UNIVERSITI PUTRA MALAYSIA

HYBRIDFLOOD ALGORITHMS MINIMIZING REDUNDANT MESSAGES AND MAXIMIZING EFFICIENCY OF SEARCH IN UNSTRUCTURED P2P NETWORKS

HASSAN BARJINI

FSKTM 2012 12

HYBRIDFLOOD ALGORITHMS MINIMIZING REDUNDANT MESSAGES AND MAXIMIZING EFFICIENCY OF SEARCH IN UNSTRUCTURED P2P NETWORKS

HASSAN BARJINI

DOCTOR OF PHILOSOPHY
UNIVERSITI PUTRA MALAYSIA

2012

HYBRIDFLOOD ALGORITHMS MINIMIZING REDUNDANT MESSAGES AND MAXIMIZING EFFICIENCY OF SEARCH IN UNSTRUCTURED P2P NETWORKS

By

HASSAN BARJINI

Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfilment of the Requirements for the Degree of Doctor of Philosophy

July 2012

DEDICATIONS

To My dear wife for her encouragement,

and My sweet daughters

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfillment of the requirement for the degree of Doctor of Philosophy

HYBRIDFLOOD ALGORITHMS MINIMIZING REDUNDANT MESSAGES AND MAXIMIZING EFFICIENCY OF SEARCH IN UNSTRUCTURED P2P NETWORKS

By

HASSAN BARJINI

July 2012

Chairman: Professor Mohamed Othman, PhD

Faculty: Computer Science and Information Technology

Unstructured peer-to-peer (P2P) networks, aggregate the slack resources on each peer, which may include bandwidth, storage space, and computing power. As a peer joins this P2P network, the total demand and total capacity of the system simultaneously increase. However, in a typical client-server network, as a client joins the network it only shares its demands, not its resources. Thus as more clients join the client-server network fewer resources are available to serve each client. Besides, the decentralized structure of a P2P network increases its robustness because it removes the single point of failure that can be inherent in a client-server based system.

Therefore the unstructured model of the P2P network has attracted the greatest attention from both users and the researcher communities. Searching is an essential and basic activity for all P2P applications. Thus, there are a large number of research works have focused on unstructured P2P search facilities. There are two main reasons driving the research interest in this area.

First, the upward trend of digital information production, such as HTML, music and image files, requires a scalable information infrastructure that is capable of indexing and searching. Recent studies have shown that more than 97% of information produced worldwide is in a digital form. The amount of digital information is expected to grow exponentially. There are many challenges posed by such a huge amount of information for existing search systems.

Second, compared to traditional centralized networks, unstructured P2P networks are particularly attractive and promising due to their scalability, availability, low cost, easy deployment, and data freshness. Meanwhile the fundamental property of existing and scalable unstructured P2P networks is the high heterogeneity of the peers that participate in the network. The heterogeneity of peers in unstructured P2Ps introduces both challenges and opportunities when designing a P2P network. Flooding is a basic file search procedure in unstructured P2P file-sharing systems. In flooding a peer initiates the file search operation by broadcasting a query to its neighbors, who continue to propagate it to their neighbors. Flooding has no knowledge about network topology nor files or resources distribution, so it offers an attractive method for file discovery in dynamic and developing networks. In the meantime, flooding produces exponentially redundant messages at each hop. Consequently, the growth of redundant messages limits the system's scalability and causes unnecessary traffic in networks.

In this thesis, we combine two search techniques to tackle this issue and improve P2P search performance in terms of search efficiency and the quality of the search results. We proposed two novel search algorithms named QuickFlood and HybridFlood. QuickFlood combines two flood-based searches; flooding and teeming. QuickFlood is performed in two steps, in a first step the algorithm performs flooding in a limited number of hops. In the second step the algorithm follows a teeming search. QuickFlood compared with blocking expanding ring search decreased re-

dundant messages by 70%, increased two-time success rate and decreased latency by 30%. Therefore, the algorithm enhanced unstructured P2P search by increasing scalability, efficiency and reliability of search.

HybridFlood is also performed in two steps. The first step performs flooding with a limited number of hops. In the second step, nosey nodes are selected in each searching horizon. The nosey nodes are nodes which have the most links to other nodes. These nodes maintain the data index of all client nodes. HybridFlood in comparison to the blocking expanding ring search decreased redundant messages by 80%, increased the success rate by 2.5, and decreased of latency by 80%. In the other word the algorithm improved unstructured P2P search's scalability, efficiency and reliability.

We provided analytical studies for flooding, QuickFlood and HybridFlood. The analytical results provided the best hop threshold point for the optimum growth rate coverage and redundant messages from the three systems. It also proved that in HybridFlood, broadcasting messages are reduced by at least an order of magnitude. Thus, the proposed algorithms enhance the performance of the search by reducing redundant messages, increasing the success rate and decreasing latency. The simulation experiments validated the analytical results.

Abstrak tesis dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Doktor Falsafah

ALGORITMA HYBRIDFLOOD UNTUK MEMINIMAKAN KEBERULANGAN MESEJ DAN MEMAKSIMAKAN KECEKAPAN DALAM RANGKAIAN TIDAK BERSTRUKTUR P2P

Oleh

HASSAN BARJINI

Julai 2012

Pengerusi: Profesor Mohamed Othman, PhD

Fakulti: Sains Komputer dan Teknologi Maklumat

Rangkaian tidak berstruktur rakan-kepada-rakan (P2P), mengumpulkan sumber-sumber yang bekerja perlahan pada setiap rakan, yang mungkin merangkumi jalur lebar, ruang simpanan, dan kuasa pengkomputan. Apabila rakan menyertai rangkaian P2P ini, jumlah permintaan dan jumlah kapasiti sistem meningkat secara serentak. Walau bagaimanapun, dalam rangkaian pelanggan-pelayan biasa, apabila pelanggan menyertai rangkaian ia hanya berkongsi permintaan, bukan sumber. Jadi apabila lebih banyak pelanggan menyertai rangkaian pelanggan-pelayan kurang sumber-sumber yang ada untuk melayani setiap pelanggan. Selain itu, struktur tidak berpusat rangkaian P2P meningkatkan keteguhannya kerana ia membuang titik tunggal kegagalan yang boleh wujud dalam sistem berasaskan pelanggan-pelayan.

Oleh itu, model tidak berstruktur rangkaian P2P telah menarik perhatian yang paling besar daripada kedua-dua pengguna dan komuniti penyelidik. Carian meru-

pakan aktiviti asas dan penting untuk semua aplikasi P2P. Oleh itu, terdapat sebilangan besar kerja-kerja penyelidikan tertumpu kepada kemudahan carian tidak berstruktur P2P. Terdapat dua sebab utama yang menggalakkan minat penyelidikan dalam bidang ini.

Pertama, aliran menaik penghasilan maklumat digital, seperti HTM fail-fail muzik dan imej, memerlukan infrastruktur maklumat berskala yang mampu mengindeks dan mencari. Kajian terkini menunjukkan bahawa lebih daripada 97% maklumat yang dihasilkan di selaruh dunia cdalah dalam bentuk digital. Bilangan maklumat digital dijangkakan akan bertambah secara eksponen. Terdapat bangak cabaran ditimbulkan oleh maklumat sebegitu banyak bagi sistem carian yang sedia ada.

Kedua, daripada maklumat yang dihasilkan di seluruh dunia adalah dalam bentuk digital. Jumlah maklumat digital dijangka berkembang dengan pesat. Terdapat banyak cabaran yang ditimbulkan seperti jumlah maklumat yang besar untuk sistem carian yang sedia ada. Kedua, dibandingkan dengan rangkaian berpusat tradisional, rangkaian P2P tidak berstruktur khususnya menarik dan menjanjikan kebolehskalaan, ketersediaan, kos rendah, penggunaan mudah, dan kesegaran data. Sementara itu, perkara asas rangkaian tidak berstruktur dan berskala P2P yang sedia ada adalah kepelbagaian yang tinggi daripada rakan-rakan sebaya yang mengambil bahagian dalam rangkaian. Kepelbagaian rakan-rakan dalam P2P tidak berstruktur mendedahkan kedua-dua cabaran dan peluang-peluang apabila mereka mereka bentuk bentuk satu rangkaian P2P.

Pembanjiran adalah prosedur asas carian fail dalam sistem perkongsian fail P2P yang tidak berstruktur. Dalam pembanjiran rakan sebaya memulakan operasi fail carian dengan menyebarkan permintaan kepada jiran-jiran, yang terus menyebarkan kepada jiran-jiran mereka yang lain. Pembanjiran tidak mempunyai pengetahuan mengenai topologi rangkaian mahu pun fail atau pengagihan sumber, jadi ia menawarkan satu kaedah yang menarik untuk penemuan fail dalam rangkaian

yang dinamik dan membangun. Sementara itu, pembanjiran menghasilkan mesej-mesej lewah di setiap loncatan masing-masing. Akibatnya, pertambahan mesej berlebihan menghadkan kebolehskalaan sistem dan menyebabkan lalu lintas yang tidak perlu dalam rangkaian.

Dalam tesis ini, kami menggabungkan dua teknik carian untuk menangani isu ini dan meningkatkan prestasi carian P2P dari segi kecekapan carian dan kualiti hasil carian. Kami mencadangkan dua algorithma carian baharu yang dinamakan QuickFlood dan HybridFlood. QuickFlood menggabungkan dua carian berasaskan pembanjiran; pembanjiran dan penuh sesak. QuickFlood dilaksanakan dalam dua langkah: dalam langkah pertama algorithma ini melaksanakan pembanjiran di beberapa loncatan terhad. Dalam langkah kedua prosedur pengiraan mengikut satu carian penuh sesak. QuickFlood dibandingkan dengan carian cincin sekatan berkembang mangurangkan mesej lewah sebanyak 70%, meningkat kan kadar kejayaan dua-kali dan pendaman menurun kan sebanyak 30%. Oleh itu, prosedur pengiraan dipertingkatkan carian P2P tidak berstruktur dengan meningkatkan kebolehan untuk diskala, kecekapan dan kebolehpercayaan carian.

HybridFlood juga dilaksanakan dalam dua langkah. Langkah pertama melaksanakan pembanjiran dengan bilangan loncatan yang terhad. Pada langkah kedua, nod-nod yang ingin tahu dipilih dalam setiap lapisan carian. Nod-nod ingin tahu adalah nod yang mempunyai pautan kepada nod-nod lain. Nod ini mengekalkan indeks data semua nod pelanggan. HybridFlood, jika dibandingkan dencan carian cincin sekatan berkembang menurunkan mesej lewah sebanyak 80%, meningkatkan kadar kejayaan sebanyak 2.5, dan menurunkan pendaman sebanyah 80%. Dengan kata lain, algorithma ini meningkatkan kebolehskalaan, kecekapan dan kebolehpercayaan carian tidak berstruktur P2P.

Kami menyediakan kajian analisis untuk pembanjiran, QuickFlood dan HybridFlood. Keputusan analisis memberikan titik ambang loncatan yang terbaik untuk

pertumbuhan kadar liputan yang optima dan mesej lewah dari ketiga-tiga sistem. Ia juga membuktikan bahawa dalam HybridFlood, penyebaran mesej dapat dikurangkan sekurang-kurangnya satu tertib magnitud. Oleh itu, algoritma yang dicadangkan meningkatkan prestasi carian dengan mengurangkan mesej lewah, meningkatkan kadar kejayaan dan mengurangkan pendaman. Eksperimen simulasi mengesahkan keputusan analisis.

ACKNOWLEDGEMENTS

First all, praise is for *Allah Subhanahu Wa Taala* for giving me the strength, guidance and patience to complete this thesis.

I would like to express my sincere gratitude to my supervisor Professor Dr. Mohamed Othman and also my supervisory committee members Associate Professor Dr. Hamidah Ibrahim and Dr. Nur Izura Udzir for their guidance and advice throughout this work in making this a success.

I certify that a Thesis Examination Committee has met on **19 July 2012** to conduct the final examination of **Hassan Barjini** on his thesis entitled "**Hybrid-Flood Algorithms Minimizing Redundant Messages and Maximizing Efficiency of Search in Unstructured P2P Networks**" in accordance with the Universities and University Colleges Act 1971 and the Constitution of the Universiti Putra Malaysia [P.U.(A) 106] 15 March 1998. The Committee recommends that the student be awarded the Doctor of Philosophy.

Members of the Thesis Examination Committee were as follows:

Ali b Mamat, PhD
Associate Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Chairman)

Azizol b Hj Abdullah, PhD
Senior Lecturer
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Internal Examiner)

Kasmiran Jumari, PhD
Professor
Faculty of Engineering and Built Environment
Universiti Kebangsaan Malaysia
(External Examiner)

Xiaodong Zhang, PhD
Professor
Faculty of Computer Science and Engineering
College of Engineering
The Ohio State University United State
(External Examiner)

ZULKARNAIN ZAINAL, PhD
Professor and Deputy Dean
School of Graduate Studies
Universiti Putra Malaysia

Date: 27 August 2012

This thesis was submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfilment of the requirement for the degree of Doctor of Philosophy. The members of Supervisory Committee were as follows:

Mohamed Othman, PhD
Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Chairperson)

Hamidah Ibrahim, PhD
Associate Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

Nur Izura Udzir, PhD
Lecturer
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

BUJANG BIN KIM HUAT, PhD
Professor and Dean
School of Graduate Studies
Universiti Putra Malaysia

Date:

DECLARATION

I declare that the thesis is my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously, and is not concurrently, submitted for any other degree at Universiti Putra Malaysia or at any other institution.

HASSAN BARJINI

Date: 19 July 2012

TABLE OF CONTENTS

	Page
DEDICATIONS	i
ABSTRACT	ii
ABSTRAK	v
ACKNOWLEDGEMENTS	ix
APPROVAL	x
DECLARATION	xii
LIST OF TABLES	xvi
LIST OF ABBREVIATIONS	xx

CHAPTER

1	**INTRODUCTION**	1
	1.1 Background	1
	1.2 P2P Search Architecture	1
	1.2.1 P2P Structured	1
	1.2.2 P2P Unstructured	2
	1.2.3 P2P Hybrid	4
	1.3 Significant of the Research	5
	1.4 Problem Statement	7
	1.5 Objectives of the Research	8
	1.6 Research Scope	8
	1.7 Contributions of the Research	9
	1.8 Thesis Organization	10
2	**LITERATURE REVIEW**	11
	2.1 Introduction	11
	2.2 Fundamental Terms and Concepts	12
	2.3 Overly Network	12
	2.3.1 Flat P2P Network Topology	12
	2.3.2 Tree-based P2P Network Topology	14
	2.3.3 Hybrid P2P Network Topology	15
	2.4 Search Techniques in Unstructured P2P Networks	16
	2.4.1 Blind Search Algorithms	17
	2.4.2 Informed Search Algorithms	27
	2.5 Super-Peer Selection and Searching Algorithms	30
	2.5.1 OceanStore	31
	2.5.2 Brocade	32
	2.5.3 Gnutella	34
	2.5.4 KaZaa	35

2.5.5 Gnutella 0.6 36
2.5.6 eDonkey 37
2.5.7 Supernode Based P2P File-Sharing Networks 37
2.6 Related Works on Search Mechanisms 38
2.6.1 Blind Search Algorithms Review 38
2.6.2 Informed Search Algorithms Review 45
2.6.3 Super-peer Search Algorithms Review 45
2.7 Summary 48

3 **RESEARCH METHODOLOGY** 50
3.1 Introduction 50
3.2 An Overview 50
3.3 Research Framework 51
3.3.1 Problem Formulation 51
3.3.2 Previous Flood-Based Search Algorithm 52
3.3.3 Proposed Schemes 54
3.3.4 Perform Analytical Studies and Simulation Experiments 55
3.4 Experiment Environment 55
3.4.1 Data Collection 55
3.4.2 Network Model Assumptions and Parameters 56
3.4.3 Experimental Setup 63
3.5 Performance Metrics 63
3.5.1 Query Success Rate 63
3.5.2 Number of Redundant Messages 64
3.5.3 Number of Latency 64
3.6 Simulator Overview 64
3.7 Summary 65

4 **ANALYTICAL STUDIES AND EXPERIMENTAL EXAMINES FOR FLOODING-BASED SEARCH ALGORITHMS** 67
4.1 Introduction 67
4.2 Flooding Algorithm across the Hops 67
4.2.1 Trend of Coverage Growth Rate 68
4.2.2 Trend of Redundant Messages 71
4.3 Proposed Performance Metric 75
4.3.1 Critical Metric 76
4.3.2 Critical Metrics in Flood-Based Algorithms 77
4.3.3 Evaluation of Critical Metrics in Flood-Based Algorithms 83
4.4 Analytical Results 86
4.5 Experimental Results 87
4.6 Summary 89

5 **PROPOSED QUICKFLOOD SEARCH ALGORITHM** 93
5.1 Introduction 93
5.2 Proposed QuickFlood Model 94
5.3 First Step of Algorithm 94
5.4 Second Step Description 96
5.5 Second Step Algorithm 97
5.6 Analytical Studies 98
5.7 Experimental Results 101
5.8 Summary 104

6 **PROPOSED HYBRIDFLOOD SEARCH ALGORITHM** 106
6.1 Introduction 106
6.2 Proposed HybridFlood Search Model 106
6.3 First Step of Algorithm 107
6.4 Second Step Description 108
6.4.1 Nosey Node Selection 108
6.4.2 Indexing Procedure in Nosey Nodes 110
6.4.3 Nosey Node Overlay Network 111
6.4.4 Nosey Node Redundancy 112
6.4.5 Discussion on Nosey Nodes 112
6.5 Second Step of Algorithm 113
6.6 Analytical Study 114
6.6.1 Optimum Hop Count to Switch between Two Steps 114
6.6.2 Analytical Study in the Second Step 116
6.7 Experimental Results 118
6.8 Summary 122

7 **CONCLUSION AND FUTURE WORKS** 123
7.1 Conclusion 123
7.2 Future Works 124

REFERENCES 126
APPENDIX I 134
APPENDIX II 150
BIODATA OF STUDENT 165
LIST OF PUBLICATIONS 166

LIST OF TABLES

Table		Page
2.1	Fundamental terms and descriptions	13
2.2	Advantage and disadvantage of blind search algorithms for flat topology	41
2.3	Advantage and disadvantage of blind search algorithms for super-peer	44
2.4	Advantage and disadvantage of informed search algorithms	46
2.5	Compare important super-peer selections and searching techniques	49
3.1	The clip2 topology used in simulation experiments	56
4.1	Notations and description used in the research	91
4.2	The reduced percentages of redundant messages for each algorithm compare with flooding at different topologies (%)	91
4.3	The increased times of success rate for each algorithm compare with flooding at different topologies	92
4.4	The reduced percentages of number of latencies for each algorithm compare with flooding at different topologies (%)	92
5.1	The reduced percentages of redundant messages for each algorithm compare with BER algorithm at different topologies (%)	102
5.2	The increased times of success rate for each algorithm compare with BER algorithm at different topologies	103
5.3	The reduced percentages of number of latencies for each algorithm compare with BER algorithm at different topologies (%)	104
6.1	The reduced percentages of redundant messages for each algorithm compare with BER algorithm at different topologies (%)	119

6.2 The increased times of success rate for each algorithm compare with BER algorithm at different topologies 120

6.3 The reduced percentage of number of latencies in each algorithm compare with BER algorithm at different topologies (%) 121

LIST OF FIGURES

Figure		Page
1.1	P2P applications	2
1.2	P2P search architecture	3
2.1	Flat network topology each nodes act as a client, a server and a router simultaneously	14
2.2	Part of tree-based P2P network topology	15
2.3	Hybrid P2P network	16
2.4	Search techniques in unstructured P2P network	17
2.5	Flooding procedure	19
2.6	Expanding ring procedure	20
2.7	Blocking expanding ring procedure	21
2.8	Teeming procedure with assumed probability = 0.5	21
2.9	Random walk procedure	23
3.1	The research frame works	53
3.2	Peer's fields	56
3.3	Source peer's fields	59
3.4	Query's fields	60
3.5	Nosey node's fields	62
4.1	Compare the coverage growth rate in two topologies	74
4.2	Compare the number of redundant messages in each hop of two topologies	75
4.3	Compare the percentage of redundant messages in two topologies	75
4.4	Number of redundant messages for each algorithm at different topologies	88

4.5 Success Rate for each algorithm at different topologies 89
4.6 Number of latency for each algorithm at different topologies 90

5.1 Pseudo code Flooding(S_k, Q_m, TTL_F, MAX) 95
5.2 First step of QuickFlood procedure 96
5.3 Second step of QuickFlood procedure 97
5.4 Pseudo code QuickFlood_Second_Step($S_k, Q_m, TTL_S, MAX, LN, \theta$) 99
5.5 Compare the number of redundant messages of QuickFlood with different arrangements by other algorithms at different topologies 102
5.6 Compare the values of success rate of QuickFlood with different arrangements by other algorithms at different topologies 103
5.7 Compare the number of latency of QuickFlood with different arrangements by other algorithms at different topologies 104

6.1 Nosey nodes selection 109
6.2 Pseudo code Select_Nosey_Nodes(S_i, LP_i, Q_m) 110
6.3 Pseudo code Select_Nosey_Nodes_Clusters(LP_i, Q_m) 111
6.4 Pseudo code Collect_Index_Nosey_Nodes($NN_i.ID$) 111
6.5 Pseudo code Refresh_Index_Nosey_Nodes($NN_i.ID$) 112
6.6 Pseudo code HybridFlood_Second_Step(LP, Q_m, TTL_S, MAX) 115
6.7 Compare the number of redundant messages of QuickFlood and HybridFlood with different arrangements by other algorithms at different topologies 119
6.8 Compare the values of success rate of QuickFlood and HybridFlood with different arrangements by other algorithm at different topologies 120
6.9 Compare the number latency of QuickFlood and HybridFlood with different arrangements by other algorithms at different topologies 121

LIST OF ABBREVIATIONS

AOL	American On line
APS	Adaptive Probabilistic Search
BER	Blocking Expanding Ring
BFS	Breadth First Search
CAN	Content Addressable Network
CGR	Coverage Growth Rate
CPU	Central Processing Unit
CRI	Compound routing indices
DBFS	Directed Breadth first search
DDS	Distributed Search Solution
DFS	Depth-First Search
DHT	Distributed Hash Table
DNS	Domain Name System
DoS	Denial of Service
DQ	Dynamic Query
DQ+	Enhanced Dynamic Query
ER	Expanding Ring
GUSS	Gnutella UDP Extension for Scalable Search
HGP	Human Genome Project
HLBF	Hybrid Limit-Based Flooding
HTML	HyperText Markup Language
IDC	International Data Corporation
ISP	Internet Service Provider
LI	Local Indices
LRM	Local Relational Model
P2P	Peer-to-peer
PIER	P2P Information Exchange & Retrieval
PLBF	Probabilistic Limit-Based Flooding
QSR	Query Success Rate
RBFS	Random Breadth-First-Search
RW	Random Walk
SBARC	Supernode Based Peer-to-Peer File Sharing System
SETI	Search for Extraterrestrial Intelligence
SR	Success Rate
TLBF	Time To Live Limit-Based Flooding
TTL	Time To Live
ZB	Zettabyte = 10^{21} bytes= 1 trillion Gigabytes

CHAPTER 1
INTRODUCTION

1.1 Background

The peer-to-peer (P2P) system is the best-known application in the area of distributed computing. It consists of interconnected nodes that can self-organize themselves without requiring intermediation or a centralized server or authority [1]. The main goal in a P2P system is to tie together and combine small resources, such as storage, bandwidth, and CPU, from each node. The system then constructs a large-scale distributed application, instant messaging, Internet telephony, distributed database system, and content distribution.

P2P systems have therefore become extremely popular, and resulted in an attractive application for millions of users around the world. Thus, the major portion of Internet traffic belongs to P2P applications. For example, Napster, with about 50 million users around the world, appears to be one of the most famous music stores on the web [2]. Figure 1.1 presented P2P applications.

1.2 P2P Search Architecture

A P2P system from a search perspective is classified as: structured, unstructured and hybrid [3]. Figure 1.2 presents peer-to-peer searching architecture.

1.2.1 P2P Structured

This type of architecture, like CAN [4] and Chord [5], has a tightly defined overlay topology, which assigns a unique search path to any lookup request. In other words, they have a close connection between the P2P topology and location of the data, which commonly makes use of Distributed Hash Tables (DHTs [6]). In this architecture, no servers are used, and resources are distributed evenly between

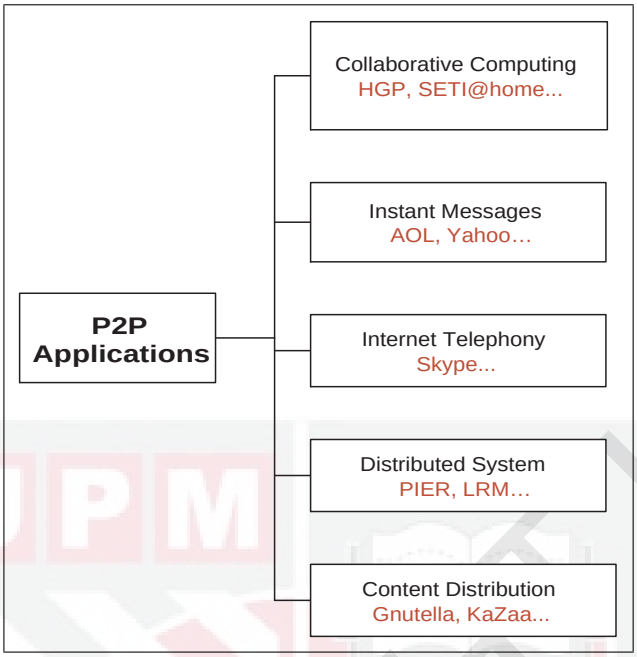

Figure 1.1: P2P applications

all participating nodes in a predefined manner. Thus, when peer looks up certain information, it uses distributed indexes or (DHTs) to find the information's storage place. Although its search overhead is small, the maintenance overhead is huge especially for highly dynamic system [7]. Structured P2P networks are more suitable for static environments [8] than for dynamic, evolving P2P networks.

1.2.2 P2P Unstructured

The unstructured P2P system has no central directory or any precise control over the network topology. The advantage of eliminating the server is that it provides freedom for the participating peers to swap information between each other. The designs of unstructured P2P are extremely resilient to nodes entering and leaving the system. Unstructured systems can be further classified into centralized and decentralized ones.

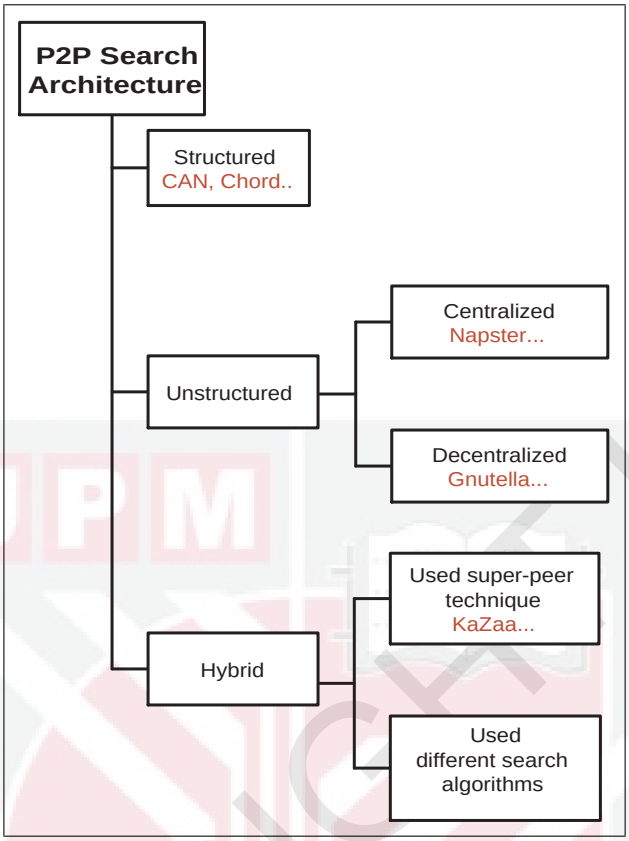

Figure 1.2: P2P search architecture

P2P Centralized Unstructured

Centralized unstructured P2P networks follow the classical client-server architecture. In general, servers store information about resources that other peers provide. The central servers use keep-alive messages to check the state of each node in the network and keep the document indexes up-to-date [9]. A well-known type of this network is Napster. In Napster, when a peer wants to find a certain resource it sends a query to the server (often called a directory) that returns a search result. The centralized unstructured network has the fastest search mechanism, but it has a single point of failure and is vulnerable to Denial of Service (DoS) attacks.

P2P Decentralized Unstructured

In this type of P2P network there is no centralized directory (server) or control over the network topology. A peer can freely join and leave the network at

any time. Furthermore, it selects its neighbors arbitrarily. Due to the absence of topological constraints such a system produces a better performance. Decentralized unstructured P2P networks need less maintenance overhead due to their extremely dynamic condition where peers can join and leave frequently and concurrently. Thus the unstructured P2P network topology is highly robust to failures or node transience. Gnutella [10] is an example of such a system.

1.2.3 P2P Hybrid

The hybrid strategy of P2P has two different approaches in the literature. The first approach proposed is a two-tiered overlay structure which divides peers into two groups, such as ultrapeers or super-peers and leaf peers or clients. The second approach simply tries to combine two different architecture types to boost the overall system performance.

In the first approach, clients are located at the edge of the network. They have no responsibility for any routing. The leaf nodes are connected to the overlay through a few super-peers or ultrapeers. The super-peers keep an index of its clients' data, in order to process the query for its clients. Thus, whenever a client receives a query it forwards the query to its associated super-peer. Then that specific super-peer will process the query on its clients' behalf. If the super-peer locates the answers, it will return a response message. This response message or query hit message contains the results and the address of each client whose collection generated the result. If the super-peer did not find the result, it will forward the query message to other super-peers. Gnutella 2 [11, 12] or KaZaa [3] are both based on these hybrid architectures.

In the second approach of the hybrid strategy there is no difference between the peers. Peers in this architecture are acting the same or equally. They are sometime acting as a server, client or router. The hybrid frameworks here try to combine

the advantages offered by the two different types of algorithm.

1.3 Significant of the Research

A recent report conducted by IDC in 2010 [13] has shown that there is an upward trend to digital information production. The volume of digital information from 0.8 ZB in 2009 will increase to 35 ZB in 2020. For the first time ever the total volume of digital content will exceed the entire storage capacity. It is speculated that by 2011 only half of the digital universe will be stored. This trend towards the production of information requires a scalable infrastructure capable of indexing and searching rich content, such as HTML, music and image files [14].

A traditional solution is to construct a centralized searching server, such as that which all current search engines operate. This kind of solution needs to maintain an enormous centralized database about all the online information. They require a large amount of sophisticated hardware and high-performance software to pretend to be scalable and available. The only advantage of centralized search servers is that they can provide exact-match queries [15]. However, keyword searches are more prevalent, and more important than exact-match queries. The disadvantages of centralized searching servers are that they impose their censorship policy, use privacy of users, there is no freshness of responding guarantee, and there is the issue of the threat of a single point of failure.

Therefore, the reasonable solution is to use a scalable, available, low cost, and easily deployed system. P2P networks are distributed systems consisting of interconnected nodes, which provide scalability, fault tolerance, decentralized coordination, anonymity, and self-organization. It is defined as an overlay network built by a set of nodes on top of a physical network infrastructure. Its operating protocols acts as a distributed network, which do not rely on any specific dedicated server for communication; instead it consists of interconnected nodes. All peers in such

a network are symmetric and corporative: they can act as a client, a server, or a router [9]. The P2P system is an interesting and promising alternative for the following reasons.

- Scalability: In a P2P system, maintaining the performance attributes is independent of the number of nodes or documents. There is a minimal effect on performance with a dramatic increase in the number of nodes or documents.
- Availability: Availability requires stability in the presence of failure or changing of the node population. The self-organization and fault tolerance nature of the P2P system guarantees its stability and persistence. Authorized users are ensured access to data and related resources when required.
- Low cost and deployments: A P2P system is practically connected together, with very low resources in its nodes. It can increment its deployment when any new node joins the system.
- Data freshness: Once a node appears it can publish its documents immediately. Thus the freshness of data in P2P systems is guaranteed.

Flooding broadcasts the query message from the source peer to all its immediate neighbors, who continue to propagate it to their neighbors [16]. Flooding is the most frequently used [17, 18] search scheme in unstructured P2P networks. Flooding and unstructured P2P networks follow the same natures. Both have no knowledge about network topology or file distribution. Thus flooding offers an attractive method for file discovery in dynamic and evolving networks. Flooding with these interesting features is widely used in unstructured P2P networks, such as Gnutella2, KaZaa [19], and BitTorrent [20].

1.4 Problem Statement

Due to the advantages of unstructured P2P networks that were highlighted in the preceding section, it has scalable storage for saving enormous volumes of digital information. It can support the availability and freshness of its information, and uses the slack in its resources in its nodes for deployment. Despite this advantage, searching and indexing in unstructured P2P networks is the big challenge to be tackled in the literature [21, 22].

The main problem of searching in an unstructured P2P network with such a huge amount of information is its performance. The existing searching methods have low success rate, enormous volume of redundant messages, high latency, and small coverage.

The fundamental search technique used in unstructured P2P networks is flooding. Although it has well-known merits, such as a simple algorithm, high reliability, moderate latency, and large coverage, it seriously limits the system's scalability. In order to tackle this negative aspect of flooding, the main issue can be decomposed into two well-known and solvable sub-problems.

- Although flooding has considerable merits, it produces a huge amount of redundant messages. The number of query messages grows exponentially with the hop count [23]. These redundant messages increase the peer processing in the network, without enlarging the propagation scope.
- In addition, flooding follows a blind search algorithm discipline and therefore has no opportunity to make use of the advantage of node heterogeneity from participating in unstructured P2P networks.

1.5 Objectives of the Research

The main objective of this research is to propose a new search algorithm in order to take advantage of the flooding search algorithm and super-peer merits to enhance the search performance in unstructured P2P networks. The secondary objectives of the research are:

- To propose an optimal point of a hop in the pure flooding search algorithm to provide the minimum amount of redundant messages and maximize the search efficiency.
- To propose an optimal threshold for a hop to switch from the flooding search algorithm to the proposed QuickFlood and HybridFlood algorithms.
- To compare our novel algorithm with prevalent search algorithms.

1.6 Research Scope

This research mainly focuses on the search content in unstructured P2P networks. The research excludes structured and centralized unstructured P2P networks. The study uses an overlay network as an interconnecting layout for nodes, and thus the physical layer is not considered in the study.

It is assumed that the overlay network is an undirected random graph in which each node is represented as a peer, and they are connected to each other by edges. Forwarding messages are in the form of bidirectional fashion between nodes. This means that messages may be transferred in either direction along the edges. There is no difference in broadband between peers in this study. A message that travels from node A to node B must travel along a path in the graph. Each hop is the distance from any nodes to its immediate neighbors.

The length of a path is known as the number of hops taken by the message. Two nodes are said to be "*n hops apart*" if the shortest path between them has the

length n. The number of hops inside the overlay is completely different from the number of hops in the physical network.

The source node sends a query message to its neighbors by using the routing policy mechanism. Always positive respond to message back along the same path as a query message was carried. It is assumed that peers are honest and cooperative, thus malicious and free-rider peers are excluded and are out of scope of this research.

1.7 Contributions of the Research

This study accounts for both analytical and algorithmic contributions.

- By the analytical contributions it has provided:

  A. The optimum point of a hop count for the minimum number of redundant messages and maximum rate of coverage growth in a pure flooding search algorithm.

  B. The optimal threshold of hop counts to switch from the flooding search algorithm to the QuickFlood algorithm.

  C. The optimal threshold of hop counts to switch from the flooding search algorithm to the HybridFlood algorithm, and for the number of query messages propagated to be cut down by at least an order of magnitude in the HybridFlood algorithm compared to the flooding search algorithm.

- By the algorithmic contributions it has provided:

  A. QuickFlood and HybridFlood proposed novel algorithms for minimum cost and maximum search efficiency.

1.8 Thesis Organization

This study is divided into seven chapters. Chapter 1 serves as an essential introduction to the work. Chapter 2 gives an overview of the basic concepts and related works and reviews, and discusses the existing search procedure in unstructured P2P networks. Chapter 3 introduces the methodology. Chapter 4 explains the optimum hop count in flood-based algorithms. Chapters 5 and 6 design and describe the QuickFlood and HybridFlood search algorithms, and finally Chapter 7 concludes this study and discusses potential areas of future research.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The most popular application in P2P, until now, is file-sharing [24]. There are three different types of functional unit in a file-sharing P2P network. These functional units may be called *provider*, *consumer* and *service*. The *provider* provides the information, the *consumer* is the one who requests the information, and *service* is what provides the efficient facility and effective search for the relevant information. In the other words, *consumer* and *provider* may also be called *client* and *server*, and the *service* is similar to a *router*. A peer may act as a single practical unit or as a combination of multiple functional units. It can act as both, or even as all functional units, at the same time. A peer may act as a *provider* (*server*) and *consumer* (*client*), or as a *provider* (*server*), *consumer* (*client*) and even *service* (*router*). There are logical connections between peers which are established at a protocol layer. The logical connections serve as data channels. This channel exchanges information in the form of messages between peers.

Search is a basic and essential activity for all P2P applications, especially in a file-sharing system. The two components of a P2P network are basic to the searching procedure: *overlay network topology* and *search mechanism*. The *overlay network topology* defines the practicality and responsibilities of each type of functional unit, as well as the relations between peers with different types of functional unit. A *search mechanism* specifies the process of the search with a set of protocols describing how the contents are represented and used for searching.

The purpose of this chapter is a comprehensive review of the literature that is relevant to the present study. The chapter begins with identifying the fundamental terms and concepts, then overlay networks are explained, and the most common search techniques used in unstructured P2P networks are reviewed.

2.2 Fundamental Terms and Concepts

P2P definitions: P2P has many definitions in the literature; however one of the most complete definitions has been stated in [1] as: "*Peer-to-peer systems are distributed systems consisting of interconnected nodes able to self-organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority.*"

Table 2.1 presents fundamental terms and concepts used in this study. Throughout this study, the terms *node*, *peer*, *user*, and *servent* are used interchangeably.

