en.wikipedia.org/wiki/Hostname>.
- [43] Wikipedia. (2012, April) [cited Jun.25, 2012] Operation Aurora. [Online].
http://en.wikipedia.org/wiki/Operation_Aurora.
- [44] Wikipedia. (2012, June) [cited Jun.25, 2012] Yahoo! [Online].
<http://en.wikipedia.org/wiki/Yahoo!>
- [45] Wolfgang Nagele, Emile Aben, Daniel Karrenberg (2011, July) [cited Oct. 19, 2011] Analysis of Increased Query Load on Root Name Servers. [Online].
<https://labs.ripe.net/Members/wnagele/analysis-of-increased-query-load-on-root-name-servers>.
- [46] Y. Jiang and J. You, "A low latency Chord routing algorithm for DHT," in: Proceedings of the 1st International Symposium on Pervasive Computing and Applications, August 2006, pp. 825-830.
- [47] Yahoo. [cited Jun.25, 2012] Yahoo Products. [Online].
<http://info.yahoo.com/privacy/us/yahoo/products.html>.
- [48] Yu Dan, Chen XinMeng and Chang YunLei, "An improved P2P model based on chord," PDCAT 2005: Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies, Proceedings on 2005, pp. 825-830, August 2006.

Vitae

Name : Mahmoud Salman Al-Saba

Nationality : Saudi

Date of Birth : 9/24/1984

Email : saba2006@gmail.com

Address : Dhahran 31311 P.O.Box 10318

Academic Background : Information & Computer Science