2.3 Overly Network

An overlay network is a set of logical connections used to organize the peers in the network in a protocol layer. In the other words, the overlay network or P2P network [33] is a virtual network of nodes and logical links that is built on top of an existing network with the purpose of implementing a network service that is not available in the existing network. The functioning protocol in the overlay network is such as the Internet. In a P2P network, each node keeps a limited number of connections with other peers, which are called its immediate neighbors, the graph of a peer's connections establishes the overlay network. The overlay network can be classified [34] as: flat [35], tree-based [36], and hybrid-based P2P network topologies [37–40].

2.3.1 Flat P2P Network Topology

In this category [35, 41], all nodes play the same role, and they have equal responsibility. The nodes keep a set of *neighbors* to create an overlay network. This node

Table 2.1: Fundamental terms and descriptions

Term	***Description***
Peer	Also called *node* [25] or *user* [26], is a participating entity in the network. Peers act as a *server* or *client* and even *router*, so is also called a *servent* [27].
Overlay network	*Overlay network* [28] is a virtual network which consists of nodes and logical links that are built on top of the physical network. The purpose of overlay network is to implement a network service that is not available in the existing network.
Hop	*Hop* is the virtual distance between a peer with its immediate neighbors in an overlay network.
Hop count	*Hop count* is the number of hops that are counted in an overlay network.
Low-hops	*Low-hops* are the sequence of initial hops [16].
High-hops	*High-hops* are the sequence of final hops [16].
PeerId	*PeerId* is a unique random number which is assigned to any peer for identification.
Source peer	*Source peer* also called *routing peer* [29] or *query originator* [30] is a peer who publishes a query message and broadcasts it to other nodes, and receives results.
Query message	The *query message* [29] is a message that is sent to other peers in order to find the object. It is published by the source peer and contains the following fields; queryId, requested-keywords, TTL value, and source-peer' s Id.
QueryId	The *QueryId* is a unique random number assigned to each query for identification.
Requested-keywords	The *Requested-keywords* field contains search keywords, which are received from a user.
Source-peer's Id	*Source-peer's Id* is identical to the peer's Id for a typical peer who is now a source peer.
TTL	The TTL is an the abbreviation for Time-to-Live, which is indicated by the number of hops or life time that a query message can traverse.
QueryHit	A peer receiving a query which has math within its data generates a *QueryHit*. The *QueryHit* message back along the source peer same path that received query [31].
Churn rate	The *Churn rate* is the average fraction of peers who move or are out of the network for a specific period [32].
Time step	The *time step* is the amount of time a cycle of the simulation takes to complete.

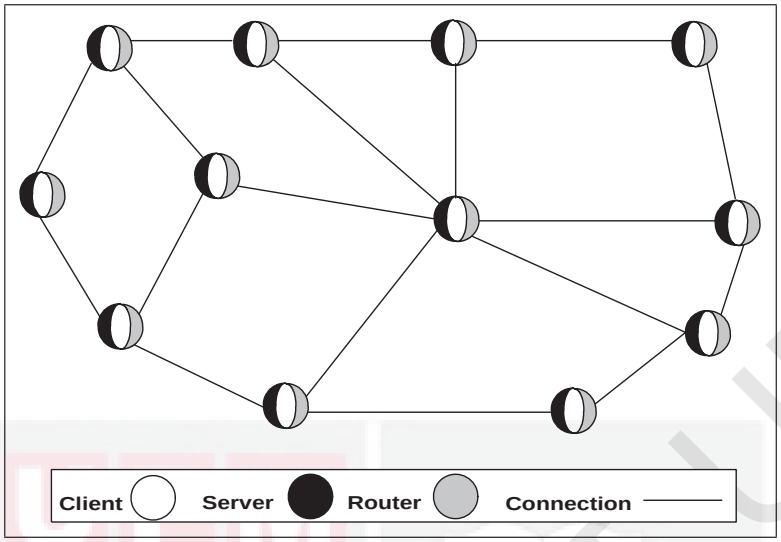

Figure 2.1: Flat network topology each nodes act as a client, a server and a router simultaneously

structure may simultaneously act as a *client*, *server*, and also as a *router*. Nodes in a flat network topology have no knowledge about the content of other neighbors. Hence, as well as a node request for an object that is not available in its content it has to forward the query to other nodes. A flat P2P network topology is an unstructured P2P network [34]. A typical flat topology is presented in Figure 2.1.

2.3.2 Tree-based P2P Network Topology

In this kind of topology [36, 42], data are propagated along a distributed tree that is rooted at the source peer. This type of topology cannot take full advantage of the network's resources, and the system can easily become unbalanced due to the bandwidth of the leaf peers not being utilized. Furthermore, the system is vulnerable to peer churn since at any time any middle peer's departure will disconnect all descendants of the peer from the content forwarding tree. Many attempts have been made to resolve this limitation, such as multiple tree-based topologies [43]. Such study is out of this scope Figure 2.2 presents part of a typical tree-based network topology.

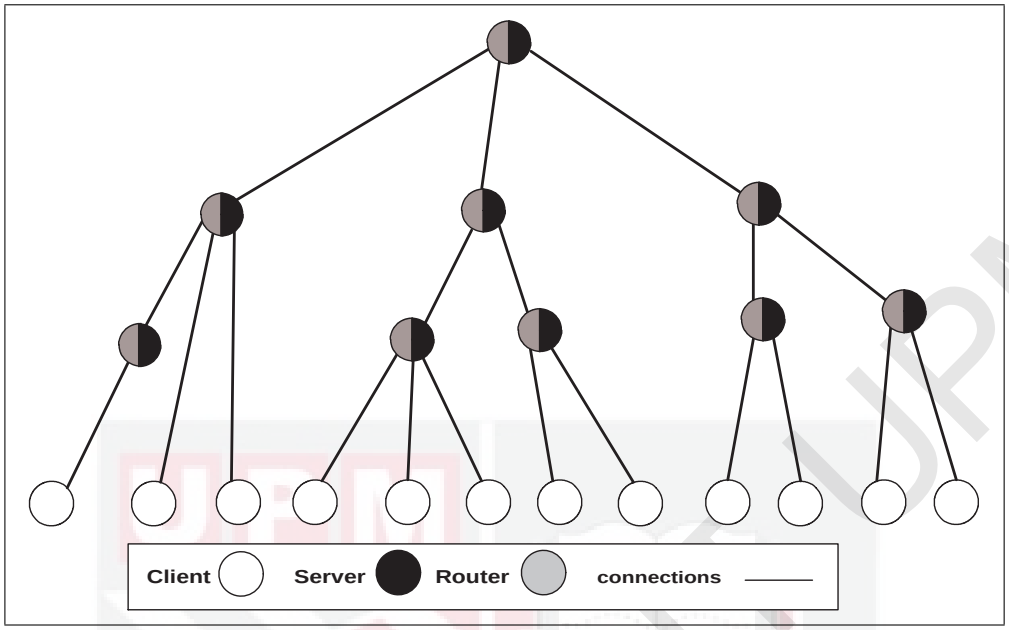

Figure 2.2: Part of tree-based P2P network topology

2.3.3 Hybrid P2P Network Topology

In a Hybrid P2P Network Topology [37–40, 44, 45], the nodes are classified into two classes, super-peers and ordinary peers. Each super-peer and its clients are connected and formed a cluster. Ordinary nodes are connected to super-peers and provide their local objects to the super-peer. Super-peers maintain the index of all the objects which are available in their cluster. Super-peers are connected to themselves and form an overlay network similar to the overlay of the flat P2P network. The search begins with a request which is published by an ordinary node. The ordinary node forwards its query message to the relevant super-peer. The super-peer first checks its indexes, and if it is found then sends the object's location to the requesting node. Otherwise the super-peer begins to search among the other super-peers, in exactly the same way as performed in a flat P2P network. Figure 2.3 presents a typical hybrid topology.

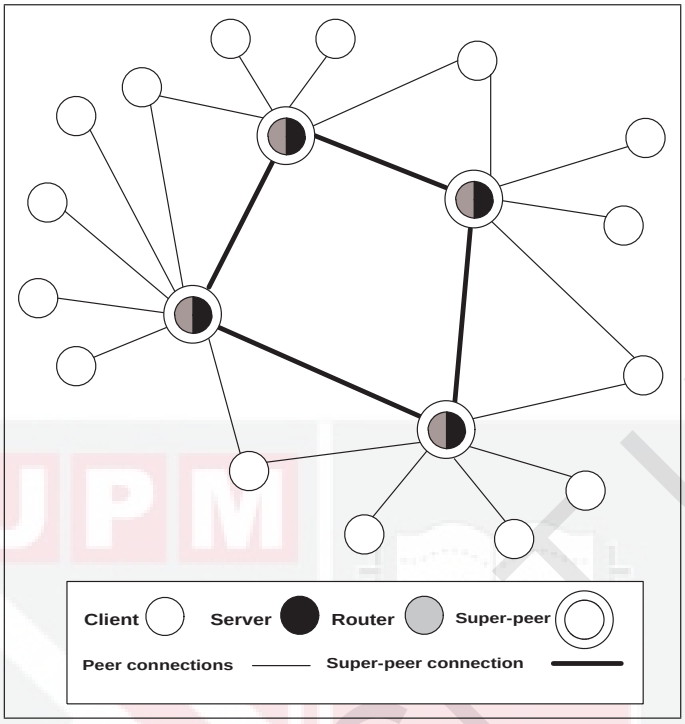

Figure 2.3: Hybrid P2P network

2.4 Search Techniques in Unstructured P2P Networks

The main objective of each search algorithm is to successfully locate resources while incurring low overheads and delay [46]. In an unstructured P2P network, there is neither a centralized index nor strict control over the network's topology or file placement. Designing an efficient search in such a system is difficult, due to its scale and the unreliability of individual peers. Many attempts have been made to develop an efficient search system for this type of P2P network [32, 47, 48]. We can classify search in this area as blind and informed search. The blind search can be further classified as a blind search in flat topology or flood-based algorithms, and a blind search in super-peer topology [49].

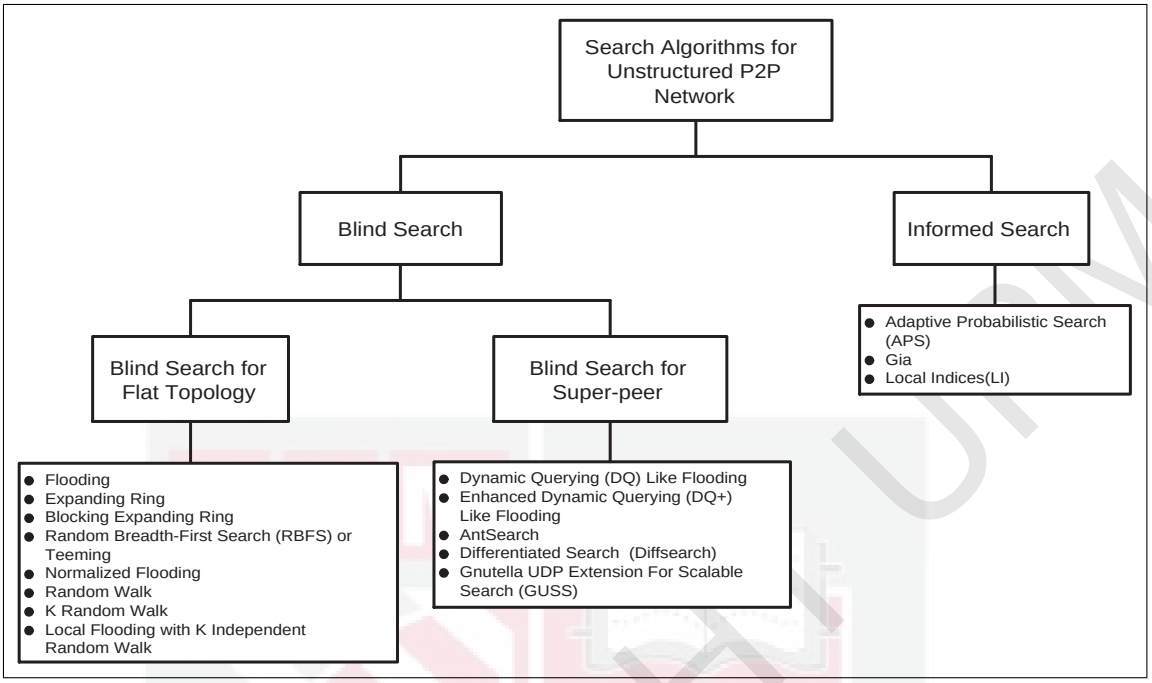

Figure 2.4: Search techniques in unstructured P2P network

2.4.1 Blind Search Algorithms

Blind search schemes utilize flooding methods to broadcast the queries to peers in the network. Peers keep no information about the P2P network or object locations for routing queries. Features such as scalability, success rate, and latency are employed to evaluate the performance of a search algorithm. This searching scheme can be classified as a blind search for flat topology and blind search for super-peer. Figure 2.4 presents search algorithms in unstructured P2P networks.

Blind Search Algorithms for Flat Topology

This kind of searching technique employs a flooding algorithm to forward queries. Peers have no information on the network topology and file distribution. Metrics such as scalability, success rate and latency are employed to evaluate the performance of these search techniques. Scalability of search can be measured by the number of redundant messages produce by the search algorithm.

Flood-based [35] techniques are searching techniques that substantially follow the

Breadth First Searching [50] (BFS) algorithm. The aim of these methods is to expand and examine all nodes of a network. All nodes in this approach are utilized equally, and they do not have the opportunity to exploit the node's heterogeneities and source locations. This kind of search does not use the heuristic algorithm.

Flooding

Flooding, or the BFS algorithm, has no knowledge about the overlay topology and file distribution. Therefore it is attractive for searching in unstructured P2P networks [51]. It starts from a source peer with an initial Time-to-Live value (for instance, $TTL = k$), and broadcasts a query message in a hop-to-hop fashion counted by the TTL count. TTL is decremented by each hop. Each node acts as both a sender and receiver, and each node tries to forward every message to every one of its neighbors except the source node. A message comes to end when it either receives sufficient information to respond, its TTL reaches zero, or because it becomes a redundant message. Figure 2.5 presents a sample flooding procedure in an unstructured P2P network.

Expanding Ring

The Expanding Ring (ER) search is the first control-TTL-based flooding algorithm [52]. It uses successive flooding searches, with different TTL value. The source peers start by sending query messages to its immediate neighbors with the initial TTL value. The query messages come to end when either it receives sufficient information to respond, its TTL value reaches to zero, or because the messages are redundant. If no adequate query message is received within a time-out interval, the source peer assumes that this attempt has failed. Therefore, it starts a new round of searches with a larger TTL value. This process could be repeated over multiple rounds. Obviously, by controlling the TTL value this scheme reduces the huge broadcasting overhead. It is widely used in multi-hop networks [53], such as ad-hoc and sensor networks [54]. Figure 2.6 presents a sample of an expanding

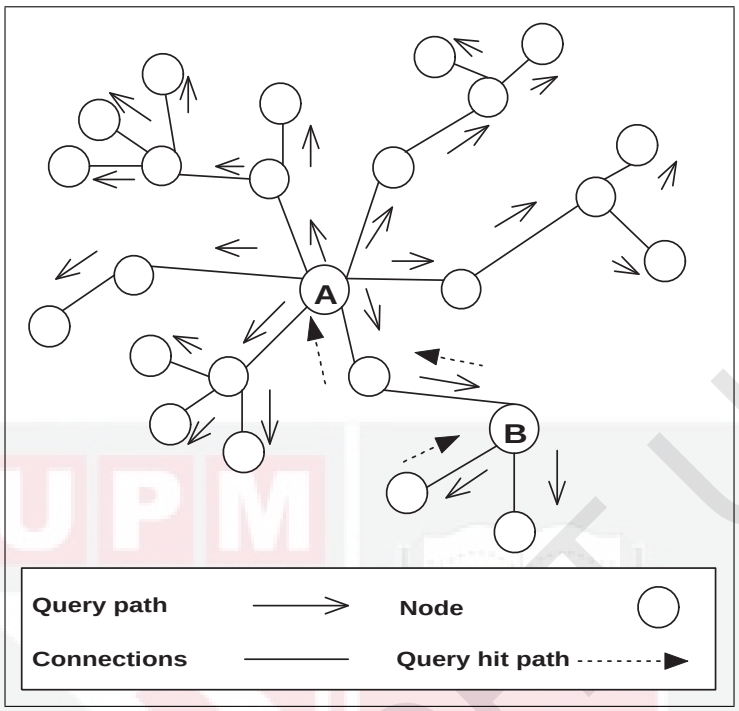

Figure 2.5: Flooding procedure

ring.

Blocking Expanding Ring

The Blocking Expanding Ring (BER) search [55] is an adapted version of the expanding ring search and is similar to iterative deepening [34, 49]. This is also a successive flooding search with a different TTL value. It starts at the source peer with a query message with an initial TTL value. The default message comes to end when it either becomes a redundant message or because its TTL value has expired or it has received an adequate response. In this process, if no sufficient response is received within a time-out interval, the source node assumes that this attempt has failed. Hence, it starts a new round of searches with a larger TTL value, but not from the source node, but rather from all nodes that took part in the last attempt. The main difference between this algorithm and ER concerns the rebroadcast procedure. In BER the rebroadcast procedure starts from all the nodes involved in the previous attempt. Figure 2.7 presents the blocking expanding

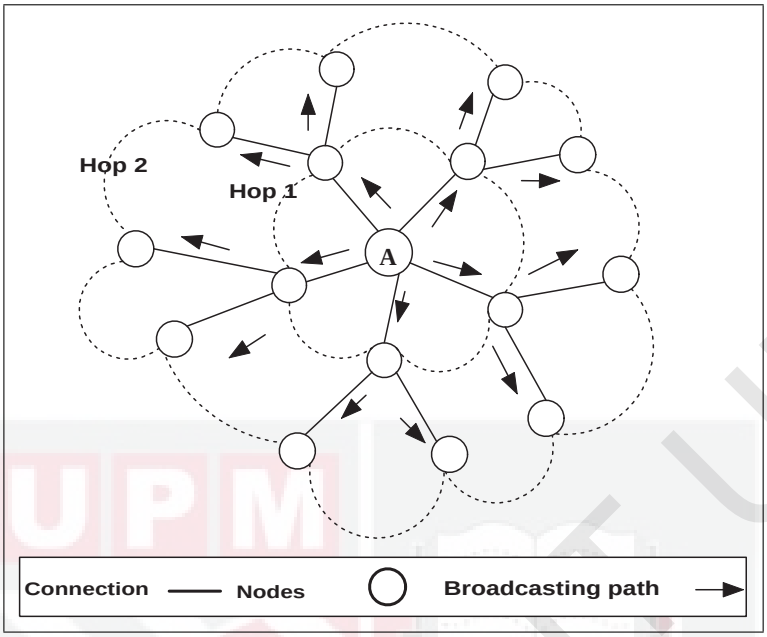

Figure 2.6: Expanding ring procedure

ring technique.

Random Breadth-First-Search (RBFS) or Teeming

Random Breadth-First-Search [56] or teeming [10] is an uninformed search algorithm that has been proposed as an alternative to basic flooding. This variation of the flooding algorithm restricts the search space, such as by random walkers or paths [56, 57]. It is probabilistic flooding, which based on flooding, but forwarded the query to only certain percentage (θ) of the node's neighbors at each step.

In this algorithm, at each step if the files or data are not found in the local cache of a node then the node only propagates the inquiring message to a random subset of its neighbors. This technique uses a fixed probability such as θ for selecting a particular neighbor. The teeming algorithm, in contrast with flooding, is no longer d_ary but is $d{\times}\theta\_ary$. Figure 2.8 presents a sample of a teeming algorithm.

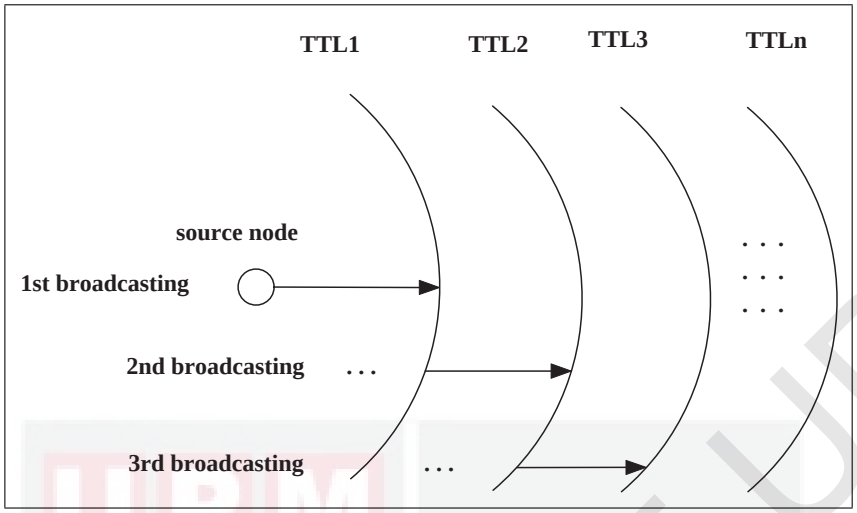

Figure 2.7: Blocking expanding ring procedure

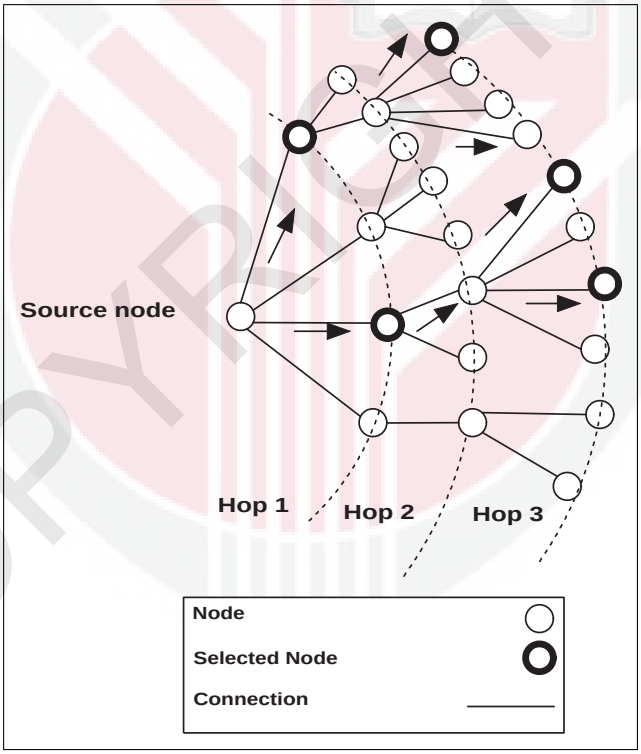

Figure 2.8: Teeming procedure with assumed probability = 0.5

Normalized Flooding

This technique [58] is same as flooding; each node only sends a message to a subset of its neighbors. The subset is selected based on the minimum degree in the

network. Assume that δ is the minimum degree of a node in the network. If a new visiting node has a degree of more than δ, then the query will only be forwarded to a δ subset of its neighbors. This subset node is selected uniformly in a random fashion. In this algorithm for d is not properly selected then low degree nodes that are hosting good content will be left out during searching. Other features of nodes such as a number of objects, storage and bandwidth are not considered in this algorithm.

Random Walk

A Random Walker is modified version of flooding, which forwards a query message (walker) [57] to only one randomly chosen neighbor. The message comes to end when either the number of results is satisfied, a message becomes redundant, or when its Time-to-Live expires. It is a well-known searching protocol based on Depth-First Search (DFS) search algorithms.

To speed up the respond time in Random Walker, versions of the random walk are adopted such as *k-walker* [57] and two level random walkers. The *k-walker* [58] (Random Breadth-First Search (RBFS) or teeming) forwards a query message (walker) to a k random subset of its neighbors. Figure 2.9 presents the random walk technique.

K Random Walk

Qin Lv et al. in [57], have proposed a new version of the random walker to reduce the delay and increase the number of walkers. This method is called the *k-walker* algorithm. In this new technique, instead of just sending out one query messages in each hop, the requesting peer sends out K query messages, and each query continues its own random walk. This algorithm after T hops expects KT walkers. Therefore, the delay should be reduced by a factor of K.

To terminate the walkers, this algorithm proposes two methods; TTL and checking. TTL is similar to flooding. Each walker is terminated after a certain number of

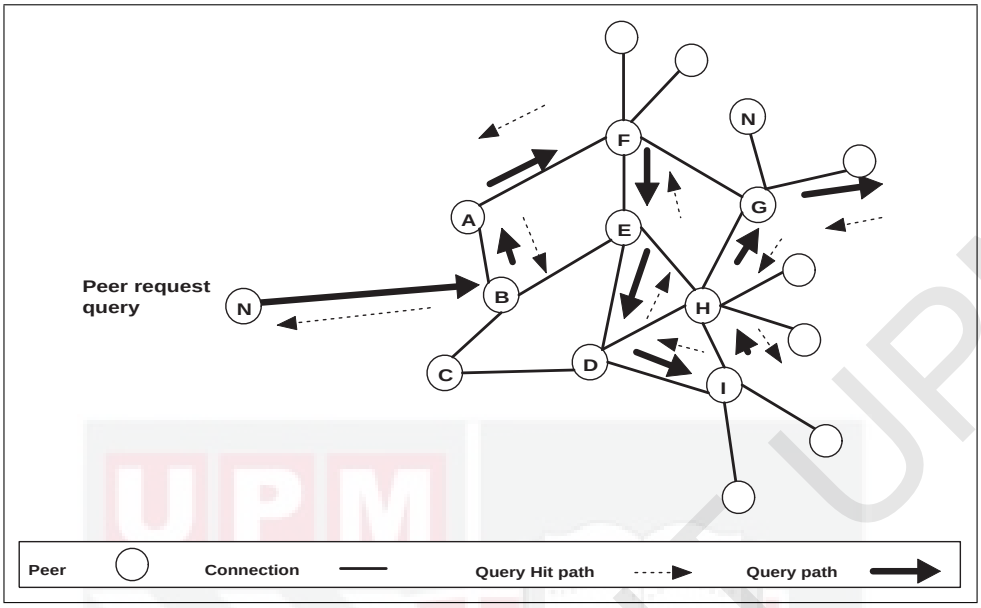

Figure 2.9: Random walk procedure

hops. For checking, each Walker periodically checks with the source peer before walking to the next node.

There is another version of the random walker called two-level [59]. In the first step of the two-level query the nodes select k_1 random walks with $TTL_1 = l_1$, when this step is finished the last node chooses k_2 random walks with $TTL_2 = l_2$. This technique generates fewer duplicate messages but has a longer search delay than the *k-walker*.

Local Flooding With K Independent Random Walks

This search method [58] is a compromise between flooding and random walk. In the first step the algorithm follows flooding until it precisely discovers k new outer nodes, for some predefined value of k. If one of these nodes hosts the object the search is successful and the process is terminated. Otherwise, each k node starts an independent random walk. If the files or data are located close to the source peer, then local flooding would be sufficient to locate it quickly with just few messages exchanged. If the file is located far away from the source peer, it is expected that it will be located by one of the random walks. Meanwhile, the flooding occurs

only locally the message complexity is small. If the search is continued by the independent random walk after local flooding with large TTL values, the message production will be huge and the performance will be decreased.

Blind Search Algorithms for Super-Peer

The few blind search methods designed for unstructured P2P networks are based on super-peer to controlled flooding. Dynamic Querying (DQ) like flooding, Enhanced Dynamic Querying (DQ+) like flooding, AntSearch, Differentiated, and Gnutella UDP extensions for Scalable Search are the nominated techniques for review

Dynamic Querying (DQ) Like Flooding

This flooding technique is designed for super-peer architecture [24]. The main goal of this method is to estimate the size of the query's popularity. In DQ the source peer sends a probe query to a few neighbors with a small TTL. If the probe does not receive the number of results. Then the source peer begins an iterative process to dynamically evaluate the TTL for the remaining neighbors.

Such controlled flooding reduces much of the network traffic [60, 61]. DQ has two benefits, first it avoids sending query packets too far. Second, it does not send query packets repeatedly to the same subset of peers. This controlled flooding technique has a minimum search cost, while it increases the latency perceived by the peers. The reason that this algorithm increases the latency is that it is very *conservative* in propagating the query packets to the network

Enhanced Dynamic Querying (DQ+) Like Flooding

Jiang et al. in [62] proposed a new algorithm to achieve a lower search cost. This new algorithm is called Enhanced Dynamic Querying (DQ+). The main difference between these two algorithms (DQ and DQ+) concerns the iterative process of the enhanced algorithms. In DQ+ each iterative process follows two main polices. First, the iterative process is *greedy*: the source peer propagates the query to a new neighbor wanting to determine all the required results from this

neighbor alone. Second, the iterative process is *conservative*: at the same time the source peer avoids propagating the messages to all peers, which made overshooting problem. The source peer uses a confidence interval method to provide a safety barrier on the estimation of the popularity of the search item. Compared to the DQ, the DQ+ query is only flooded to a small number of the peers. Thus, this technique performs well with respect to search latency. The main defect of this algorithm is when searching for unpopular items, as the search cost for this item is high.

AntSearch

Eytan et al. in [63] shows that 70% of users are not sharing any files, these kind of users are called free-riders [64]. Free-riders are those who use the system for free. The problem is that free-riders produce a large amount of redundant messages in the system. AntSearch [24] is a developed version of Enhanced Dynamic Querying. It designed for solving the problems of free-riding while searching unstructured P2P networks. AntSearch performs its algorithm in three phases:

- Assignment phase: In this phase pheromone value is assigned to the immediate nodes. Each peer maintains its hit rate of the previous queries, and records a list of pheromone values for its immediate neighbors.

- Probe phase: In this stage the requester peer sends queries to a few neighbors with $TTL = 2$. From this probe's queries it estimates the popularity of the target file. After this phase, the requester peer collects the statistical information about the search file. From this information, the requester peer can predict how many results may be retrieved when each step only floods the query to k% of its immediate neighbors. All the information will be collected in the probe table.

- Flooding phase: The requester peer has to decide the two parameters for the query flooding, the k value and TTL value. The k value identifies the percentage of neighbors that must be chosen to flood a query. The TTL value represents

the upper bound of the flooding hops. The requester peer will now propagate the query to the $k\%$ of neighbors by the estimated TTL, and only to neighbors which have higher pheromone values.

The AntSearch algorithm significantly reduces redundant messages during query flooding. Thus it decreases more overhead messages, and its performance improves over time because the node has learnt more information about its neighbors. However, it has a high search cost, because each time it must evaluate the pheromone of its neighbors and produce a probe table. **Differentiated Search (DiffSearch)** *Wang et al.* in [65] proposed a new search algorithm based on an ultrapeer overlay. The DiffSearch algorithm selects an ultrapeer as a peer with a high querying reply capability. Ultrapeers form an overlay and ordinary peers serve as leaf peers. Leaf peers upload their indices to the ultrapeer, and allow them to be shared by differentiated searches. The main idea behind using an ultrapeer is based on the work of *E. Adar et al.* [63], which shows that only 1% of peers answer the main portion of a query. Hence, by routing queries to selected peers it is possible to save up to 90% of query traffic. Search using this algorithm is performed in two rounds; in the first round the query is only forwarded to the ultrapeer overlay. If the search result in the first round fails, the second round starts by querying the entire network. The precondition in the DiffSearch algorithm is that the ultrapeer overlay must consist of content-rich peers who are well formed in a P2P network. The load-balancing problem in DiffSearch has been solved by uploading indices of the leaf nodes to the ultrapeers. The main advantage of the differentiated search technique is that it extensively reduces the search traffic due to a reduced search space. However, the pure flooding search in the second round produces more overheads on the network.

Gnutella UDP Extension for Scalable Search (GUSS)

Search using this technique [57, 66] is conducted by iteratively contacting various super-peers and having them question all their leaves, until sufficient objects are found. The important feature of GUESS is that query messages are not broadcast by a flooding-based algorithm. In addition the peer merely iterates through the entries in its link cache, and performs a search on the target peer. In GUESS, the number of leaves per super-peer must be kept high, in order to receive sufficient results. However, the larger number of probes produces a larger cache size, but they do not guarantee more satisfied queries. There are fewer numbers of peers who wish to share the large number of objects, thus many queries will go unsatisfied.

2.4.2 Informed Search Algorithms

In this type of searching technique peers retain some kind of routing information about forwarding the query to suitable peers. The range of this information is based on different parameters, such as the popularity of objects, success rate and so on. This class of search techniques refers to the *weighted selection of relay neighbors* [67]. In this class, instead of randomly selecting neighbors, some techniques have been proposed to select neighbors more objectively.

The informed search offers a smaller response time to blind searching. while it increases the cost of the search and the overhead of maintaining indexes. Although the focus of the thesis is outside informed searches, the following section reviews and compares the most prevalent informed searching techniques in unstructured P2P networks.

Adaptive Probabilistic Search (APS)

In this method, quantitative data is implemented as probabilistic information in order to facilitate the search operation [68–70]. The main difference between this method and the random walker is that in this method a node is utilized from an earlier response to probabilistically direct future walkers. Whereas in the random walker technique the future walk is selected by random. Each node in the APS keeps a table for the forwarding probability to each neighbor for each resource. Each value in the table evaluates the probability of the node's neighbor to be selected as the next hop in a future request for the specific object. The APS method implements k random walkers to search for the required object with a probability given by its table index. These index values use feedback after each Walker process and updated. The value of the relative probabilities of the walker succeeding or failing are increased or decreased. In APS the discipline duplicate messages are considered a failure state. The performance of this method in comparison with the random walker is improved. This method results in more discovered objects, a higher success rate, low bandwidth consumption and an adaptation to changing topologies. The search process in APS does not involve object placement and P2P overlay topology. It is just assumed that the storage of objects and their copies in the network follows a replication distribution. The method for evaluating the probability of selecting a node for query forwarding has a major fault. The evaluation does not consider some important parameters such as bandwidth and storage availability when choosing the target peers.

Gia

The search protocol in Gia [15], in using a biased random walk, uses an active flow-control. The flow-control allows a sender to direct queries to a neighbor, but only if the neighbor has notified the sender that it is willing to accept queries from the sender.

In Gia, a node selects the highest capacity neighbor for which it has flow-control tokens. It then sends the query to a selected neighbor. In cases where the node does not have tokens from any neighbor it queues the query until new tokens arrive. Gia uses a bookkeeping procedure to avoid redundant paths. With the bookkeeping procedure, each query is assigned a unique global identifier (GUID) with its source node. A node remembers the neighbors to which it has previously forwarded queries for a given GUID. If a query with an equal GUID arrives back at the node, it is forwarded to a different neighbor. This procedure reduces the likelihood that a query traverses the same path twice.

Gia improves flooding-based search methods by many orders of magnitude in terms of the aggregate query load. All nodes provide one-hop replication with keeping pointer to the content offered by their immediate neighbors. There are two main problems, which are not considered in Gia. First, the indexing resources of the neighbors increase the responsibilities of each peer, plus the communication overhead. Second is how fast can the algorithm work for joining peers, and at what cost to their vicinity.

Local Indices (LI)

In Local Indices [10], each node maintains an index of the content of all neighbors within an r hop distance from itself. The distance is called the radius of the index (for instance, for $r = 0$ it exactly follows the flooding algorithm, where a node only indexes its own meta data). To minimize the overhead, the hop-distance between two consecutive depths must be $2r+1$. Thus when a node receives a query message

it can be processed on behalf of every node within r hops. Therefore, the data of many nodes can be searched by a few nodes. This approach resembles the two search schemes for hybrid networks. The search procedure in local indices is performed in a BFS-like method. There is a policy which specifies the depths at which the query should be processed. All nodes at depths not listed in the policy simply forward the query to a next path.

The advantage of this technique is that it increases the search performance for power-law topologies where only a few nodes have a very large number of neighbors. It also reduces the search bandwidth costs, and improves scalability. The disadvantages of this technique are: first, it increases the storage cost for maintaining an index, second it is fault tolerant, third its reliability is low, and finally it takes extra steps to update the indexes every time a node joins or leaves the network.

2.5 Super-Peer Selection and Searching Algorithms

There are three types of P2P system [38] that express varying levels of centralization.

First, centralized unstructured P2P systems function as a centralized component. In this type of system, the search is performed over a centralized directory; however downloads still occurs in a P2P fashion. Hence, peers are only equal in download. Napster is a well-known example of this type of unstructured P2P system.

Second, decentralized or pure unstructured P2P systems function as a decentralized component. Two samples of this system are Gnutella [71] and Freenet [72]. In this type of system, all peers have equal roles and responsibilities in all aspects. All the peers in this category may act as a client, server or router, and they all can query and download, etc.

The third type of P2P system, which reflect a varying degree of centralization,

are called super-peers. The super-peer systems function as a cross between pure centralized and decentralized systems. They have two advantages, first compared with pure or decentralized P2P systems they can exploit the heterogeneity in peers. Second, compared with centralized systems they do not have any unique centralized components, thus they are not threatened by a single point failure of the server.

This section reviews P2P systems with the simplest super-peer selection. This category does not include super-peer selection relay on higher-level applications, or super-peer sets that are hardcoded. The review does not include systems where super-peer are selected manually, or systems that use a centralized component for manipulating super-peer management.

The reviews start with OceanStore, one of the first P2P systems that assumed the use of super-peers. The section then describes some file-sharing systems that depend on super-peers, such as Gnutella, KaZaa, Gnutella 0.6, eDonkey, and supernode-based P2P file-sharing networks. The review investigates and focuses on two main factors in each super-peer system:

- The method of super-peer selection or the determination of which peers are the best candidates for super-peers.
- The search procedure in the selected super-peer system.

2.5.1 OceanStore

OceanStore is a global-scale distributed storage system for persistent data and is one of the first systems to propose the use of a super-peer system [73, 74]. It was published in 2000, and can be called as a predecessor to P2P systems, which was designed to run on a large number of nodes distributed around the world. OceanStore is maintained by several independent autonomous providers. In order to improve its reliability and performance it used a proactive replication algorithm.

The algorithm replicated and spread the data stored in OceanStore evenly across all nodes. Each data object has a primary replica, which updates periodically to verify access, and constructs a broadcasting tree between secondary replicas. These primary replicas are hosted on a selected set of nodes. These nodes are called the primary tier or inner ring, which functions in OceanStore in a similar way to super-peers in many P2P systems.

This primary tier consists of a small number of replicas located in high-bandwidth, high-connectivity regions of the network [73]. OceanStore does not provide any algorithm for the selection of nodes in the primary tier, but in order to the replication protocol specification [75], nodes that participate in the inner ring are selected by system operators.

The search and resource discovery in OceanStore is controlled by a modified version of the Plaxton algorithm [76]. This algorithm later evolved into Tapestry [77], which is one of the first distributed hash table systems. In Tapestry, each data item, as well as every peer in the system, is assigned a unique identifier. The data items are assigned to peers based on their identifiers independent of their physical locations. Thus peers maintain a system topology and routing tables that enable efficient query routing and data access. Tapestry and the original Plaxton algorithm distribute data and traffic equally between all peers in the system. They deal with all peers in the system, as if they possessed uniform resources.

2.5.2 Brocade

Brocade is an extension of Tapestry, which uses peer heterogeneity to improve the routing performance [78]. In this system, high-capability super-peers, which are titled *landmark* nodes, are used for routing messages through a wide-area network on behalf of other peers who act as their clients. These *landmark* nodes maintain a Tapestry network between each other and use the Tapestry routing algorithm.

The *landmark* nodes also maintain lists of their clients.

Super-peers are selected in Brocade among the nodes with certain characteristics by the Internet Service Provider (ISP) in each local network. The selected nodes have remarkable processing power, a minimum number of IP hops to the wide-area network, and high-bandwidth upload links [78]. The gateways routers or machines that are close to the ISP are attractive candidates for super-peers [78]. In this system, although an election algorithm is mentioned, no detail of such an algorithm is provided.

The search and resource discovery in Brocade includes two mechanisms. In the first mechanism, super-peers monitor the traffic in their local networks and intercept all messages destined to peers located in remote networks. These messages are subsequently tunneled and forwarded over the Tapestry overlay by the super-peers. However, this approach necessitates that every local network in the system has at least one super-peer, and the network must be configured in such a way that the super-peers can intercept messages from all local peers. Such a requirement may be a serious obstruction in the system's deployment.

The second mechanism counts on the use of predefined names in the Domain Name System (DNS) to identify super-peers. Each super-peer, when selected, binds its address to a fixed name in the local DNS domain. Every client can find its super-peer by resolving the fixed name in its own local DNS domain. If there is no super-peer or no super-peer is found, the client can itself become a super-peer. However, this technique again requires that at least one super-peer must be created for every local DNS domain in the system, and also in addition super-peers must be allowed to alter their DNS domains in order to register.

2.5.3 Gnutella

Gnutella was one of the first file-sharing applications that introduced super-peers. In Gnutella, each peer specifies a number of files that it agrees to with other peers. Therefore, in this system each peer can potentially download any available file by other peers. Every file sharing system provides a search facility that allows peers to discover files shared by other peers. In this regard, Napster uses a centralized server that keeps track of all files hosted by peers in the system. However, most P2P file-sharing systems offer a decentralized search scheme facility which performs a search by generating a search query. The query message is broadcast on the P2P overlay, and peers who store files that match the query reply back to the searching peer.

The first file-sharing systems, such as the early version of Gnutella, used a flooding scheme to propagate queries across all peers. Further search techniques frequently used include random walks, Breadth First Search (BFS), Depth First Search (DFS), and Expanding Ring (ER). However, these search schemes, called blind search methods, have the drawback that they potentially need to broadcast search queries to all peers in the system in order to find matching results. Accordingly, when the size of the system grows and more search queries are produced, most of the peers receive more search messages. Therefore, the system becomes unscalable as the search overhead is inevitable.

Gnutella, in addressing this problem [11,12], divides peers into two sets: leaves and hubs. Leave nodes maintain one or two links to hubs, while hubs allow hundreds of leaves, and numerous connections to other hubs. When a search is started the node contacts the hubs in the list, noting which have been searched, until the list is exhausted, or a predefined search margin has been reached. This permits a user to find a popular file without loading the entire network. Hubs index the files that a leaf has been using with a Query Routing Table. Gnutella also has a meta

data system for more complete category, ranking, and quality information to be provided in the search results.

Gnutella uses a compression system in its network connections to reduce the bandwidth used by the network. Gnutella is more efficient, as continuing a search does not exponentially increase the network traffic. Queries are not directed through as many nodes, and the granularity of a search grows, allowing a client to stop once a pre-defined threshold of results has been obtained more effectively than in Gnutella. However, the scheme increase the complexity of the network and the network maintenance required, because extra indices need to be maintained in super peers.

2.5.4 KaZaa

KaZaa has solved the scalability problems that Gnutella and other file-sharing systems have experienced. It is the first file-sharing application that introduced super-peers for handling searches [3, 79].

KaZaa divided peers into two classes, high-capability peers and ordinary peers. The high-capability peers or super-peers are also called supernodes, and ordinary peers are called clients. The algorithm of super-peer selection in KaZaa is not publicly available. Some evidence has identified that KaZaa uses local knowledge in its super-peer selection [80]. Each super-peer keeps an index of all the files or data stored on its clients.

KaZaa counts on the FastTrack protocol for search and resource discovery. The search process in KaZaa begins with a client submitting a search query to its super-peer. Then the super-peer broadcasts the query to other super-peers. If the super-peer receives results from the other super-peers, it forward the results back to the client.

The search strategy in KaZaa has two advantages compared with the traditional

search algorithms of P2P networks. First, the search message is just exchanged between super-peers. Thus, the number of peers who participate in the search is limited. Second, the search is handled by selected, high-capability peers (super-peers) and prevents low-capability peers (clients) being involved, hence, the performance of the search is improved.

2.5.5 Gnutella 0.6

As the super-peer approach in KaZaa became popular and proved its validity, Gnutella version 0.6 also used super-peers as its coordinate based [81].

Gnutella 0.6 divided nodes into two categories: ultrapeer-capable and ultrapeer-incapable. The difference between these two categories is based on minimum performance requirements. The protocol of Gnutella 0.6 indicates that a peer is capable of becoming an ultrapeer if it has, first of all, non-firewalled connections to the Internet, and secondly has at least 20 KB/s downstream and 10 KB/s upstream bandwidth. In Gnutella 0.6 each ultrapeer can accept up to 32 connections from leaves and up to 30 connections from other ultrapeers. Based on the Gnutella 0.6 protocol, if a leaf peer obtains 90% of an ultrapeer's requirements it can change to an ultrapeer. Usually, ultrapeers are generated when new peers join the network, and ultrapeers are destroyed when peers leave the network. Furthermore, each Gnutella user can force its peer to operate as an ultrapeer or a leaf. The Gnutella 0.6 protocol is backward compatible with its earlier versions. Hence, it is possible for ultrapeers and leaves to stay in one Gnutella overlay with peers that do not distinguish between super-peers and clients, and still connect to all of them in the same manner.

Gnutella 0.6 employ super-peers (ultrapeers) to index the files stored by clients (leaves) for performing a search. The super-peer technology in Gnutella 0.6 reduces the load on the lowest-performaning peers and improves the scalability of the

Gnutella network.

2.5.6 eDonkey

One of the most successful file-sharing systems is the eDonkey P2P system [82,83]. There are more than 4 million users [84] connected online at any given time to eDonkey and its clients such as eMule [84] or MLDonkey [83].

The essentials of the system are based on the eDonkey protocol. The eDonkey introduced super-peers, which are called eDonkey servers [85, 86]. There is no clear algorithm to identify super-peer selection in this file-sharing system. Each peer in the eDonkey network is qualified to be setup as a server, and it may decide manually by each eDonkey user [85].

The network technique in eDonkey follows a client-server architecture. Each of its server indexes the resources which its clients provide. Servers in eDonkey do not share any files, and they do not initiate any downloads. In eDonkey 2000, there are three types of communications: server to server, client to server, and client to client. The servers communicate over the UDP protocol to maintain a list of other known servers. Clients login onto a server via a TCP protocol. They provide their username, IP address, connection port, and the list of files that they want to offer to the system. Then the server adds information about these files to their database and consecutively assigns the user ID. Then each server sends the list of other known servers to the user. This list usually contains 100-200 entries. After this data is exchanged between clients and servers, the users can search and download their required files.

2.5.7 Supernode Based P2P File-Sharing Networks

In [87], the authors do not specify any super-peer selection algorithm. However, they describe a routing algorithm and a data caching scheme that takes advantage

of super-peers. They state the following criteria for super-peers:

- High bandwidth connection
- Have enough computation power
- Cannot join/leave network frequently
- Have a large amount of storage

However, they have not specified the amount of bandwidth or computational power or storage space required. In addition, they have not clarified how to estimate peer stability [88]. The Supernode Based P2P File-Sharing Network (SBARC) used the Pastry DHT [89] by routing messages through high-capability super-peers.

2.6 Related Works on Search Mechanisms

In this section, we have reviewed and analyzed all searching algorithms that are proposed as state-of-the-art for unstructured P2P systems. Search algorithms in this area can be classified as blind and informed search algorithms. The blind search algorithms are further classified into the blind using flat topology and blind search using super-peers.

2.6.1 Blind Search Algorithms Review

Blind search schemes use a flooding technique to broadcast the query messages to the peers in the network. In this method, the peer has no knowledge about P2P topology and the distribution of a resource. Flooding schemes have been widely used in the communication network because of their good performance with respect to the metrics [90,91]. Flooding has significant merits [7,46], such as simple algorithms, low maintenance, large coverage, reliability, and dependency. Based on these merits, it is used mostly in unstructured P2P networks [16,58,92].

Despite these merits, it generates exponentially increasing redundant messages which threaten the network's scalability [93]. Jiang *et al.* [16], show that more than 70% of messages generated by flooding are redundant messages within the TTL of 7. Consequently, fully decentralized Gnutella-network like flooding does not scale [94]. Flooding is efficient in low-hops, but by increasing the number of hops it generates a huge amount of redundant messages. All searching techniques reviewed here employed a flooding algorithm because of its simple algorithm, which is easy to support and easy to implement. Table 2.2 represents the advantages and disadvantages of flood-based search algorithms that used a flat topology.

To alleviate the harm of flooding, there are three major solutions in the literature, which can be classified as:

- TTL Limit-Based Flooding (TLBF).
- Probabilistic Limit-Based Flooding (PLBF).
- Hybrid Limit-Based Flooding (HLBF).

TTL Limit-Based Flooding (TLBF)

TLBF are those algorithms that limit the Time-to-Live value for controlling the flooding of overshooting messages. There are three major search algorithms in this area, Expanding Ring, Iterative Deepening and Blocking Expanding Ring. These algorithms propagate messages to the limited number of hops. Expending ring or Iterative Deepening is the first control-TTL based flooding algorithm [52].

Indeed, this scheme uses successive floods with increasing TTL values in each round. Thus with each new iteration more messages cover a larger area of the network. Although expanding ring has a simple algorithm and low overhead, it is non-deterministic in returning results, and for resources that are far away from the source node this approach could even generate more sever overshooting messages than flooding [95].

The main difference between Expanding Ring and Blocking Expanding Ring is with respect to its rebroadcast procedure. In BER, the rebroadcast procedure started from all nodes of the last attempt. The BER has gained better performance in query success rate and a decrease in overshooting messages.

Although TLBF solutions have successfully controlled many overshooting messages, still they produce considerable amounts redundant messages, which limit the system's scalability. The reviewed schemes in this area indicated that this type of solution has an uncertain performance for resources that are far away from the source node and for unpopular objects.

Table 2.2: Advantage and disadvantage of blind search algorithms for flat topology[1]

Algorithm	Description	Advantage	Disadvantage
Flooding	Flood queries to all of its logical neighbours with certain TTL value (BFS)	1. Simple algorithm (popular in practice) 2. Large coverage 3. High reliability 4. Moderate latency 5. Locate rare resources	1. Produce huge amount of redundant messages (unnecessary traffic) 2. Very large overhead 3. Wasting bandwidth and processing resources 4. Limit system scalability
Expanding Ring (ER)	Successive flooding (BFS) with increment of TTL value	1. Successfully control flooding 2. More bandwidth efficient when queried resources are nearby 3. Low overhead	1. There are duplicate messages 2. None deterministic 3. For resource far away from source has sever duplicate messages 4. Performance could be uncertain for less popular files
Blocking Expanding Ring (BER)	Successive flooding with increment of TTL value from last nodes in each round	1. Control flooding better than expanding ring 2. Almost no duplicate messages	1. For resource far away from source has sever duplicate messages 2. Performance could be uncertain for less popular files
Random Breadth First Search (RBFS) or Teeming	Select portion of its neighbors with fix random value (such as θ)	1. Decrease more messages propagating 2. Acceptable coverage	1. High performance nodes may be skipped 2. Success rate decrease since target nodes are selected randomly
Normalized flooding	Nodes are selected based on the minimum degree of nodes in the network.(such as δ)	1. As nodes forward messages to at most m neighbors. Thus, message overhead is reduced	1. The value of minimum degree (δ) in network is not clear 2. The problem of random selection may cause high performance nodes omitted 3. Low degree nodes may let out from receiving queries

[1] The rest of table is in the next page.

Algorithm	Description	Advantage	Disadvantage
Random walks	The k number of nodes randomly selected in each hop. k is one for the standard random walks and is k for k *random walk*. utilized probabilistic routing procedure.	1. Low message complexity. It cut messages over head an order to magnitude 2. Algorithm scale well 3. Gains local load-balancing	1. The problem of random selection may cause high performance nodes omitted. 2. Variable (unknown) performance 3. For unpopular files almost give up (no result) 4. High latency (an order of magnitude increase in user-perceived)
Local flooding with k independent random walk	Compromise between flooding and random walks	1. Integrates the advantage of flooding and random walk 2. If flooding happens locally, the message complexity is low.	1. In order walker to more hops, messages overhead is high 2. The value of k in unclear

Probabilistic Limit-Based Flooding (PLBF)

PLBF are those algorithms for which a limited numbers of peers are visited in each hop with a random selection discipline for controlling overshooting messages. There are four major searches in this category: Random Breadth-First-Search (RBFS), normalized flooding, Random Walks (RW), and k Random Walk, propagates messages into randomly selected neighbors.

The RBFS or teeming algorithm is dramatically reduces the number of overshooting messages. Although this method gains acceptable coverage, due to the nodes chosen it may skip the chance of using high-performance nodes.

Normalized flooding is close to teeming in selecting the subset of nodes. The subset is unique and equal to the least degree of nodes in the network. This technique decreases the forwarding of messages in the network, hence the message overhead reduces. However the proper selection of δ is critical; it may leave out low degree nodes that are hosting good content.

The main idea behind the Random Walker is to limit the broadcasting area by controlling the number of peers. This technique can reduce by an order of mag-

nitude the number of overshooting messages compared to expanding ring and flooding. However, there is also an order of magnitude increase in user-perceived delay. Although its search cost is low, its performance is highly variable, it is non-deterministic, and it is not reliable.

The proposed solutions of the PLBF type confirm the main problem of a randomly chosen peers, such as variable (random) performance because it is possible to skip high capability nodes due to chance. The schemes in this area analyzed have shown that although these techniques cut the message overhead by an order to magnitude, they also increase user-perceived delay by an order of magnitude.

Hybrid Limit-Based Flooding (HLBF)

The main goal of this group of search algorithms is to limit the number of peers in each hop by either utilizing hybrid overlay networks or super-peer techniques. There are many searching schemes in this category, such as the all bind search technique for super-peer structures and super-peer search techniques in unstructured P2P networks.

The all super-peer search reviewed here are dynamic query-like flooding (DQ), enhanced dynamic query-like flooding (DQ+), AntSearch, Diffsearch, GUSS and Gnutella 2 that employ a super-peer as an ordinary node without any prominent features. They simply dispatch information between so-called super-peers and their unknown clients. In all these systems, there is no policy for the selection of super-peers. Despite this, these systems have significantly controlled the issues of flooding and improved search efficiency with a low search cost and high success rate. Table 2.3 presents the advantages and disadvantages of blind search algorithms using a super-peer.

Table 2.3: Advantage and disadvantage of blind search algorithms for super-peer[1]

Algorithm	Description	Advantage	Disadvantage
Dynamic querying like flooding (DQ,)	Used hybrid structure As an ultrapeer first, forwards a probe query via a few neighbors with small TTL to estimate popularity of query. This procedure repeats and estimate popularity parameter each time. (dynamic fashion)	1. Control flooding overshooting messages 2. Reducing search cost 3. Avoid sending query packets too far 4. Avoid send query packets to same subset of peers	1. Selection of ultrapeers ambiguous. 2. The iterative process can introduce long delay 3. The query is propagated to just a small fraction of required number of peers 4. Method is doomed to have high latency 5. The estimating number of all nodes in network is not possible due to high churn rate of nodes
Enhanced Dynamic querying like flooding (DQ+)	The iterative process here is greedy and conservative estimation of popularity used confidence interval.	1. Same advantage as DQ plus 2. Reduce latency more than four time 3. Low degree peers often find right number of result	1. Selection of ultrapeers ambiguous. 2. Search cost for less and none popular items are high 3. The estimating number of all nodes in network is not possible due to high churn rate of nodes
AntSearch	Use same query relying as (DQ, DQ+) And consider free riding problem.	1. Successfully control flooding (reduce redundant messages) 2. latency is same as DQ 3. Low overhead	1. Selection of ultrapeers ambiguous. 2. Overshooting problem because the physical number of search peer is larger than estimated number of peers.
Differentiated search (DiffSearch)	Consist of two rounds First, query forward to ultrapeer overly if not satisfied forward to entire network.	1. Search traffic is significantly reduced due shrunken search space.	1. The second round of search causes overhead in network due to flood of messages 2. The index uploading adds a small cost to the overhead

[1] The rest of table is in the next page.

Algorithm	Description	Advantage	Disadvantage
Gnutella UDP extension for scalable search (GUESS)	Iterative contacting various super-peers and having them inquire all their leaves.	1. Controlled flooding	1. Even there is larger cache size in a larger number of probes, they do not guarantee to more satisfied queries 2. Since there are few number of peers that wish to share a large number of objects, many queries will go unsatisfied

2.6.2 Informed Search Algorithms Review

Informed search algorithms are those who nodes save some kind of routing information for forwarding the query to the appropriate nodes.

The sort of this information is based on different parameters, such as the popularity and similarity of their objects, and other relevant factors. In this class of search, each peer is ranked related to its parameters and queries routed to nodes that have top rank. Normally, most informed search methods improve the success rate. However, they produce a lot of update messages in dynamic P2P environments. Table 2.4 presents the advantages and disadvantages of informed search algorithms.

2.6.3 Super-peer Search Algorithms Review

Yang and Garcia-Molina [38] divided P2P into three categories. First, Pure P2P networks such as Freenet and the initial version of Gnutella, where all peers are similar and have the same roles and responsibilities in all aspects, and thus the system's functionality is fully decentralized.

Second, hybrid systems such as Napster where some functionality is handled by a main directory or server (such as search). However, downloading in Napster is decentralized because downloads are performed directly between peers. Third, super-peer networks [96–98] such as KaZaa, which act like a cross between pure and hybrid P2P networks. In these networks, the super-peer is a node that operates as a centralized server to a set of clients. Each super peer with its clients is called a

Table 2.4: Advantage and disadvantage of informed search algorithms

Algorithm	Description	Advantage	Disadvantage
Adaptive Probabilistic Search (APS)	Searching is according to the uses of k independent walker and probabilistic forwarding. Each intermediate neighbor sends the query to one of its immediate neighbor with probability given by its local index. The index values are updated by using feedback from the walkers	1. Bandwidth efficient 2. Probabilistic selection of peers instead of random selection 3. Shows robustness when topology changes. 4. Nodes finally share, process and update their search knowledge with the time.	1. Popular files have more chance to located than other files 2. The peer discovered first might used more for future routine and might experience more load. Concurrently, other peers, which are closer, could be ignored. 3. The load-balancing is not considered 4. There is not considered other features of node such as bandwidth, storage, degree. And all nodes given equal status. 5. Free rider problems not consider 6. There exists partial coverage problem
Gia	A search protocol is according to biased random walk, it directed queries toward high-capacity nodes, which usually able to supply best answer.	1. Uses an active flow control to avoid hotspots 2. By bookkeeping reduce the query-dropping rate in flooding 3. Effectively manage load balancing time.	1. Produce more communication overhead 2. Low-capacity nodes has no chance to select, thus, unpopular objects may not queried
Local Indices	Every peer keeps a local index of the content of all its neighboring peers within hop distance called the radius of index	1. Displays good performance for power-law networks (few nodes have very large number of neighbors) 2. Decreased search cost and bandwidth and improve scalability	1. In order dynamic network, extra steps must be taken for update indices every time a node joins or leave 2. In creased storage cost and keeping index 3. Reliability and fault-tolerance are low

cluster; the cluster size is the number of nodes in the cluster, including the super-peer. Clients in each cluster communicate with their super-peer as in a traditional client-server. Super-peers construct a secondary topology for communication with each other. The advantage of the super-peer compared with pure flooding is that a super-peer can exploit the heterogeneity in peers. The advantage of super-peers compared with a hybrid network is that the super-peer does not have any centralized part. The main reason in this type of solution is the chance to improve the quality of the search the decreasing or increasing the role of one scheme in combination. Introducing a hybrid technique can improve the performance and scalability of the search technique in P2P networks.

The systems reviewed here all employ very simple approaches to super-peer selection and management. Centralized systems prepared reliability and scalability in search, which is in contradiction with the nature of a P2P network. Manually selection of the super-peer is impossible due to the large scale, dynamism, and complexity of P2P networks.

Most of the systems reviewed have indicated criteria for super-peer selection. They have described their criteria as a high amount of available bandwidth, storage space, processing power, and long session time. However, they did not provide a specific mechanism for the estimation of the amount of the high in any of criteria. In several reviewed systems, the super-peer characteristic is defined in a static mode, which is not acceptable due to the P2P's dynamic nature. The performance of systems and the number of super-peers have a mutual relationship, in all applications the number of super-peers must be the optimum. For example, in file-sharing applications, the number of super-peers in order to reduce traffic must be low, while it should by high enough in order to be load-balancing.

To limit the number of super-peers to a desirable level requires a global knowledge of peer characteristics. Many P2P networks assumed that the peer's properties

followed a certain distribution. There are many reasons why the distribution of a peer's characteristics has been changed.

By developing new technologies and the growth of the Internet over time, a natural change has occurred. The capabilities of computers are increasing exponentially every two years [99]. The changeable network conditions and general behavior of users are other important and unpredictable factors. In this new era, the properties of peers depend on the time of the day, day of a week, and many external events. The properties of peers may change in different environments. The properties of peers are not same in different countries and even between different groups of users. Introducing super-peers can improve the performance and scalability of many P2P networks. There is an essential requirement to control the method of selection and the number of super-peers required to improve searching on a P2P network.

2.7 Summary

This chapter introduced overlay network constraints in unstructured P2P networks. There are three classes of overlay in this area; flat, tree-based, and hybrid. In flat topology, all nodes have a same role, in operating as a client, server, and router. This kind of topology is robust to the churn rate of nodes due to their open architecture and self-organizing structure.

In tree-based topology, data are forwarded from a source root to leaf nodes. This kind of topology cannot take full advantage of the network resources, and the system can easily become unbalanced due to the bandwidth of the leaf peers not being utilized. Moreover, the system is vulnerable to peer churn since any middle peer's departure will disconnect all descendants of that peer from the content forwarding tree.

In hybrid topology, peers are divided into two group: ordinary peers and super-peers. Super-peers act as a server which index all the files of clients and the search

Table 2.5: Compare important super-peer selections and searching techniques

Super-peer	Selection technique	Searching technique
OceanStone	1. There is not any algorithm for selection	1. Rely on Plaxton algorithm. 2. Peers maintain a network topology and routing tables.
KaZaa	1. Select high-capability peers. 2. Selection algorithm is not publicly.	1. Super-peers maintain an index of all files stored on their client. 2. The search just exchange between super-peers.
Gnutella 0.6	1. Peers who has acceptable capabilities 20 Kb/s downstream and 10Kb/s upstream.	1. Super-peers maintain an index of all files stored on their client. 2. The search just exchange between super-peers.
eDonkey	1. Selection algorithm is not clear.	1. Super-peers maintain an index of all files stored on their client. 2. Search perform by super-peers technique.
Supernode Based Peer-to-peer file sharing network	1. Do not specified the criteria for selection. 2. High bandwidth, enough computation, cannot join/leave network frequently and have large amount of storage	1. Pastry (DHT).

is performed by super-peers.

This chapter, reviewed, compared and critiqued the work carried out in the searching of unstructured P2P networks and super-peer selection and search.

CHAPTER 3
RESEARCH METHODOLOGY

3.1 Introduction

Analytical modeling, simulation and measurement are common tools for evaluating the performance of techniques that used as methodologies. There are several factors influencing the selection of evaluation technique, such as life cycle, time available, amount of developed tools available and the accuracy of the results produced. This research utilized both analytical studies and simulation techniques to evaluate the performance of the optimum hop count in flood-based, QuickFlood and HybridFlood searches in unstructured P2P networks.

The main goal of this chapter is to explain the research methodology that is intended to be applied for the optimum hop count in flood-based, QuickFlood and HybridFlood searching algorithms.

3.2 An Overview

Unstructured P2P networks are large scale distributed systems, which must be used for digital information storage and retrieval in this new era. Search or content location is the main activity which utilizes these applications. Searching in unstructured P2P network is big challenges in state-of-the-art. An unstructured P2P network, in contrast with the client server, has no knowledge about resources locations and network topology. Due to this differentiation the searching techniques employed in the client server would not be suitable for unstructured P2P networks.

The nature of the flooding search is same as the unstructured P2P network topology. Both assume no knowledge about resource locations and network topology. Thus, flooding offers an attractive method for resource location in dynamic and

evolving unstructured P2P networks. However, flooding incurs large network overloads which threaten the system's scalability.

Due to this drawback, the research introduces two novel hybrid search algorithms based on the flooding algorithm. These hybrids propose that the search algorithms are performed in two steps. They utilize flooding in the first step to gain a high coverage growth rate of the networks. In the second step, one of the proposed algorithms, which is called QuickFlood, used another search technique to obtain a low level of redundant messages. The next proposal, an algorithm called HybridFlood, used a super-peer technique to exploit the peer's heterogeneity, and decrease the shortcomings of flooding. This research also proposed an optimum hop count to switch between first step and second step in both algorithms proposed.

3.3 Research Framework

The thesis focuses on improving the searching technique in unstructured P2P networks. Accordingly, the following independent steps have been taken to satisfy the research objectives. The steps include the problem formulation, previous flooding-based search algorithm implementation and analysis, proposed scheme model, analytical study and experimental simulation, and then a comparison of the results and performance evaluation. Figure 3.1 presents the research framework.

3.3.1 Problem Formulation

The primary step of study starts with an intensive review of two subjects. The first review is of existing unstructured P2P search algorithms that have always used the flooding search algorithm, and the second review is of searching in hybrid P2P structures. In this step, many papers were studied to collect the required information about previous solutions, and thus suggest future work. The studied focused on three main areas:

- Concepts and applications of unstructured P2P networks.
- Overlay networks in unstructured and hybrid P2P networks.
- Searching algorithms using flooding schemes in unstructured P2P networks.

Hence, many searching features being designed, include the search performance such as; scalability, success rate, and latency. Thus, it is hard to find all the required merits in one scheme. Therefore, at this step the research problem and assumed scope were identified. This step also stated the shortages and limitations of the current search algorithms.

3.3.2 Previous Flood-Based Search Algorithm

In this step, two main P2P search algorithms, informed and blind searches, were investigated. In informed searching [100], a number of node cache information that related to files or data, therefore, can select "*good*" neighbors to forward query to. In general, most informed search methods improve the success rate. However they produced a lot of update messages in the dynamic and evolving unstructured P2P environment.

The blind search can be further classified as a blind search for flat topology and blind search for super-peer topology.

- In blind search for flat topology, nodes have no knowledge about the file or data location, thus queries are forwarded without any knowledge of its neighbors. Existing blind methods waste a lot of bandwidth to obtain a high success rate, such as with flooding search, or have low success rate, such as in a random walk search.
- There are a few blind searches designed for unstructured P2P networks that are based on super-peers to control the negative aspects of flooding searches. All super-peers employed in this category are an ordinary node without any

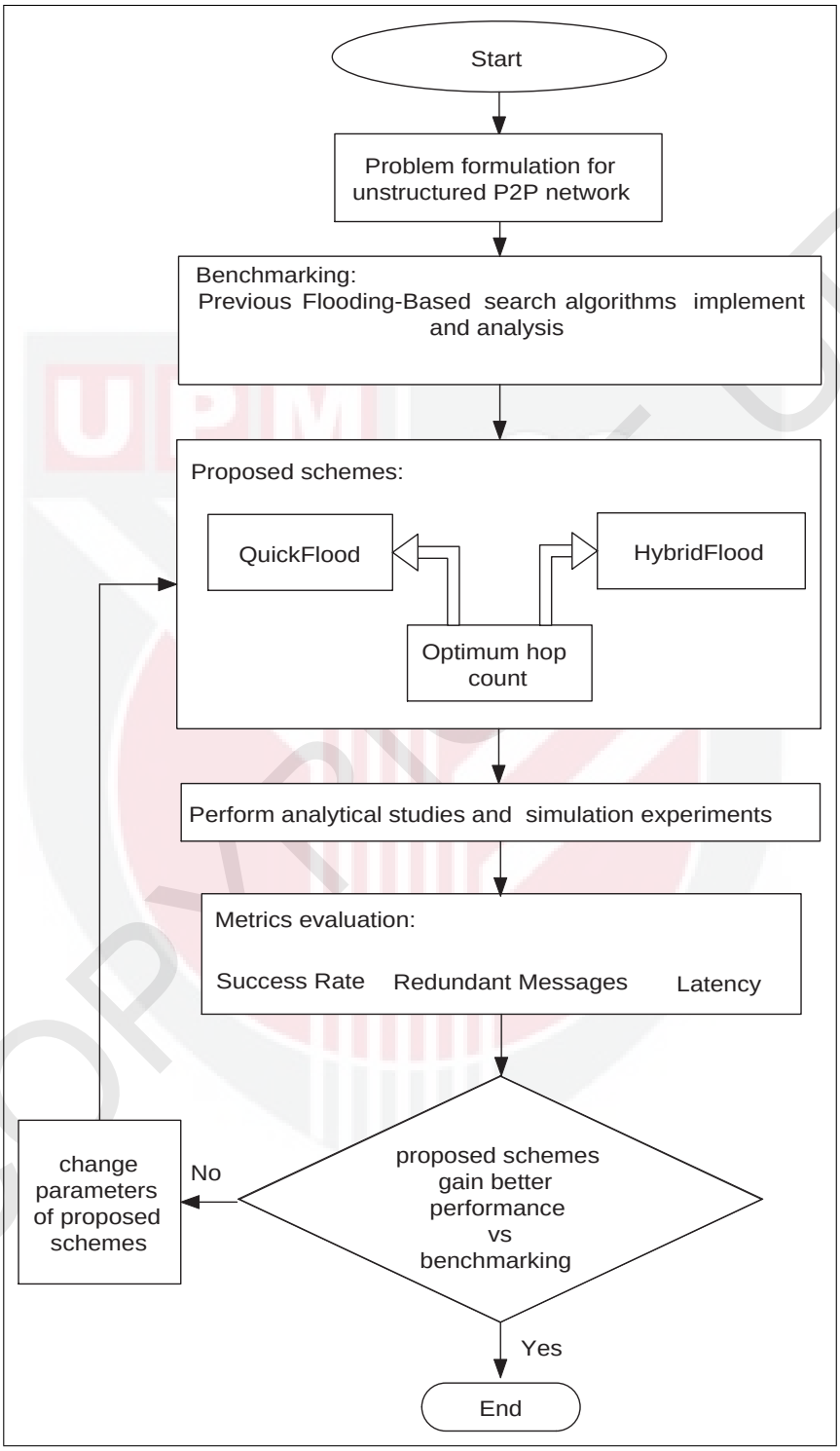

Figure 3.1: The research frame works

prominent features. They simply broadcast information between so-called super-peers and their unknown clients. These systems have significantly controlled the issues of flooding and have improved search efficiency with a low search cost and high success rate.

In the super-peer search, the peers are classified as super-peer and ordinary peers or clients. Super-peers are nodes that operate as a centralized server to a set of clients. Each super-peer and its clients is called a cluster. Super-peers create a secondary topology for communication with each other. The advantage of the super-peer compared with pure flooding is that a super-peer can exploit the heterogeneity in peers. The advantage of the super-peer compared with a client-server network is that the super-peer does not have any centralized part. The super-peer act likes a cross between a pure and client-server network.

The main reason for the super-peer type of solution is the chance to improve the quality of the search by decreasing or increasing the role of one scheme in combination. The hybrid technique can improve the performance and scalability of search techniques in P2P networks.

3.3.3 Proposed Schemes

The main goals of the proposed schemes are to focus on decreasing the number of redundant messages and increasing the schemes success rate in terms of improving search efficiency. The structures of the proposed methods are based on combining two flooding-based algorithms and also combined flooding with a super-peer technique. This research introduced two novel search algorithms based on the hybrid structure, called QuickFlood and HybridFlood. Both algorithms use the proposed optimum hop count method for switching between the two steps. The methodology of each proposed scheme will be discussed in detail in Chapters 4, 5 and 6. Chapter 4 will study the optimum hop count in flooding-based search

algorithms, while Chapter 5 and 6 will consider the new algorithms, QuickFlood and HybridFlood.

3.3.4 Perform Analytical Studies and Simulation Experiments

The proposed methods and benchmark flooding-based algorithms are compared analytically and experimentally. Many experiments have been conducted in order to investigate the performance of the proposed algorithms.
To evaluate the proposed methods and compare them with other methods a custom built simulator is developed to measure metrics such as query success rate, number of redundant messages, and amount of latency.

3.4 Experiment Environment

To examine and evaluate the performance of the proposed schemes, many resources and parameters need to be set and identified. This section identified the computer resources, data collection, parameters setup, and experimental setup that are used in this research.

3.4.1 Data Collection

This study used Gnutella topologies collected during the first six months of 2001, which are provided by Clip2 Distributed Search Solution [101]. The research mainly uses two topology traces, which are listed in Table 3.1, as the samples are of different topology of size and average degrees of connectivity. The original name in the table refers to the trace file name used in [101]. The average of the two-hop neighboring peers indicates the average of the number of peers within two hops from each peer.

Table 3.1: The clip2 topology used in simulation experiments [1]

Topology	Name	Average degree	Average 2-hop neighbouring peers	Number of peers
T1	Graph1 [2]	3.4	34.1	42822
T2	Graph2 [3]	4.7	54.8	28895

[1] The clip2 topology [101].
[2] Original name = 052701104502.xml samples are in appendix.
[3] Original name = 050301100618.xml samples are in appendix.

3.4.2 Network Model Assumptions and Parameters

To assist the discussion, we presented the assumptions and parameters of the basic components of a network. The basic components of a network include the network topology, peers, and query messages. The network topology defines the connectivity between peers. This connectivity is assumed using the Gnutella topologies collected during the first six month of 2001 by Clip2 Distributed Search Solution [101].

Peers

Each peer has unique *id*, which is stored in its *P.ID* field. There is a field in peers such as *P.CM*, which stores file names and documents, which the peer likes to share. A list of the *ids* of all immediate neighbors of the peer is saved in *P.NI*. When a query visits a peer, its *id* saved in *P.TR*. The peer's departure and arrival are recorded in *P.ST*. Figure 3.2 presents fields that are assumed in a peer.

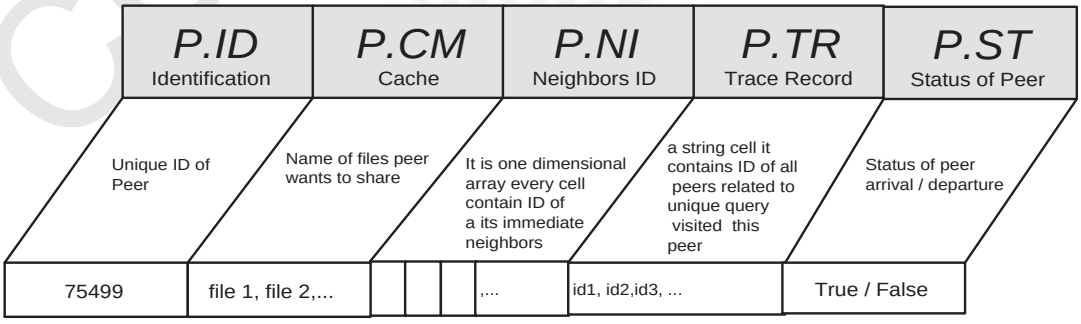

Figure 3.2: Peer's fields

- Peer consist of following fields

    1. $P.ID$ = unique id, select a unique number
    2. $P.CM$ = stores file names and documents, which the peer likes to share
    3. $P.NI$ = an array, store the id of all its immediate neighbours
    4. $P.TR$ = a string cell, store the id of all peers related to unique query that have visited this peer
    5. $P.ST$ = status of peer: active or inactive

Source Peer

The source peer is a typical peer, which publish a query(s). This kind of peer creates two extra fields for saving the query(s) and the nosey node(s) information. Thus when a peer becomes a source it establishes two new fields, called $S.TB$ and $S.NT$.

Thus, when a peer begins to publish a query it first selects an array field in $S.TB$ for each query. In the selected row, it stores a query id and the maximum number of queries that must be found. It stores them respectively in ID and MAX cells of the selected row. In the selected row of the $S.TB$ there are also other cells for storing the total numbers of found, not found, and redundant messages of the specific query. The second field $S.NT$ is also an array, which stores information about nosey nodes. It saves the id and $hop\ count$ of each nosey node, which is created by the source peer. Figure 3.3 shows the fields that are assumed in a source peer. At the same time, it is possible that a source peer acts as an ordinary peer for other source peers.

- Source peer consist of following fields

    1. $S.ID = P.ID$
    2. $S.CM = P.CM$

3. $S.NI = P.NI$
4. $S.TR = P.TR$
5. $S.ST = P.ST$
6. $S.TB$ = a table, store the *ID*, *MAX*, *Find*, Not_Find, *Redundant* of each query in a separate row
7. $S.NT$ = an array, store the *ID* and *HP* (*hop count*) of each nosey node

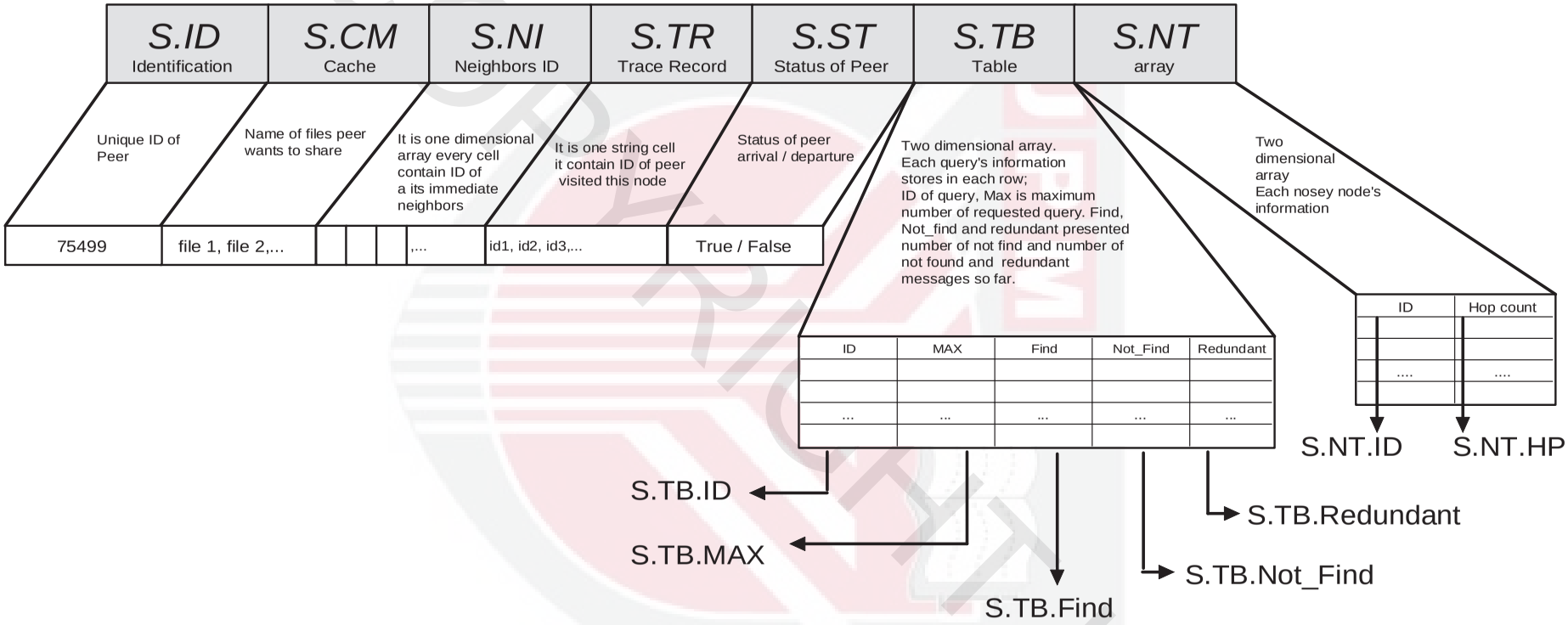

Figure 3.3: Source peer's fields

Query Message

Query messages have four fields for saving a query's information. Each query has a unique *id* that it has published by the source peer, and it is exactly the same as *S.TB.ID*. Each query saves the *id* of the peer that issued this query. The query saves its current Time-To-Live value in $Q.TL$. The keyword received from a user is stored in $Q.KW$. Figure 3.4 provides the fields that are assumed in a query.

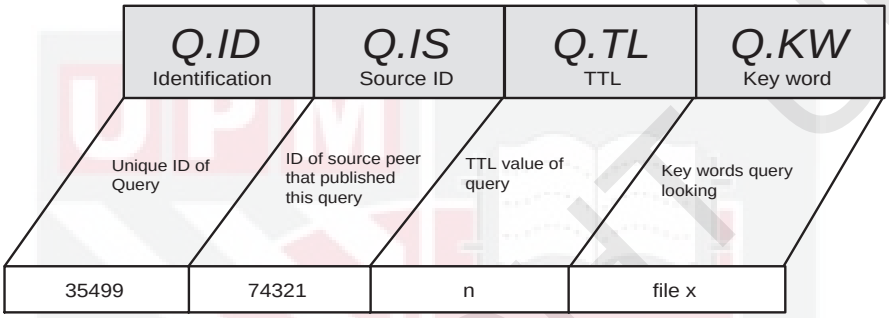

Figure 3.4: Query's fields

- Query message consist of following fields

  1. $Q.ID = S.TB.ID$
  2. $Q.IS$ = *id* of the source peer, that published the query
  3. $Q.TL$ = current TTL value
  4. $Q.KW$ = search keywords received from the user

Nosey Node

The nosey node of any selected peer is the node among its immediate neighbors that has the highest number of links. The procedure for selecting nosey nodes is explained by detail in section 6.4.1, the nosey node acts as a super-peer.
This kind of peer has three extra fields $NN.CT$, $NN.TA$, and $NN.RD$ for storing their client's information that of the reserved nosey nodes. All indices of files that clients like to share are saved in the $NN.CT$ field. The *ids* of clients are stored in $NN.TA$, and the *id* of the reserved nosey nodes is saved in $NN.RD$. Figure 3.5 presents the fields that are assumed in a nosey node.

- Nosey node consist of following fields

    1. $NN.ID = P.ID$
    2. $NN.CM = P.CM$
    3. $NN.NI = P.NI$
    4. $NN.TR = P.TR$
    5. $NN.ST = P.ST$
    6. $NN.CT$ = index of all files its clients like to share
    7. $NN.TA$ = an array store of its clients'*id*
    8. $NN.RD$ = store the *id* of the reserved nosey node (redundancy nosey node)

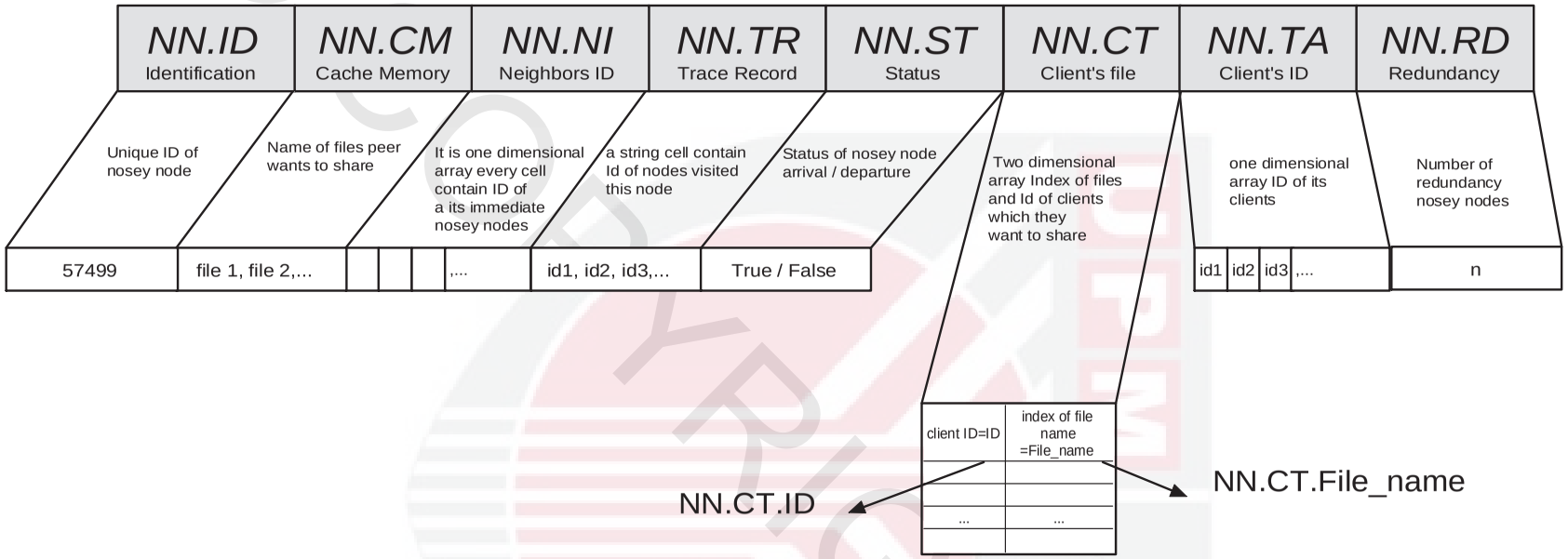

Figure 3.5: Nosey node's fields

3.4.3 Experimental Setup

To perform all evaluations in this study, the research used the data set in section 3.4.1 as base topology. The replication ratio was set to 1/800=0.00125 which means that for each of 800 nodes one has object. This distribution of objects is conducted in 20 different random sets and is called the *number of placement* of the object. The query is started by 50 different random nodes in each topology it means *number of query*. Thus each experiment is conducted under *number of placement* $\times$ *number of query* = 1000 different situations. This guarantees the small standard deviation in our results [57].

3.5 Performance Metrics

In this study, there are three standard types of measurement metrics to be applied in unstructured P2P networks.

- Query success rate (success rate)
- Number of redundant messages
- Number of latency

3.5.1 Query Success Rate

One of the well-known measuring metrics for evaluating the performance of P2P searches is the query success rate or success rate. In the remainder of this research, the term success rate will be used instead of query success rate (QSR). The success rate of each algorithm is the ratio of the number of successful queries over the total number of queries broadcast by a typical source peer for an object [102].

$$QSR = SR = \frac{N_{Success\_query}}{N_{Broadcasted\_query}} \tag{3.1}$$

The success rate (SR) is also defined as the probability that a query is successful [23]. This metric evaluates the efficiency and quality of the search algorithm. The major concern of this metric is the end user perspective [103]. As this metric increases the quality and the efficiency of search improves, and vice versa.

3.5.2 Number of Redundant Messages

When a multiple message with the same message *id* in each algorithm is sent to a peer by its multiple neighbors, all except the first message are considered a redundant message. Unnecessary traffic overhead is produced by a large number of redundant message forwarding, particularly in a network with a high-connectivity topology. Redundant messages increase peer processing and network transfer, without expanding the propagation scope. This metric represents the rate of a network's scalability. As the number of redundant message increases the scalability of the system decreases as it causes increased load and traffic on the network.

3.5.3 Number of Latency

Latency is a measure of the time delay experienced in a network. Latency is the time from the source sending a packet to the destination and vice versa. Latency can be measured as the number of rounds of flooding required to discover the target [60]. This metric is used to measure the search algorithm's efficiency, because the enhanced algorithm should effectively limit the search latency.

3.6 Simulator Overview

Performance evaluation is a basic part of research. There are three approaches to evaluating the performance of a study; theoretical system analysis, test-bed experiments, and simulation experiments. Although this study used theoretical system analysis for some of the research, it is infeasible to assess performance

metrics in this way, due to the system complexity. The deployment of a P2P test-bed or realistic scale is also impossible, because it requires extremely large amounts of resource, such as machines and users. Therefore, the approach followed in this thesis is simulation experiments.

The research used a custom-built P2P simulator which was developed with Java and MatLab. The simulator followed the cycle model, which evaluated an algorithm is periodic. There is a global loop which controls the flow of time in the simulator. Every operation of the peers, such as neighbor selection, super-peer selection, aggregation and routing algorithm, is performed in a cycle time step.

Churn Rate Model

Churn rate is the average portion of peers leaving and joining the network per time unit. In a dynamic P2P network, peers are continually arriving and departing from a network. The churn rate can be evaluated with session time. The session times in P2P networks are varied from six hours to 2 minutes [104,105]. In [27], the mean session time for 40% of the nodes is less than 4 hours. This research assumed a mean session time of $\mu = 60$ minutes, thus the churn rate per minute is equal to $1/\mu = 1.67\%$. Assuming each time step in the simulation is 5 seconds, hence there are 12 time steps in a minute so the churn rate is equal to $1.67/12 = 0.14\%$. Therefore, the research accepted the churn rate as 0.14% of peers per time step.

3.7 Summary

This chapter has explained the methodologies are used in the performance analysis and conducted in this thesis, including many of the procedures. The methodological steps which are described in this chapter include data set collection; applying standard flooding searches, optimum hop count in flood-based searches, design of the QuickFlood and HybridFlood algorithms, and the analytical study of QuickFlood and HybridFlood. The data sets which are used in this study and three

types of measurement metrics are selected to evaluate the results, i.e. query success rate, number of redundant messages, and amount of latency. The thesis used a custom-built simulator to evaluate the performance of the algorithms.

CHAPTER 4
ANALYTICAL STUDIES AND EXPERIMENTAL EXAMINES FOR FLOODING-BASED SEARCH ALGORITHMS

4.1 Introduction

In the second chapter we studied many search algorithms for unstructured P2P networks. The flooding search algorithm with its significant merits is an attractive method for expanding a dynamic unstructured P2P network. The reputation of the flooding algorithm is mostly due to its simplicity it eases of implementation and ease of support. Furthermore, in its unlimited version (or when its TTL is set to a sufficiently large value) it will always succeed in the search.

The primary study of flooding search algorithm shows two different characteristics, in terms of low-hops and high-hops. First, in its low-hops it gains a high coverage growth rate. However, in its high-hops it resulted in an almost non coverage growth rate. Second, it incurs huge numbers of redundant messages in its high-hops, but there are few redundant messages in its low-hops.

The main goals of this chapter are: first to analyze analytical studies for flood-based search algorithms. Second, to examine simulation experiments for flood-based search algorithms.

4.2 Flooding Algorithm across the Hops

The flooding algorithm is the simplest search scheme and one of the most commonly used in P2P networks. The procedure of flooding is conducted in a hop-by-hop fashion. By increasing the hops, new peers are gained and more messages generated. A part of these messages are redundant messages. This section investigated the trend of coverage growth rate and redundant messages in flooding. The summary of all notations used in this research are presented in Table 4.1.

4.2.1 Trend of Coverage Growth Rate

An overlay network as a Random Graph $G_{n,p}$ (n is the number of vertices and p is the number of edges). Each vertex or node is represented by a peer, and they are connected to each other by edges. The degree of each peer represents the number of its immediate neighbors. Supposing that the graph has n nodes with an average degree d, and assuming that d is greater than 3, the total messages broadcast from each peer to hop t, as given in [106], is:

$$\begin{aligned} TM_{F,t} &= \sum_{i=1}^{t} d(d-1)^{(i-1)} \\ &= \frac{d(d-1)^t - 1}{d-2} \end{aligned} \tag{4.1}$$

Loop nodes or cyclic paths are groupings of nodes linked together in a ring fashion. In Gnutella and other internet topology, there are many cyclic paths [107]. If there are no cyclic paths or loop nodes in the topology then the total number of new peers visited so far is equal to:

$$\begin{aligned} TP_{F,t} &= \sum_{i=1}^{t} d(d-1)^{(i-1)} \\ &= \frac{d(d-1)^t - 1}{d-2} \end{aligned} \tag{4.2}$$

Thus, the coverage growth rate of messages [16] in hop t is given as:

$$CGR_{F,t} = \frac{TP_t}{TP_{t-1}} \tag{4.3}$$

By substituting Equation (4.2) in Equation (4.3) the value of coverage growth rate becomes:

$$\begin{aligned} CGR_{F,t} &= \frac{TP_t}{TP_{t-1}} \\ &= \frac{\sum_{i=1}^{t} d(d-1)^{(i-1)}}{\sum_{i=1}^{t-1} d(d-1)^{(i-1)}} \end{aligned} \tag{4.4}$$

By simplifying Equation (4.4) the coverage growth rate of messages in flooding up to hop t is given as:

$$\begin{aligned} CGR_{F,t} &= 1 + \frac{(d-1)^{(t-1)}}{\sum_{i=1}^{(t-1)} (d-1)^{(i-1)}} \\ &= 1 + \frac{(d-1)^{(t-1)}}{\frac{(d-1)^{(t-1)}-1}{(d-2)}} \\ &= 1 + \frac{(d-1)^{(t-1)}(d-2)}{(d-1)^{(t-1)}-1} \\ &= 1 + \frac{(d-2)}{1 - \frac{1}{(d-1)^{(t-1)}}} \end{aligned} \tag{4.5}$$

For investigating the trend of the coverage growth rate in each hop assume:

$$A = d - 1 \tag{4.6}$$

Thus, the value of coverage growth rate in hop t with references to Equation (4.5), and then yield:

$$\begin{aligned} CGR_{F,t} &= 1 + \frac{(A-1)}{1 - \frac{1}{A^{(t-1)}}} \\ &= 1 + \frac{(A-1)}{\frac{A^{(t-1)}-1}{A^{(t-1)}}} = 1 + \frac{(A-1)A^{(t-1)}}{A^{(t-1)}-1} \\ &= \frac{A^{(t-1)} - 1 + (A-1)A^{(t-1)}}{A^{(t-1)}-1} = \frac{A^t - 1}{A^{(t-1)}-1} \end{aligned} \tag{4.7}$$

The discrete derivative of a function $f(n)$, with respect to n, define as:

$$\Delta_n f(n) = f(n) - f(n-1) \tag{4.8}$$

Thus, derivative of $CGR_{F,t}$ with respect to t, lead to:

$$\begin{aligned}
\Delta_t(CGR_{F,t}) &= CGR_{F,t} - CGR_{F,t-1} \\
&= \frac{A^t - 1}{A^{t-1} - 1} - \frac{A^{t-1} - 1}{A^{t-2} - 1} \\
&= \frac{(A^t - 1) \times (A^{t-2} - 1) - (A^{t-1} - 1)^2}{(A^{t-1} - 1) \times (A^{t-2} - 1)} \\
&= \frac{A^{2t-2} - A^t - A^{t-2} + 1 - (A^{2t-2} - 2A^{t-1} + 1)}{(A^{t-1} - 1) \times (A^{t-2} - 1)} \\
&= -\frac{(A^t - 2A^{t-1} + A^{t-2})}{(A^{t-1} - 1) \times (A^{t-2} - 1)} \\
&= -\frac{A^{t-2}(A^2 - 2A + 1)}{(A^{t-1} - 1) \times (A^{t-2} - 1)} \\
&= -\frac{A^{t-2}(A - 1)^2}{(A^{t-1} - 1) \times (A^{t-2} - 1)}
\end{aligned} \tag{4.9}$$

The value of $\Delta_t(CGR_{F,t})$ is always negative (because A is greater than 1).

Lemma 4.1 *If the derivative* $\Delta_t(CGR_{F,t})$ *of a discrete function* $CGR_{F,t}$ *satisfies* $\Delta_t(CGR_{F,t}) < 0$ *(negative) on an interval* $t \in \{a, ..., b\}$*, then* $CGR_{F,t}$ *is decreasing on* $\{a, ..., b\}$. **Proof:**

A discrete function $CGR_{F,t}$ *is said to be decreasing, if for any* $b-a = 1$, $CGR_{F,b} - CGR_{F,a} < 0$. *Or* $CGR_{F,b} - CGR_{F,b-1} < 0$ *for any* $b > 0$. *Therefor,* $\Delta_b(CGR_{F,b}) < 0$. *So at any point* $x = b$, *if* $\Delta_x(CGR_{F,x}) < 0$, *then* $CGR_{F,x}$ *is said to be decreasing.*

Hence, it can be shown as:

$$CGR_{F,2} > CGR_{F,3} > CGR_{F,4} > CGR_{F,5} > \cdots \tag{4.10}$$

Therefore, $CGR_{F,t}$ is always in descending order. By increasing t hops, the value of $CGR_{F,t}$ decreases. Thus the maximum value of $CGR_{F,t}$ is presented in the second hop. Therefore by increasing the hops t, the value of the coverage growth rate decreases.

4.2.2 Trend of Redundant Messages

The redundant messages in each topology are generated by loop nodes. *A* loop is a grouping of nodes linked together in a ring fashion. In Gnutella and other Internet topologies, there are too many loop nodes in high-hops [16].

Assuming that there is a loop in each hop of the defined topology and that the loop started from second hop. Thus the number of new peers in hop two becomes:

$$P_{F,2} = d(d-1)^2 - 1 \tag{4.11}$$

By considering a loop, the number of new peers in the third hop is equal to:

$$\begin{aligned} P_{F,3} &= [(d(d-1)^2-1)(d-1)]-1 \\ &= [(d(d-1)^3-(d-1)]-1 \end{aligned} \tag{4.12}$$

The number of new peers in the fourth hop by considering a loop is:

$$\begin{aligned} P_{F,4} &= [(d(d-1)^3-(d-1)-1)](d-1)-1 \\ &= [(d(d-1)^4-(d-1)^2-(d-1)]-1 \end{aligned} \tag{4.13}$$

By induction, the number of new peers visited in hop t becomes:

$$P_{F,t} = d(d-1)^t - \sum_{i=0}^{t-2}(d-1)^i \tag{4.14}$$

Obviously, there are more loops in each hop topology. This is the minimum number of loops that is being considered here. The number of redundant messages can be defined as the difference between the number of messages and the number of new peers visited in hop t.

$$R_{F,t} = M_{F,t} - P_{F,t} \tag{4.15}$$

Thus the redundant messages generated in hop $t+1$ is equal to:

$$\begin{aligned} R_{F,t} &= d(d-1)^t - [d(d-1)^t - \sum_{i=0}^{t-2}(d-1)^i] \\ &= \sum_{i=0}^{t-2}(d-1)^i = \frac{(d-1)^{t-1}-1}{(d-2)} \end{aligned} \tag{4.16}$$

By substituting Equation (4.6) in Equation (4.16), leads to:

$$R_{F,t} = \frac{A^{(t-1)}-1}{A-1} \tag{4.17}$$

The derivative of $R_{F,t}$ with respect to t, refer to Equation (4.8) can be shown as:

$$\begin{aligned}
\Delta_t R_{F,t} &= R_{F,t} - R_{F,t-1} \\
&= \frac{A^{t-1}-1}{A-1} - \frac{A^{t-2}-1}{A-1} \\
&= \frac{A^{t-1}-1-A^{t-2}+1}{A-1} \\
&= \frac{A^{t-2}(A-1)}{A-1} \\
&= A^{(t-2)} \qquad (4.18)
\end{aligned}$$

The value of Equation (4.18) is always positive.

Lemma 4.2 *If the derivative $\Delta_t(R_{F,t})$ of a discrete function $R_{F,t}$ satisfies $\Delta_t(R_{F,t}) > 0$ (positive) on an interval $t \in \{a, ..., b\}$, then $R_{F,t}$ is increasing on $\{a, ..., b\}$.*

Proof:

A discrete function $R_{F,t}$ is said to be increasing, if for any $b-a=1$, $R_{F,b}-R_{F,a} > 0$. Or $R_{F,b} - R_{F,b-1} > 0$ for any $b > 0$. Therefor, $\Delta_b(R_{F,b}) > 0$. So at any point $x = b$, if $\Delta_x(R_{F,x}) > 0$, then $R_{F,x}$ is said to be increasing.

Thus the function is always in ascending order. By increasing the value of t (hops) the value of $R_{F,t}$ increases. Hence, the minimum value of $R_{F,t}$ is visited in the second hop. Therefore, it can be shown as:

$$R_{F,2} < R_{F,3} < R_{F,4} < R_{F,5} < \cdots \qquad (4.19)$$

The percentage of redundant messages in hop t based on seven hops is equal to:

$$\frac{R_{F,t}}{\sum_{i=1}^{7}(R_{F,i})} \times 100 \qquad (4.20)$$

We have examined these facts with our data sets of T1 and T2 topology for 1000 random nodes. Figure 4.1 presents coverage growth rate in each hop based on

Equation (4.3) and Figure 4.2 shows the number of redundant messages in each hop. To show the up trend of redundant message in each hop the Figure 4.3 presents the percentage of redundant messages in each hop based on Equation (4.20). Observations have confirmed the analytical results from Equations (4.10),

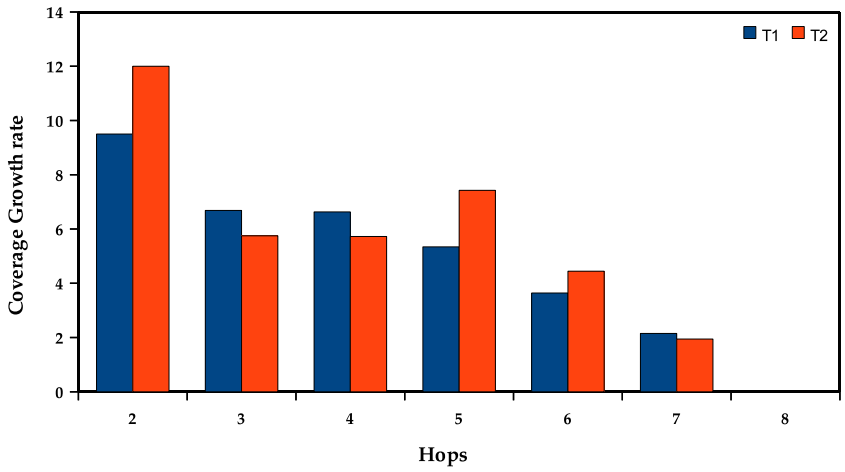

Figure 4.1: Compare the coverage growth rate in two topologies

(4.16) and (4.19). In addition, the analytical study identified three important points:

1. The coverage growth rate of messages for flooding in defined topologies has a maximum value in the second hop, by increasing the hops the value of coverage growth rate decreases.

2. The redundant messages for flooding in the same topologies at low-hops are very low but by increasing hops their values increase exponentially.

3. The trend in the coverage growth rate and redundant messages in each topology is in reverse order. As far as one is high in a hop the other is low in the same hop and vice versa.

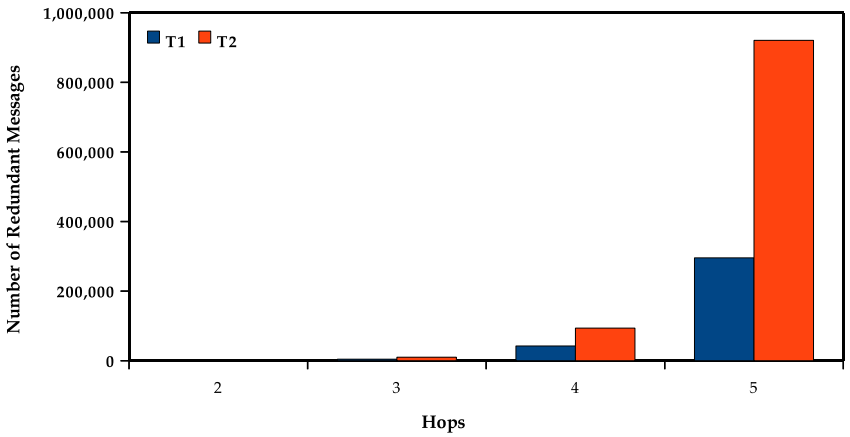

Figure 4.2: Compare the number of redundant messages in each hop of two topologies

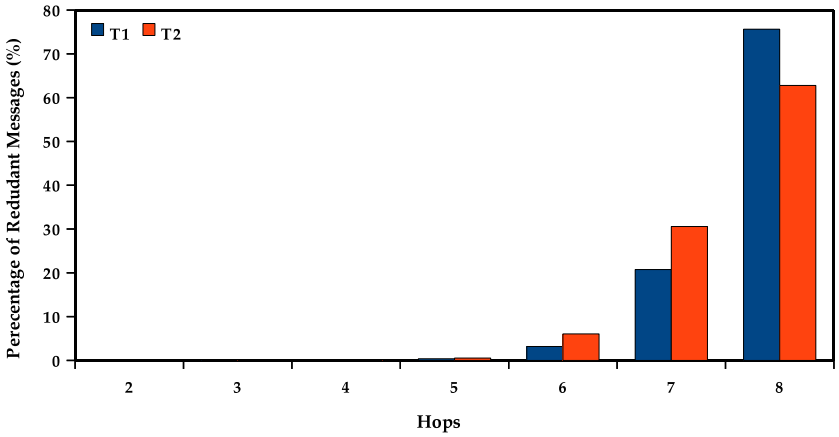

Figure 4.3: Compare the percentage of redundant messages in two topologies

4.3 Proposed Performance Metric

The most important metrics, which can evaluate the performance of the search algorithms, is the number of redundant messages and coverage growth rate of messages that are generated with a search algorithm. As the number of redundant messages increases the scalability of the search algorithm decreases. Thus, the search algorithms with low scalability cannot be employed in dynamic and evolving

unstructured P2P networks. A search algorithm with low coverage is not reliable and cannot locate rare objects. Thus, the performance of a search algorithm with large coverage is more efficient than one with low coverage.

Inspired by the reverse trend in coverage growth rate and redundant messages in flooding schemes, we propose a new metric, which called the *critical metric*. The *critical metric* combines the coverage growth rate and redundant messages in each hop of the flood-based algorithms to present the state of the search performance at the same hop. Therefore the new metric can evaluate the performance of different algorithms and also hybrid flood-based algorithms.

4.3.1 Critical Metric

Here is the description of *critical matric*. The definition is:

Definition 4.1 ***(Critical Metric*** $CM_{x,t}$***)*** *Let x represent any algorithm in flood-base category and t be the number of hop in such a algorithm. Let $R_{x,t}$ and $CGR_{x,t}$ be respectively the number of redundant message and number of coverage growth rate in any x flood-based algorithm at hop t. For any x algorithm in flood-base category the critical metric is defined as:*

$$CM_{x,t} = \frac{R_{x,t}}{CGR_{x,t}} \tag{4.21}$$

This metric is valid for low-hops, because in high-hops the value of coverage growth rate becomes close to 0. Obviously, if as a consequence the amount of coverage growth rate becomes zero, the value of the *critical metric* tends to infinity, and is hence not valid. The value of the *critical metric* in low-hops is always greater than zero.

Critical Metric discussion

Critical metric represents the state of the redundant messages and coverage growth rate at each hop of a flood-based algorithm. Thus it can evaluate the performance

of a search algorithm in a hop. Here the research evaluated the *critical metric* in two ranges of value: when it is less than one and then when it is greater than one. First when the value of the *critical metric* is less than one: in this state the amount of redundant messages must be less than the coverage growth rates, hence when the number of redundant message is low and the coverage growth is high, the search algorithm has an acceptable performance.

Second, when the value of the *critical metric* is greater than one; in this situation the amount of redundant messages is greater than the coverage growth rate, thus when the amount of redundant messages is the high and coverage growth rate is low, the search algorithm has a low performance.

Consequently, this characteristic of the *critical metric* is a suitable tool for comparing the performance of flood-based search algorithms. It is also a proper tool to estimate the optimum threshold point of hops in order to switch from one algorithm to the other in a hybrid search algorithm that uses flood-based algorithms. The following sections first calculate *critical metric* for flood-based algorithms. The following sections first calculate the *critical metric* for flood-based algorithms, and then compare and evaluate them.

4.3.2 Critical Metrics in Flood-Based Algorithms

This section calculated the values of *critical metrics* for flood-based search algorithms. Flood-based search algorithms include flooding, and other search algorithms used flooding as a base in its algorithms. The main goal of flood-based algorithms is to control the issues caused by flooding to improve the performance of a search.

Flooding

The flooding search algorithm performs the following: the source of the search sends the requested messages to all its neighbors, which in turn propagate the

message to their own neighbors except for the neighbor from which they have received the message and so on. Flooding is unrestricted in the sense that there is no constraint on the number of messages generated by the search request.

To calculate the *critical metric* in flooding, the coverage growth rate and redundant messages are calculated in detail. As given in Equation (4.7) the coverage growth rate of flooding is equal to:

$$CGR_{F,t} = \frac{A^t - 1}{A^{(t-1)} - 1} \tag{4.22}$$

Equation (4.22) can be presented as:

$$\begin{aligned} CGR_{F,t} &= \frac{A^t - 1}{A^{(t-1)} - 1} \\ &= \frac{A^t}{A^{(t-1)} - 1} - \frac{1}{A^{(t-1)} - 1} \end{aligned} \tag{4.23}$$

Equation (4.23) shows:

$$CGR_{F,t} > \frac{A^t}{A^{(t-1)} - 1} \tag{4.24}$$

The right side of Equation (4.24) can be presented as:

$$\frac{A^t}{A^{(t-1)} - 1} > \frac{A^t}{A^{(t-1)}} = A \tag{4.25}$$

Equations (4.24) and (4.25) lead to:

$$CGR_{F,t} > A \tag{4.26}$$

Equation (4.26) presents the minimum value of coverage growth rate in hop t. By assumption, the average degrees of nodes are always greater than 3 ($d > 3$). Thus the value of the coverage growth rate in hop t is greater than A. As a result, the

value of the redundant messages as given in Equation (4.17) is equal to:

$$R_{F,t} = \frac{A^{(t-1)} - 1}{A - 1} \tag{4.27}$$

The right hand side of Equation (4.27) can be shown as:

$$\frac{A^{(t-1)} - 1}{A - 1} < A^{(t-1)} \tag{4.28}$$

By considering Equations (4.27) and (4.28) the maximum value of redundant messages in hop t can be presented as:

$$R_{F,t} < A^{(t-1)} \tag{4.29}$$

Equation (4.29) shows that the values of redundant messages in the first two hops of flooding are negligible, thus it confirms the result in relation to Equation(4.19). By substituting the results of Equations (4.26) and (4.29) in Equation (4.21) the *critical metric* for the flooding algorithm under the most pessimistic condition where the coverage growth rate is minimum and the amount of redundant messages is maximum is equal to:

$$\begin{aligned} CM_{F,t} &= \frac{R_{F,t}}{CGR_{F,t}} \\ &= \frac{A^{(t-1)}}{A} = A^{(t-2)} \end{aligned} \tag{4.30}$$

Assume that the Time-To-Live (TTL) value used in flooding is k, thus the total

critical metric for all hops up to k is equal to:

$$\begin{aligned} TCM_{F,k} &= \sum_{i=1}^{k} CM_{F,i} = \sum_{i=1}^{k} A^{(i-2)} \\ &= \frac{A^k - 1}{A^2 - A} \end{aligned} \tag{4.31}$$

Expanding Ring

The expanding ring is successive flooding; assuming that the TTL value in expanding ring started from 1 and is incremented by 1 up to l, where l is less than k and greater than 2. Thus the *critical metric* for the expanding ring from 1 to l is equal to:

$$TCM_{ER,l} = \sum_{t=1}^{l} \sum_{i=1}^{t} CM_{F,i} \tag{4.32}$$

By substituting the Equation (4.31) in Equation (4.32) the value of the *critical metric* for the expanding ring becomes:

$$\begin{aligned} TCM_{ER,l} &= \sum_{t=1}^{l} \frac{A^t - 1}{A^2 - A} \\ &= \frac{\sum_{t=1}^{l} (A^t) - \sum_{t=1}^{l} 1}{A^2 - A} \\ &= \frac{\frac{A^{(l+1)} - A}{A-1} - l}{A^2 - A} \\ &= \frac{A^{(l+1)} - A - l(A-1)}{(A^2 - A)(A-1)} \end{aligned} \tag{4.33}$$

Blocking Expanding Ring

The blocking expanding ring is an extended version of the expanding ring, which is not rebroadcast from source node, but rather in each new round it is rebroadcast from the nodes of the last attempts. Hence, its *critical metric* for $(l < k)$ referring to Equation (4.30) is equal to:

$$\begin{aligned} TCM_{BER,l} &= \sum_{i=1}^{l} CM_{F,i} \\ &= \sum_{i=1}^{l} A^{(i-2)} \\ &= \frac{A^l - 1}{A^2 - A} \end{aligned} \tag{4.34}$$

Random Walker

Random walker is modified version of flooding. It is a Probabilistic Limit-Based Flooding (PLBF) type, which forwards a query message (walker) to only one randomly chosen neighbour. In the Random walker, the average degree of d is equal to 2. Because the query messages from each node are only forwarded to one randomly selected node. The value of the *critical metric* in references to Equation (4.30) is equal to:

$$TCM_{RW,l} = \sum_{i=2}^{l} A^{(i-2)} \tag{4.35}$$

Let $A = d - 1$ thus the value of A becomes $A = 2 - 1 = 1$. Therefore, the total *critical metric* for random walk becomes:

$$\begin{aligned} TCM_{RW,l} &= \sum_{i=2}^{l} A^{(i-2)} \\ &= \sum_{i=2}^{l} 1^{(i-2)} = l - 2 \end{aligned} \tag{4.36}$$

Teeming

Modified Breadth-First-Search or teeming is also a Probabilistic Limit-Based Flooding (PLBF) type. There is a fixed probability denoted by θ for selecting a particular neighbors. Let d_i denote the number of nodes in hop i, and d_T the average degree in the teeming algorithm. At each hop, a node propagates an inquiry message to $\theta \times d_i$ of its neighbors. Thus, d_T is equal to:

$$d_T = \frac{\sum_{i=1}^{n} d_i \times \theta}{n} \tag{4.37}$$

Thus, given by Equation (4.37) the average degree in teeming is equal to:

$$d_T = \frac{\sum_{i=1}^{n} d_i \times \theta}{n} = \frac{\theta \times \sum_{i=1}^{n} d_i}{n} = \theta \times d \tag{4.38}$$

Let $A_T = d_T - \theta$ thus, the *critical metric* for teeming referring to Equation (4.30) is equal to:

$$TCM_{T,l} = \sum_{i=2}^{l} (A_T)^{(i-2)} \tag{4.39}$$

$$\begin{aligned} A_T &= d_T - \theta \\ &= \theta \times d - \theta \\ &= \theta(d-1) \\ &= \theta \times A \end{aligned} \tag{4.40}$$

Considering Equations (4.38), (4.6) and (4.40) the value of A_T becomes equal to

$\theta \times A$. By substituting A_T with $\theta \times A$ Equation (4.38) can be shown as:

$$\begin{aligned} TCM_{T,l} &= \sum_{i=2}^{l} (A_T)^{(i-2)} \\ &= \sum_{i=2}^{l} (A \times \theta)^{(i-2)} \end{aligned} \tag{4.41}$$

The result shows that the value of *critical metric* in the teeming algorithm in each hop t is $\theta^{(t-2)}$ of the *critical metric* in the flooding algorithm for the same algorithm.

4.3.3 Evaluation of Critical Metrics in Flood-Based Algorithms

This section evaluates the *critical metrics* of flood-based searching algorithms in order to compare the performance of these algorithms. To perform this evaluation the bench-mark assumed is the value of the *critical metric* in the flooding algorithm.

Comparing the Performance of Expanding ring

The values of the *critical metric* in standard flooding for k hops and the expanding ring for l hops are respectively:

$$TCM_{F,k} = \frac{A^K - 1}{A^2 - A} \tag{4.42}$$

$$TCM_{ER,l} = \frac{A^{(l+1)} - A - l(A-1)}{(A^2 - A)(A-1)} \tag{4.43}$$

By default K is almost greater than l because the number of hops in flooding are always greater than the number of hops in the expanding ring thus $K > l$. To compare Equations (4.42) and (4.43):

$$\begin{aligned}
TCM_{ER,l} &\neq TCM_{F,k} \\
\frac{A^{(l+1)} - A - l(A-1)}{(A^2 - A)(A-1)} &\neq \frac{A^K - 1}{A^2 - A} \\
\frac{A^{(l+1)} - A - l(A-1)}{(A^2 - A)(A-1)} &\neq \frac{A^K - 1}{A^2 - A} \times \frac{A-1}{A-1} \\
A^{(l+1)} - A - l(A-1) &\neq (A^K - 1)(A-1) \\
A^{(l+1)} - A - l(A-1) &\neq (A^K)(A-1) - (A-1)
\end{aligned} \tag{4.44}$$

Given that $l < K$:

$$A^{(l+1)} \approx (A^K)(A-1) \tag{4.45}$$

By deleting both sides of Equation (4.45) from Equation (4.44):

$$-A - l(A-1) \neq -(A-1) \tag{4.46}$$

Therefore, it is clear that $(A-1) < A + l(A-1)$ and it proves that the right side of Equation (4.44) is greater than the left side. Hence, it can be concluded the

critical metric in flooding are almost greater than in the expanding ring.

$$TCM_{ER,l} < TCM_{F,k} \tag{4.47}$$

Therefore, the expanding ring algorithm is more efficient than the flooding algorithm.

Comparing the Performance of Blocking Expanding Ring

The values of the *critical metric* in expanding ring for l hops is equal to:

$$TCM_{ER,l} = \frac{A^{(l+1)} - A - l(A-1)}{(A^2 - A)(A-1)} \tag{4.48}$$

For the blocking expanding ring l hops is equal to:

$$TCM_{BER,l} = \frac{A^l - 1}{A^2 - 1} \tag{4.49}$$

The comparison between equations (4.48) and (4.49) produces:

$$\begin{aligned} TCM_{BER,l} &\neq TCM_{ER,l} \\ \frac{A^l - 1}{A^2 - 1} &\neq \frac{A^{(l+1)} - A - l(A-1)}{(A^2 - A)(A-1)} \end{aligned}$$

$$\begin{aligned} \frac{A^l - 1}{A^2 - 1} \times \frac{A-1}{A-1} &\neq \frac{A^{(l+1)} - A - l(A-1)}{(A^2 - A)(A-1)} \\ (A^l - 1) \times (A-1) &\neq A^{(l+1)} - A - l(A-1) \\ (A^{l+1} - A) - (A^l - 1) &\neq A^{(l+1)} - A - l(A-1) \\ A^{l+1} - A - A^l + 1 &\neq A^{(l+1)} - A - lA + l \\ -A^l + 1 &\neq -lA + l \end{aligned} \tag{4.50}$$

For l greater than 3 the value of $(-A^l + 1)$ is less than $(-lA + l)$. Thus, as far as

l is greater than 3 the *critical metric* in blocking expanding rings is less than the *critical metric* of the expanding ring. Thus, for hops greater than 3 the blocking expanding rings are more efficient than expanding rings.

$$TCM_{BER,l} < TCM_{ER,l} \tag{4.51}$$

The critical metrics shows that for hops less than three there is no difference between expanding ring and blocking expanding rings, since there are few redundant messages in low-hops.

Comparing the Performance of Random Walk

As mentioned in the description of the random walker the average degree d is two, thus the *critical metric* of the random walker as shown before in Equation (4.36) is equal to the number of hops it expands. Therefore, this algorithm has often no redundant messages as well as minimum coverage growth rate. This algorithm is more efficient than its predecessors.

Comparing the Performance of Teeming

The teeming algorithm is the adopted version of a random walker with a fixed probability of chosen nodes at each step. As proved by Equation (4.41) its *critical metric* is dependent on the value of fixed probability θ. Its critical metric is equal to θ^{t-2} times the *critical metric* of flooding with same *hop count*.

4.4 Analytical Results

Our analytical study in this stage showed that with a TLBF methods the *critical metric* decreases linearly refer to Equations (4.46) and (4.50) for expanding ring and blocking expanding ring. However, refer to Equations (4.36) and (4.41) in PLBF methods for random walker and teeming it decreases exponentially. It proved that PLBF has a better performance compared to TLBF. The main drawback of the PLBF type is they incur high latency.

Therefore, by considering Equations (4.36), (4.41), (4.47), and (4.51) it can be concluded:

$$TCM_{F,l} > TCM_{ER,l} > TCM_{BER,l} > TCM_{RW,l} > TCM_{T,l} \tag{4.52}$$

This evaluation proved for the same conditions the *critical metrics* in PLBF methods are less than TLBF methods.

4.5 Experimental Results

To confirm the analytical study, we used our data set in section 3.4.1 as the base topologies, and followed the set of experiments as detailed in the experimental setup section 3.4.3 to measure the query success rate, number of redundant messages, and amount of latency. The evaluation compared the performance of flooding, Expanding Ring (ER), Blocking Expanding Ring (BER) and teeming algorithms with predefined metrics. The value of probability θ in the teeming algorithm is set to 0.3 or 30% (Teeming_30).

Teeming algorithm is modified version of random walk. Random walk algorithm forward a message (walker) to only one randomly chosen neighbours. While teeming at each step forward message to subset of its neighbours. The value of θ shows the number of nodes can be chosen in each step of teeming. Obviously the minimum number of nodes can participate in teeming is one. Thus as far as we assume the average degree of node (d) is greater than three, so the minimum number of node can find by $\theta = 0.30$.

Figure 4.4 and Table 4.2 present the trend of the redundant messages for all algorithms. They show that the teeming algorithm reduces redundant messages by almost 90% compared to flooding. It shows that the blocking expanding ring and expanding ring reduce the redundant messages by over 70% compared to flooding. The experiment showed that the trends of the decrease in the expanding ring

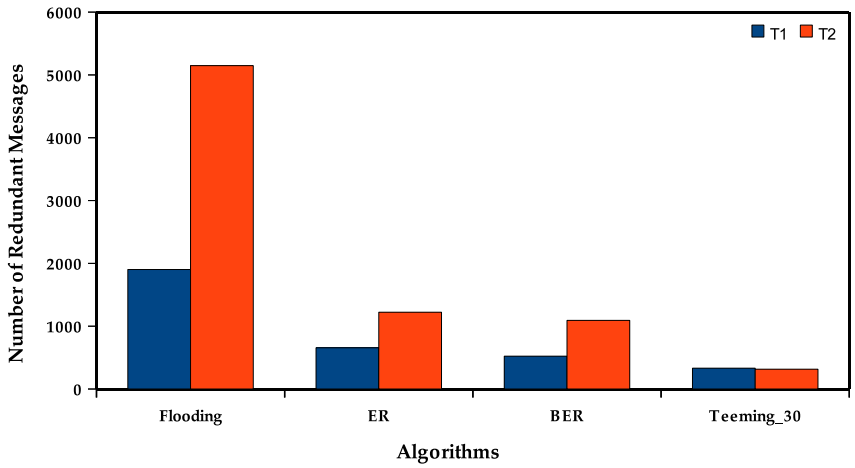

Figure 4.4: Number of redundant messages for each algorithm at different topologies

and blocking expanding ring are close to each other, but that by applying the teeming algorithm this trend decreases further. Thus, as expected, teeming has dramatically decreased the number of redundant messages.

The trend in the success rate for each algorithm in different topologies is presented in Figure 4.5 and Table 4.3. As expected, the expending ring and blocking expanding ring improved the success rate by 2.0 and 2.7 times compared with flooding, the reason refers to their limitation of the TTL values. The result illustrates the exact difference between the expanding ring and blocking expanding ring algorithms. The expanding ring collected many duplicate messages because it must start from a source peer in each round of processing, whereas the blocking expanding ring does not follow this procedure.

The reflection of this difference is evident in their success rate results, thus the blocking expanding ring achieves a better success rate than the expanding ring. On average the teeming algorithm achieves a success rate more than four times that of flooding. The main reason is that its algorithm refers to our analytical analysis exponentially reduced redundant messages compared with flooding Equation

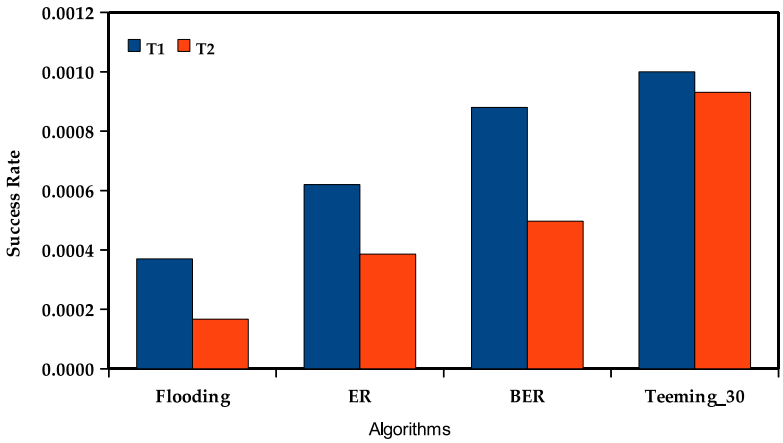

Figure 4.5: Success Rate for each algorithm at different topologies

(4.41).

The average amounts of latency in each algorithm for the different topologies are presented in Figure 4.6 and Table 4.4. Latency in the expanding ring and blocking expanding ring decrease by about 70%. The results show that the blocking expanding ring is more efficient than the expanding ring. As expected, the latency in teeming is not significant and shows an average decreases of about 37%.

The reason behind this result is that teeming follows the procedure of the random algorithm. The random algorithm incurs more latency than the others.

4.6 Summary

The chapter began with an analytical study of the flooding search in P2P unstructured networks. The research analyzes the trend in coverage growth rate and redundant messages across the hops. The study proved that the network coverage growth rate in low-hops is much higher than in high-hops, and in contrast the majority of redundant messages are generated in high-hops. The idea was examined and proven for two Internet topologies, T1 and T2. We were then interested in

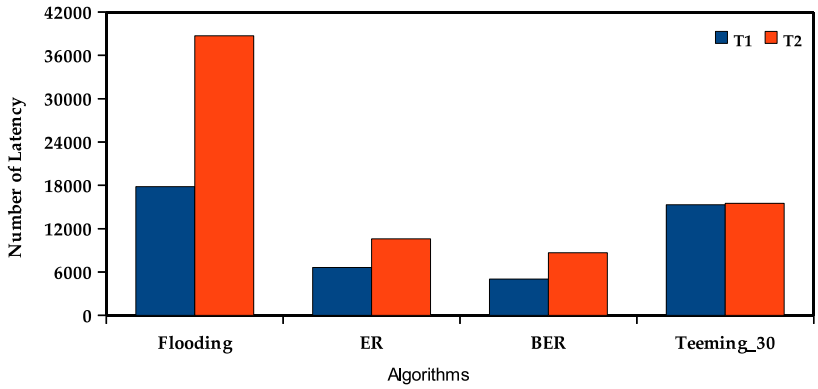

Figure 4.6: Number of latency for each algorithm at different topologies

comparing the flooding-based searches for indications of their shortcomings and strengths. For comparing the algorithms, the thesis defined a new custom-built metric, which is called the critical metric. The critical metric is the result of the coverage growth rate over the redundant messages in each hop. When the value of the critical metric is low the search algorithm has a high performance, and vice versa. The reason is that when the critical metric is low its coverage is high and it amount of redundant messages is low. Thus when a search algorithm gains more coverage and a low number of redundant messages it is obviously more efficient than one which has low coverage and more redundant messages. The study classified flooding-based searches into three classes; TTL Limited-Based Flooding (TLBF), Probability Limited-Based Flooding (PLBF), and Hybrid Limited-Based Flooding (HLBF). The comparison proves that PLBF performs better than the others. In the meantime we examined all the algorithms discussed and proved analytical studies. The experimental results show that the teeming algorithm has a better performance than the others. The teeming algorithm belongs to class of PLBF.

Table 4.1: Notations and description used in the research

Symbols	Description
n	*Number of vertices or nodes or peers in graph*
d	*Average degree of nodes, (assumed that $d > 3$)*
nbr	*neighbor*
$G_{n,p}$	*Random graph G with n vertices and p edges*
t	*Number of hops ($t > 1$)*
$M_{x,t}$	*Number of messages propagate in (x algorithm) at hop t*
$P_{x,t}$	*Number of new peer visited in (x algorithm) at hop t*
$R_{x,t}$	*Number of redundant messages in (x algorithm) at hop t*
$TM_{x,t}$	*Total number of messages propagate in (x algorithm) up to hop t*
$TM_{x,h,t}$	*Total number of messages propagate in (x algorithm) between hop h until hop t*
$TP_{x,t}$	*Total Number of new peer visited in (x algorithm) up to hop t*
$TR_{x,t}$	*Total number of redundant messages in (x algorithm) up to hop t*
$CGR_{x,t}$	*Coverage growth rate in (x algorithm) at hop t*
$CM_{x,t}$	*Critical metric in (x algorithm) at hop t*
$TCM_{x,t}$	*Total critical metric in (x algorithm) up to hop t*
F	*Flooding algorithm*
ER	*Expanding Ring algorithm*
BER	*Blocking Expanding Ring algorithm*
RW	*Random Walk algorithm*
LI	*Local Indices algorithm*
T	*Teeming algorithm*
T_θ	*Teeming algorithm with fix θ probability*
QSR	*Query Success Rate*
SR	*Success Rate*
$QF_{h,\theta}$	*QuickFlood algorithm with h hop flooding and θ probability*
$HF_{h,n}$	*HybridFlood algorithm with h hop flooding and n redundancy nosey nodes*
TTL	*Time To Live*
TTL_F	*TTL in first step*
TTL_S	*TTL in second step*

Table 4.2: The reduced percentages of redundant messages for each algorithm compare with flooding at different topologies (%)

Algorithm \ Topology	T1	T2	Average
ExpandingRing	66	76	71
BlockingExpandingRing	73	79	76
Teeming_30	83	94	88

Table 4.3: The increased times of success rate for each algorithm compare with flooding at different topologies

Algorithm \ Topology	T1	T2	Average
ExpandingRing	1.7	2.3	2.0
BlockingExpandingRing	2.4	3.0	2.7
Teeming_30	2.7	5.6	4.1

Table 4.4: The reduced percentages of number of latencies for each algorithm compare with flooding at different topologies (%)

Algorithm \ Topology	T1	T2	Average
ExpandingRing	63	73	68
BlockingExpandingRing	72	78	75
Teeming_30	14	60	73

CHAPTER 5
PROPOSED QUICKFLOOD SEARCH ALGORITHM

5.1 Introduction

Efficient search schemes are the main interest for complete and appropriate materials searching. A P2P system from a search aspect can be classified as structured, unstructured or hybrid.

Structured systems use classical client-server architecture. In this strategy, the locations of the resources that generate the search results for a query are fixed and known to the users as files or data that are stored in a centralized repository. Although this architecture is the fastest search mechanism, it is vulnerable to a single point of failure and they require tight controls on the data's location and the topology of the network.

Unstructured systems utilize blind and informed search algorithms. The blind search algorithms use a flooding algorithm as the basic scheme in their strategy. Flooding algorithms, although they have significant merits, produces a huge amount of unnecessary redundant messages, which cause the system to be inefficient and unsalable. The informed search exploits some kind of routing information, which is saved in the peers, about forwarding the queries to suitable peers. Although the informed search offers a faster response time than blind search algorithms, they increase the cost of the search and the overhead of maintaining the indexes.

Hybrid strategies try to combine the advantages offered by the different architecture types to boost the overall system performance. This type of strategy can combine two or more search algorithms. These algorithms may include a structured strategy or unstructured architecture, or even both strategies. The main challenge for a hybrid strategy is to find the optimum formula for the combination of its original searching algorithms. The formula should present the best threshold

points of hops to switch from one algorithm to other. This chapter proposed a hybrid search algorithm to prove two objectives.

- The hybrid search algorithm gains better results for unstructured P2P networks.
- The *critical metric* can determine the best threshold to switch from one algorithm to another in a hybrid search.

The proposed search algorithm combines two flooding-based search algorithms to benefit from their merits and to limit their drawbacks. The algorithm used flooding and teeming search algorithms.

5.2 Proposed QuickFlood Model

QuickFlood has been designed in two steps: in the first step it starts with a flooding algorithm with a limited number of hops to benefit from the high coverage growth rate of messages and low number of redundant messages. Whenever flooding starts to generate a high amount of redundant messages it switches from flooding to a teeming algorithm to benefit from the low number of redundant messages generated in teeming algorithms.
The main objective here is to find an optimum threshold of hops for the switch from flooding to teeming.

5.3 First Step of Algorithm

QuickFlood is started by source nodes such as S_i, where $i = \{1, 2, 3, ..., k\}$, and actions flooding by the limiting number of the Time-To-Live value, TTL_F. The typical source node S_i sends the requested query message to all its immediate neighbors, which, in turn, forward the message to all their neighbors except the message sender. The query reaches its end when either its TTL_F becomes zero, or because it becomes a redundant message or the number of found messages reaches

its maximum. If the procedure is ended by latter case (number of found messages is sufficient) the search is finished. Otherwise when $TTL_F = 0$ those peers reached on last hop of this step become last peers of the first step $\{n_{i1}, n_{i2}, ..., n_{ik}\}$, and thus the second step of search begins with these peers.

It is assumed that the source peer S_k, publishes a query message Q_m, with a TTL value for querying equal to TTL_F and the maximum number of found queries is set to MAX.

Figure 5.1 illustrates the flooding algorithm when the query message encounters a typical peer, and Figure 5.2 presents the first step procedure of QuickFlood.

Figure 5.1: Pseudo code Flooding(S_k, Q_m, TTL_F, MAX)

```
1  Q_m.TTL ← TTL_F;
2  Peer ← all nbr of S_k;
3  while Q_m.TTL > 0 do
4      Q_m.TTL ← Q_m.TTL − 1;
5      NewPeer ← ∅; {Initialize NewPeer as an empty set}
6      {For each neighbor}
7      for ∀ P_i ∈ Peers do
8          if Q_m.ID ∈ S_k.TB.ID then
9              {Peer visited before}
10             Redundant ← Redundant + 1;
11         else
12             P_i.TR ← P_i.TR ∪ Q_m.ID;
13             {Add the id of query to trace array of peer then check its content}
14             if Q_m.KM ∈ P_i.CM then
15                 Find ← Find + 1;
16                 if Find > MAX then
17                     Return;
18                 end
19             else
20                 Not_Find ← Not_Find + 1;
21             end
22             NewPeer ← NewPeer ∪ P_i.ID; {Add nbrs of P_i to NewPeer}
23         end
24     end
25     Peer ← NewPeer; {Assign NewPeer to Peer for next run}
26 end
```

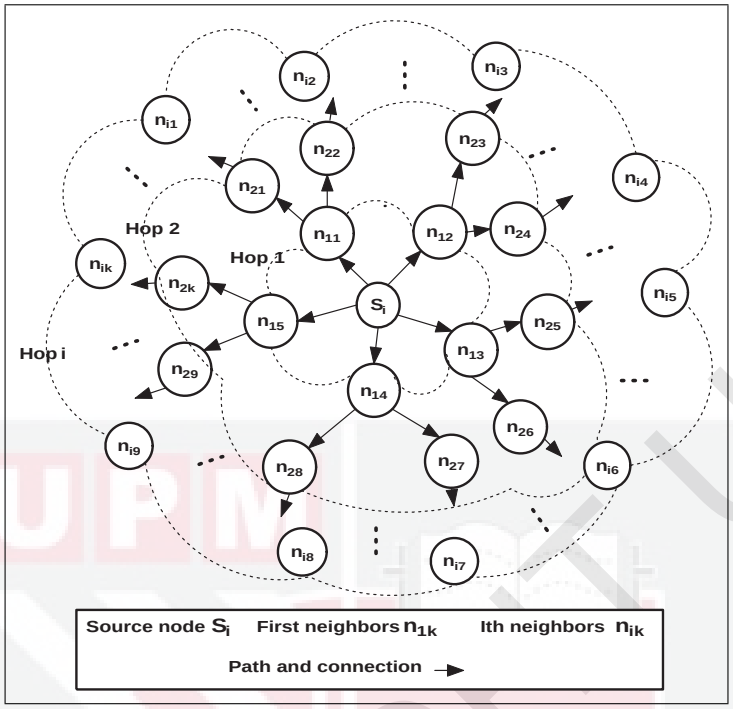

Figure 5.2: First step of QuickFlood procedure

5.4 Second Step Description

In this step, QuickFlood begins with the last peers visited in the first step $\{n_{i1}, n_{i2}, ..., n_{ik}\}$. These peers are dispersed uniformly across the overlay network. In this step, every last peer acts as in teeming algorithm with a fixed probability of θ.

Nodes Selection

In a teeming algorithm each node selects θ subset of its immediate neighbors for searching. The main assumption is that if the subset becomes less than one it has selected at least one neighbor for searching. Thus every node in the second step selects at least one node for searching. Figure 5.3 shows the procedure for one selected node in the second step of QuickFlood.

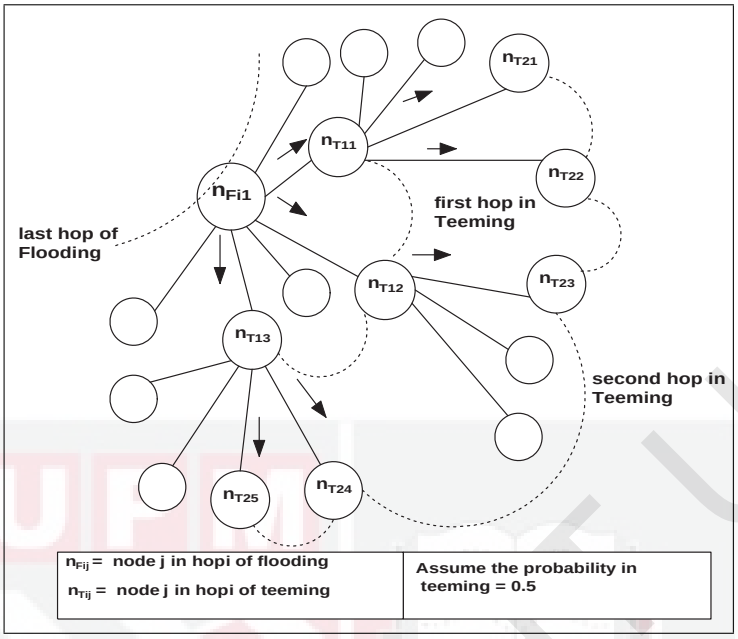

Figure 5.3: Second step of QuickFlood procedure

5.5 Second Step Algorithm

When queries reach the last nodes of the first step, and the maximum number of found is not satisfied, the search process of the first step has finished. The QuickFlood scheme marks these nodes as the starting nodes for the second step of the search.

The search procedure in this step initiates by replacing the Time-To-Live value of each query in marked peers with a second Time-To-Live value $Q.TL = TTL_S$ and begins the second step of the search algorithm. The search then continues simultaneously from this group of peers that are randomly and uniformly located in the P2P network. This algorithm is explained in Figure 5.4. It continues until its Time-To-Live value is valid, or obtains enough number of found messages.

5.6 Analytical Studies

This section analytically evaluated the performance of the proposed search algorithm by using the *critical metric*.

The *critical metric* in the QuickFlood algorithm is a combination of the *critical metrics* from flooding and teeming. It assumes that the coverage growth rate in flooding at hop t is equal to K and the number of redundant message in flooding at hop t is equal to L.

Assumptions:

- Coverage growth rate in hop t for flooding is: $CGR_{F,t} = K$
- Number of redundant messages in hop t for flooding is: $R_{F,t} = L$

Thus the *critical metric* at hop t for flooding can be shown as:

$$CM_{F,t} = \frac{L}{K} \tag{5.1}$$

Optimum Hop Count to Switch between Two Steps

By Equation (4.41) the *critical metric* for teeming in hop t with a fixed probability of θ is equal to:

$$CM_{T,\theta,t} = \frac{L}{K}\theta^{(t-2)} \tag{5.2}$$

The *critical metric* in QuickFlood is a combinations of the *critical metric* in flooding and teeming. Thus the *critical metric* for QuickFlood can be shown as:

$$CM_{QF,t} = \frac{L}{K} + \frac{L}{K}\theta^{(t-2)} = \frac{L}{K}(1 + \theta^{(t-2)}) \tag{5.3}$$

Hence, since θ is less than one (because it is a probability) the value of $\frac{L}{K}\theta^{(t-2)}$ is

Figure 5.4: Pseudo code QuickFlood_Second_Step($S_k, Q_m, TTL_S, MAX, LN, \theta$)

```
1  S_K ← is source peer;
2  Q_m ← is defult query message;
3  Q_m.TL ← TTL_S;{Assign TTL value of query as TTL value for second step}
4  MAX ← is maximum number of requested query;
5  LN ← is a set contain all last nodes remined from first step;
6  θ ← is a rate of selecting node in second step;
7  while Q_m.TL > 0 do
8      Temp_array ← ∅; {Initialize an empty set or array}
9      Q_m.TL ← Q_m.TL − 1;
10     {n_ij is a typical node in LN}
11     {For each n_ij do}
12     for ∀ n_ij ∈ LN do
13         if Q_m.ID ∈ n_ij.TR then
14             {n_ij visited before}
15             Redundant ← Redundant + 1;
16         else
17             { Check the content of n_ij }
18             if Q_m.KW ∈ n_ij.CM then
19                 Find ← Find + 1;
20                 if Find > MAX then
21                     Return;
22                 end
23             else
24                 Not_Find ← Not_Find + 1;
25             end
26         end
27         {Procedure selecting new nodes for next turn.}
28         {This Procedure performs in two steps.}
29         { 1-Find number of nodes according to θ and number of nbrs of n_ij}
30         Number_of_node ← round(θ × number of nbrs of n_ij);
31         if Number_of_node < 1 then
32             Number_of_node ← 1;
33         end
34         {2-Choose Number_of_node selected nodes randomly form neighbors of n_ij }
35         i ← 1;
36         Node_Selected ← ∅; {Assign as an empty set or array }
37         for i ≤ Number_of_node do
38             Node_Selected ← Node_Selected ∪ select a node uniformly from nbrs of
               n_ij;
39         end
40         Temp_array ← Temp_array ∪ Node_selected;
41     end
42     LN ← Temp_array;
43 end
```

not less than $\frac{L}{K}$ for $t = 1$, and 2. By considering $\theta < 1$:

$$\frac{L}{K}\theta^{(t-2)} = \begin{cases} \frac{L}{K}\theta^{(1-2)} = \frac{L}{K}\theta^{(-1)} = \frac{L}{K\times\theta^1} > \frac{L}{K} & \text{if} \quad t = 1 \\ \frac{L}{K}\theta^{(2-2)} = \frac{L}{K}\theta^{(0)} = \frac{L\times\theta^0}{K} = \frac{L}{K} & \text{if} \quad t = 2 \\ \frac{L}{K}\theta^{(3-2)} = \frac{L}{K}\theta^{(1)} = \frac{L\times\theta^1}{K} < \frac{L}{K} & \text{if} \quad t = 3 \\ \frac{L}{K}\theta^{(4-2)} = \frac{L}{K}\theta^{(2)} = \frac{L\times\theta^2}{K} < \frac{L}{K}\theta^{(1)} & \text{if} \quad t = 4 \\ \ldots \end{cases}$$

Thus it is not rational to use the teeming algorithm in hops 1, and 2 in this combination. But for $t \geqq 2$ the value of $\frac{L}{K}\theta^{(t-2)}$ starts to decrease compared to $\frac{L}{K}$ and the rate of decrease depends on the value of θ.

So the optimum threshold for switching from flooding to teeming is when $t \geqq 2$. Therefore, the best combination for QuickFlood is to use the flooding algorithm in the first two hops, and the teeming algorithm for the rest of the hops.

The *critical metric* in QuickFlood is equal to:

$$\begin{aligned} CM_{QF,\theta,t} &= CR_{F,t} + CR_{T,\theta,t} \\ &= A^{t-2} + (A \times \theta)^{t-2} \\ &= A^{t-2}(1 + \theta^{t-2}) \end{aligned} \tag{5.4}$$

Here we calculate the value of *critical metric* for $t = 3$, 4, and 5.

$$CM_{QF,\theta,t} = A^{t-2}(1+\theta^{t-2}) = \begin{cases} A^{2-2}(1+\theta^{2-2}) = A^0(1+\theta^0) & \text{if} \quad t=2 \\ A^{3-2}(1+\theta^{3-2}) = A^1(1+\theta^1) & \text{if} \quad t=3 \\ A^{4-2}(1+\theta^{4-2}) = A^2(1+\theta^2) & \text{if} \quad t=4 \\ A^{5-2}(1+\theta^{5-2}) = A^3(1+\theta^3) & \text{if} \quad t=5 \\ \ldots \end{cases}$$

The results show the value of *critical metric* in $t = 2$ is minimum compare with $t > 2$. Thus, the optimum threshold for switching from flooding to teeming is when $t = 2$, because for $t > 2$ the value of *critical metric* increases exponentially.

5.7 Experimental Results

The goal of the evaluation is to study the performance of QuickFlood compared with flooding, expanding ring, blocking expanding ring, and teeming algorithms. The proposed algorithm is implemented in two steps. To perform this evaluation we have used the same experimental setup and metrics, which are explained in section 4.5. In the first step, M hops are performed with the flooding algorithm, and in the second step it continues N hops with the teeming algorithm by a fixed probability of θ.

Hence, there is an interesting question which must be investigated in the (M and N) arrangement.

- What is the effect of increasing or decreasing M and N on the performance of QuickFlood?

In the previous studies include analytically, and experimentally we have proven the blocking expanding ring algorithm produces better results than standard flooding

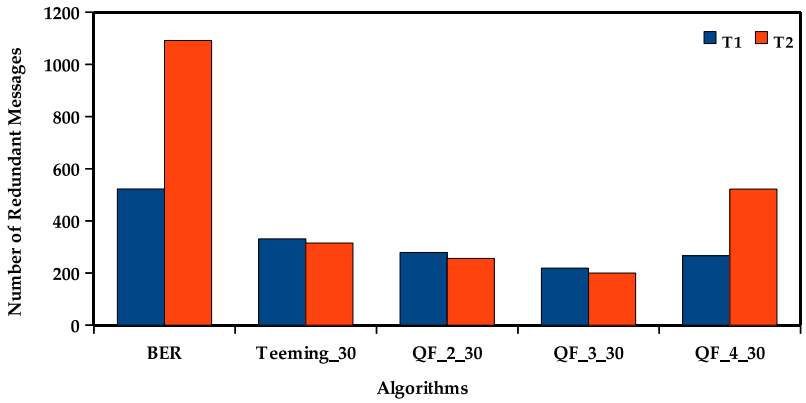

Figure 5.5: Compare the number of redundant messages of QuickFlood with different arrangements by other algorithms at different topologies

and expanding ring algorithms. Thus in the remainder of the experiments we compare the results with the blocking expanding ring.

Table 5.1 and Figure 5.5 compare the number of redundant messages produced under the same conditions for blocking expanding ring (BER), teeming and different combinations of hops for the QuickFlood algorithm. QuickFlood has the lowest number of redundant messages compared with the other algorithms. It proved the analytical results that QuickFlood has the best arrangement in hop three. The experiments prove that by increasing the number of hops in flooding the number of redundant messages increases. The main reason is that flooding in these hops starts to produce more redundant messages.

Table 5.1: The reduced percentages of redundant messages for each algorithm compare with BER algorithm at different topologies (%)

Topology / Algorithm	T1	T2	Average
*Teeming*_30	37	71	54
*QF*_2_30	47	77	62
*QF*_3_30	58	82	70
*QF*_4_30	49	52	51

Table 5.2 and Figure 5.6 show the success rate from the blocking expanding ring,

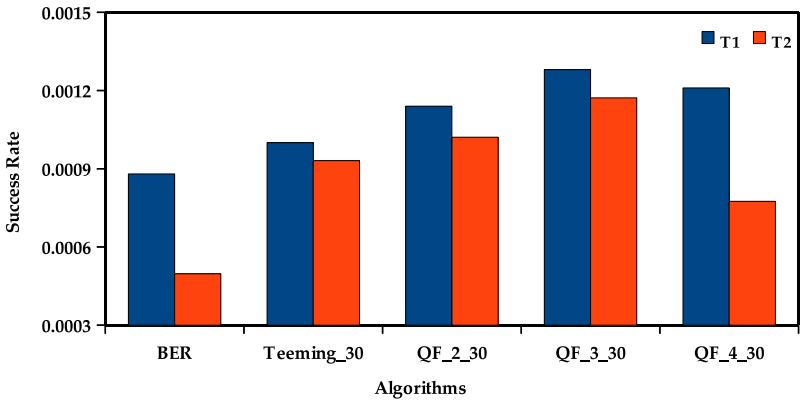

Figure 5.6: Compare the values of success rate of QuickFlood with different arrangements by other algorithms at different topologies

teeming, and different combinations of QuickFlood. The QuickFlood algorithms with ($h = 3$ and $\theta = 0.3$) achieve better success rates compared with others, and has on average a 1.9 times success rate compared with the blocking expanding ring. It proves the analytical results that this algorithm achieves the best performance in the combination of four hops flooding and then the rest teeming.

Table 5.2: The increased times of success rate for each algorithm compare with BER algorithm at different topologies

Algorithm \ Topology	T1	T2	Average
$Teeming\_30$	1.1	1.9	1.5
QF_2_30	1.3	2.1	1.7
QF_3_30	1.5	2.4	1.9
QF_4_30	1.4	1.6	1.5

Table 5.3 and Figure 5.7 presented the amount of latency for the different algorithms. QuickFlood with the arrangement ($hop = 3$ and $\theta = 0.3$) achieves a better result than the teeming algorithms. The randomly chosen nodes have more latencies than the other algorithms, because there are many high performing nodes that may be omitted, and thus it took more time to capture the target nodes. The results show that the QuickFlood by three hops of flooding gained better results

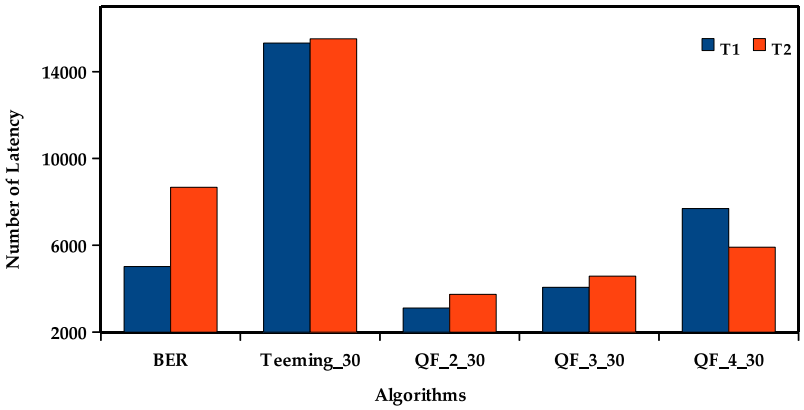

Figure 5.7: Compare the number of latency of QuickFlood with different arrangements by other algorithms at different topologies

than QuickFlood using 2 hops and teeming.

Table 5.3: The reduced percentages of number of latencies for each algorithm compare with BER algorithm at different topologies (%)

Algorithm \ Topology	T1	T2	Average
*Teeming*_30	-205	-79	-142
*QF*_2_30	38	57	47
*QF*_3_30	19	47	33
*QF*_4_30	-53	32	-11

5.8 Summary

This chapter introduced a new searching algorithm, called QuickFlood. QuickFlood combined two flood-based search algorithms (flooding and teeming) to benefit from their merits and to limit their drawbacks. QuickFlood follows a hybrid search discipline. Flooding in low-hops has fewer redundant messages and a large coverage growth rate. Thus the algorithm in the first step used flooding to achieve a large coverage growth rate in the network and in the meantime a low number of redundant messages. Whenever flooding starts to generate a high volume of re-

dundant messages it switches from the flooding to the teeming algorithm to benefit from its merits. The teeming algorithm produces fewer redundant messages but incurs a high latency. The main problem of this class of algorithm is determining which hop must switch from one algorithm to another to benefit from their merits and limit their drawbacks. In this chapter thesis analytically and experimentally proved that the best threshold to switch from flooding to teeming is in hop three. The results show that QuickFlood achieves a better performance compared to the other flood-based algorithms.

CHAPTER 6
PROPOSED HYBRIDFLOOD SEARCH ALGORITHM

6.1 Introduction

Flooding has an essential problem when faced with high aggregate query rate. Nodes become overloaded and the network closes down to function unsatisfactorily. Furthermore, this problem gets worse as the size of the network increases. As mentioned in section 2.6.1 there are three major solutions to alleviate the effects of flooding. This thesis has named them TTL Limit-Based Flooding (TLBF), Probability Limit-Based Flooding (PLBF), and Hybrid Limit-Based Flooding (HLBF). The main goal in the hybrid search is to combine two searching protocols to benefit from their merits and limit their drawbacks. The significant question is how to combine the search protocols to gain the maximum benefit and minimum drawback?

This chapter introduces a hybrid search algorithm for an unstructured P2P network, which combines flooding with super-peers. Super-peers exploit the heterogeneity of the peers participating in the network.

6.2 Proposed HybridFlood Search Model

The essential problem of flooding starts when it is faced with a high aggregate query rate, as nodes become overloaded and the network closes down to function satisfactorily. Furthermore, this problem gets worse as the size of the network increases. Our main idea behind the HybridFlood search is to enhance the flooding search so that it can handle a much higher aggregate query rate, and to function well with an increasing network size. To achieve this scalability, our HybridFlood integrated flooding and super-peer searching schemes to cope with this problem. HybridFlood is designed in two steps:

1. In low-hops, it processes flooding to benefit the high coverage growth rate of messages and the low number of redundant messages.

2. In high-hops, when there are too many redundant messages and a low coverage growth rate of messages, it does not follow the flooding algorithm. Instead, it starts selecting nosey nodes as super-peers to obtain the advantage of the heterogeneity of capabilities across peers. These super-peers provide good scalability, the opportunity to take advantage of node heterogeneity, and a high routing efficiency.

6.3 First Step of Algorithm

In this step, HybridFlood starts with source peers for instance S_i, $i = \{1, 2, 3, \ldots, k\}$, and actions flooding by the limiting number of the Time-To-Live value TTL_F the typical source peer S_i send the requested query message to all its immediate neighbors $\{P_1, P_2, \ldots, P_n\}$, which, in turn, forward the message to all their neighbors except the message sender. This procedure is conducted in a hop-by-hop fashion counted by the TTL_F count. The query reaches is end when its TTL_F becomes zero, it becomes a redundant message, or the number of found messages reaches the maximum number of found. If the procedure has ended due to latter case (number of found messages is enough) the search is finished. Otherwise, when $TTL_F = 0$ those peers reached at the last hop of this step become the last peers of the first step $\{LP_1, LP_2, \ldots, LP_n\}$, so the second step of search begins with these peers.

It is assumed that the source peer is S_k, query message Q_m, a Time-To-Live value equal to TTL_F and the maximum number of found messages are equal to $S_k.TB.MAX$. The algorithm of flooding is same as in Figure 5.1 in chapter 5.

6.4 Second Step Description

In this step, HybridFlood begins with the last peers that have been visited in the first step $\{LP_1, LP_2, \ldots, LP_n\}$. These peers are dispersed uniformly across the overlay network. In this step every last peer is picked its relevant nosey node, and collects its meta data and then start processing the second step of the HybridFlood search.

This section specifically describes the nosey nodes selection, indexing procedures in nosey nodes, nosey node overlay network, redundancy nosey nodes, the search procedure in the second step, and discusses the nosey node.

6.4.1 Nosey Node Selection

For selecting nosey nodes; each last peer such as LP_i has checked its immediate neighbors, which were not queried previously and has called them valid neighbors, $Valid.P_i$. Then from these valid neighbors it selects the neighbor with the most links. This neighbor is called a nosey node, NN_i. The high-degree nodes actually have more pointers to a larger number of files and hence will be more likely to have an answer that matches the query. Each nosey node is related to at least one specific source node, such as S_i. It may be a possible that one nosey node is related to more than one source node.

The nosey node exploits its heterogeneity by introducing a two level hierarchy of peers. These nosey nodes arrange an independent sub-topology within the existing P2P network. Nosey nodes operate as super-peers. They act as a centralized server to a subset of its clients. Figure 6.1 illustrates the nosey node selection in the second step source node start querying by flooding in 2 hops. Each final node of the flooding step in the second hop selects a neighbor with the highest degree of connections and call it a nosey node.

Each nosey nodes connect to its clients and caches the addresses of their files.

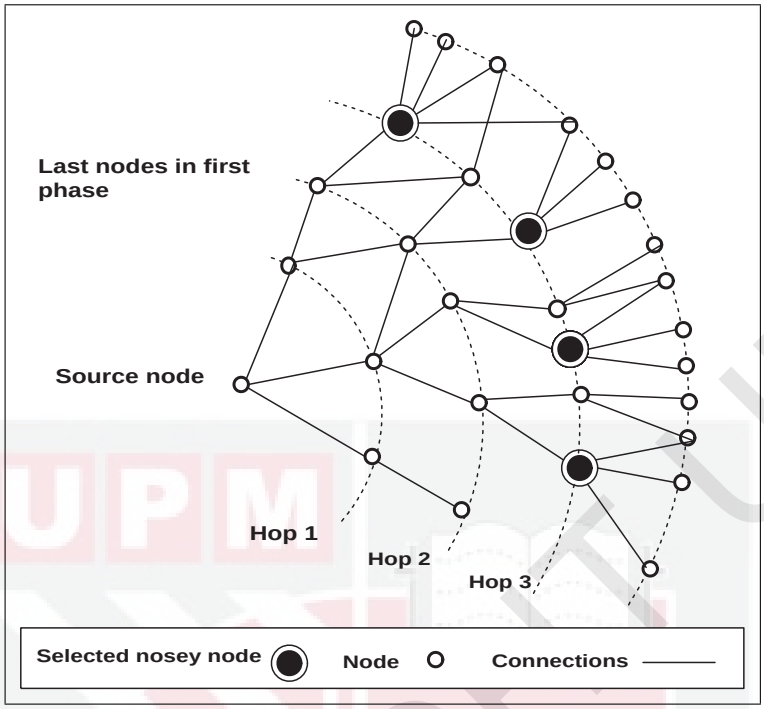

Figure 6.1: Nosey nodes selection

Nosey nodes have two responsibilities: first they collect data index of their clients, and second they answer behalf of their clients for query(s). Figure 6.2 explains the algorithm which select nosey nodes.

Nosey nodes create three more fields for storing information about their indices, clients, and the reserved nosey node $NN.CT$, $NN.TA$, and $NN.RD$. Nosey nodes collect its meta data indexes in $NN.CT$. The id of its clients and id of the reserved nosey node are stored respectively in $NN_i.TA$ and $NN_i.RD$.

Each nosey node has a unique id, and it is stored in $NN.ID$. All nosey nodes created in each round have the same *hop count* numbers. The *ids* and *hop count* numbers of nosey nodes are stored in the relevant source peer field $S_i.NT$. Figure 6.3 presents the algorithm which returns the clusters of a typical nosey node.

Based on the algorithm, each client can be linked to more than one nosey node. Thus, if one nosey node fails the clients may be connected to other nosey nodes, this improves the network's reliability and fault tolerance. To guarantee the reliability,

Figure 6.2: Pseudo code Select_Nosey_Nodes(S_i, LP_i, Q_m)

```
1  LP_i ← One typical last node remind from first step;
2  |P_i.NI| ← Represent the degree of node P_i;
3  Link ← 1;
4  { For all neighbors of LP_i}
5  for ∀ P_i ∈ nbr LP_i do
6      if Q_m.ID ∈ P_i.TR then
7          Continue; {The node visited before therefore, by pass it}
8      else
9          {Check the degree of neighbors }
10         Link1 ← |P_i.NI|;
11         if Link1 > Link then
12             Temp ← P_i.ID;
13             Link ← Link1;
14         end
15     end
16 end
17 {Assign the id of neighbor with most degree to nosey node}
18 NN_i.ID ← Temp;
19 {Assign the id and hop count of nosey node to relevant source node}
20 S_i.NT.ID ← NN_i.ID;
21 S_i.NT.HP ← Q_m.TL;
22 Return(NN_i.ID);
```

we propose a redundancy nosey node. At the same time, it is possible that a nosey node operates as an ordinary node for other source peers.

6.4.2 Indexing Procedure in Nosey Nodes

After selecting each nosey node in the region, the nosey node then creates an index table, which indexes all the files of their clients. Figure 6.4 illustrates the collection of meta data in nosey nodes. This meta data contains the names and addresses of the files which belong to their clients.

In order to maintain this index, when a client joins leaves or updates its data, it sends new meta data to the selected nosey node. Figure 6.5 explains the algorithm for refreshing and updating indexes of nosey nodes.

Figure 6.3: Pseudo code Select_Nosey_Nodes_Clusters(LP_i, Q_m)

```
1  LP_i ← One typical last node remind from first step;
2  Valid.LP_i ← ∅; {Valid neighbors of LP_i assign as empty set}
3  { For all neighbors of LP_i}
4  for ∀ P_i ∈ nbr LP_i do
5      if Q_m.ID ∈ P_i.TR then
6          Continue; {The node visited before, so, by pass it}
7      else
8          Valid.LP_i ← Valid.LP_i ∪ P_i.ID; {Add this peer to valid neighbors }
9      end
10 end
11 Return(Valid.LP_i);
```

Figure 6.4: Pseudo code Collect_Index_Nosey_Nodes($NN_i.ID$)

```
1  {For all client of NN_i}
2  for ∀ P_i ∈ NN_i do
3      {If node is active }
4      if P_i.ST = True then
5          {Assign id of client to related field of nosey node}
6          NN_i.CT.ID ← NN_i.CT.ID ∪ P_i.ID;

7          {Assign file names of client to related field of nosey node cache}
8          NN_i.CT.File_name ← NN_i.CT.File_name ∪ P_i.CM;
9      end
10 end
11 Return(NN_i.ID);
```

6.4.3 Nosey Node Overlay Network

Nosey nodes construct a secondary overlay, as a super-peer overlay. It is similar to the original P2P overlay except that it is completely composed of the nosey nodes created in each round. Nosey nodes with the same hop count numbers from specific source peers link and construct a nosey node overlay. These nosey nodes have multiple roles: first, they are ordinary peers in the original overlay. Second, they are super-peers in the secondary overlay; where they act as a centralized server for their subset of clients. Clients submit queries to them and receive results from them.

Figure 6.5: Pseudo code Refresh_Index_Nosey_Nodes($NN_i.ID$)

```
1  {If node x join to nosey node's cluster }
2  if Join (node_x) then
3  |   {Assign id of client to related field of nosey node cache}
4  |   NN_i.CT.ID ← NN_i.CT.ID ∪ node_x.ID;
5  |   {Assign file names of client to related field of nosey node cache}
6  |   NN_i.CT.File_name ← NN_i.CT.File_name ∪ node_x.CM;
7  end

8  {If node x leave the nosey node's cluster }
9  if Leave (node_x) then
10 |   {Delete id of client from related field of nosey node}
11 |   NN_i.CT.ID ← NN_i.CT.ID − node_x.ID;
12 |   {Delete file names of client from related field of nosey node cache}
13 |   NN_i.CT.File_name ← NN_i.CT.File_name − node_x.CM;
14 end

15 {If node x Update its files }
16 if Update (node_x) then
17 |   {Assign id of client to related field of nosey node}
18 |   NN_i.CT.ID ← NN_i.CT.ID ∪ node_x.ID;
19 |   {Assign file names of client to related field of nosey node cache}
20 |   NN_i.CT.File_name ← NN_i.CT.File_name ∪ node_x.CM;
21 end
22 Return(NN_i.ID);
```

6.4.4 Nosey Node Redundancy

The main problem in a hierarchical structure is its single point failure. When a nosey node fails or leaves all its clients become disconnected. To provide reliability, we propose a redundancy nosey node in each cluster. The redundancy nosey node is a node which has the most links in the cluster. Therefore, if the main nosey node leaves or fails, this redundancy nosey node acts as the main nosey node.

6.4.5 Discussion on Nosey Nodes

Those nodes reached on the last hop of the first step $TTL_F = 0$ become kernels, from which nosey nodes are initiated in the second step. Nosey nodes are selected

at little cost by using local information. Each kernel only uses local information. It checks its immediate neighbors for their degree and selects the node that has the highest degree and calls it a nosey node. In this fully autonomous and highly dynamic network, only local information can be inexpensively available. Nosey nodes a have high degree and a large number of files and hence will be more likely to contain an answer that matches the query. In addition, nosey nodes maintain the index of their immediate neighbors which improves its accountability.

Its cache requires only negligible memory and has no overhead to network. The nosey nodes are selected after limited hops of flooding. Thus, the kernels of nosey nodes are dispersed randomly and uniformly across the network. Therefore, the coverage of nosey nodes in the network is considerable. Nosey nodes decrease the number of messages broadcast in the second step by an order of magnitude. Thus, it significantly decreases the network traffic and improves the search efficiency.

6.5 Second Step of Algorithm

This step started by last peers visited in first step. When queries reach these last nodes, and the maximum number of found is not satisfied, the search process in the first step has finished. Our HybridFlood scheme marks these nodes as starting points for the second step of the search.

The search procedure in this step initiates by replacing the Time-To-Live value of each query in the marked peers with a second Time-To-Live value $Q.TL = TTL_S$. Then the search simultaneously continues from this group of peers, which are randomly and uniformly located across the P2P networks. This procedure continues in the following two steps until its Time-To-Live value is valid, or it has obtained enough found messages.

First, the TTL value of each query in marked nodes is decremented by one. Then the query is forwarded to the new nosey nodes' overlay. The search in this step

follows the super-peer search discipline. The query checks the index field of each nosey node in the new overlay, if found it returns the address of the relevant client(s) and increments the number of found messages by one, otherwise it increment the number of messages not found by one.

Second, all cluster nodes of all nosey nodes become the last nodes in this step. Thus each of these nodes operates the same as the last node in the first step. So, again, each of these last nodes selects new nosey nodes, collect their meta data, and begin a new search. This scheme obviously decreases the volume of traffic and load on the network and increases the efficiency and quality of the search. The Figure 6.6 explain the search algorithm used in the second step of HybridFlood.

6.6 Analytical Study

The analytical study of HybridFlood search has two objectives. First it estimates the optimum threshold point of hops for switching from the first step of the algorithm to the second step. Second, it compares the number of redundant messages produces in the second step of HybridFlood with standard flooding.

6.6.1 Optimum Hop Count to Switch between Two Steps

To evaluate the threshold point of a hop, we have used the critical metric. The critical metric in HybridFlood is a combination of a critical metric in the first and second step of the algorithm. The first step of HybridFlood follows a flooding search. Thus its critical metric, with reference to Equation (4.30), is equal to $A^{(t-2)}$. In the second step, HybridFlood follows the super-peer search discipline. Thus in the second step the critical metric is equal to the critical metric in the local indexes. This local index has just one hop domain, because it is collecting the data of its immediate neighbors.

The critical metric in the second step is negligible, because the number of redun-

Figure 6.6: Pseudo code HybridFlood_Second_Step(LP, Q_m, TTL_S, MAX)

```
1  {Initialization}
2  LP ← Set of last nodes remain from first step;
3  Q_m ← Default query message;
4  TTL_S ← TTL value in second step;
5  Q_m.TL ← TTL_s;
6  while Q_m.TL > 0 do
7      Temp_array ← ∅; {Assign empty set}
8      Q_m.TL ← Q_m.TL - 1 ;
9      {for all typical last node in second step}
10     for ∀ LP_i ∈ LP do
11         {Check the node visited before? }
12         if Q_m.ID ∈ LP_i.TR then
13             Redundant ← Redundant + 1;
14         end
15         {do following functions}
16         {Select its nosey node}
17         NN_i.ID ← Select_Nosey_Nodes(LP_i, Q_m, S_i);
18         {Collect index of its client}
19         NN_i.ID ← Collect_Index_Nosey_Nodes(NN_i.ID);
20         {Refresh its index}
21         NN_i.ID ← Refresh_Index_Nosey_Nodes(NN_i.ID);
22         {Get clusters of nosey node}
23         Cluster_nodes ← Select_Nosey_Nodes_Clusters(LP_i, Q_m);
24         {Now collect clusters of nosey nodes }
25         Temp_array ← Temp_array ∪ Cluster_nodes;
26         {Check content of selected nosey nodes }
27         if Q_m.KW ∈ NN_i.CT then
28             Find ← Find + 1;
29             if Find > MAX then
30                 Return;
31             end
32         else
33             Not_Find ← Not_Find + 1;
34         end
35     end
36     {Last nodes for next turn }
37     LP ← Temp_array;
38 end
```

dant message in the first hop is almost 0. Hence, the critical value of HybridFlood depends on first step or just flooding. Equations (4.13), (4.19), and (4.30) proved that the optimum point of hops in flooding are its first two hops. Hence the best threshold point of a hop to switch between the two steps is in hop three.

6.6.2 Analytical Study in the Second Step

This section compares the number of messages broadcast in the same hops for flooding and in the second step of the HybridFlood search. In the second step of HybridFlood, messages are broadcast in alternate hops. Assuming that the HybridFlood search in the second step begins at hop h: thus the number of messages broadcast in this hop is equal to zero because it is selecting nosey nodes. However, in the next hop the number of message broadcast is equal to:

$$M_{HF,h+1} = d(d-1)^h \tag{6.1}$$

Thus, the number of messages broadcast from this hop to hop t, is equal to:

$$\begin{aligned} TM_{HF,h,t} &= \sum_{i=h}^{\lfloor t/2 \rfloor} d(d-1)^{i-1} \\ &= \frac{d(d(d-1)^{\lfloor t/2 \rfloor} - (d-1)^h - (d-1)^{\lfloor t/2 \rfloor})}{(d-1)(d-2)} \\ &= \frac{d((d-1)^{\lfloor t/2 \rfloor+1} - (d-1)^h)}{(d-1)(d-2)} \end{aligned} \tag{6.2}$$

The value of t must be greater than two, because in HybridFlood first two hops are process as flooding. In the same hops, the number of messages broadcast in

flooding is equal to:

$$\begin{aligned} TM_{F,h,t} &= \sum_{i=h}^{t} d(d-1)^{i-1} \\ &= \frac{d(d(d-1)^t - (d-1)^h - (d-1)^t)}{(d-1)(d-2)} \\ &= \frac{d((d-1)^{t+1} - (d-1)^h)}{(d-1)(d-2)} \end{aligned} \tag{6.3}$$

Thus:

$$\begin{aligned} \frac{TM_{HF,h,t}}{TM_{F,h,t}} &= \frac{\frac{d((d-1)^{\lfloor t/2 \rfloor+1} - (d-1)^h)}{(d-1)(d-2)}}{\frac{d((d-1)^{t+1} - (d-1)^h)}{(d-1)(d-2)}} \\ &= \frac{(d-1)^{\lfloor t/2 \rfloor+1} - (d-1)^h}{(d-1)^{t+1} - (d-1)^h} \end{aligned} \tag{6.4}$$

By simplifying Equation (6.4):

$$\begin{aligned} \frac{TM_{HF,h,t}}{TM_{F,h,t}} &= \frac{\frac{(d-1)^{\lfloor t/2 \rfloor+1} - (d-1)^h}{(d-1)^h}}{\frac{(d-1)^{t+1} - (d-1)^h}{(d-1)^h}} \\ &= \frac{(d-1)^{\lfloor t/2 \rfloor+1-h} - 1}{(d-1)^{t+1-h} - 1} > \frac{(d-1)^{\lfloor t/2 \rfloor+1-h} - 1}{(d-1)^{t+1-h}} \end{aligned} \tag{6.5}$$

$$\begin{aligned} \frac{(d-1)^{\lfloor t/2 \rfloor+1-h} - 1}{(d-1)^{t+1-h}} &= \frac{(d-1)^{\lfloor t/2 \rfloor+1-h}}{(d-1)^{t+1-h}} - \frac{1}{(d-1)^{t+1-h}} \\ &\approx \frac{(d-1)^{\lfloor t/2 \rfloor+1-h}}{(d-1)^{t+1-h}} \\ &\approx \frac{1}{(d-1)^{\lfloor t/2 \rfloor}} \end{aligned} \tag{6.6}$$

With respect to Equations (6.5) and (6.6), can conclude:

$$\frac{TM_{HF,h,t}}{TM_{F,h,t}} \approx \frac{1}{(d-1)^{\lfloor t/2 \rfloor}} \tag{6.7}$$

Equation (6.7) proves that our HybridFlood search in the second step cuts down the number of message propagated by at least an order of magnitude compared with flooding.

6.7 Experimental Results

The goal of this evaluation is to compare HybridFlood with blocking expanding ring, teeming and QuickFlood algorithms. HybridFlood is implemented in two steps. In the first step it performs M hops by flooding, and in the second step it continues with N hops by choosing nosey nodes in each hop.
Hence, there are two interesting questions which must be investigated with an M, N arrangement.

1. What is the effect of increasing or decreasing M on the performance of HybridFlood?

2. How does the redundancy nosey node impact on the searching efficiency?

To perform this evaluation, we have used the same experimental setup and metrics explained in section 3.4.3.
Table 6.1 and Figure 6.7 compare the number of redundant messages produced under the same conditions for Blocking Expanding Ring (BER), Teeming($\theta = 0.30$), QuickFlood ($t = 4$, $\theta = 0.30$), and the proposed HybridFlood with a different number of hops and different number of redundancy nosey nodes. HybridFlood achieves the lowest number of redundant messages compared with the other algorithms. It confirmed our analytical results in which HybridFlood has the best arrangement at $t = 3$. That means three hops of flooding and the rest choosing super-peers as nosey nodes. Our proposed algorithm reduced redundant messages by more than 85% compared with blocking expanding ring.
The experiments show that increasing the number of hops in flooding increases the number of redundant messages. The results also show that HybridFlood with one

Table 6.1: The reduced percentages of redundant messages for each algorithm compare with BER algorithm at different topologies (%)

Algorithm \ Topology	T1	T2	Average
*Teeming*_30	37	71	54
*QF*_4_30	58	82	70
*HF*_3_0	85	89	87
*HF*_3_1	76	85	80
*HF*_3_2	59	31	45
*HF*_4_0	59	57	58
*HF*_4_1	54	56	55
*HF*_4_2	44	32	38

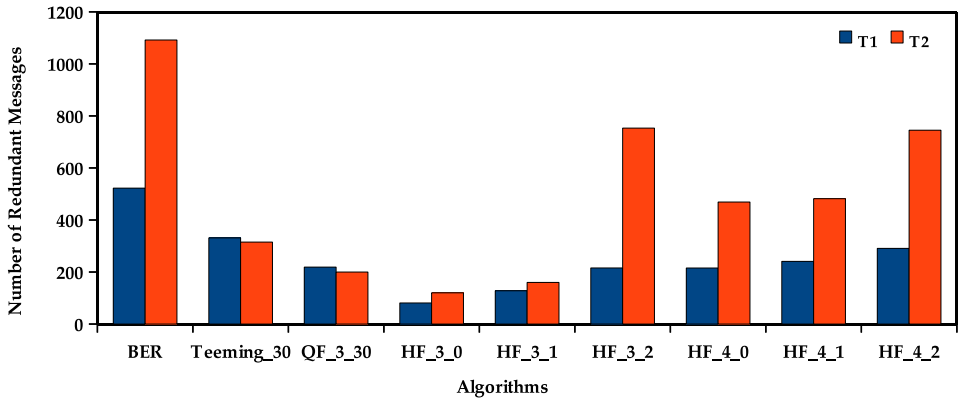

Figure 6.7: Compare the number of redundant messages of QuickFlood and HybridFlood with different arrangements by other algorithms at different topologies

redundancy nosey node produces better results compared with other algorithms, in that by increasing the number of redundancy nosey nodes the number of redundant messages increases.

Table 6.2 and Figure 6.8 show the success rate in blocking expanding ring, teeming, QuickFlood ($t = 4$, $\theta = 0.30$), and HybridFlood ($t = 3$ and 4 with 0, 1, and 2 redundancy nosey node) algorithms.

The HybridFlood ($t = 3$) algorithms achieves the best success rate compared with the others, on average its success rate is more than twice that of the blocking

expanding ring. It confirms our analytical results and proves that this algorithm achieves the best performance with a combination of three hops in flooding and the rest using nosey nodes.

Table 6.2: The increased times of success rate for each algorithm compare with BER algorithm at different topologies

Algorithm \ Topology	T1	T2	Average
*Teeming*_30	1.1	1.9	1.5
*QF*_4_30	1.5	2.4	1.9
*HF*_3_0	1.7	2.7	2.2
*HF*_3_1	1.5	2.3	1.9
*HF*_3_2	1.0	0.8	0.9
*HF*_4_0	1.4	1.7	1.5
*HF*_4_1	1.3	1.6	1.5
*HF*_4_2	1.0	1.0	1.0

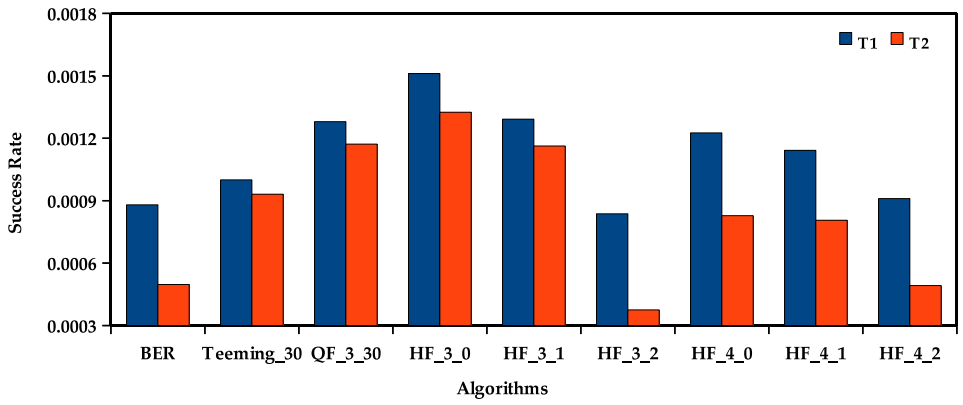

Figure 6.8: Compare the values of success rate of QuickFlood and HybridFlood with different arrangements by other algorithm at different topologies

The proposed algorithm with one redundancy nosey node even performs better than teeming, its success rate is almost same as QuickFlood with arrangements ($t = 4$ and $\theta = 0.3$). The results show that increasing the number of redundancy nosey nodes decreases the success rate. Table 6.3 and Figure 6.9 presented the

amount of latencies from the different algorithms. As expected, teeming again has the highest latency compared with the other algorithms. It gains more than two times the latency compared with the blocking expanding ring.

The proposed algorithm achieves the best latency compared with the other algorithms, it reduces the latency by approximately 80% compared with the blocking expanding rings. The results show that HybridFlood (with an arrangement of $t = 3$ and $redundancy = 1$) also achieves the best result.

Table 6.3: The reduced percentage of number of latencies in each algorithm compare with BER algorithm at different topologies (%)

Algorithm \ Topology	T1	T2	Average
Teeming_30	-205	-79	-142
QF_4_30	19	47	33
HF_3_0	77	82	79
HF_3_1	74	80	77
HF_3_2	64	40	52
HF_4_0	50	52	51
HF_4_1	48	51	50
HF_4_2	45	38	41

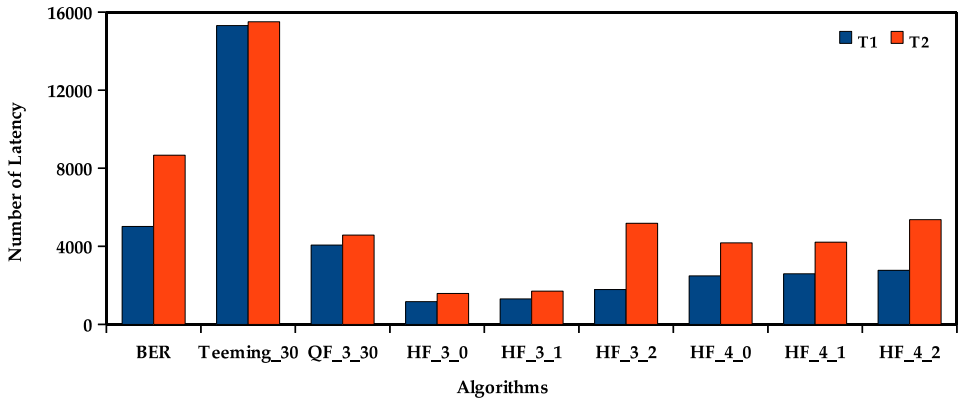

Figure 6.9: Compare the number latency of QuickFlood and HybridFlood with different arrangements by other algorithms at different topologies

6.8 Summary

The chapter started with the proposed HybridFlood search algorithm. The algorithm used two different classes of searching protocols; flooding and super-peers. The main objective of the hybrid search is to combine two or more searching protocols to benefit from their merits and limit their drawbacks. The essential question is how to combine the search protocols to gain the maximum benefit and minimum drawback. The thesis describes the design of the HybridFlood algorithm. HybridFlood includes two main steps; it starts with a flooding search in low-hops and then continues with nosey nodes in a super-peer search. The chapter describes in details the first and second steps of the HybridFlood algorithms. The first step is flooding for limited number of hops.

The second step includes algorithms for selecting nosey nodes, caching information for nosey nodes, selecting reserved nosey nodes and comprise them and performing the search algorithm in the second steps. Then the search performed by HybridFlood is analytically studied for two objectives. First for estimating the optimum threshold point of hops for switching from the first step to the second step. Second, to compare the number of messages propagating in the second step of HybridFlood with flooding at the same condition. The analytical study proved that the best threshold point to switch between the two steps of the hybrid flood is in hop three. The results of the comparison study proved that the number of messages that were propagated in the second step of HybridFlood were reduced by at least an order of magnitude compared with flooding.

The experimental results confirmed that HybridFlood, with an arrangement of $t = 3$ with zero and one redundancy nosey nodes, achieved the best performances compare with the other algorithms.

CHAPTER 7
CONCLUSION AND FUTURE WORKS

7.1 Conclusion

The upward trend of digital information production requires a scalable infrastructure that is capable of indexing and searching reaches content. The traditional solutions such as all search engines act need to maintain an enormous central database and a large amount of sophisticated hardware and high performance software. The main disadvantage of this solution is they are always threatened by scalability and availability.

A reasonable solution is thus to use unstructured P2P networks which are scalable, available, low cost and easily deployed. The main challenge in unstructured P2P is in searching or locating resources.

Search is basic activity for all P2P applications. The fundamental search for unstructured P2P networks is flooding because they both have similar characteristics. Both have no knowledge regarding the network topology and distribution of files. Thus the flooding search is an attractive method for resource discovery in dynamically evolving networks. Although flooding has a large coverage growth rate in low-hops, it produces a huge amount of redundant messages in high-hops. There was no reasonable threshold point between low-hops and high-hops in flooding searches to obtain the maximum coverage growth rate and minimum number of redundant messages.

This thesis introduced a custom-built metric which we called the *critical metric*. With the *critical metric* the best threshold point for flooding and other combinations of search algorithms can be estimated.

The research proposed a hybrid algorithm which combines flooding and teeming to benefit from their advantages and limit their disadvantages. The proposed algorithm is called the QuickFlood algorithm. QuickFlood was used to improve

the performance of flooding search. QuickFlood used the arrangement $t = 2$, which means two hops are processed with flooding and rest use the teeming algorithm. The performances of QuickFlood compared with the blocking expanding ring shows a 70% reduction in redundant messages, doubled the success rate, and reduced the amount of latency by 30%.

The thesis proposed the HybridFlood search algorithm, which has combined flooding with a super peer technique to benefit from their merits and to limit their drawbacks. The algorithm uses flooding in the first step to gain a greater coverage growth rate and fewer redundant messages. In the second step it used super peers to gain low number of broadcast and redundant messages, and to achieve a high search speed. HybridFlood is applied to enhance the flooding search. The performance of HybridFlood compared with the blocking expanding ring shows an 80% reduction in redundant messages, 2.5 time increase in the success rate, and an 80% reduction in the amount of latency. The HybridFlood search algorithm is simple and did not need any powerful resource nodes. The selection of a super peer is simple and dynamic, it simply selects a neighbor with the highest degree and calls it a nosey node (super-peer). The construction of HybridFlood is scalable, simple and easy to implement in real systems.

7.2 Future Works

The future work can be summarized as follows:

- Consider the effective role of loop and cyclic paths in flooding searches and counteract them to enhance flooding, QuickFlood, and HybridFlood searches.
- Study the effective role of the replication ratio of items to enhance QuickFlood and HybridFlood searches.
- Consider the free riding malicious effects in P2P networks and overcome them

efficiently to improve QuickFlood and HybridFlood searches.

BIBLIOGRAPHY

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

[2] F. Bergmann. Napster & the music industry. *Global eManagement*, 2001.

[3] E. Meshkova, J. Riihijarvi, M. Petrova, and P. Mahonen. A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. *Computer networks*, 52(11):2097–2128, 2008.

[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, volume 31, pages 161–172, 2001.

[5] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer communication review*, 31(4):149–160, 2001.

[6] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications surveys and tutorials*, 7(2):72–93, 2005.

[7] S. Chen, Z. Zhang, S. Chen, and B. Shi. Efficient file search in non-dht peer-to-peer networks. *Computer communications*, 31(2):304–317, 2008.

[8] L.B. Oliveira, I.G. Siqueira, D.F. Macedo, A.A.F. Loureiro, H.C. Wong, and J.M. Nogueira. Evaluation of peer-to-peer network content discovery techniques over mobile ad-hoc networks. In *World of wireless mobile and multimedia networks, 2005. WoWMoM 2005. sixth IEEE international symposium*, pages 51–56. IEEE, 2005.

[9] Y. Zhu. *Enhancing Search Performance in Peer-to-peer Networks.* PhD thesis, University of Cincinnati, 2005.

[10] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *Distributed computing systems, 2002. proceedings. 22nd international conference*, pages 5–14. IEEE, 2002.

[11] Gnutella2. Available: http://www.gnutella2.com (jan, 2006). [last visited 01/19/2012], 2003.

[12] A. Gud, M. Mizuno, and D. Andresen. A scalable search algorithm for unstructured peer-to-peer networks. pages 120–126, 2008.

[13] J.F. Gantz, J. Mcarthur, and S. Minton. The expanding digital universe. *Director*, 285(6), 2007.

[14] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proceedings of the 2003 conference on applications, technologies, architectures, and protocols for computer communications*, pages 175–186. ACM, 2003.

[15] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like peer-to-peer systems scalable. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 407–418. ACM, 2003.

[16] S. Jiang, L. Guo, X. Zhang, and H. Wang. Lightflood: Minimizing redundant messages and maximizing scope of peer-to-peer search. *IEEE Transactions on parallel and distributed systems*, pages 601–614, 2007.

[17] X. Liu, Y. Liu, and L. Xiao. Improving query response delivery quality in peer-to-peer systems. *IEEE Transactions on Parallel and Distributed Systems*, pages 1335 – 1347, 2006.

[18] C. Wang and L. Xiao. An effective peer-to-peer search scheme to exploit file sharing heterogeneity. *IEEE Transactions on parallel and distributed systems*, pages 145–157, 2007.

[19] KaZaa. http://www.kazaa.com/ [last visited 01/19/2012].

[20] B. Cohen. Incentives build robustness in bittorrent. In *Workshop on economics of peer-to-Peer systems*, volume 6, pages 68–72. Citeseer, 2003.

[21] Reaz Ahmed and Raouf Boutaba. Distributed pattern matching: a key to flexible and efficient peer-to-peer search. *Selected areas in communications, IEEE Journal*, 25(1):73–83, Jan. 2007.

[22] C. Wang and L. Xiao. An effective p2p search scheme to exploit file sharing heterogeneity. *IEEE Transactions on Parallel and Distributed Systems*, pages 145–157, 2007.

[23] T. Lin, P. Lin, H. Wang, and C. Chen. Dynamic search algorithm in unstructured peer-to-peer networks. *IEEE Transactions on parallel and distributed systems*, pages 654–666, 2008.

[24] K.H. Yang, C. Wu, and J.M. Ho. Antsearch: An ant search algorithm in unstructured peer-to-peer networks. *IEICE transactions on communications*, 89(9):2300–2308, 2006.

[25] C.J. Wu, D.K. Liu, and R.H. Hwang. A location-aware peer-to-peer overlay network. *International Journal of Communication Systems*, 20(1):83–102, 2007.

[26] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and whitewashing in peer-to-peer systems. *Selected Areas in Communications, IEEE Journal on*, 24(5):1010–1019, 2006.

[27] R. Matei, A. Iamnitchi, and P. Foster. Mapping the gnutella network. *Internet computing, IEEE*, 6:50–57, 2002.

[28] F.S. Annexstein, K.A. Berman, and M.A. Jovanovic. Broadcasting in unstructured peer-to-peer overlay networks. *Theoretical computer science*, 355(1):25–36, 2006.

[29] M. Karakaya, I. Korpeoglu, and O. Ulusoy. Gnusim: A general purpose simulator for gnutella and unstructured peer-to-peer networks. *Department of computer engineering, Bilkent university, Tech. Rep*, 2005.

[30] K.W. Kwong and D.H.K. Tsang. A congestion-aware search protocol for heterogeneous peer-to-peer networks. *Supercomputing*, 36(3):265–282, 2006.

[31] D. Hughes, G. Coulson, and J. Walkerdine. Free riding on gnutella revisited: the bell tolls? *IEEE Distributed systems Online*, 6:1–12, 2005.

[32] D. Zeinalipour-Yazti, V. Kalogeraki, and D. Gunopulos. pfusion: A peer-to-peer architecture for internet-scale content-based search and retrieval. *Parallel and distributed systems, IEEE Transactions*, 18(6):804–817, june 2007.

[33] R. Steinmetz and K. Wehrle. *Peer-to-peer Systems and Applications*, volume 3485. Springer-verlag new york inc, 2005.

[34] X. Li and J. Wu. Searching techniques in peer-to-peer networks. *Handbook of theoretical and algorithmic aspects of Ad Hoc, sensor, and Peer-to-Peer networks*, pages 613–642, 2006.

[35] D. Stutzbach, R. Rejaie, and S. Sen. Characterizing unstructured overlay topologies in modern peer-to-peer file-sharing systems. *Networking, IEEE/ACM Transactions*, 16(2):267–280, 2008.

[36] M. Marzolla, M. Mordacchini, and S. Orlando. Resource discovery in a dynamic grid environment. In *Database and expert systems applications, 2005. Proceedings. sixteenth international workshop*, pages 356–360. IEEE, 2005.

[37] D. Talia and P. Trunfio. Peer-to-peer protocols and grid services for resource discovery on grids. *Advances in parallel computing*, 14:83–103, 2005.

[38] B. Yang and H. Garcia-Molina. Designing a super-peer network. *Proceedings. 19th international conference on data engineering, 2003.*, pages 49–60, 2003.

[39] A. Iamnitchi, I. Foster, and D.C. Nurmi. A peer-to-peer approach to resource location in grid environments. *International series in opertions research and management science*, pages 413–430, 2003.

[40] X. Tang, J. Xu, and W.C. Lee. Analysis of ttl-based consistency in unstructured peer-to-peer networks. *IEEE Transactions on parallel and distributed systems*, pages 1683–1694, 2008.

[41] K. Kant. An analytic model for peer-to-peer file sharing networks. In *Communications, 2003. ICC'03. IEEE international conference*, volume 3, pages 1801–1805. IEEE, 2003.

[42] H.C. Hsiao and C.P. He. A tree-based peer-to-peer network with quality guarantees. *IEEE Transactions on parallel and distributed systems*, pages 1099–1110, 2008.

[43] M. Castro, P. Druschel, A.M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *ACM SIGOPS Operating systems review*, volume 37, pages 298–313. ACM, 2003.

[44] T. Nguyen, K. Kolazhi, R. Kamath, and S. Cheung. Efficient video dissemination in structured hybrid peer-to-peer networks. In *2006 IEEE International conference on multimedia and expo*, pages 1673–1676. IEEE, 2006.

[45] M. Demirbas and H. Ferhatosmanoglu. Peer-to-peer spatial queries in sensor networks. *Proceedings third international conference on peer-to-peer computing.*, pages 32–39, 2003.

[46] N. Bisnik and A. Abouzeid. Modeling and analysis of random walk search algorithms in peer-to-peer networks. *Second international workshop on hot topics in peer-to-peer systems.*, pages 95–103, 2005.

[47] Rongmei Zhang and Y.C. Hu. Assisted peer-to-peer search with partial indexing. *Parallel and distributed systems, IEEE Transactions*, 18(8):1146–1158, aug. 2007.

[48] Gang Chen, Chor Ping Low, and Zhonghua Yang. Enhancing search performance in unstructured peer-to-peer networks based on users' common interest. *Parallel and distributed systems, IEEE Transactions*, 19:821–836, 2008.

[49] S.M. Thampi. Survey of search and replication schemes in unstructured p2p networks. *Arxiv preprint arXiv:1008.1629*, 2010.

[50] F. Otto and S. Ouyang. Improving search in unstructured peer-to-peer systems: Intelligent walks (i-walks). *Intelligent data engineering and automated learning-IDEAL 2006*, pages 1312–1319, 2006.

[51] V.V. Dimakopoulos and E. Pitoura. On the performance of flooding-based resource discovery. *IEEE Transactions on parallel and distributed systems*, pages 1242–1252, 2006.

[52] N. Chang and M. Liu. Revisiting the ttl-based controlled flooding search: Optimality and randomization. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 85–99. ACM, 2004.

[53] J. Hassan and S. Jha. Performance analysis of expanding ring search for multi-hop wireless networks. In *Vehicular technology conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, volume 5, pages 3615–3619, Sept. 2004.

[54] Kui Wu, Hong-Chuan Yang, and Fulu Li. Wsn08-3: Pessimism is mostly the best for the expanding ring search in wireless networks. In *Global telecommunications conference. GLOBECOM '06. IEEE*, pages 1–5, 27 Dec 2006.

[55] I.M. Pu and Y. Shen. Enhanced blocking expanding ring search in mobile ad-hoc networks. In *New technologies, mobility and security (NTMS), 2009 3rd international conference*, pages 1–5. IEEE, 2009.

[56] V.V. Dimakopoulos and E. Pitoura. Performance analysis of distributed search in open agent systems. pages 45–52, 2003.

[57] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing*, pages 84–95. ACM, 2002.

[58] R. Dorrigiv, A. Lopez-Ortiz, and P. Pralat. Search algorithms for unstructured peer-to-peer networks. In *Local computer networks, 2007. LCN 2007. 32nd IEEE Conference on*, pages 343–352. IEEE, 2007.

[59] I. Jawhar and J. Wu. A two-level random walk search protocol for peer-to-peer networks. In *Proc. of the 8th world multi-conference on systemics, cybernetics and informatics*, pages 160–164, 2004.

[60] S. Jin and H. Jiang. Novel approaches to efficient flooding search in peer-to-peer networks. *Computer networks*, 51(10):2818–2832, 2007.

[61] C. Tian, H. Jiang, X. Liu, W. Liu, and Y. Wang. Towards minimum traffic cost and minimum response latency: A novel dynamic query protocol in unstructured peer-to-peer networks. In *37th International conference on parallel processing*, pages 1–8. IEEE, 2008.

[62] H. Jiang and S. Jin. Exploiting dynamic querying like flooding techniques in unstructured peer-to-peer networks. In *Network protocols, 2005. ICNP 2005. 13th IEEE International Conference*, pages 10–20. IEEE, 2005.

[63] E. Adar and B.A. Huberman. Free riding on gnutella. *First Monday*, 5(10):2–13, 2000.

[64] M. Karakaya, I. Korpeoglu, and Ulusoy. Counteracting free riding in peer-to-peer networks. *Computer Networks*, 52(3):675–694, 2008.

[65] Chen Wang and Li Xiao. An effective peer-to-peer search scheme to exploit file sharing heterogeneity. *Parallel and Distributed Systems, IEEE Transactions*, 18(2):145–157, Feb. 2007.

[66] B. Yang, P. Vinograd, and H. Garcia-Molina. Evaluating guess and non-forwarding peer-to-peer search. *proceedings. 24th International conference on distributed computing systems*, pages 209 – 218, 2004.

[67] Z. Zhuang, Y. Liu, L. Xiao, and L.M. Ni. Hybrid periodical flooding in unstructured peer-to-peer networks. In *Parallel processing, international conference*, pages 171–178. IEEE, 2003.

[68] D. Tsoumakos and N. Roussopoulos. Adaptive probabilistic search for peer-to-peer networks. *Proceedings. third international conference on peer-to-peer computing*, pages 102–109, 2003.

[69] D. Tsoumakos and N. Roussopoulos. Analysis and comparison of peer-to-peer search methods. In *Proceedings of the 1st international conference on scalable information systems*, pages 25–37. ACM, 2006.

[70] D. Tsoumakos and N. Roussopoulos. Probabilistic knowledge discovery and management for peer-to-peer networks. *P2P Journal*, pages 15–20, 2003.

[71] Gnutella. http://www.zeropaid.com/software/file-sharing/gnutella/ [last visited 01/19/2012].

[72] Freenet. http://freenet.sourceforge.net [last visited 01/19/2012].

[73] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, et al. Oceanstore: An architecture for global-scale persistent storage. *ACM SIGARCH Computer architecture news*, 28(5):190–201, 2000.

[74] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: The oceanstore prototype. In *Proceedings of the 2nd USENIX conference on file and storage technologies*, pages 1–14, 2003.

[75] D. M. Geels. Data replication in oceanstore. Technical report, UCB/CSD-02-1217, EECS Department, University of California, Berkeley, December 2002.

[76] C.G. Plaxton, R. Rajaraman, and A.W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory of computing systems*, 32(3):241–280, 1999.

[77] B.Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph, and J.D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *Selected areas in communications, IEEE Journal*, 22(1):41–53, 2004.

[78] B. Zhao, Y. Duan, L. Huang, A. Joseph, and J. Kubiatowicz. Brocade: Landmark routing on overlay networks. *Peer-to-Peer Systems*, pages 34–44, 2002.

[79] M. Fetscherin. Movie piracy on peer-to-peer networks-the case of kazaa. *Telematics and informatics*, 22(number 2):57–70, 2005.

[80] J. Liang, R. Kumar, and K.W. Ross. The kazaa overlay: A measurement study. *Computer Networks*, 49(6), 2005.

[81] A. Singla and C. ROHRS. Ultrapeers: Another step towards gnutella scalability, working draft. *Limewire LLC. Dezembro de*, 2001.

[82] P. Byrne A. Olivré A. OŠConnor, C. Brady. Characterising the edonkey peer-to-peer file sharing network. Technical report, Computer Science Department, Trinity College Dublin, Ireland, 2004.

[83] eDonkey. http://heanet.dl.sourceforge.net/sourceforge/pdonkey/edonkey-protocol-0.6.2.html [last visited 01/19/2012].

[84] eMule. http://www.emule-project.net/home/perl/general.cgi?l = 1 [last visited 01/19/2012].

[85] O. Heckmann, A. Bock, A. Mauthe, and R. Steinmetz. The edonkey file-sharing network. In *Proceedings of the workshop on algorithms and protocols for efficient Peer-to-Peer applications (Informatik)*, volume 51, pages 224–228, 2004.

[86] B. Traversat, A. Arora, M. Abdelaziz, M. Duigou, C. Haywood, J.C. Hugly, E. Pouyoul, and B. Yeager. Project jxta 2.0 super-peer virtual network. *Sun microsystem white paper. Available at www. jxta. org/project/www/docs [Last visited 01/19/2012]*, 2003.

[87] Z. Xu and Y. Hu. Sbarc: A supernode based peer-to-peer file sharing system. *Proceedings. eighth IEEE international symposium on computers and communication. (ISCC 2003).*, 2:1503–1508, 2003.

[88] Y. Zhu, H. Wang, and Y. Hu. A super-peer based lookup in structured peer-to-peer systems. In *Proceedings of the 16th international conference on parallel and distributed computing systems (PDCS)*, pages 465–470. Citeseer, 2003.

[89] B. Traversat, M. Abdelaziz, and E. Pouyoul. Project jxta: A loosely-consistent dht rendezvous walker. *At http://research. sun. com/spotlight/misc/jxta-dht. pdf [Last visited 01/19/2012]*, 2003.

[90] A.O. Stauffer and V.C. Barbosa. Probabilistic heuristics for disseminating information in networks. *IEEE/ACM Transactions on networking (TON)*, 15(2):425–435, 2007.

[91] N.B. Chang and M. Liu. Controlled flooding search in a large network. *Networking, IEEE/ACM Transactions*, 15(number 2):436–449, 2007.

[92] S. Jiang and X. Zhang. Floodtrail: An efficient file search technique in unstructured peer-to-peer systems. In *Global telecommunications conference, 2003. GLOBECOM'03. IEEE*, volume 5, pages 2891–2895. IEEE, 2003.

[93] D. Zeinalipour-Yazti, V. Kalogeraki, and D. Gunopulos. Information retrieval techniques for peer-to-peer networks. *Computing in science and engineering*, pages 20–26, 2004.

[94] Y.J. Pyun and D.S. Reeves. Constructing a balanced,(log (n)/loglog (n))-diameter super-peer topology for scalable p2p systems. *Proceedings of the 4th international conference on peer-to-peer computing*, pages 210–218, 2004.

[95] C.C. Chou. *Techniques for Peer-to-peer Content Distribution Over Mobile Ad-hoc Networks*. PhD thesis, University of southern california, 2007.

[96] S.K. Kim, K.J. Lee, and S.B. Yang. An enhanced super-peer system considering mobility and energy in mobile environments. In *Wireless and pervasive computing (ISWPC), 2011 6th International Symposium*, pages 1–5. IEEE, 2011.

[97] C. Mastroianni, P. Cozza, D. Talia, I. Kelley, and I. Taylor. A scalable super-peer approach for public scientific computation. *Future generation computer systems*, 25(3):213–223, 2009.

[98] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Loser. Super-peer-based routing strategies for rdf-based peer-to-peer networks. *Web Semantics: Science, services and agents on the world wide web*, 1(2):177–186, 2004.

[99] G.E. Moore et al. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.

[100] Z. Jia, X. Tang, J. You, and M. Li. Search in unstructured peer-to-peer networks. *Web Information Systems–WISE 2004*, pages 694–705, 2004.

[101] Clip2. Distributed search solutions available: http://public.yahoo.com/ lguo/download/gnutella-trace/xmlfiles.tar.gz and http://public.yahoo.com/ lguo/download/gnutella-trace/sample-query.dat [last visited 01/19/2012].

[102] AH Farooqi, SB Kazmi, and FA Khan. Performance analysis of peer-to-peer overlay architectures for mobile ad hoc networks. In *Emerging technologies, 2009. ICET 2009. International conference*, pages 471–475. IEEE, 2009.

[103] L. Guo, S. Jiang, L. Xiao, and X. Zhang. Fast and low-cost search schemes by exploiting localities in peer-to-peer networks. *Parallel and distributed computing*, 65(6):729–742, 2005.

[104] K. Kutzner and T. Fuhrmann. Measuring large overlay networks-the overnet example. In *Kommunikation in verteilten systemen (KiVS)*, pages 193–204. Springer, 2005.

[105] K.P. Gummadi, R.J. Dunn, S. Saroiu, S.D. Gribble, H.M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proceedings of the nineteenth ACM symposium on operating systems principles*, pages 314–329. ACM, 2003.

[106] K. Aberer and M. Hauswirth. An overview on peer-to-peer information systems. In *Workshop on distributed data and structures WDAS-2002*, pages 1–14, 2002.

[107] Z. Zhu, P. Kalnis, and S. Bakiras. Dcmp: A distributed cycle minimization protocol for peer-to-peer networks. *IEEE Transactions on parallel and distributed systems*, pages 363–377, 2007.

APPENDIX I

Samples of GraphT1 Traces

<graph>
<node id="0" ip="217.81.26.19" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0"
ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="1" ip="12.240.0.222" hostname="null" port="6346"
pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/>
<node id="2" ip="4.42.92.183" hostname="crtntx1-ar8-092-183.crtntx1.dsl.gtei.net" port="6346" pingtime="18140" speed="0"
numfilesshared="8828" numkbytesshared="34163807" ttl="0" queryrate="6.4333334" firewalled="no" responsive="yes"/> <node
id="3" ip="65.4.98.8" hostname="c1266262-a.lwstn1.id.home.com" port="6346" pingtime="1688" speed="0" numfilesshared="6613"
numkbytesshared="16466309" ttl="0" queryrate="3.2666667" firewalled="no" responsive="yes"/> <node id="4" ip="209.86.176.193"
hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"
firewalled="yes" responsive="no"/> <node id="5" ip="63.228.146.99" hostname="phnxdslgw6poolC99.phnx.uswest.net" port="6346"
pingtime="2312" speed="0" numfilesshared="5687" numkbytesshared="12839008" ttl="0" queryrate="7.133333" firewalled="no"
responsive="yes"/> <node id="6" ip="212.76.43.117" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0"
numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="7" ip="24.176.98.46" hostname="c1569638-
b.bllngs1.mt.home.com" port="6346" pingtime="23156" speed="0" numfilesshared="5542" numkbytesshared="18743010" ttl="0"
queryrate="10.6" firewalled="no" responsive="yes"/><node id="8" ip="213.51.181.116" hostname="null" port="6346" ping-
time="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/>
<node id="9" ip="172.153.179.249" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0"
ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/><node id="10" ip="200.185.92.57" hostname="null" port="6346"
pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/>
<node id="11" ip="213.112.188.235" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numk-
bytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="12" ip="213.65.118.125" host-
name="h125n2fls22o906.telia.com" port="6346" pingtime="1969" speed="0" numfilesshared="3835" numkbytesshared="31046404"
ttl="0" queryrate="2.5" firewalled="no" responsive="yes"/> <node id="13" ip="64.34.249.37" hostname="null" port="9550"
pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/>
<node id="14" ip="4.60.96.171" hostname="evrtwa1-ar7-4-60-096-171.vz.dsl.gtei.net" port="6346" pingtime="13828" speed="0"
numfilesshared="3568" numkbytesshared="12450012" ttl="0" queryrate="3.1333334" firewalled="no" responsive="yes"/> <node
id="15" ip="138.88.68.238" hostname="adsl-138-88-68-238.dc.adsl.bellatlantic.net" port="6346" pingtime="0" speed="0" num-
filesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/><node id="16" ip="24.21.182.120"
hostname="cx833780-a.pnscla1.fl.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0"
ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="17" ip="202.161.98.159" hostname="null" port="6346"
pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/>
<node id="18" ip="24.146.2.52" hostname="pc52.pool2.e-velocity.net" port="6346" pingtime="485" speed="0" numfilesshared="3371"
numkbytesshared="7572545" ttl="0" queryrate="10.8" firewalled="no" responsive="yes"/> <node id="19" ip="24.234.166.16"
hostname="cm016.166.234.24.lvcm.com" port="5500" pingtime="8734" speed="0" numfilesshared="3332" numkbytesshared="14162641"

ttl="0" queryrate="2.2333333" firewalled="no" responsive="yes"/> <node id="20" ip="212.144.158.9" hostname="nbgdi6-212-144-158-009.arcor-ip.net" port="6346" pingtime="23750" speed="0" numfilesshared="3215" numkbytesshared="16958301" ttl="0" queryrate="4.0" firewalled="no" responsive="yes"/> <node id="21" ip="217.80.188.109" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="22" ip="24.2.35.24" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="23" ip="62.5.10.109" hostname="bb-62-5-10-109.bb.tninet.se" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="24" ip="207.30.207.92" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="25" ip="217.80.239.209" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="26" ip="24.6.27.27" hostname="cx370006-c.okc1.ok.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="27" ip="62.31.73.20" hostname="pc-62-31-73-20-ed.blueyonder.co.uk" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="28" ip="24.154.5.54" hostname="acs-24-154-5-54.zoominternet.net" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="29" ip="65.29.142.179" hostname="mke-65-29-142-179.wi.rr.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="30" ip="4.60.26.28" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="31" ip="202.67.90.124" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="32" ip="66.26.6.168" hostname="ilm26-6-168.ec.rr.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="33" ip="193.251.57.220" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="34" ip="24.67.68.58" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="35" ip="194.25.81.132" hostname="adialin65.asamnet.de" port="6346" pingtime="2781" speed="0" numfilesshared="2278" numkbytesshared="9550169" ttl="0" queryrate="0.33333334" firewalled="no" responsive="yes"/> <node id="36" ip="66.66.198.97" hostname="alb-66-66-198-97.nycap.rr.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="37" ip="4.35.68.179" hostname="lsanca1-ar8-068-179.biz.dsl.gtei.net" port="6346" pingtime="5891" speed="0" numfilesshared="2140" numkbytesshared="8174677" ttl="0" queryrate="4.0666666" firewalled="no" responsive="yes"/> <node id="38" ip="63.205.14.143" hostname="adsl-63-205-14-143.dsl.scrm01.pacbell.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="39" ip="63.164.226.18" hostname="63.164.226.18" port="6346" pingtime="0"

speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="40" ip="165.121.83.91" hostname="user-2inikqr.dialup.mindspring.com" port="6346" pingtime="17578" speed="0" numfilesshared="2018" numkbytesshared="2097151" ttl="0" queryrate="0.93333334" firewalled="no" responsive="yes"/> <node id="41" ip="66.43.231.158" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="42" ip="66.1.82.237" hostname="cpe-66-1-82-237.az.sprintbbd.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="9.166667" firewalled="no" responsive="yes"/> <node id="43" ip="4.40.132.47" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="44" ip="24.40.20.10" hostname="cn200818-a.wall1.pa.home.com" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="45" ip="24.9.183.76" hostname="c1017010-a.salem1.or.home.com" port="6348" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="46" ip="172.184.222.181" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="47" ip="62.211.34.68" hostname="62.211.34.68" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="48" ip="24.11.231.39" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="49" ip="66.41.70.19" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="50" ip="212.229.169.10" hostname="scope.demon.co.uk" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="6.5" firewalled="no" responsive="yes"/> <node id="51" ip="65.80.120.227" hostname="adsl-80-120-227.dab.bellsouth.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="52" ip="65.0.195.183" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="53" ip="65.5.229.89" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="54" ip="24.108.147.67" hostname="24-108-147-67.ivideon.com" port="6346" pingtime="3735" speed="0" numfilesshared="1720" numkbytesshared="5048857" ttl="0" queryrate="7.3333335" firewalled="no" responsive="yes"/> <node id="55" ip="172.162.152.146" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="56" ip="63.87.133.21" hostname="63.87.133.21" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="57" ip="24.112.71.207" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="58" ip="24.42.46.4" hostname="cr594688-a.rchrd1.on.wave.home.com" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="59" ip="165.121.198.34" hostname="user-2injhh2.dialup.mindspring.com" port="6346" pingtime="0" speed="0"

numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="60" ip="24.17.126.168" hostname="cx449755-a.crsv1.fl.home.com" port="6346" pingtime="6734" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="8.166667" firewalled="no" responsive="yes"/> <node id="61" ip="24.42.1.168" hostname="cr704458-a.wlfdle1.on.wave.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="62" ip="211.132.71.176" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="63" ip="63.94.223.29" hostname="host44-29.prestige.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="64" ip="212.31.34.8" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="65" ip="24.234.8.27" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="66" ip="24.22.133.191" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="67" ip="24.92.164.232" hostname="dt041ne8.tampabay.rr.com" port="6346" pingtime="609" speed="0" numfilesshared="1427" numkbytesshared="2097151" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="68" ip="61.114.14.176" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="69" ip="63.230.49.170" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="70" ip="217.80.214.163" hostname="pD950D6A3.dip.t-dialin.net" port="6346" pingtime="2687" speed="0" numfilesshared="1362" numkbytesshared="6397493" ttl="0" queryrate="5.8333335" firewalled="no" responsive="yes"/> <node id="71" ip="172.161.99.208" hostname="ACA163D0.ipt.aol.com" port="6346" pingtime="26078" speed="0" numfilesshared="1349" numkbytesshared="4195864" ttl="0" queryrate="2.7333333" firewalled="no" responsive="yes"/> <node id="72" ip="193.253.32.154" hostname="ATuileries-103-1-2-154.abo.wanadoo.fr" port="6346" pingtime="4984" speed="0" numfilesshared="1309" numkbytesshared="4793927" ttl="0" queryrate="1.7666667" firewalled="no" responsive="yes"/> <node id="73" ip="217.82.188.211" hostname="pD952BCD3.dip.t-dialin.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="3.5666666" firewalled="no" responsive="yes"/> <node id="74" ip="12.37.155.15" hostname="15.fwsgrp12.als.att.net" port="6347" pingtime="10171" speed="0" numfilesshared="1300" numkbytesshared="2097151" ttl="0" queryrate="4.733333" firewalled="no" responsive="yes"/> <node id="75" ip="199.224.95.183" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="76" ip="213.200.141.237" hostname="c213.200.141.237.cm-upc.chello.se" port="6346" pingtime="6515" speed="0" numfilesshared="1270" numkbytesshared="5467287" ttl="0" queryrate="9.1" firewalled="no" responsive="yes"/> <node id="77" ip="213.104.75.115" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="78" ip="213.20.48.225" hostname="213.20.48.225" port="6346" pingtime="22281" speed="0" numfilesshared="1265" numkbytesshared="6303569" ttl="0" queryrate="10.633333" firewalled="no" responsive="yes"/> <node id="79" ip="213.123.167.154"

hostname="host213-123-167-154.btopenworld.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="80" ip="64.161.29.133" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="81" ip="207.172.166.163" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="82" ip="24.182.145.53" hostname="24.182.145.53" port="6346" pingtime="5657" speed="0" numfilesshared="1251" numkbytesshared="7449529" ttl="0" queryrate="4.3333335" firewalled="no" responsive="yes"/> <node id="83" ip="206.206.164.173" hostname="dialup42.pm3.caverns.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="2.0" firewalled="no" responsive="yes"/> <node id="84" ip="172.132.99.247" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="85" ip="62.40.128.150" hostname="h062040128150.stp.cm.kabsi.at" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="86" ip="65.28.133.56" hostname="okc-65-28-133-56.mmcable.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="87" ip="208.61.154.178" hostname="adsl-61-154-178.atl.bellsouth.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="88" ip="212.227.15.108" hostname="ccxxxiv.yapay.inka.de" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="89" ip="24.88.221.253" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="90" ip="24.132.18.145" hostname="node11291.a2000.nl" port="5634" pingtime="4937" speed="0" numfilesshared="1193" numkbytesshared="5956340" ttl="0" queryrate="2.2666667" firewalled="no" responsive="yes"/> <node id="91" ip="172.144.15.229" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="92" ip="138.23.62.123" hostname="resnet62-123.ucr.edu" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="93" ip="24.48.220.7" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="94" ip="203.164.134.25" hostname="co3040170-a.sunsh1.vic.optushome.com.au" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="95" ip="138.23.62.123" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="96" ip="208.63.220.138" hostname="adsl-63-220-138.asm.bellsouth.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="97" ip="203.164.27.199" hostname="co3013348-a.kelvn1.qld.optushome.com.au" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="98" ip="62.137.129.241" hostname="vdnltd.fsnet.co.uk" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="99" ip="64.89.105.37"

hostname="null" port="6348" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="100" ip="65.13.148.12" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="101" ip="172.149.250.80" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="102" ip="217.80.202.11" hostname="pD950CA0B.dip.t-dialin.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="103" ip="24.91.222.66" hostname="h0010b5732cb5.ne.mediaone.net" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="104" ip="24.150.73.241" hostname="d150-73-241.home.cgocable.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="105" ip="206.45.122.10" hostname="brnd-122-10-res.mts.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="106" ip="217.80.145.251" hostname="pD95091FB.dip.t-dialin.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="5.5" firewalled="no" responsive="yes"/> <node id="107" ip="217.82.166.93" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="108" ip="213.200.145.204" hostname="c213.200.145.204.cm-upc.chello.se" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="5.866667" firewalled="no" responsive="yes"/> <node id="109" ip="216.26.45.192" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="110" ip="24.40.3.36" hostname="cn974131-a.bens1.pa.home.com" port="7908" pingtime="5094" speed="0" numfilesshared="1079" numkbytesshared="3674377" ttl="0" queryrate="1.2666667" firewalled="no" responsive="yes"/> <node id="111" ip="66.25.22.238" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="112" ip="64.217.144.218" hostname="adsl-64-217-144-218.dsl.eulstx.swbell.net" port="6346" pingtime="2937" speed="0" numfilesshared="1079" numkbytesshared="4478239" ttl="0" queryrate="10.4" firewalled="no" responsive="yes"/> <node id="113" ip="62.31.205.205" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="114" ip="217.80.130.93" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="115" ip="128.61.106.202" hostname="tke10.eastnet.gatech.edu" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="116" ip="65.0.55.115" hostname="c1384394-a.potlnd1.or.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="117" ip="64.108.212.3" hostname="adsl-pool42-3.chicago.il.ameritech.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="118" ip="65.3.148.191" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"

firewalled="yes" responsive="no"/> <node id="119" ip="66.108.18.34" hostname="66.108.18.34" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="120" ip="217.128.89.60" hostname="null" port="10307" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="121" ip="213.93.116.3" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="122" ip="172.129.151.17" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="123" ip="149.225.124.45" hostname="pec-124-45.tnt8.m2.uunet.de" port="6346" pingtime="8797" speed="0" numfilesshared="1004" numkbytesshared="2097151" ttl="0" queryrate="1.2225589" firewalled="no" responsive="yes"/> <node id="124" ip="66.31.55.104" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="125" ip="172.158.121.202" hostname="AC9E79CA.ipt.aol.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="126" ip="24.183.218.74" hostname="cj494898-a.indpdnce1.mo.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="127" ip="203.164.135.2" hostname="co3036961-a.sunsh1.vic.optushome.com.au" port="6346" pingtime="1891" speed="0" numfilesshared="987" numkbytesshared="10262665" ttl="0" queryrate="1.3" firewalled="no" responsive="yes"/> <node id="128" ip="172.185.189.189" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="129" ip="66.20.126.4" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="130" ip="4.40.16.45" hostname="washdc3-ar3-016-045.washdc3.dsl.gtei.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="7.0666666" firewalled="no" responsive="yes"/> <node id="131" ip="62.252.59.42" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="132" ip="66.1.110.214" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="133" ip="209.179.222.184" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="134" ip="62.178.73.17" hostname="chello062178073017.25.11.vie.surfer.at" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="135" ip="62.252.197.29" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="136" ip="24.43.118.81" hostname="cr175027-a.slnt1.on.wave.home.com" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="137" ip="24.6.27.103" hostname="cx240951-f.okc1.ok.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="138" ip="213.47.75.70" hostname="chello213047075070.14.vie.surfer.at" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0"

ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="139" ip="62.224.246.197" hostname="p3EE0F6C5.dip.t-dialin.net" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="140" ip="213.64.131.3" hostname="h3n1fls21o913.telia.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="141" ip="66.43.211.71" hostname="huxdsl-71.pcpartner.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="142" ip="24.28.4.74" hostname="24284hfc74.tampabay.rr.com" port="6346" pingtime="485" speed="0" numfilesshared="932" numkbytesshared="6769312" ttl="0" queryrate="6.1" firewalled="no" responsive="yes"/> <node id="143" ip="200.241.84.202" hostname="pppD29.elo.com.br" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="6.296642" firewalled="no" responsive="yes"/> <node id="144" ip="65.2.81.198" hostname="cx687197-a.hmpt1.va.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="145" ip="217.81.254.60" hostname="pD951FE3C.dip.t-dialin.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="146" ip="128.253.4.100" hostname="cmm33.resnet.cornell.edu" port="5634" pingtime="2468" speed="0" numfilesshared="909" numkbytesshared="3234216" ttl="0" queryrate="0.46666667" firewalled="no" responsive="yes"/> <node id="147" ip="208.61.132.102" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="148" ip="213.51.38.11" hostname="213.51.38.11" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.76666665" firewalled="no" responsive="no"/> <node id="149" ip="24.10.157.55" hostname="cy15191-b.sanmon1.ca.home.com" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="150" ip="62.2.113.66" hostname="dclient113-66.hispeed.ch" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="151" ip="24.91.94.184" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="152" ip="65.28.152.97" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="153" ip="172.176.76.165" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="154" ip="24.13.140.192" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="155" ip="172.177.67.239" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="156" ip="62.122.22.131" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="157" ip="217.80.55.136" hostname="null" port="6355" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="158" ip="217.82.95.170" hostname="pD9525FAA.dip.t-dialin.net" port="6346" pingtime="0" speed="0" numfi-

lesshared="0" numkbytesshared="0" ttl="0" queryrate="4.9666667" firewalled="no" responsive="yes"/> <node id="159" ip="217.85.39.239" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="160" ip="64.229.14.162" hostname="HSE-Kitchener-ppp194829.sympatico.ca" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="161" ip="24.178.41.172" hostname="cc214784-a.mcmb1.mi.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="162" ip="213.69.89.24" hostname="213.69.89.24" port="6346" pingtime="5484" speed="0" numfilesshared="826" numkbytesshared="3892099" ttl="0" queryrate="4.4" firewalled="no" responsive="yes"/> <node id="163" ip="213.10.12.117" hostname="ipd50a0c75.speed.planet.nl" port="6346" pingtime="2938" speed="0" numfilesshared="809" numkbytesshared="3244948" ttl="0" queryrate="0.5996802" firewalled="no" responsive="yes"/> <node id="164" ip="131.111.235.179" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="165" ip="217.84.116.9" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="166" ip="65.6.134.174" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="167" ip="203.198.123.27" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="168" ip="213.47.228.197" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="169" ip="217.5.80.254" hostname="pD90550FE.dip.t-dialin.net" port="6346" pingtime="4797" speed="0" numfilesshared="808" numkbytesshared="8282497" ttl="0" queryrate="6.9333334" firewalled="no" responsive="yes"/> <node id="170" ip="203.109.201.234" hostname="203-109-201-234.nzl.ihugultra.co.nz" port="6346" pingtime="3407" speed="0" numfilesshared="793" numkbytesshared="5025049" ttl="0" queryrate="4.9333334" firewalled="no" responsive="yes"/> <node id="171" ip="62.2.182.178" hostname="dclient182-178.hispeed.ch" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="172" ip="62.156.63.138" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="173" ip="147.86.64.83" hostname="dbg3350n.dbg.fh-aargau.ch" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="174" ip="217.120.39.141" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="175" ip="195.34.157.85" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="176" ip="212.181.55.21" hostname="as1-3-4.ens.s.bonet.se" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="177" ip="24.167.140.196" hostname="we-24-167-140-196.we.mediaone.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/>

<node id="178" ip="65.13.144.232" hostname="c620263-a.alntn1.tx.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="6.8" firewalled="no" responsive="yes"/> <node id="179" ip="213.120.96.163" hostname="host213-120-96-163.btopenworld.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="180" ip="63.228.47.173" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="181" ip="24.147.204.11" hostname="null" port="6651" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="182" ip="66.56.20.215" hostname="rr-56-20-215.atl.mediaone.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="183" ip="24.51.23.125" hostname="ny-lackawanna6a-381.buf.adelphia.net" port="6346" pingtime="7078" speed="0" numfilesshared="744" numkbytesshared="5137898" ttl="0" queryrate="4.5666666" firewalled="no" responsive="yes"/> <node id="184" ip="64.229.102.24" hostname="HSE-Toronto-ppp183261.sympatico.ca" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="185" ip="203.101.104.154" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="186" ip="213.75.77.230" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="187" ip="212.204.154.5" hostname="cp20853-a.tilbu1.nb.nl.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="188" ip="212.58.180.223" hostname="qn-212-58-180-223.quicknet.nl" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.8" firewalled="no" responsive="no"/> <node id="189" ip="24.70.174.144" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="190" ip="63.42.108.50" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="191" ip="213.237.157.109" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="192" ip="172.135.232.101" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="193" ip="64.231.96.90" hostname="HSE-Toronto-ppp311444.sympatico.ca" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="194" ip="172.157.151.49" hostname="AC9D9731.ipt.aol.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="195" ip="213.51.202.164" hostname="cc75838-a.ensch1.ov.nl.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="196" ip="24.102.73.143" hostname="cr308554-a.etob1.on.wave.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="197" ip="24.19.196.92" hostname="c85746-a.sttln1.wa.home.com" port="6346" pingtime="6578" speed="0" num-

filesshared="695" numkbytesshared="2026700" ttl="0" queryrate="4.0" firewalled="no" responsive="yes"/> <node id="198" ip="217.81.110.9" hostname="pD9516E09.dip.t-dialin.net" port="6700" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="199" ip="212.186.14.44" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/>

Samples of GraphT1 Connections

<edge startnode="42" endnode="9658"/> <edge startnode="63" endnode="9659"/>
<edge startnode="63" endnode="9660"/> <edge startnode="63" endnode="9661"/>
<edge startnode="63" endnode="9662"/> <edge startnode="63" endnode="1448"/>
<edge startnode="63" endnode="9663"/> <edge startnode="63" endnode="9664"/>
<edge startnode="63" endnode="9665"/> <edge startnode="63" endnode="2974"/>
<edge startnode="63" endnode="9666"/> <edge startnode="28" endnode="5806"/>
<edge startnode="28" endnode="9667"/> <edge startnode="28" endnode="9668"/>
<edge startnode="28" endnode="9669"/> <edge startnode="28" endnode="9670"/>
<edge startnode="28" endnode="3552"/> <edge startnode="28" endnode="9671"/>
<edge startnode="28" endnode="9672"/> <edge startnode="28" endnode="9673"/>
<edge startnode="28" endnode="9674"/> <edge startnode="96" endnode="9675"/>
<edge startnode="96" endnode="4095"/> <edge startnode="96" endnode="9676"/>
<edge startnode="96" endnode="9677"/> <edge startnode="96" endnode="7629"/>
<edge startnode="96" endnode="9678"/> <edge startnode="96" endnode="9679"/>
<edge startnode="96" endnode="2896"/> <edge startnode="96" endnode="9680"/>
<edge startnode="96" endnode="9681"/> <edge startnode="27" endnode="9682"/>
<edge startnode="27" endnode="9683"/> <edge startnode="27" endnode="3607"/>
<edge startnode="27" endnode="9684"/> <edge startnode="27" endnode="9685"/>
<edge startnode="27" endnode="9686"/> <edge startnode="27" endnode="9687"/>
<edge startnode="27" endnode="4097"/> <edge startnode="27" endnode="4483"/>
<edge startnode="27" endnode="9688"/> <edge startnode="51" endnode="9689"/>
<edge startnode="51" endnode="9690"/> <edge startnode="51" endnode="9691"/>
<edge startnode="51" endnode="9692"/> <edge startnode="51" endnode="9693"/>
<edge startnode="51" endnode="9694"/> <edge startnode="51" endnode="9695"/>
<edge startnode="51" endnode="9696"/> <edge startnode="51" endnode="9697"/>
<edge startnode="51" endnode="4308"/> <edge startnode="103" endnode="2208"/>
<edge startnode="103" endnode="9698"/> <edge startnode="103" endnode="9699"/>
<edge startnode="103" endnode="9700"/> <edge startnode="103" endnode="9701"/>
<edge startnode="103" endnode="864"/> <edge startnode="103" endnode="9702"/>
<edge startnode="103" endnode="9703"/> <edge startnode="103" endnode="4346"/>
<edge startnode="103" endnode="9704"/> <edge startnode="90" endnode="9705"/>
<edge startnode="90" endnode="9706"/> <edge startnode="90" endnode="9707"/>
<edge startnode="92" endnode="9708"/> <edge startnode="92" endnode="9709"/>
<edge startnode="92" endnode="9710"/> <edge startnode="92" endnode="9711"/>
<edge startnode="92" endnode="9712"/> <edge startnode="92" endnode="9713"/>
<edge startnode="92" endnode="9714"/> <edge startnode="92" endnode="9715"/>
<edge startnode="92" endnode="9716"/> <edge startnode="92" endnode="9717"/>
<edge startnode="61" endnode="9718"/> <edge startnode="61" endnode="9719"/>
<edge startnode="61" endnode="9720"/> <edge startnode="61" endnode="9721"/>
<edge startnode="61" endnode="9722"/> <edge startnode="61" endnode="9723"/>
<edge startnode="61" endnode="9724"/> <edge startnode="61" endnode="9725"/>
<edge startnode="61" endnode="9726"/> <edge startnode="58" endnode="9727"/>
<edge startnode="44" endnode="9729"/> <edge startnode="61" endnode="9728"/>
<edge startnode="58" endnode="9730"/> <edge startnode="44" endnode="9731"/>
<edge startnode="58" endnode="9732"/> <edge startnode="44" endnode="9733"/>
<edge startnode="58" endnode="9734"/> <edge startnode="44" endnode="9735"/>
<edge startnode="44" endnode="9736"/> <edge startnode="58" endnode="9737"/>
<edge startnode="58" endnode="9739"/> <edge startnode="44" endnode="9738"/>

<edge startnode="58" endnode="9740"/> <edge startnode="44" endnode="9741"/>
<edge startnode="44" endnode="5520"/> <edge startnode="58" endnode="9742"/>
<edge startnode="44" endnode="9743"/> <edge startnode="58" endnode="9744"/>
<edge startnode="58" endnode="1521"/> <edge startnode="44" endnode="9745"/>
<edge startnode="73" endnode="9658"/> <edge startnode="23" endnode="9746"/>
<edge startnode="23" endnode="7272"/> <edge startnode="23" endnode="4922"/>
<edge startnode="23" endnode="7763"/> <edge startnode="23" endnode="9747"/>
<edge startnode="23" endnode="9748"/> <edge startnode="23" endnode="9749"/>
<edge startnode="23" endnode="8984"/> <edge startnode="23" endnode="9315"/>
<edge startnode="23" endnode="9295"/> <edge startnode="59" endnode="9750"/>
<edge startnode="59" endnode="9751"/> <edge startnode="59" endnode="9752"/>
<edge startnode="59" endnode="9753"/> <edge startnode="59" endnode="9666"/>
<edge startnode="59" endnode="9754"/> <edge startnode="59" endnode="9755"/>
<edge startnode="59" endnode="9756"/> <edge startnode="59" endnode="9757"/>
<edge startnode="59" endnode="9758"/> <edge startnode="56" endnode="9759"/>
<edge startnode="56" endnode="9760"/> <edge startnode="56" endnode="9761"/>
<edge startnode="56" endnode="9762"/> <edge startnode="56" endnode="9763"/>
<edge startnode="56" endnode="9764"/> <edge startnode="56" endnode="9765"/>
<edge startnode="56" endnode="9766"/> <edge startnode="56" endnode="9767"/>
<edge startnode="56" endnode="9768"/> <edge startnode="115" endnode="9769"/>
<edge startnode="115" endnode="4295"/> <edge startnode="115" endnode="9770"/>
<edge startnode="115" endnode="9771"/> <edge startnode="115" endnode="3869"/>
<edge startnode="115" endnode="3373"/> <edge startnode="115" endnode="9772"/>
<edge startnode="115" endnode="9773"/> <edge startnode="115" endnode="9774"/>
<edge startnode="115" endnode="9775"/> <edge startnode="105" endnode="9776"/>
<edge startnode="105" endnode="9777"/> <edge startnode="105" endnode="9778"/>
<edge startnode="105" endnode="9779"/> <edge startnode="105" endnode="9780"/>
<edge startnode="105" endnode="9781"/> <edge startnode="105" endnode="7515"/>
<edge startnode="105" endnode="9782"/> <edge startnode="105" endnode="9783"/>
<edge startnode="105" endnode="9784"/> <edge startnode="97" endnode="9785"/>
<edge startnode="97" endnode="9786"/> <edge startnode="97" endnode="9787"/>
<edge startnode="97" endnode="9788"/> <edge startnode="97" endnode="9789"/>
<edge startnode="97" endnode="9790"/> <edge startnode="97" endnode="9791"/>
<edge startnode="97" endnode="9792"/> <edge startnode="97" endnode="9793"/>
<edge startnode="97" endnode="9794"/> <edge startnode="146" endnode="633"/>
<edge startnode="45" endnode="9795"/> <edge startnode="45" endnode="9796"/>
<edge startnode="45" endnode="9797"/> <edge startnode="45" endnode="9798"/>
<edge startnode="45" endnode="9799"/> <edge startnode="45" endnode="9800"/>
<edge startnode="45" endnode="9801"/> <edge startnode="45" endnode="9802"/>
<edge startnode="45" endnode="9803"/> <edge startnode="45" endnode="9804"/>
<edge startnode="108" endnode="9658"/> <edge startnode="146" endnode="9805"/>
<edge startnode="143" endnode="9658"/> <edge startnode="150" endnode="9806"/>
<edge startnode="150" endnode="6803"/> <edge startnode="150" endnode="9807"/>
<edge startnode="150" endnode="9808"/> <edge startnode="150" endnode="3660"/>
<edge startnode="150" endnode="5184"/> <edge startnode="150" endnode="9809"/>
<edge startnode="150" endnode="9810"/> <edge startnode="150" endnode="9811"/>
<edge startnode="150" endnode="9812"/> <edge startnode="83" endnode="9658"/>
<edge startnode="136" endnode="9813"/> <edge startnode="136" endnode="9814"/>
<edge startnode="136" endnode="9815"/> <edge startnode="136" endnode="9816"/>
<edge startnode="136" endnode="9817"/> <edge startnode="136" endnode="9818"/>

<edge startnode="136" endnode="9819"/> <edge startnode="136" endnode="9820"/>
<edge startnode="136" endnode="9821"/> <edge startnode="136" endnode="3290"/>

APPENDIX II

Samples of GraphT2 Traces

<graph>
<node id="0" ip="63.147.108.191" hostname="cm191.cablemodem.nebi.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="1" ip="62.0.166.175" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="2" ip="202.248.109.84" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="3" ip="131.156.156.120" hostname="131.156.156.120" port="6346" pingtime="1313" speed="0" numfilesshared="17196" numkbytesshared="4293758434" ttl="0" queryrate="4.3333335" firewalled="no" responsive="yes"/> <node id="4" ip="152.7.23.109" hostname="Brag-02-109.rh.ncsu.edu" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="5" ip="195.162.174.225" hostname="195.162.174.225" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/>
<node id="6" ip="62.179.4.191" hostname="chello062179004191.chello.pl" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="7" ip="217.0.73.89" hostname="pD9004959.dip.t-dialin.net" port="4346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="8" ip="24.164.176.104" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="9" ip="203.164.78.147" hostname="co3021787-a.belrs1.nsw.optushome.com.au" port="5634" pingtime="891" speed="0" numfilesshared="7589" numkbytesshared="28347261" ttl="0" queryrate="0.06666667" firewalled="no" responsive="yes"/> <node id="10" ip="24.219.0.73" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="11" ip="206.25.218.153" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="12" ip="24.3.49.195" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="13" ip="194.182.82.134" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="14" ip="172.151.123.188" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="15" ip="62.122.98.99" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="16" ip="128.171.254.220" hostname="resnet4-220.housing.hawaii.edu" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="17" ip="213.10.19.32" hostname="ipd50a1320.speed.planet.nl" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="3.9" firewalled="no" responsive="yes"/> <node id="18" ip="62.224.243.240" hostname="p3EE0F3F0.dip.t-dialin.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="4.8333335" firewalled="no" responsive="yes"/>
<node id="19" ip="24.48.235.118" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0"

ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="20" ip="195.68.30.22" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="21" ip="129.177.102.200" hostname="129.177.102.200" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="22" ip="63.25.127.88" hostname="1Cust88.tnt4.lancaster.ca.da.uu.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="23" ip="207.172.150.45" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="24" ip="213.123.121.190" hostname="host213-123-121-190.btopenworld.com" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="25" ip="212.171.250.39" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="26" ip="200.185.92.199" hostname="200.185.92.199" port="6346" pingtime="2078" speed="0" numfilesshared="3442" numkbytesshared="19264479" ttl="0" queryrate="3.4666667" firewalled="no" responsive="yes"/> <node id="27" ip="128.104.112.218" hostname="c-112218.cmdm.wisc.edu" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="28" ip="24.95.18.28" hostname="dt052n1c.maine.rr.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="29" ip="195.252.201.138" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="30" ip="64.231.5.240" hostname="HSE-Toronto-ppp285685.sympatico.ca" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="31" ip="172.140.252.4" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="32" ip="4.41.65.118" hostname="chcgil2-ar3-065-118.dsl.gtei.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="33" ip="130.64.135.142" hostname="resnet135-142.medford.tufts.edu" port="6346" pingtime="422" speed="0" numfilesshared="2547" numkbytesshared="11364762" ttl="0" queryrate="7.866667" firewalled="no" responsive="yes"/> <node id="34" ip="151.199.153.94" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="35" ip="217.0.237.126" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="36" ip="63.39.49.201" hostname="1Cust201.tnt1.auburn.wa.da.uu.net" port="6346" pingtime="13203" speed="0" numfilesshared="2466" numkbytesshared="12684170" ttl="0" queryrate="1.1666666" firewalled="no" responsive="yes"/> <node id="37" ip="24.11.200.160" hostname="cc865735-a.bnapk1.occa.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="38" ip="24.1.66.180" hostname="c1576260-a.wntck1.sfba.home.com" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="39" ip="212.84.222.161" hostname="drogenberatung.uni-neukoelln.de" port="5634" pingtime="656" speed="0"

numfilesshared="2303" numkbytesshared="144159898" ttl="0" queryrate="0.26666668" firewalled="no" responsive="yes"/>
<node id="40" ip="198.143.251.145" hostname="198.143.251.145" port="6346" pingtime="0" speed="0" numfilesshared="0"
numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="41" ip="216.28.53.110" host-
name="d366.as1.clmb.oh.voyager.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0"
queryrate="2.4" firewalled="no" responsive="yes"/> <node id="42" ip="35.10.128.13" hostname="null" port="6346" ping-
time="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/>
<node id="43" ip="24.182.226.194" hostname="c1399913-a.aurora1.co.home.com" port="6350" pingtime="0" speed="0" num-
filesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="44" ip="65.34.144.95"
hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"
firewalled="yes" responsive="no"/> <node id="45" ip="65.34.144.95" hostname="null" port="6347" pingtime="0" speed="0"
numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="46" ip="172.185.55.98"
hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"
firewalled="yes" responsive="no"/> <node id="47" ip="213.89.164.92" hostname="c213.89.164.92.cm-upc.chello.se" port="8080"
pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/>
<node id="48" ip="128.180.108.122" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numk-
bytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="49" ip="24.22.237.13" hostname="null"
port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes"
responsive="no"/> <node id="50" ip="172.149.94.44" hostname="AC955E2C.ipt.aol.com" port="6346" pingtime="0" speed="0"
numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="3.6666667" firewalled="no" responsive="yes"/> <node id="51"
ip="194.8.137.48" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0"
queryrate="0.0" firewalled="yes" responsive="no"/> <node id="52" ip="61.124.43.49" hostname="null" port="6346" ping-
time="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/>
<node id="53" ip="217.127.4.136" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0"
ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="54" ip="62.227.170.113" hostname="null" port="6346"
pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/>
<node id="55" ip="211.125.11.125" hostname="nttsitm02125.ppp.infoweb.ne.jp" port="6346" pingtime="6641" speed="0"
numfilesshared="1798" numkbytesshared="2871476" ttl="0" queryrate="3.5666666" firewalled="no" responsive="yes"/> <node
id="56" ip="207.253.15.171" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0"
ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="57" ip="24.128.136.238" hostname="bluebox.ne.mediaone.net"
port="5634" pingtime="469" speed="0" numfilesshared="1781" numkbytesshared="17128437" ttl="0" queryrate="2.8" fire-
walled="no" responsive="yes"/> <node id="58" ip="38.202.148.242" hostname="38.202.148.242" port="6346" pingtime="0"
speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node
id="59" ip="172.180.82.15" hostname="ACB4520F.ipt.aol.com" port="6346" pingtime="5718" speed="0" numfilesshared="1707"

numkbytesshared="6583391" ttl="0" queryrate="6.6" firewalled="no" responsive="yes"/> <node id="60" ip="62.211.34.118"
hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"
firewalled="yes" responsive="no"/> <node id="61" ip="62.211.34.56" hostname="62.211.34.56" port="6346" pingtime="8187"
speed="0" numfilesshared="1493" numkbytesshared="5498637" ttl="0" queryrate="8.1" firewalled="no" responsive="yes"/>
<node id="62" ip="24.15.99.105" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0"
ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="63" ip="4.43.170.31" hostname="lsanca1-ar10-170-
031.dsl.gtei.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="7.233333"
firewalled="no" responsive="yes"/> <node id="64" ip="213.113.34.230" hostname="null" port="6346" pingtime="0" speed="0"
numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="65" ip="152.66.72.106"
hostname="levedi.eet.bme.hu" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"
firewalled="no" responsive="no"/> <node id="66" ip="204.97.175.6" hostname="ip-204-97-175-6.dsl.logical.net" port="6346"
pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/>
<node id="67" ip="66.24.100.160" hostname="roc-66-24-100-160.rochester.rr.com" port="6347" pingtime="0" speed="0" num-
filesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="68" ip="203.164.224.235"
hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"
firewalled="yes" responsive="no"/> <node id="69" ip="213.122.130.3" hostname="null" port="6347" pingtime="0" speed="0"
numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="70" ip="129.116.34.46"
hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"
firewalled="yes" responsive="no"/> <node id="71" ip="24.132.48.74" hostname="node1304a.a2000.nl" port="6346" ping-
time="5984" speed="0" numfilesshared="1389" numkbytesshared="1447320" ttl="0" queryrate="5.1666665" firewalled="no"
responsive="yes"/> <node id="72" ip="24.19.89.146" hostname="c1408159-a.elnsng1.mi.home.com" port="6346" pingtime="3329"
speed="0" numfilesshared="1367" numkbytesshared="4293564113" ttl="0" queryrate="4.0" firewalled="no" responsive="yes"/>
<node id="73" ip="64.171.0.221" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0"
ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="74" ip="217.84.99.42" hostname="pD954632A.dip.t-
dialin.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" fire-
walled="no" responsive="no"/> <node id="75" ip="64.192.163.73" hostname="null" port="6346" pingtime="0" speed="0"
numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="76" ip="208.16.252.66"
hostname="sfk1-pm2-66.dial.up.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0"
queryrate="2.6666667" firewalled="no" responsive="yes"/> <node id="77" ip="195.22.71.141" hostname="null" port="6346"
pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/>
<node id="78" ip="213.64.170.130" hostname="h130n3fls34o896.telia.com" port="6346" pingtime="0" speed="0" numfi-
lesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="79" ip="63.215.218.213"
hostname="dialup-63.215.218.213.Dial1.Tampa1.Level3.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numk-

bytesshared="0" ttl="0" queryrate="1.2205753" firewalled="no" responsive="yes"/> <node id="80" ip="195.186.129.98"
hostname="null" port="6348" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"
firewalled="yes" responsive="no"/><node id="81" ip="198.82.58.190" hostname="null" port="6347" pingtime="0" speed="0"
numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="82" ip="65.193.144.169"
hostname="65.193.144.169" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"
firewalled="no" responsive="yes"/> <node id="83" ip="62.46.46.227" hostname="null" port="6346" pingtime="0" speed="0"
numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="84" ip="195.252.185.202"
hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"
firewalled="yes" responsive="no"/> <node id="85" ip="64.218.174.225" hostname="null" port="6346" pingtime="0" speed="0"
numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="86" ip="130.203.167.14"
hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"
firewalled="yes" responsive="no"/> <node id="87" ip="213.120.104.225" hostname="host213-120-104-225.btopenworld.com"
port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" re-
sponsive="no"/> <node id="88" ip="168.150.192.49" hostname="pm3a-49.dcn.davis.ca.us" port="6347" pingtime="0" speed="0"
numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="89" ip="62.158.121.41"
hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"
firewalled="yes" responsive="no"/> <node id="90" ip="64.229.195.104" hostname="HSE-MTL-ppp71025.qc.sympatico.ca"
port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" re-
sponsive="yes"/> <node id="91" ip="216.66.147.32" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0"
numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="92" ip="207.175.226.15" hostname="calnet3-
15.gtecablemodem.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"
firewalled="no" responsive="no"/> <node id="93" ip="65.28.134.98" hostname="65.28.134.98" port="6346" pingtime="0"
speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node
id="94" ip="213.93.15.203" hostname="e15203.upc-e.chello.nl" port="6346" pingtime="6344" speed="0" numfilesshared="1122"
numkbytesshared="3685535" ttl="0" queryrate="8.666667" firewalled="no" responsive="yes"/> <node id="95" ip="172.135.50.39"
hostname="AC873227.ipt.aol.com" port="6346" pingtime="29391" speed="0" numfilesshared="1114" numkbytesshared="5435742"
ttl="0" queryrate="1.6" firewalled="no" responsive="yes"/> <node id="96" ip="62.226.240.117" hostname="p3EE2F075.dip.t-
dialin.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" fire-
walled="no" responsive="no"/> <node id="97" ip="172.155.81.47" hostname="null" port="6346" pingtime="0" speed="0"
numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/><node id="98" ip="172.183.147.31"
hostname="ACB7931F.ipt.aol.com" port="6346" pingtime="21766" speed="0" numfilesshared="1101" numkbytesshared="4515832"
ttl="0" queryrate="2.0333333" firewalled="no" responsive="yes"/> <node id="99" ip="209.114.220.54" hostname="209.114.220.54"
port="6346" pingtime="1906" speed="0" numfilesshared="1088" numkbytesshared="3476180" ttl="0" queryrate="2.9333334"

firewalled="no" responsive="yes"/> <node id="100" ip="62.30.84.2" hostname="pc-62-30-84-2-hf.blueyonder.co.uk" port="6346" pingtime="344" speed="0" numfilesshared="1084" numkbytesshared="4904256" ttl="0" queryrate="10.0" firewalled="no" responsive="yes"/> <node id="101" ip="134.114.15.167" hostname="dorm167.resnet.nau.edu" port="6346" pingtime="3531" speed="0" numfilesshared="1066" numkbytesshared="4093828" ttl="0" queryrate="3.3666666" firewalled="no" responsive="yes"/> <node id="102" ip="212.120.98.167" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="103" ip="66.27.20.116" hostname="sc-66-27-20-116.socal.rr.com" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="104" ip="142.179.220.86" hostname="142.179.220.86" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="105" ip="64.31.3.35" hostname="host35.64-31-3.bignet.net" port="6347" pingtime="594" speed="0" numfilesshared="132" numkbytesshared="1228943" ttl="0" queryrate="0.20124774" firewalled="no" responsive="yes"/> <node id="106" ip="63.10.134.2" hostname="1Cust2.tnt1.phoenix2.az.da.uu.net" port="6346" pingtime="19297" speed="0" numfilesshared="996" numkbytesshared="3386247" ttl="0" queryrate="1.1333333" firewalled="no" responsive="yes"/> <node id="107" ip="209.211.110.110" hostname="PAGDI110.rica.net" port="6346" pingtime="23344" speed="0" numfilesshared="980" numkbytesshared="3950106" ttl="0" queryrate="0.46666667" firewalled="no" responsive="yes"/> <node id="108" ip="199.247.232.41" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="109" ip="169.226.226.139" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="110" ip="24.22.252.88" hostname="ct40860-a.lafayt1.in.home.com" port="6346" pingtime="1719" speed="0" numfilesshared="947" numkbytesshared="6116594" ttl="0" queryrate="5.9666667" firewalled="no" responsive="yes"/> <node id="111" ip="217.80.203.72" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="112" ip="24.30.229.194" hostname="va-24-30-229-194.va.mediaone.net" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="113" ip="64.167.76.88" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="114" ip="213.93.20.84" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="115" ip="155.58.208.161" hostname="155.58.208.161" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="116" ip="24.176.97.6" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="117" ip="129.252.162.213" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="118" ip="62.30.96.108" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="119" ip="136.142.149.169" hostname="null" port="6346" pingtime="0" speed="0" numfi-

lesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="120" ip="64.123.118.133" hostname="adsl-64-123-118-133.dsl.wchtks.swbell.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="4.1666665" firewalled="no" responsive="yes"/> <node id="121" ip="64.194.8.181" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="122" ip="217.96.198.151" hostname="pa151.swietochlowice.sdi.tpnet.pl" port="6346" pingtime="2407" speed="0" numfilesshared="826" numkbytesshared="1512158" ttl="0" queryrate="0.6" firewalled="no" responsive="yes"/> <node id="123" ip="4.41.165.147" hostname="lsanca1-ar15-165-147.dsl.gtei.net" port="6346" pingtime="7562" speed="0" numfilesshared="823" numkbytesshared="4184699" ttl="0" queryrate="4.8333335" firewalled="no" responsive="yes"/> <node id="124" ip="128.39.144.111" hostname="hybel111.grm.hia.no" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="8.366667" firewalled="no" responsive="yes"/> <node id="125" ip="63.168.40.218" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="126" ip="208.61.89.154" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="127" ip="203.165.202.158" hostname="cj3078477-a.fkoka1.ky.home.ne.jp" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="128" ip="193.91.151.88" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="129" ip="208.248.193.211" hostname="red-pc-211.cablenet-va.com" port="6347" pingtime="563" speed="0" numfilesshared="807" numkbytesshared="4293843349" ttl="0" queryrate="3.3" firewalled="no" responsive="yes"/> <node id="130" ip="24.91.15.159" hostname="h00207802d202.ne.mediaone.net" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="131" ip="172.156.36.51" hostname="AC9C2433.ipt.aol.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="132" ip="151.200.9.209" hostname="adsl-151-200-9-209.dc.adsl.bellatlantic.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="133" ip="65.66.22.130" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="134" ip="209.122.154.71" hostname="209.122.154.71" port="6346" pingtime="1594" speed="0" numfilesshared="823" numkbytesshared="3957324" ttl="0" queryrate="3.7666667" firewalled="no" responsive="yes"/> <node id="135" ip="64.252.0.47" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="136" ip="195.116.155.86" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="137" ip="152.7.6.190" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="138" ip="62.155.143.193" hostname="p3E9B8FC1.dip.t-dialin.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no"

responsive="no"/> <node id="139" ip="208.224.174.170" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="140" ip="195.34.150.112" hostname="TK150112.tuwien.teleweb.at" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="141" ip="24.23.229.49" hostname="c1242410-a.alntn1.tx.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="142" ip="209.226.108.15" hostname="HSE-MTL-ppp4832.qc.sympatico.ca" port="6699" pingtime="8250" speed="0" numfilesshared="731" numkbytesshared="3257230" ttl="0" queryrate="0.56666666" firewalled="no" responsive="yes"/> <node id="143" ip="130.60.71.144" hostname="null" port="5634" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="144" ip="203.173.238.66" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="145" ip="212.127.164.83" hostname="qn-212-127-164-83.quicknet.nl" port="6346" pingtime="734" speed="0" numfilesshared="726" numkbytesshared="4293960538" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="146" ip="65.81.227.139" hostname="adsl-81-227-139.bhm.bellsouth.net" port="7200" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="3.1" firewalled="no" responsive="yes"/> <node id="147" ip="130.83.236.244" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="148" ip="128.175.11.141" hostname="host11-141.student.udel.edu" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="149" ip="129.64.167.46" hostname="dhcp-129-64-167-46.dorm.brandeis.edu" port="6346" pingtime="703" speed="0" numfilesshared="720" numkbytesshared="2585265" ttl="0" queryrate="12.733334" firewalled="no" responsive="yes"/> <node id="150" ip="209.8.198.82" hostname="ppp18.ts5-2.NewportNews.visi.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="151" ip="207.16.59.104" hostname="d104.htonline.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="3.5666666" firewalled="no" responsive="yes"/> <node id="152" ip="63.16.114.32" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="153" ip="213.132.141.54" hostname="cable-213-132-141-54.upc.chello.be" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="154" ip="24.188.186.66" hostname="ool-18bcba42.dyn.optonline.net" port="6345" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="4.5333333" firewalled="no" responsive="yes"/> <node id="155" ip="213.73.145.86" hostname="qn-213-73-145-86.quicknet.nl" port="6346" pingtime="14953" speed="0" numfilesshared="695" numkbytesshared="10647105" ttl="0" queryrate="3.1666667" firewalled="no" responsive="yes"/> <node id="156" ip="63.14.67.23" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="157" ip="211.133.170.189" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="158"

ip="24.8.233.58" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="159" ip="64.229.77.90" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="160" ip="208.180.96.127" hostname="cdm-96-127-bent.cox-internet.com" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="161" ip="213.141.54.42" hostname="null" port="3337" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="162" ip="172.184.10.6" hostname="ACB80A06.ipt.aol.com" port="6347" pingtime="7125" speed="0" numfilesshared="685" numkbytesshared="4294303978" ttl="0" queryrate="1.9666667" firewalled="no" responsive="yes"/> <node id="163" ip="65.81.138.31" hostname="adsl-81-138-31.asm.bellsouth.net" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="164" ip="24.22.21.236" hostname="c309208-a.chmpgn1.il.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="165" ip="128.122.114.84" hostname="MATADOR.PSYCH.NYU.EDU" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="166" ip="209.101.24.76" hostname="209.101.24.76" port="6346" pingtime="1531" speed="0" numfilesshared="665" numkbytesshared="909820" ttl="0" queryrate="5.633333" firewalled="no" responsive="yes"/> <node id="167" ip="24.232.30.130" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="168" ip="217.80.41.137" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="169" ip="172.153.23.191" hostname="AC9917BF.ipt.aol.com" port="6346" pingtime="11312" speed="0" numfilesshared="648" numkbytesshared="4292943549" ttl="0" queryrate="1.3" firewalled="no" responsive="yes"/> <node id="170" ip="216.114.215.161" hostname="kato-asc12-cs-33.dial.hickorytech.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="171" ip="64.228.87.29" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="172" ip="134.124.42.2" hostname="TIGER.umsl.edu" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="173" ip="24.77.10.157" hostname="h24-77-10-157.gv.shawcable.net" port="6347" pingtime="625" speed="0" numfilesshared="630" numkbytesshared="4292933233" ttl="0" queryrate="3.4333334" firewalled="no" responsive="yes"/> <node id="174" ip="64.168.22.248" hostname="64.168.22.248" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="1562.4999" firewalled="no" responsive="no"/> <node id="175" ip="172.149.169.188" hostname="AC95A9BC.ipt.aol.com" port="6347" pingtime="1157" speed="0" numfilesshared="616" numkbytesshared="4293193225" ttl="0" queryrate="3.1" firewalled="no" responsive="yes"/> <node id="176" ip="212.204.186.212" hostname="cc14313-a.groni1.gr.nl.home.com" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="yes"/> <node id="177" ip="208.180.92.149" hostname="cdm-92-149-fran.cox-internet.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0"

firewalled="no" responsive="no"/> <node id="178" ip="24.17.173.215" hostname="c1329063-a.rchdsn1.tx.home.com" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="179" ip="24.179.47.10" hostname="c391643-a.alntn1.tx.home.com" port="6346" pingtime="750" speed="0" numfilesshared="617" numkbytesshared="4294450691" ttl="0" queryrate="1.4333333" firewalled="no" responsive="yes"/> <node id="180" ip="64.229.131.231" hostname="HSE-Ottawa-ppp157814.sympatico.ca" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="181" ip="217.80.27.164" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="182" ip="172.172.94.84" hostname="ACAC5E54.ipt.aol.com" port="6346" pingtime="13047" speed="0" numfilesshared="601" numkbytesshared="1642675" ttl="0" queryrate="2.9333334" firewalled="no" responsive="yes"/> <node id="183" ip="204.168.44.27" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="184" ip="172.144.149.163" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="185" ip="195.184.174.108" hostname="gep17-3783.miskcatv.broadband.hu" port="6347" pingtime="22672" speed="0" numfilesshared="590" numkbytesshared="4294666953" ttl="0" queryrate="2.5666666" firewalled="no" responsive="yes"/> <node id="186" ip="24.51.234.12" hostname="ma-northadams2-524.nad.adelphia.net" port="6346" pingtime="3078" speed="0" numfilesshared="583" numkbytesshared="4286930" ttl="0" queryrate="3.5666666" firewalled="no" responsive="yes"/> <node id="187" ip="195.215.251.7" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="188" ip="128.153.151.91" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="189" ip="213.66.185.143" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="190" ip="24.184.47.31" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="191" ip="62.30.171.27" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="192" ip="172.181.160.13" hostname="ACB5A00D.ipt.aol.com" port="6346" pingtime="21343" speed="0" numfilesshared="578" numkbytesshared="2469287" ttl="0" queryrate="2.0666666" firewalled="no" responsive="yes"/> <node id="193" ip="64.168.100.221" hostname="adsl-64-168-100-221.dsl.scrm01.pacbell.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="194" ip="172.130.91.154" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="195" ip="63.227.23.53" hostname="63.227.23.53" port="6346" pingtime="3781" speed="0" numfilesshared="571" numkbytesshared="2076269" ttl="0" queryrate="20.466667" firewalled="no" responsive="yes"/> <node id="196" ip="66.31.96.69" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="197" ip="64.229.245.145" hostname="null" port="6346"

pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/>
<node id="198" ip="206.172.57.173" hostname="null" port="6347" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/> <node id="199" ip="24.221.153.189" hostname="cpe-24-221-153-189.az.sprintbbd.net" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="no" responsive="no"/> <node id="200" ip="172.132.36.228" hostname="null" port="6346" pingtime="0" speed="0" numfilesshared="0" numkbytesshared="0" ttl="0" queryrate="0.0" firewalled="yes" responsive="no"/>

Samples of GraphT2 Connections

<edge startnode="28" endnode="8168"/> <edge startnode="33" endnode="8169"/>
<edge startnode="57" endnode="8170"/> <edge startnode="33" endnode="8171"/>
<edge startnode="33" endnode="8172"/> <edge startnode="33" endnode="8173"/>
<edge startnode="33" endnode="8174"/> <edge startnode="33" endnode="8175"/>
<edge startnode="33" endnode="8176"/> <edge startnode="33" endnode="8177"/>
<edge startnode="33" endnode="8178"/> <edge startnode="33" endnode="8179"/>
<edge startnode="33" endnode="8180"/> <edge startnode="33" endnode="8181"/>
<edge startnode="33" endnode="8182"/> <edge startnode="33" endnode="8183"/>
<edge startnode="33" endnode="357"/> <edge startnode="33" endnode="8184"/>
<edge startnode="57" endnode="772"/> <edge startnode="9" endnode="8185"/>
<edge startnode="120" endnode="8186"/> <edge startnode="78" endnode="8187"/>
<edge startnode="78" endnode="5060"/> <edge startnode="78" endnode="8188"/>
<edge startnode="78" endnode="8189"/> <edge startnode="78" endnode="8190"/>
<edge startnode="78" endnode="8191"/> <edge startnode="78" endnode="8192"/>
<edge startnode="78" endnode="1642"/> <edge startnode="78" endnode="1176"/>
<edge startnode="78" endnode="3742"/> <edge startnode="4" endnode="8193"/>
<edge startnode="4" endnode="8194"/> <edge startnode="4" endnode="8195"/>
<edge startnode="4" endnode="3406"/> <edge startnode="4" endnode="8196"/>
<edge startnode="4" endnode="8197"/> <edge startnode="4" endnode="8198"/>
<edge startnode="4" endnode="8199"/> <edge startnode="4" endnode="8200"/>
<edge startnode="4" endnode="7410"/> <edge startnode="103" endnode="8201"/>
<edge startnode="103" endnode="8202"/> <edge startnode="103" endnode="8203"/>
<edge startnode="103" endnode="8204"/> <edge startnode="103" endnode="8205"/>
<edge startnode="103" endnode="8206"/> <edge startnode="103" endnode="8207"/>
<edge startnode="103" endnode="2465"/> <edge startnode="103" endnode="8208"/>
<edge startnode="103" endnode="8209"/> <edge startnode="67" endnode="3484"/>
<edge startnode="67" endnode="8210"/> <edge startnode="67" endnode="8211"/>
<edge startnode="67" endnode="8212"/> <edge startnode="67" endnode="8213"/>
<edge startnode="67" endnode="8214"/> <edge startnode="67" endnode="5655"/>
<edge startnode="67" endnode="8215"/> <edge startnode="67" endnode="8216"/>
<edge startnode="67" endnode="8217"/> <edge startnode="141" endnode="8218"/>
<edge startnode="141" endnode="5758"/> <edge startnode="141" endnode="8219"/>
<edge startnode="141" endnode="8220"/> <edge startnode="141" endnode="8221"/>
<edge startnode="141" endnode="8222"/> <edge startnode="141" endnode="8223"/>
<edge startnode="141" endnode="8224"/> <edge startnode="141" endnode="8225"/>
<edge startnode="141" endnode="8226"/> <edge startnode="33" endnode="8227"/>
<edge startnode="124" endnode="8182"/> <edge startnode="38" endnode="8228"/>
<edge startnode="38" endnode="8229"/> <edge startnode="38" endnode="8230"/>
<edge startnode="38" endnode="4458"/> <edge startnode="38" endnode="8231"/>
<edge startnode="38" endnode="8232"/> <edge startnode="38" endnode="8233"/>
<edge startnode="38" endnode="8234"/> <edge startnode="38" endnode="8235"/>
<edge startnode="38" endnode="8236"/> <edge startnode="18" endnode="8182"/>
<edge startnode="6" endnode="2548"/> <edge startnode="6" endnode="8237"/>
<edge startnode="6" endnode="8238"/> <edge startnode="6" endnode="8239"/>
<edge startnode="6" endnode="6565"/> <edge startnode="6" endnode="8240"/>
<edge startnode="6" endnode="8241"/> <edge startnode="6" endnode="8242"/>
<edge startnode="6" endnode="8243"/> <edge startnode="6" endnode="8244"/>
<edge startnode="112" endnode="88"/> <edge startnode="112" endnode="8245"/>

<edge startnode="112" endnode="8246"/> <edge startnode="112" endnode="8247"/>
<edge startnode="112" endnode="8248"/> <edge startnode="112" endnode="8249"/>
<edge startnode="112" endnode="8250"/> <edge startnode="112" endnode="4015"/>
<edge startnode="112" endnode="8251"/> <edge startnode="112" endnode="8252"/>
<edge startnode="24" endnode="8253"/> <edge startnode="24" endnode="8254"/>
<edge startnode="24" endnode="5171"/> <edge startnode="24" endnode="8255"/>
<edge startnode="24" endnode="8256"/> <edge startnode="24" endnode="8257"/>
<edge startnode="24" endnode="3486"/> <edge startnode="24" endnode="3980"/>
<edge startnode="24" endnode="8258"/> <edge startnode="24" endnode="8259"/>
<edge startnode="149" endnode="8260"/> <edge startnode="146" endnode="8261"/>
<edge startnode="30" endnode="8262"/> <edge startnode="30" endnode="8263"/>
<edge startnode="30" endnode="8264"/> <edge startnode="30" endnode="8265"/>
<edge startnode="30" endnode="8266"/> <edge startnode="30" endnode="8267"/>
<edge startnode="30" endnode="8268"/> <edge startnode="30" endnode="8269"/>
<edge startnode="30" endnode="3008"/> <edge startnode="30" endnode="385"/>
<edge startnode="154" endnode="8186"/> <edge startnode="88" endnode="8270"/>
<edge startnode="88" endnode="2080"/> <edge startnode="88" endnode="8271"/>
<edge startnode="88" endnode="8272"/> <edge startnode="88" endnode="8273"/>
<edge startnode="88" endnode="6040"/> <edge startnode="88" endnode="8274"/>
<edge startnode="88" endnode="8275"/> <edge startnode="88" endnode="8276"/>
<edge startnode="88" endnode="330"/> <edge startnode="79" endnode="8277"/>
<edge startnode="79" endnode="8278"/> <edge startnode="79" endnode="8279"/>
<edge startnode="79" endnode="8280"/> <edge startnode="79" endnode="8281"/>
<edge startnode="79" endnode="8282"/> <edge startnode="79" endnode="8283"/>
<edge startnode="79" endnode="2092"/> <edge startnode="79" endnode="2568"/>
<edge startnode="79" endnode="8284"/> <edge startnode="79" endnode="8285"/>
<edge startnode="79" endnode="1507"/> <edge startnode="79" endnode="8286"/>
<edge startnode="79" endnode="8287"/> <edge startnode="72" endnode="8288"/>
<edge startnode="72" endnode="8289"/> <edge startnode="72" endnode="8290"/>
<edge startnode="7" endnode="8291"/> <edge startnode="7" endnode="8292"/>
<edge startnode="7" endnode="8293"/> <edge startnode="7" endnode="8294"/>
<edge startnode="7" endnode="8295"/> <edge startnode="7" endnode="7251"/>
<edge startnode="7" endnode="5734"/> <edge startnode="7" endnode="8296"/>
<edge startnode="7" endnode="8297"/> <edge startnode="7" endnode="8298"/>
<edge startnode="90" endnode="8299"/> <edge startnode="90" endnode="8300"/>
<edge startnode="90" endnode="8301"/> <edge startnode="90" endnode="8302"/>
<edge startnode="90" endnode="8303"/> <edge startnode="90" endnode="8304"/>
<edge startnode="90" endnode="486"/> <edge startnode="90" endnode="8305"/>
<edge startnode="90" endnode="8306"/> <edge startnode="90" endnode="8307"/>
<edge startnode="130" endnode="8308"/> <edge startnode="130" endnode="8309"/>
<edge startnode="130" endnode="8310"/> <edge startnode="130" endnode="2359"/>
<edge startnode="130" endnode="8311"/> <edge startnode="130" endnode="8230"/>
<edge startnode="130" endnode="2197"/> <edge startnode="130" endnode="7485"/>
<edge startnode="130" endnode="8312"/> <edge startnode="130" endnode="5160"/>
<edge startnode="129" endnode="8313"/> <edge startnode="153" endnode="8314"/>
<edge startnode="153" endnode="7235"/> <edge startnode="153" endnode="8315"/>
<edge startnode="153" endnode="5702"/> <edge startnode="153" endnode="8316"/>
<edge startnode="153" endnode="3787"/> <edge startnode="153" endnode="8317"/>
<edge startnode="153" endnode="8318"/> <edge startnode="153" endnode="4656"/>
<edge startnode="153" endnode="8319"/> <edge startnode="131" endnode="8320"/>

<edge startnode="129" endnode="8321"/> <edge startnode="131" endnode="8322"/>
<edge startnode="131" endnode="8323"/> <edge startnode="131" endnode="8324"/>

BIODATA OF STUDENT

Hassan Barjini received the BS degree in statistic and information from the Institute of Statistic and Information Tehran, Iran 1973, and the MS Degree in computer science, from University of Detroit Michigan (USA) 1979. He started as a lecturer in Computer Science department in Imam Khomeini International University Qazvin *(www.ikiu.ac.ir)*, Iran since 1986. In 2007 he joined the faculty of computer science and information technology (FSKTM), University Putra Malaysia (UPM) to pursue his PhD. His current research interest include Parallel and distributed algorithms and peer-to-peer content locations.

LIST OF PUPLICATIONS

Publications in Journals

- Hassan Barjini, Mohamed Othman, Hamidah Ibrahim, and Nur Izura Udzir "Shortcoming, Problems and Analytical Comparison for Flooding-Based Search Techniques in Unstructured P2P Networks", Peer-to-peer Network Applications, 5(1): 1-13, 2012. (IF=0.417).
- Hassan Barjini, Mohamed Othman, Hamidah Ibrahim, and Nur Izura Udzir "HybridFlood: Minimizing the effects of redundant messages and Maximizing search efficiency of unstructured Peer-to-peer networks", Cluster Computing, 2011,(under second revision) (IF=0.679).

Publications in Book Chapters

- Hassan Barjini, Mohamed Othman, and Hamidah Ibrahim "An efficient hybridflood searching algorithm for unstructured peer-to-peer networks", LNCS 6377, 2011. pp.173-180, 2011.
- Hassan Barjini, Mohamed Othman, Hamidah Ibrahim, and Nur Izura Udzir "Analytical Studies and Experimental Examines for Flooding-Based Search Algorithms" Accepted for LNCS will publish at September 2012. ICICA 2012

Publications in Proceeding

- Hassan Barjini, Mohamed Othman, Hamidah Ibrahim, and Nur Izura Udzir "QuickFlood: An Efficient Search Algorithm for Unstructured Peer-to-Peer Networks", CCIS 136 The International conference on Networked Digital Technologies 2011, pp. 81-90

- Hassan Barjini, Mohamed Othman, and Hamidah Ibrahim "SmoothFlood: Decreasing Redundant Messages and Increasing Search Quality of Service in Peer-to-Peer Networks", Information Retrieval & Knowledge Management, (CAMP), 2010 International Conference, pp. 138-142, 2010

Award

- A silver medal in PRPI11 (Exhibition in Design, Research and Innovation) University of Putra Malaysia (UPM) 2011.